摘 要

传统的无线通信系统的设计主要围绕硬件来展开，其产品的开发和制造也只是针对专门的标准。这样的开发模式已经不能满足当今通信系统的要求，并严重地阻碍了无线通信技术的发展。新一代无线通信系统，应该具备通用的硬件平台，能够兼容多种通信体制，具有软件可重构的能力，能灵活地根据网络环境，动态构建不同的通信系统，以实现不同通信体制之间的无缝切换。

为了解决上述问题，软件通信结构(SCA)应运而生。SCA(Software Communication Architecture)是美军为开发未来的战术无线通信系统而发布的。它作为软件无线电(SDR)的一种实现方式，为无线通信系统的设计和开发提供了详细的规范，建立了独立于设备的结构框架。为了保证软件和硬件的可移植性和可配置性，SCA 详细定义了无线电系统的硬件结构、软件结构、安全结构以及公共服务和配置。基于 SCA 的无线通信技术研究，旨在摆脱传统的面向用途的设计思想，通过一种高度模块化的通用信号处理平台，把各种通信装备提供的业务从基于硬件特性的方式中解放出来，通过装载不同的软件来动态配置系统的功能，实现系统硬件平台的通用化、模块化和软件的可移植性，从而满足新一代无线通信统一平台的需要。

本文描述了 SCA 的一些基本的核心规则，得出了自己对 SCA 的理解。结合统一平台，对通信体制识别、核心框架实现、CORBA(Common Object Request Broker Architecture)的应用、波形组件的开发以及软件的构建等关键技术进行了较为详细的阐述。在研究过程中，严格遵循 SCA 规范，采用嵌入式处理模块与通用 PC 相结合的架构，运用面向对象的思想对平台软件的重构性进行研究。通过 Winpcap 开源库，实现了 PC 与嵌入式处理模块之间的高速数据通信。在 PC 环境下，通过 Visual C++，成功实现了新一代无线通信系统的接收端和发射端在 GSM、WCDMA 和 WiMAX 等多种通信体制环境下的软件重构。最后，给出了研究过程中所涉及到的部分程序代码。

关键字：新一代无线通信；统一平台；软件通信结构；软件可重构

ABSTRACT

The focus of designing conventional wireless communication system is the hardware. And the manufacture and development of the products are in accordance with specialized standard. This kind of development mode can not meet the requirements of communication system nowadays. Moreover, it hinders the further development of wireless communication technique severely. The wireless communication system of next generation is supposed to possess unified hardware platform, be compatible with different communication standards, be software reconfigurable, and be adaptive to the network environment by constructing different systems dynamically to implement seamless handover among different standards.

In order to solve the problems above, software communication architecture (SCA) emerged. SCA is developed by the U.S. army in the purpose of developing future tactic wireless communication system. As one way to realize software defined radio (SDR), it provides detailed specifications to the design and development of wireless communication system, and establishes the framework which is independent of the hardware. To ensure the transplantable and reconfigurable capability, SCA defines the framework of hardware, software and security in detail.Based on the research of SCA in the background of wireless communication, in order to get rid of the design idea oriented for application, we can realize the generalization, modularization and software transplantable capability of hardware platform by the highly modularized common signal processing platform to free the services of communication equipment from the hardware characteristic and loading different software to configure to systems dynamically, hereby satisfying the requirements of the new generation wireless communication system.

This article described the SCA some basic core rule, has obtained itself to SCA the understanding. The union unified the platform to the correspondence system recognition, the realization of the core framework, application of CORBA (Common Object Request Broker Architecture), the development of waveform component as well as software's construction and so on , have carried on a more detailed elaboration. In the research process, follows the SCA standard strictly, uses construction which the embedded processing module and general PC unify, utilizes the object-oriented thought to conduct the research to platform software's restructuring. Opens the source

storehouse through Winpcap, has realized PC and between the embedded processing module high speed data correspondence. Under the PC environment, through Visual C++, has realized the new generation wireless communication system's receiving end and the transmitting end in GSM, WCDMA and WiMAX and so on under many kinds of correspondence system environment software restructurin, and gives part of the relevant codes during the research.

Key words: New Generation Wireless Communication; The Unified Platform; Software Communication Architecture; Software Reconfiguration

学位论文独创性声明

本人声明所呈交的硕士学位论文《第一代无线通信统一软件无线电相关研究》是我个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的地方外，论文中不包含其他人已经发表或撰写过的研究成果。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。

学位论文作者签名：李晓宝　　签字日期：2009.6.4

导师签名：[illegible]　　签字日期：2009.6.3

学位论文使用授权书

本人完全了解重庆大学有关保留、使用学位论文的规定。本人完全同意《中国博士学位论文全文数据库、中国优秀硕士学位论文全文数据库出版章程》（以下简称“章程”），愿意将本人的___士学位论文《第一代无线通信统一软件无线电相关研究》提交中国学术期刊（光盘版）电子杂志社（CNKI）在《中国博士学位论文全文数据库》、《中国优秀硕士学位论文全文数据库》以及《重庆大学博硕学位论文全文数据库》中全文发表。《中国博士学位论文全文数据库》、《中国优秀硕士学位论文全文数据库》可以以电子、网络及其他数字媒体形式公开出版，并同意编入CNKI《中国知识资源总库》，在《中国博硕士学位论文评价数据库》中使用和在互联网上传播，同意按“章程”规定享受相关权益和承担相应义务。本人授权重庆大学可以采用影印、缩印或其他复制手段保存论文，可以公开论文的全部或部分内容。

作者签名：李晓宝　　导师签名：[illegible]

2009年6月3日

备注：审核通过的**涉密论文不得签署“授权书”**，须填写以下内容：

该论文属于涉密论文，其密级是_____，涉密期限至_____年___月___日。

说明：本声明及授权书**必须**装订在提交的学位论文最后一页。

1 绪 论

1.1 课题的研究背景

1.1.1 问题的提出

近年来，3G 技术逐渐步入人们的生活，4G 的一些标准的制定也正在如火如荼地进行。当新技术给大家带来便捷的同时，我们也慢慢发现目前的无线通信系统面临着诸多困难：系统的设计主要围绕硬件来展开，产品只是针对特定的标准来开发和制造，每出台一种新标准都要重新制造新的专用芯片；多种通信体制不兼容、网络覆盖范围有限、运营商初期投资过大、扩展升级能力有限等。其结果不仅限制了新技术和新业务的使用，而且给制造商、运营商带来更大的投资风险，给用户带来诸多不便。同时，由于各国的经济利益，全球标准的统一又难以实现。标准的种类在不断增加，造成频率分配与管理更为困难、需要的频率资源增加、多频多模手持机制造困难、成本增加。

一般而言，传统的通信设备仅能满足单模式和窄频段的通信需求。不同通信系统之间很难实现信息交互，通常只能采用组合式无线电的方式来实现，通过激活硬件的不同状态来完成不同系统之间的切换。这种组合无线电的体系结构需要不断的硬件扩展才能支持更多的系统，结果导致系统尺寸的无限膨胀。另外，随着技术的发展，用户不得不因为设备的"落后"而更换通信设备。这些问题严重地限制了用户对通信业务的功能性需求[10]。

新一代无线通信系统，应该能够兼容多种通信体制，具有软硬件可重构的能力，能灵活地根据网络环境，通过软件控制并调动各个模块组件，动态构建不同的通信系统，达到无缝切换模式的目的。这种无线通信系统与传统无线通信系统的不同之处，在于它以软件无线电技术为支撑，采用软件定义系统功能的方式，勿须引入新的硬件，即可改变运行特性或参数。如工作频率范围、调制类型、信道编码方式、带宽、最大发射或输出功率和体制协议等。

1.1.2 国内外研究现状

2003 年 7 月，IEEE Communication Magazine 集中报导了欧洲走向 4G 的研发情况[1]，发现随着社会的进步和通信技术的发展，要求骨干网提供多种不同的接入技术，使其能连接大量各种类型的终端，以满足人们的需求。在各种网络环境中，从微微网到蜂窝网，以及卫星网等，每一种无线接入技术都能在基于公共的 IP 平台上独立地演进，以支持端到端的 IP 连接。面临的挑战是核心网如何根据用户的需求和市场的演进方向提供各种无缝的无线接入技术，以及无线通信系统如何灵活配置自己，达到对网络环境的适应要求。

为此，人们提出无线通信系统应该具有可重构性，使得它成为一种灵活实现各种无线接入方式的手段，能在各种不同的环境中达到持久最佳的连接，并可优化使用稀有的频谱资源以及最有效的动态频谱分配。例如，研究认为典型的 B3G（Beyond 3 Generation）无线通信系统可同时采用一些不同的接入技术：蜂窝系统技术、无线局域网技术以及广播接入技术。对于跨越这些异种接入网络的全 IP 的应用，应具有统一的网络连接层提供 IP 连网技术。通常，这种无线通信系统与传统无线通信系统的不同之处，在于它采用软件定义系统功能，提供易于应用开发的环境，勿须引入新的硬件，即可改变运行特性或参数，如工作频率范围、调制类型、带宽、最大发射或输出功率和体制协议。这使得一个无线通信系统就能通过可重构，实现多种调制、编码和接入协议下的业务功能。

目前，国内外电子信息领域对可重构技术在电子器件和芯片的研究和应用方面已经比较普遍，例如获得较为广泛研究的有“可重构 RF”[2]、“可重构 FPGA”[3]、“可重构 DSP”[4]和“可重构企业信息系统”[6]等。这些研究主要针对的是进行数字信号处理的基本单元，通过设计实现其内部电路，达到赋予这些器件可重构能力的目的。如何采用这些具有可重构能力的器件，构作可重构的电子信息系统，尤其是针对无线通信网络和无线通信系统可重构的研究还不充分。

文献［6］提出了一种适于有线通信分布式系统的系统重构算法,它根据出发条件分别通过重新分配进程来消除节点失效,或通过转移进程来均衡系统负载。每隔一段较长的时间给出统计总表,对系统硬件的更新换代给出建议。保证了当分布式系统的某些部分失效时,整个系统能够继续正常运行,同时还保证调整系统负载,使 CPU 负载和网络通信量相对平衡。该文献针对的是一个假定的有线通信的分布式系统,讨论在其上实现可重构性的算法，对无线通信系统具有一定的借鉴意义。

文献[7]针对宽带无线接入和第三代无线通信系统的需求，提出了一种兼容 WLAN（Wireless Local Area Network）和 WCDMA（Wideband Code Division Multiple Access）两种无线通信系统的可重构软件无线电结构。分析了混合系统中可以重用的功能模块，如 FFT 模块，提出了一些解决混合系统的频率偏移、相位噪声、直流偏移、I/Q 失衡和非线性等失配问题的方法，并给出了这些方法的计算机仿真验证结果。

文献[8]采用混合可编程框架结构，提出并建立了一个承载 IEEE 802.11 WLAN 和 PHS（Personal Handy Phone System）两种体制，支持体制间切换和空中下载的实验系统。该系统由 CPU、DSP、FPGA 和辅助设备搭建而成，其 MAC 层功能在 CPU 上完成，PHY 层功能在 DSP 上完成，高速数字信号处理在 FPGA 上完成，采用了 VxWorks 操作系统。通过对系统进行发射机频谱分析、相关检测分析和误帧率分析，评估了其 PHY 层性能；通过对系统进行吞吐量和 DSP 负载

分析，评估了其 MAC 层性能。验证了 IEEE 802.11 WLAN 和 PHS 在该系统上实现的可行性，并发现专用可编程硬件芯片对于 IEEE 802.11 WLAN 严格的实时性要求是必须的。

文献[9]采用一种新的六端口结构的波导设备构作了软件定义无线电试验接收机平台。该平台可运行在 22GHz～26GHz 频段，支持 QPSK 和 16QAM 调制方式。其射频前端由低噪声放大器（LNA）、带通滤波器（BPF）、六端口连接器和 4 个功率检测器构成。从功率检测器出来的信号通过一组带通滤波器和基带放大器后，进入 DSP 部分完成基带信号处理，如解调和解码。基于六端口技术或多端口解调器，完成了新的软件定义无线电设计：直接解调架构，可以将信号直接从射频下变频到基带进行处理，使得接收机平台可运行在毫米波频段和宽带多模框架内。通过系统级的仿真和原型电路的搭建，验证了将软件定义无线电和六端口设备结合起来，可以获得灵活的系统配置、较少的系统开发工作量、潜在的系统可重用性。

从上述文献所反映出的研究情况来看，将软件无线电技术引入软硬件可重构无线通信系统的实现研究已经逐步兴起，其表现如下：

① 多模无线通信系统逐步受到关注，移动通信体制和无线接入体制的融合需要软件无线电技术的介入；

② 采用现有商用可编程器件和设备搭建起来的多模无线通信试验系统集中反映了 2G / 3G 系统的需求；

③ 可重构方法研究开始从可编程器件本身，走向出可编程器件构建的平台系统；

④ 采用 FPGA 和 DSP 组合，快速总线配合 CPU 的结构，是建立多模无线通信平台的重要而有效的手段；

⑤ 一些实验系统开始采用材料工业所研究产生的新成果、新产品，如六端口连接器，使得传统软件无线电面临的难题，如直接射频变换，开始得到解决。

1.1.3 软件无线电

面对无线通信系统软件可重构的发展趋势，各个国家对其进行了大量的研究。大部分专家认为，"软件无线电"(Software Defined Radio ，简称 SDR)将是一个解决全球无线通信需求的方案。它将成为未来无线通信设备设计的核心所在。

SDR的基本宗旨是：利用数字信号处理技术代替现在主要的模拟信号处理，利用智能天线、宽带RF器件、宽带模数转换器(ADC)及数模转换器(DAC)，通过通用可编程处理器实现IF、基带及比特流处理。因为用可重新编程的软件代替了硬件模拟电路，通过动态配置射频、中频、AD/DA、FPGA、DSP硬件和算法，并将软件对象分配到硬件组件中，使得软件无线电可以在线改变自己的特性。

SDR以软件方式实现各种空中接口，提供灵活的无线通信方式以便于实现灵活的传输机制、协议和应用。它将模块化、标准化的硬件功能单元通过一个通用的硬件平台连接起来，并且能够通过软件加载，动态组建各种不同的无线通信系统。这样，无线通信新系统、新产品的开发将逐步转到软件上来，而无线通信产业的产值将越来越多地体现在软件上。由此可见，SDR为我们新一代的无线通信技术的发展指明了方向，并提供了强有力的技术支持。随着近年SDR的迅猛发展，人们对使用SDR的无线通信系统提出了如下的要求：

① SDR 设备可简便快速的重新编程和重新设定，以支持任意传输形式的应用和在任何频率上的传输和或者接收。

② 提高兼容业务的能力，不仅支持传统的业务而且支持新业务，可以空中下载新软件，保证获得新业务的服务。

③ 支持多标准的能力，支持多频段多标准的无线通信系统。SDR 运营商能在基本不更换基站硬件的条件下，实现系统的版本更新、标准更新及升级换代。通过软件重构来定义出一个新的无线通信系统。

在计算机互联网时代，TCP/IP 的出现屏蔽了异构网络(上百种不同物理层结构的网络) 的差异，而向上层提供统一的服务，真正的实现了网络的互联，从而使 Internet 的发展有了质的飞跃。同样在无线通信领域，需要软件无线电的软件部分“屏蔽”无线信号的差异(不同的硬件平台结构，不同的信号特征) 而向上层提供统一的服务(数据、语音、图像等)。在软件开发的过程中，要求软件无线电的软件部分具有以下一些特点：

① 采用模块化结构。这样一来高级的通信应用可以利用已有的低层的程序模块，软件工程师不必将大量工作集中在系统设计的底层环节上，一些常用的程序模块都是现成的，避免了低水平的技术重复，降低了开发中无谓的浪费。

② 软件可重构。即在不同开发环境下开发的软件程序可以应用于不同的硬件平台上。对硬件平台要屏蔽软件程序的开发环境，所有的软件程序有统一的源代码；对加载软件也要屏蔽由不同厂商开发硬件平台所带来的差异。

③ 具有多模式切换，互操作性。面向未来的移动通信终端，兼容多种通信体制的功能必不可少。从而，要求我们的软件可以灵活的检测无线网络环境，通过软件控制并调动各个模块组件，重新组建一个新型的通信系统，达到无缝的模式切换目的。而且要求软件是可以互操作的，从而有利于模式切换的智能化发展。

为了达到上述目的，软件通信结构（SCA）应运而生。SCA 建立了独立于设备的结构框架，其目标是确保软件和硬件的可移植性和可配置性，并确保根据 SCA 开发的产品之间的互通。首先提出 SCA 的是联合战术无线电系统（JTRS），之后，SDR 论坛也开始接受 SCA 规范，并正在把 SCA 发展为商业应用的标准。

1.2 研究的思路与方法

本文主要应用软件无线电的研究方法，以 SDR 为理论基础，遵循 SCA 规范。为了有效地实现软件无线电的可重构性、可扩展性以及可互操作性，要求建立在软件无线电系统平台上的体系结构必须顺应其系统的内在特征，通过标准化、模块化和系列化的思路来构建整个系统。

“软硬件可重构的新一代无线通信统一平台”是由清华大学在国家“863”研究项目中提出，其整体可分为硬件平台和软件平台两部分。硬件平台采用嵌入式处理模块与通用 PC 相结合的方式搭建，把多种无线通信体制集成到一个通用系统结构中，由一个统一的硬件平台来实现。通过软件控制可编程逻辑器件和各个软件模块，并根据需要实时动态构建不同的通信体制，从而达到在不同的网络环境下，多种通信体制来回切换的目的。

“统一平台”采用开放式体系结构的设计理念，对整个无线电系统进行了层次化分析，把整个系统按照不同的类属性划分为应用层、无线电基础设施层和硬件平台层。每个层为系统提供一系列可以实现的功能，不同层次之间和同一层次之内可通过相应的接口实现连接和交互功能。系统的开发者和用户可根据规定的、公开维护的且易于使用的标准来构建这些各个层次的接口。无论是硬件还是软件，与系统对应的相应层面都是相互独立的，而且各层及其各个模块要素保持了较好的独立性，能够在不影响其他层的条件下进行改变。这样的设计理念对于系统的升级，对于提高同一系统在不同环境中的互操作性都是很有用的。

为了提高软件模块的重用性，在设计中采用了大量面向对象的方法。将具有相同功能属性的一系列现实对象从客观世界中抽象出来，并用这些抽象出来的类去定义对象，运用对象层次结构的特征和诸如类的继承、类的重载等方法提高软件组件的重用性。如把波形应用程序描述为对象，并且典型地将其分解成更小的功能对象，这样就提高了代码的可重用性。

1.3 本文的研究内容与目标

1.3.1 课题的内容

随着 3G 系统的逐步商用，移动通信领域面临着的多种通信体制不兼容、系统灵活性不足和扩展升级能力有限等问题越发突出。为了实现无所不在的、高质量的、高速率的、多媒体的信息传输目标，利用软件无线电技术，通过系统的可重构来适应不同的通信体制是有效途径之一。

统一平台体系结构包括：硬件结构、软件结构和规则集等基本的部件。通过它们将把各种无线通信体制集成到一个通用系统结构中，由一个统一的平台系统实现多种体制的功能和业务。

本文讨论的主要内容有：

① 本课题研究的背景，课题研究的思路与方法，以及通过研究希望达到的具体目标。

② 软件通信体系结构（SCA）的概念、发展情况、实现目标、结构特征以及基本构成。

③ 对“新一代无线通信统一平台”进行软硬件结构全面的分析，重点介绍了平台的软件体系结构。

④ 详细介绍了平台软件重构所涉及的各个关键技术。具体描述了软件重构的过程。

⑤ 最后，根据对新一代无线通信系统的软件重构的理解，进行实际编程验证，得出了相关结论，并对全文进行了总结。

1.3.2 研究目标

本文旨在通过理论研究和实验验证结合的方式，得出适合新一代无线通信的软件可重构的统一平台结构、重构机理、软件配置、软件加载和工作模式切换方法，验证所提出的软件可重构的新一代无线通信统一平台的可实现性。同时，通过搭建的实验验证系统，结合所提出的理论方法和现有软硬件设备，完成与实现软硬件可重构的新一代无线通信统一平台相关的研究。

2 软件通信体系结构

2.1 SCA 概述

软件通信体系结构 SCA（Software Communication Architecture），是在美军 JTRS（美军联合战术无线电系统）计划中提出，为开发未来的战术无线通信系统而发布的。SCA 为无线通信系统的设计和开发提供了详细的规范，建立了独立于设备的结构框架[15]。在 SCA 规范中定义了无线电系统的硬件结构、软件结构、安全结构以及公共服务和配置。其目的是要保证软件和硬件的可移植性和可配置性，并确保根据 SCA 开发的产品之间的互通。

SCA 是 SDR（Software Defined Radio）的一种实现方式，它在不同的层面对硬件和软件进行详细地定义，使系统具有组件的可移植性和广泛的复用性，因而非常有利于无线通信系统的快速开发。目前，SDR 论坛已经把 SCA 作为开发商用 SDR 产品的标准。截止到 2004 年，SCA 已经推出了 3.0 版本。

从内容上划分，SCA 规范包括阐述软件通信体系结构的主要文档和相关附录文档。附录文档有 JTRS 定义的应用环境描述体(AEP)以及域描述体(DP，Domain Profile)，描述体包含协议信息。SCA 规范的附录文档还包括波形配置的应用程序接口、服务定义 API、安全性说明、Rational UML（UML：Unified Modeling Language）文档，以及配置管理文档。

从逻辑上划分，SCA 由硬件结构、软件结构和规则集三个基本部件组成。SCA 使用面向对象的开发方法，定义了一个标准的、开放的、可互操作的软件平台，实现了底层硬件与上层应用软件的相互隔离。SCA 使用联合建模语言 UML 对接口进行描述，使用接口定义语言（IDL：Interface Definition Language）对接口进行定义。这两种语言都使用标准的软件开发工具，使得在结构定义完成后可以直接进行产品开发。

从层次上划分，SCA 体系结构包含：系统硬件平台、资源管理层、操作系统、核心框架、CORBA（Common Object Request Broker Architecture）中间件以及应用层波形组件。基于 SCA 的软件开发，要受到 CORBA 的接口标准和软件框架结构的限制。采用该软件体系结构主要有三个优点：(1)开放式的体系结构，可以最大化利用现成的商业产品并易于新技术的注入，极大地提高了系统的扩展性能；(2)通过开放式的分层结构将上层应用与底层硬件分开，保证了应用软件与设备无关性；(3)通过中间件技术提供分布式的处理环境，增强了软件的可移植性、重用性和扩展性[14]。

需要注意的是 SCA 并不是一个“完整”的系统设计规范。它只是为实现软件通

信结构的总体目标而制定的一系列约束系统设计的规则，因此也可以说 SCA 不是一个关于实现的设计规范，它并没有告诉软硬件设计者如何去设计硬件设备和软件程序，它仅仅是通过标准化的定义来确保总体目标的实现。系统设计者们必须按照 SCA 中的定义来研发系统，以确保与基类共用结构的统一，满足各领域系统标准化要求。SCA 规定了必要的约束条件，但是并没有限制结构的创新或在专用领域的要求。专用软件通信结构的实现框架是由专用域的技术规范方案以及其他相关的技术组合而成的[10]。

基于 SCA 的无线通信系统的软硬件体系结构设计是以 SCA 规范为基础，采用面向对象的设计思想，按照具有统一接口的"类"的模式来组织，整个系统的逻辑结构如图 2.1 所示[42]。

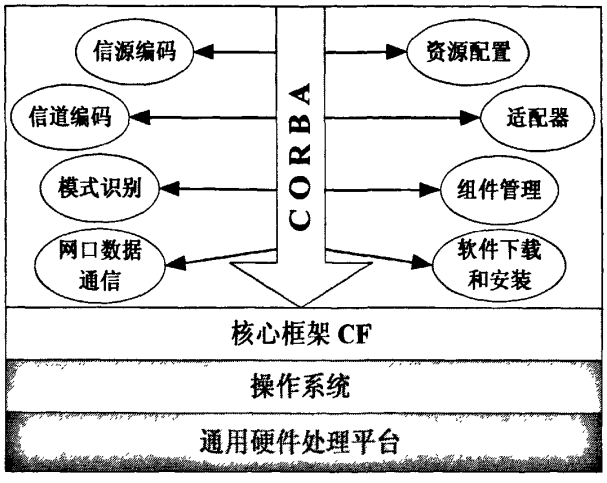

图 2.1 SCA 逻辑结构简图

Fig.2.1 Logical organization diagram of SCA

在该体系结构中，系统的配置与管理以及各功能组件间的交互是通过已定义的标准 API 接口实现的，其通信的软总线为 CORBA，从而使得"类"的内部实现方式变化（即硬件或软件的升级换代）不会影响系统的总体结构及其它模块的设计；在具体实现时，该体系结构允许通过加载不同的编码对象、交织对象、调制对象、上下变频对象来实现在不同通信模式、不同通信频段间的灵活切换，支持软件的下载和安装，以及各种设备或组件的动态配置；并通过基于 CPCI 的总线控制技术实现信道处理模块、基带处理模块、中频处理模块的互连，使得系统硬件平台具有开放性和易伸缩性，为硬件模块数目的扩展和新的通信模块的研制提供支持。可见，该体系结构是一种基于计算机技术、总线控制技术、高速 CPU/DSP/FPGA 等可编程芯片技术，以软件为核心的崭新的无线通信系统体系结

构。

软件通信体系结构需要实现八个目标[41]：

① 通用的和开放的结构，使用开放的标准化结构有利于提高竞争性、互通性、可扩展性和可升级性，便于新技术的引入、快速升级和软件的重用 。

② 支持多种环境，根据 SCA 生产的通信系统必须支持多种运行环境（多种使用域），包括空中飞行器、固定平台、舰船、陆地车辆。既要能堆叠使用，也可以手持使用。

③ 多波段多模式，根据 SCA 生产的通信系统要能取代目前使用的各种频带的无线电，从 2MHz 到 2GHz。在某个具体的频段要与目前使用的设备兼容。在模式和波形上有交叉的，要确保能互通。

④ 与现有系统兼容，新系统要与现有系统兼容，并要减少与现有系统集成过程中的相互影响。

⑤ 新技术的引入，新技术可以随时随地引进到系统中，提高系统性能，降低费用和缩短部署时间、避免故障、保持与商用技术的同步。

⑥ 保密性，SCA 结构必须解决战术通信系统中长期存在的一系列的安全问题，包括可编程的加密能力、多个独立的密级、流水线式的安全认证和结构化的部件，即密钥管理、软件管理、认证管理、用户识别和授权。

⑦ 网络化，除了支持传统的网络协议，SCA 必须支持新出现的宽带组网能力，包括语音、数据和图像。

⑧ 软件重用/通用的波形软件，SCA 应该最大限度地实现软件重用，在各种不同的实现方案中，允许使用通用的波形软件。

2.2 SCA 硬件体系结构

由于按照软件通信结构设计的无线电系统是一个以软件为核心的无线电处理平台，这就意味着该平台必须反映出软件无线电可重构和一可扩展等一系列重要的特征，这就使得采用的系统硬件必须具备通用性以满足新技术及其软组件的加载、更新或整体结构的扩展，而不能只是针对某种特定功能来设计。因此，软件通信结构的硬件构架并没有规定专门的硬件结构，而是采用了面向对象的方法划分各个功能组件的类模块，并规定与硬件设备相关联的各种属性。这些应用功能可以由硬件也可以软件来实现。在运行时，依据这些属性把软件资源分配给相应的硬件设备，从而使软件通信结构能支持软件通信结构兼容系统的重构要求，能进行动态的软件波形加载或者空中加载。

软件通信结构的硬件分成机箱和模块两大类，机箱类包含的属性都是关于与软件通信结构相兼容的硬件所具有的物理硬件属性，例如设备名称和序列号等硬

件模块类的子类包含更多特定的功能属性，而每一个子类都有特定的类型对象，其中包括射频类、调制解调类、处理器类、信息保密类、输入输出类、电源类、类、频率标准类、公共系统连接类以及天线类和同机部件类等子类[10]。

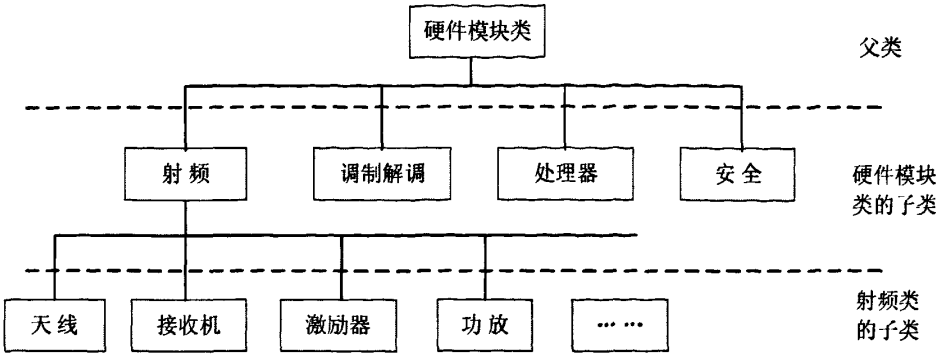

图 2.2 硬件模块类结构图

Fig.2.2 Structure diagram of hardware module class

软件通信结构硬件框架定义了与硬件设备有关的属性以及在为硬件资源分配合适的软件资源的方法。属性是一些能反应不同的使用域的硬件对象的参数。通过属性的注册实现对系统的重载。之所以采用这种方式来定义硬件是由于不同的场合对硬件的要求差异非常大，因此很难用一个统一的硬件来实现所有的功能。如果采用面向对象的方法对硬件进行描述，即硬件类(CLASS)，那么所有不同域的硬件就可以比较容易地包含在一个框架中，该框架也可以使用不同的属性行为和接口来表示不同硬件之间的实现差别。划分类的重点在于把系统分成不同的物理单元以及把这些单元组成一个功能单元。

硬件模型类从与以相兼容的硬件那里继承系统级属性，硬件模型以下的类从硬件模型中继承类属性。不同的属性值满足不同的要求，可以在实现过程中进行选择。硬件设备，即类的物理实现，具有相应的一物理平台环境和设备性能要求的属性值。一些属性由设备配置文件给出，用于产生波形应用程序，核心框架(CF)可以解读设备的配置文件。插槽类具有独立的物理结构、接口、供电和扩展环境属性，因为这些属性是最低层的，不同模块的插槽类属性不能共享。硬件模块类的继承关系如图 2.2 所示。

基于 SCA 的无线通信系统是一个以软件为核心的无线信号处理平台，硬件结构必须通用化，即硬件设备不能针对某个特定的功能而设计，而是设计成可扩展的通用平台，通过加载不同的功能软件实现特定的功能。基于 SCA 的无线通信系统的硬件体系结构如图 2.3 所示。

硬件模块主要包括：天线，射频前端，中频处理，基带处理，信道处理，信息安全，数据处理，输入输出，红边电源和黑边电源。出于电磁泄漏和信息安全的考虑，这些模块被分布在红边和黑边两个区域，并通过对不同设备采取不一样的处理措施，以有效地抑制设备的信息泄漏。红边和黑边内部模块间可以通过标准总线（CPCI 等）进行连接，构成类似于计算机的通用硬件体系结构[14]。

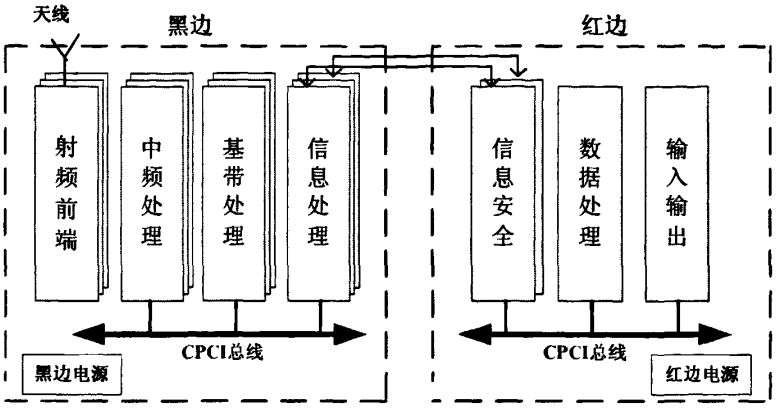

图 2.3 SCA 硬件体系结构图

Fig.2.3 SCA hardware system structure drawing

2.3 SCA 软件体系结构

2.3.1 软件体系结构简介

为了保证硬件的通用性和整个系统的开放性及可扩充性，软件体系结构非常重要，在 ISO/OSI 七层模型的基础上，参照 SCA 规范，建立了无线通信系统的软件体系结构，如图 2.4 所示。

基于 SCA 的无线通信技术研究，旨在摆脱传统的面向用途的设计思想，通过一种高度模块化的通用信号处理平台把各种通信装备提供的业务从基于硬件特性（频段、信道带宽、信道编码等）的方式中解放出来，通过装载不同的软件来动态配置系统的功能，实现系统硬件平台的通用化、模块化和软件的可移植性，从而适应新一代无线通信系统的需要，推动软件无线通信技术的发展。

软件体系结构定义了一个操作环境(OE)，该操作环境包括了集成在一个 SCA 实现中的一组核心框架服务和基础软件（包含板级支持包，操作系统和服务，CORBA 中间件服务等）。

① 总线层(板级支持包 BSP)

软件结构要求能在商业总线上操作，同时 OE 需要可靠的传输机制支持，这

就意味着需要在总线层次上支持错误的检测和纠正。目前，可能的总线包括 VME，PCI，CompactPCI，Firew ire(IEEE21394)，和 Ethernet。另外，OE 不排除在红、黑子系统中使用不同的总线体系，这就使诸如红区采用 VME 总线而黑区采用 CPCI 总线的情况成为可能，增强了系统的通用性。

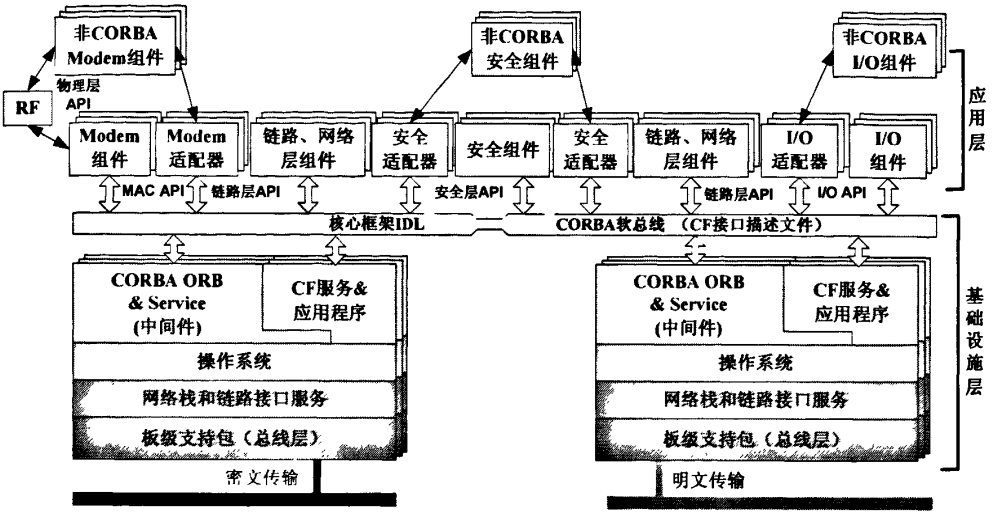

图 2.4 SCA 软件体系结构图

Fig.2.4 Software communication architecture drawing

② 网络和串行接口服务

软件结构依赖于商业构件来支持多种串行和网络接口。可能的串行和网络接口包括 RS232，RS422，RS423，RS485，Ethernet，和 8021x。而为了支持这些接口，就必须支持各种低级的网络协议，包括 PPP，SL IP，LAPx 等。

③ 操作系统层

软件体系结构包含实时嵌入式操作系统来提供对应用(包括 CF 应用)的多线程支持，同时，体系结构要求一个标准的操作系统接口来为操作系统服务，这是为了增加应用的可移植性。POSIX 是现今应用最广泛的可移植的操作系统接口，它是一个工业标准接口。POSIX 及其实时扩展与支持 OMG CORBA 规范的需求是兼容的，但支持完全的 POSIX 兼容性会比一个典型的实现包含更多的特性。因此，SCA 规范定义了一个最小化的 POSIX Profile 以满足 SCA 的需求。SCA POSIX Profile 基于在 POSIX 1003113 中定义的实时控制器系统框架(PSE52)。

该层用于为嵌入式应用程序提供多进程、多线程支持，屏蔽不同硬件平台间的差异，为上层软件提供标准的硬件访问接口和其他的基本操作系统服务，使得

上层应用软件具有设备无关性。由于可移植的操作系统接口（POSIX）规范是一个工业标准，该规范和它的实时性扩展与支持 OMG/ CORBA 规范的需求相兼容，因此在基于 SCA 的软件无线通信处理系统中，操作系统接口层应遵循 POSIX 规范。

④ CORBA 中间件

CF 中 CORBA 是作为分布式处理环境中的消息传递手段。CORBA 是一个跨平台的框架，用于标准化在分布式处理时的客户/服务器操作。分布式处理是系统体系结构的一个基础部分，而 CORBA 是一个广泛使用的中间件服务，用于提供分布式处理。CORBA 协议提供消息编码来处理发布消息时所需的位打包和握手。

嵌入式实时中间件为软件组件间的消息传递提供了统一的软总线，作为分布式应用运行平台间的通信机制，CORBA 技术可以解决硬件平台不断升级和软件需要保持相对稳定之间的矛盾，实现软件组件的即插即用、自动发现和动态部署等，从而满足无线通信系统对硬件模块和软件组件的柔性组合要求。

⑤ 核心框架

CF 是开放应用层接口和服务的重要的“核心”集，为软件设计者提供底层软件和硬件层的抽象。

核心框架服务基于开放式软件接口和描述，定义了波形应用组件的配置、管理、互联和通信等接口。它为波形应用提供了对底层软件和硬件的更高层次抽象，为波形应用组件/设备的自动装配、智能化管理定义了统一的接口，包括基本应用接口、框架控制接口、框架服务接口和配置文件。通过这些接口和配置文件可实现对整个系统中各种波形应用的安装、卸载、操作、配置和管理等，从而保持了波形应用组件实现的独立性，提高波形应用组件的可移植性。

⑥ 应用层

应用执行用户通讯功能，包括 modem 层的数字信号处理，连接层的协议处理，网络层的协议处理，互连网路由，外部 I/O 访问，安全，和嵌入效用。应用需要使用 CF 接口和服务，应用只能直接访问操作系统中在 SCA POSIX Profile 中规定的服务。而在应用层下面实现的网络功能，如商业 IP 网络层，则不受 SCA POSIX Profile 限制，因为它是存在于 OS 内核空间的。

该层主要指波形应用组件层，每个波形应用组件完成无线通信中相对完整的独立功能处理，它由核心框架中定义的一个或多个 Resource 组成，并通过 Resource 接口来控制和配置，在系统运行时，可被动态地加载到对应的硬件处理模块中，并支持动态配置功能。

2.3.2 软件分层模型

软件无线电的核心问题是用软件去完成系统的重构、升级、信号处理以及系

统管理等一系列的操作问题，其系统结构的核心部分就是软件结构。如何在软件层面上构建一个标准化操作环境，使得各种波形应用程序能够方便灵活地加载到硬件模块中，从而实现调制、中频处理、信息加密等一系列的通信问题就成为了构建整个标准化系统平台的重中之重。

在 JTRS 计划中，为了更好地使用标准化的软组件来创建一个有效的、合理的、标准化的操作环境去实现对整个无线电系统的定义过程，美军本着最大限度使用软件对象来完成系统通信和管理任务的原则，按照层次化建模的思路将具有不同系统功能的类模块进行了明确的划分，根据软件对象功能的不同和各组件的逻辑关系创建了能够满足软件无线电平台所需要的标准化软件结构，从整个软件结构上看各种类型的软件对象构成了下图所示的结构模型。APE 是可移植操作系统接口 POSIX 的子集，它为系统提供最小化操作系统限制，满足应用软件在该操作系统下的可移植性[10]。

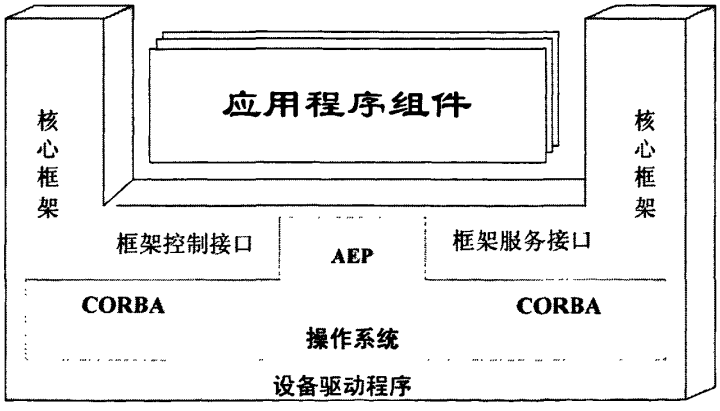

图 2.5 软件分层模型

Fig.2.5 Software lamination model drawing

其中，AEP（应用环境配置文件）是可移植操作系统接口 POSIX 的子集，它为系统提供最小化操作系统限制，满足应用软件在该操作系统下的可移植性。

整个软件结构自上而下由应用程序组件、核心框架组件、CORBA 中间件、应用环境配置文件(AEP)、操作系统和硬件设备的驱动程序组成。应用程序组件的各个模块完成系统所需的信号处理、信息加密和人机接口等顶层应用功能，其作用相当于整个系统结构中的应用层;软件分层模型中的其他部分则构成了支持顶层应用的基础设施层，该层确保了顶层结构中各种通信任务的顺利实施，并通过创建一个标准化的操作环境完成对软件无线电系统平台的标准化定义，确保整个系统结构可重构性和可扩展性[11]。软件通信结构的软件分层模型如图2.5 所示。

上面讲到分层体系结构是基于流处理的，所谓流是一个指定长度的含有数据或控制信息的信息包， 流最初在接口层形成， 然后在底层通过报头信息进行解释。所以流处理就是每个处理模块只能处理全部任务中的一部分，而处理完这部分任务后需将数据和控制信息传送到下一个处理模块进行另一部分任务的处理，依此推到全部任务完成为止，如图 2.6 所示。

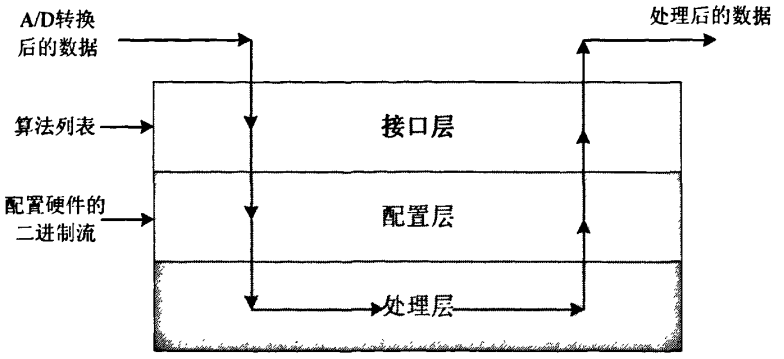

图 2.6 软件无线电的分层体系结构

Fig.2.6 SDR lamination architecture

① 接口层

在接口层的存贮器中存储有对各种无线电配置的系统级的描述。每个系统级的描述都包含有实现该系统各种算法配置的代码列表，但不包含硬件配置的代码。进入接口层的数据可以是 A/ D 转换后的数据，也可以是来自主机的控制和状态请求信息。一旦用户决定使用某一算法配置后，接口层即调用相应的代码， 并打包传送给配置层， 随后的数据将按这一预定算法进行处理，一直到用户决定改变系统的配置为止。

另外，接口层会从配置层接收到含有很多信息的信息包，这些信息包中的信息含有处理数据和控制信息，当然也不可避免地会有错误信息。因此接口层应首先检查信息包中的信息是否合法，若合法，则将信息送到主机。

② 配置层

配置层在该层的存贮器中，存储有配置处理层硬件的二进制信息包和处理层模块的状态代码列表。配置层接收来自接口层的信息包，通过报头了解信息包中的信息是控制信息还是数据信息。若是控制信息，则信息包中含有指定算法的代码。例如请求一个差分正交相移键控解调器和一个维特比解码器的代码等等。

配置层须根据控制信息调用对应的配置，将二进制流连同处理层模块的地址一起传送给处理层。当配置层接收的是数据信息时，配置层将在此数据的后面加

上本层的报头，然后传送给处理层。同样，来自处理层的信息包，经过类似的处理也会被传送到接口层。

③ 处理层

处理层是软件无线电的核心，是真正进行数据处理的层面。该层对来自配置层的数据进行处理，并将处理后的数据返回到配置层，是由一套线性相关的处理模块组成。

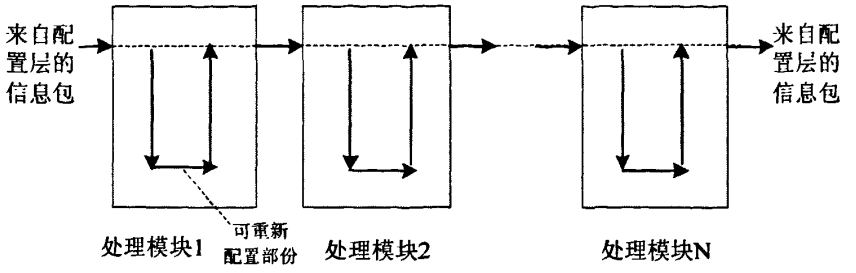

图 2.7 处理层的处理模块组

Fig.2.7 Processing module group of processing level

每一个处理模块都能通过重新配置来完成指定的功能，而且不需要打断与主流水线的同步。图 2.7 给出了处理模块的功能描述。每个处理模块都需要有三套流水线、一个信息包解释器和一个信息包重建器。配置流水线用以控制和处理相关的硬件配置，并通过执行编程信息让高版本的软件取代低版本的软件。如果信息包不是指定给该模块的，那么信息包将直接通过该模块的分路流水线绕越出该模块，以保证模块不中断不是指定给它的信息包。

另外，由于每个流水线都具有相同的时延性能，因此信息包能够保证与主时钟的周期同步。信息包通过流水线后，将在信息包重建器中加上报头，送往另一个模块。另外，每个处理模块都包括有静态的和可重新配置的两部分。所有处理模块中的静态部分都是相同的，是由信息包解释器、配置流水线和信息包重建器组成的，而处理模块的可重新配置部分则是由信息包解释器和处理流水线和信息包重建器组成的。当一信息包进入处理模块时，信息包解释器仅当该模块的有效位被设置时才能检查信息包的内容，否则信息包会被直接通过分路流水线绕越出该模块[10]，如图 2.8 所示。

信息包解释器本质上是一个开关，通过检查接收到信息包的报头，将所有控制信息传送到配置流水线，并将所有数据和信令信息传送到处理流水线。配置流水线接收到控制信息包后，检查报头的附带地址与该模块的地址是否相同，若相同， 控制信息包中的代码将在配置流水线中执行，从而改变处理流水线的硬件配

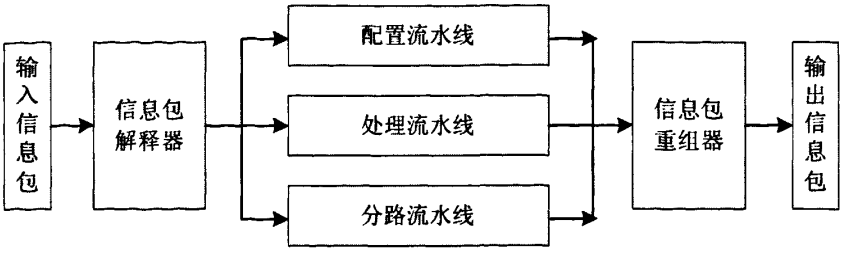

图 2.8 处理模块的功能结构

Fig.2.8 Function structure of processing module

置。若在执行代码中出现错误，则该模块将发送一个信令信息包给配置层说明错误的位置和特征，并将报头附带的地址设置为特定的值，防止随后的处理模块拦截含有错误信息的信息包[10]。

2.3.3 软件体系结构的操作环境

软件通信结构的框架是由操作环境(0E)来定义的。核心框架、CORBA 中间件、操作系统、应用环境配置文件以及硬件设备的驱动程序共同组成了系统内的操作环境。

操作环境在系统中的主要任务是:为了解决应用程序与软件通信结构兼容时产生的可移植性问题，需要对应用程序的设计加以限制，使得软件应用程序能够在采用不同软件通信结构的系统中发挥各自的功能作用而不会受到因底层硬件平台不同而带来的影响。为此操作环境一方面详细定义了核心框架和应用程序之间的接口，使得应用程序的各个功能组件可以通过核心框架来获得操作系统提供的各种服务;另外还需要对应用程序所使用的操作系统的规则以及CORBA中间件的API 接口进行必要的约束，以尽可能地满足应用程序可移植性要求。

应用环境配置文件(AEP)部分由软件通信结构来定义，其主要作用是最小化操作系统，从而提高应用软件的可移植性。AEP 通过一系列的标准化接口定义，限制操作系统对波形应用组件的访问与控制。AEP 对操作系统所提供的服务进行强制性地约束:只有当条件满足已经规定好的接口标准，操作系统才能对波形应用组件实施访问控制，从而有效地把波形组件加载到相应的硬件模块上。因此软件公司在编写波形应用程序时，并不需要考虑资源组件和底层硬件间的复杂联系，只要按照定义好了的标准接口设计波形应用软件，在安装后系统内部的实时操作系统就可以顺利地实现对各组件的访问。软件通信结构中的应用环境配置文件是按照 POSIX.13 实时控制系统配置文件 PSE52 的标准规范确立的，在系统设计过程中应注意:实时操作系统必须支持设计中应用环境配置文件的限制。只有这样才能实现通过最小化操作系统的限制达到了最大化应用程序可移植性的要求[11]。

操作环境还对 CORBA 中间件提出了限制和要求。为了满足嵌入式系统的容量需求，软件通信结构对 CORBA 中间件也做了必要的约束和限制，并提出了最小化CORBA的概念。在研发JTRS系统时，开发商必须遵守SCA规范中对CORBA的限制，不能随意地使用 CORBA 的拓展服务。现有的软件通信结构规范中已经明确的能够在最小化CORBA基础上进行拓展的三类服务是:命名服务、事件服务、日志服务。

随着软件通信结构的不断发展，互操作命名服务、实时 CORBA、CORBA 消息等拓展服务也有望被列入 SCA 规范中，成为设计者开发组件时需要考虑的服务类型。操作系统除了必需满足软件通信结构的应用环境配置文件的限制，还应当为系统提供多进程和多线程的服务。

2.3.4 核心框架和域配置文件

操作环境中的核心框架由一系列开放式软件接口和域配置文件构成，它定义了嵌入式通信系统中应用程序组件的配置信息、管理机制及其通信方式。对于所有符合软件通信结构的系统设计，其核心框架的定义都是固定的。所有的设计都必须遵守核心框架的规范定义，应用软件的编写应当遵循这些从底层硬件和软件层中抽象出来的接口和服务，不能随意添加或更改接口定义。核心框架内的所有接口是由 IDL 语言进行定义的。按照不同接口类的功能差异可将其划为基本应用接口、核心控制接口和核心服务接口。核心框架使用 CORBA 中间件来完成两个实体之间的通信。

硬件设备和软件组件共同构成一个系统域，域内硬件设备和软件组件的标识、能力、属性、相互依赖关系及位置由一组配置文件来描述，这组文件就称为域配置文件。软件通信结构中的域配置文件则是一系列由基于 CORBA 组件规范的可扩展标记语言(XML)编写的描述性文件。它对域内软件组件和硬件设备的特征进行了详细的描述，并对接口的功能、逻辑位置、连接关系和其他的一些相关参数也进行了相应地说明:如应用程序组件的属性描述、设备的启动需求等。

2.3.5 CORBA 中间件

实时中间件 CORBA 是位于操作系统和应用软件之间的通信服务，它利用对象请求代理（ORB）软总线隐蔽了真实的通信机制，提供了跨平台的消息传递。其主要作用是用来屏蔽网络硬件平台的差异性和操作系统与网络协议的异构性，使应用软件能够比较平滑地运行于不同平台上，以透明的方式实现对象间互连、互通和互操作，免去繁琐而容易出错的底层工作，使分布式软件开发提高速度并增加可靠性。CORBA 的使用让应用程序的开发者从复杂的底层环境中解脱出来，只需专注于业务的开发。因此可以实现各类分布式应用快速、可靠和高效的开发。分布式处理是系统体系结构的基本内容。

① 对象请求代理

在 ORB 结构中，ORB 并不需要作为一个单独的组件来实现，而是通过一系列接口来定义，任何一个提供适当接口的 ORB 实现均是可行的，接口分为以下三类:

1）所有 ORB 实现均相同的操作。

2）某些特定类型对象的特定操作。

3）对某些特定式样的对象实现的特定操作。

不同的 ORB 所选择的实现方式有所不同，通过与 IDL 编译器、接口库，以及各种对象适配器的协调工作，ORB 可以向客户机以及那些具备不同特性的对象的实现提供一系列服务。有可能存在多个 ORB 实现的情况，此时对象引用具有不同的表达方式，对于调用的执行可能采用不同的方法，客户可以同时访问两个由不同的 ORB 实现所管理的对象引用。当需要两个 ORB 同时工作时，ORB 必须能够区分它们自己的对象引用，而客户不必关心对象引用的区分。

ORB 核心是整个 ORB 的一部分，它负责提供基本的对象表示，并负责传送对象清求。CORBA 的设计目标是支持不同的对象机制，这种支持正是通过在 ORB 核心之上构造 ORB 组件来实现的，ORB 组件可以屏蔽不同 ORB 核心之间的差异。

② 客户

对象的客户可以访问该对象的对象引用、调用该对象的操作。客户只是根据对象接口了解其逻辑结构，并通过调用掌握对象的行为。客户通常从语言映射的角度来看待对象和 ORB 接口，也就将 ORB 直接带到程序员级别上。客户应该具有最好的可移植性，并且无需修改源码，就可以在任何 ORB 上运行，这些 ORB 拥有实现了需要接口的对象实例而且支持需要的语言映射。客户不必了解对象的具体实现方法，也不必知道这个实现采用哪个对象适配器，或是这个实现必须采用哪个 ORB 进行访问。

③ 对象实现

通常，对象实现通过定义描述对象实例的数据和对象所实现的方法的编码来提供对象的语义。对象实现往往要借助于其它的对象或软件来实现它的具体行为，某些情况下，对象的基本功能对其它的非对象部分有影响。对象实现的方法可以有多种多样，包括独立的服务器、程序库、方法的程序代码、封装的应用，以及面向对象的数据库等。实际上，通过使用附加的对象适配器，几乎可以支持所有风格的对象实现。通常，对象实现不依赖于 ORB 或客户调用对象的方式，通过对对象适配器的选择，对象实现可以选择那些依赖于 ORB 的服务的接口。

④ 对象引用

对象引用是用来指明某个 ORB 中的某个对象。按照语言映射的规定，对象

引用对于客户和对象实现仅仅是一个抽象的概念，两者可以独立于对象引用的表示。两个 ORB 实现可以选择不同的对象引用表示方式。传递给客户的对象引用的表示方式只在该客户的生命周期中有效。对于某种编程语言，所有 ORB 对于一个对象引用必须提供同样的语言映射，这种规定使得以某种语一言编写的程序对于对象引用的访问可以独立于特定的 ORB。当然，语言映射也可以提供其它方式访问对象引用，所采用的方式取决于对编程者是否方便。

⑤ OMG 接口定义语言

OMG 接口定义语言(OMG IDL，OMG Interface Definition Language)通过描述对象的接口来定义对象模型，接口包括一系列被命名的操作以及这些操作所需要的参数。尽管接口定义语言提供了相关的概念框架来描那些由 ORB 所控制的对象，但是 ORB 的正常工作不需要接口定义语言的源代码。只要等效的信息可以从根程序例程或运行时从接口库中获得，ORB 就可以正常工作。利用 IDL，一个特定的对象实现可以向其潜在的客户描述本身可提供的操作，以及如何调用这些操作。根据 IDL 定义，可将 CORBA 对象映射为特定的编程语言或对象系统[11]。

⑥ OMG IDL 到编程语言的映射

不同的面向对象或非面向对象编程语言可以采用各自喜欢的不同方式访问 CORBA 对象，面向对象语言可能倾向于将 CORBA 对象看作编程语言对象，即使是非面向对象的编程语言，将对象引用、方法名字的精确 ORB 表示屏蔽起来也是一种很好的想法。对于所有的 ORB 实现，从 OMG IDL 到一种特定编程语言的映射方法应该是相同的。语一言映射包括对该编程语言所用到的数据类型的定义，以及一些通过 ORB 能够进行对象访问的函数接口。具体而言，包括客户方根程序接口的结构(采用面向对象的语言时不必要求本项)、动态调用接口的结构，实现方程序框架、对象适配器以及直接的 ORB 接口结构。

语言映射也定义了对象调用和客户的控制线程之间的交互或实现。通常，映射采用同步调用方式，即当对象操作完成时，例程才返回。为了能够初始化调用过程，并能够实现返回主程序的控制，还需要附加其它映射规则，在这种情况下，需要提供一些专门用于这种语言的例程，以实现程序的控制线程和对象调用之间的同步。

⑦ 客户根程序

对于非面向对象语言的映射，对于每一个 IDL 接口类型，都将由一个编程接口支持对生成的根程序的调用。通常，根程序使用一种简单的方式，以便访问 OMG IDL 定义的对象操作，这种方式使得编程者只要熟悉 OMG 接口定义语言和所映射的编程语言即可。该根程序利用那些对 ORB 核心专用的、优化的接口调用 ORB 的其它部分。如果有多于一个的 ORB，则针对不同的 ORB 就有不同的根程序。

在这种情况下，ORB 与语言映射必须协调一致以便将正确的根程序和特定的对象引用相关联。面向对象的编程语言如 C++，不需要根程序接口。

⑧ 动态调用接口

接口允许动态地构造对象调用，也就是说，客户可以不调用一个专门用于特定对象的特定操作的根程序，而是通过一个调用或一连串调用来指定被调用的对象、所要执行的操作、以及这项操作的一组参数。客户代码必须提供关于所要执行的操作的信息以及要传送参数的类型(可能从接口库中或其他运行的资源中得到)。动态调用接口的特征可以因为可编程语一言映射的不同而有很大不同。

⑨ 实现程序框架

对于一种特定的语言映射，都有一个可能依赖于对象适配器的可实现仟何类型对象方法的接口。通常，这个接口是一个上行(up 一 call)接口，在这一接口中，对象实现将编写与这些接口匹配的程序，然后 ORB 通过生成的程序框架调用这些程序。程序框架的存在并不意味着一定有相应的客户根程序存在(客户也可以通过动态过程调用接口发送请求)。

⑩ 动态程序框架接口

动态程序框架接口可以实现对象调用的动态处理，这就是说，在通过接口来达到对象实现时，这一接口采用与客户端动态过程调用接口的相似的力一式访问操作名称及参数，而不必通过专用于某一操作的程序框架访问。其中的参数可以使用静态及动态(或许通过一个接口库决定的)知识来确定。

实现代码必须将所有操作参数描述提供给 ORB，在执行操作的过程中，ORB 将提供所有输入参数值。实现代码在执行操作后，将向 ORB 提供所有输出参数值或异常。随着可编程语言映射之间的不同以及对象适配器之间的不同，动态程序框架接口可能有很大差别，但普遍而言，都是上行接口。动态程序框架可以由客户根程序或动态调用接口方式实现调用，两者在效果上相同。

⑪ 对象适配器

对象适配器是对象实现访问 ORB 所提供的服务的主要方式，将来，会出现一些使用范围广泛的对象适配器，它们的接口适合于某些专门类型的对象。ORB 通过对象适配器所提供的服务通常包括:对象引用的生成与解释、方法调用、交互的安全、对象和实现的激活与终止、对象引用到对象实现的映射以及对象实现的登记对象粒度(granularity)、生命周期、策略、实现式样及其它特性的多样性使得 ORB 核心很难提供一个方便有效地用于所有对象的单一接口。因此，ORB 叮以通过对象适配器来将一组特定的对象实现作为目标，这些对象实现其有类似的功能需求和接口。

⑫ ORB 接口

所谓的ORB接口是指那些直接面对ORB的接口，它们不依赖于对象接口或对象适配器。ORB的大多数功能是通过对象适配器、根程序、程序框架或动态过程调用的方式提供的，因此只有少数操作对所有对象来说是通用的。这些操作对客户和对象的实现都是有用的。

⑬ 接口库

接口库(interface repository)是一种能够提供持久对象的服务，这些对象在运行时以某种方式表示IDL信息。ORB可以使用接口库信息来执行请求，而且，利用接口库中的信息，程序可以处理一些特定的对象，这些对象的接口描述在程序编译时未知，但却可以在运行时通过判断哪些操作对这个对象有效，然后实现该操作的调用。

接口库的功能除了表现在ORB上，还表现为可以存储与ORB对象接口相关的信息。例如，调试信息、根程序和程序框架库、能够格式化并浏览特定对象的程序等，都可以与接口库相关联。

⑭ 实现库

实现库(implementation repository)包含允许 ORB 定位和激活对象实现的信息。虽然实现库中的大部分信息与 ORB 或操作环境相关，但是实现库通常也是用于记录这些信息的地方。通常，实现的安装以及激活和执行对象实现的相关的策略控制是通过对实现库的操作实现的。实现库的功能除了表现在 ORB 上，还表现在可以存储与 ORB 对象实现相关的附加信息。例如，调试、监视、源配和安全可以与相关[10]。

2.4 SCA 规则集

JTRS 规则集为设计和实现硬件和软件框架提供了一般性指导。初始集定义了与形式参数、接口、环境要求和软件操作环境有关的规则。规则对实现开放的标准和商用单元提出了约束。规则集为项目管理者选择特殊的应用提出了严格的指导。初始集的一些规则为：

① 软件规则：

1）形式参数将从开放的商用的标准中进行选择（普遍使用的、从多个供应商可以获取的并认为是长期支持的）。

2）软件应该用高级语言开发，以便于处理器的移植。

3）新软件应该使用标准的高级语言编写，并且最大限度地独立于硬件平台和环境，便于移植和重用。

4）使用现有的软件时要通过适配器进行转换或封装，使之提供标准的接口。

② 硬件规则：

1)在硬件方面必须实现技术的特殊性和使用域特殊性与扩展性和互换性之间的平衡，硬件规则规定了 SCA 对硬件对象的需求，使硬件对象尽量实现上述平衡。

2）每个支持的硬件设备必须有一个使用域描述文件（Domain Profile），用 XML 语言编写。

3）在接口控制文档中应该定义硬件临界接口，该文档应无限制地提供给其它部门使用。

4）临界接口应该根据商用或政府标准制定。

5）硬件对象应该使用根据商用标准制定的格式化参数。

6）根据便于技术引入和模块替换的原则分割模块。

软件无线电或软件定义无线电是未来无线通信产品的重要标志，而软件通信结构是保证把软件无线电成功地应用于新系统的一个重要的规范。

2.5 SCA 安全体系

由于 JTRS 是一个政府部门研发的通信系统，因此信息传递过程中的安全问在系统研发初期就被列入到软件通信结构的设计范畴之中，也正是出于电磁泄和信息安全的考虑，美军参照可编程模块化通信系统(PMCS)指导文件的建议，在系统级上将各个模块分为红边与黑边。红边为可识别比特流，黑边为加密比流，通过对不同设备的电磁场信号采取不一样的处理措施以有效地抑制设备的息泄漏。按比特流分段，系统分为黑色总线、信息安全和红色总线三个字段，些字段在物理上是相互隔开的，从而保证红黑总线之间的隔离和信息安全的完性。

尽管系统的红边与黑边采用的可能是不同方式的信息处理技术，但是二者都必须遵守相同的通用体系结构。红边与黑边内部模块间可以通过标准总线进连接构成类似于计算机的通用硬件体系结构。软件通信结构中的核心框架与红边硬件进行连接，按照给定 SCA 所描述的系统模块类结构，实现系统双重硬件征的整合，通常情况下框架的红黑边可采用现有的商用总线如:VME，PCI，ComPactPCI，Firewire，Ethernet 等来实现框架以低层硬件的双向连接，并且黑边所采用的总线结构可以是不同的总线标准。

软件通信结构中的安全性分为 3 类:加密安全，非加密安全和系统安全。加密安全涉及到采用密码对数据进行保护，其目的是管理使用密码和密钥设备之间的接口，这可以通过空中接口和与设各直接连接的接口来实现。非加密安全涉到鉴权管理过程，确定用户\模块的权限，验证数据完整性而不论数据是偶然损还是被恶意行为故意损坏，以及分离数据从而管理对该数据的访问。最后，系统安全通过自检、算法与资源识别和选择、频率的管理、数据路径的建立和警戒、算法与过程的管理以及密钥与算法的存储来进行系统完整性检验。

随着工商部门对保守企业秘密要求的不断提高，安全性功能不再只是针对JTRS的需求，信息安全处理器的重要性也在不断地提高。利用可编程的信息安全器件，可实现软件定义的安全模块，软件通信结构的体系结构也采用了可编程的信息安全模块[10]。

3 新一代无线通信统一平台

“新一代无线通信统一平台”是由清华大学在国家“863”研究项目中提出，其整体可分为硬件平台和软件平台两部分。通过搭建和运行统一平台验证系统，可以较为全面深入地了解所设计的统一平台是否满足新一代无线通信的需求，所研究的平台体系结构中的软硬件可重构、硬件配置、软件加载和工作模式切换方法是否达到了设计要求。

平台包括：射频／信道接入模块，提供多个信号通道及跨越多个射频频段的频率变换；中频处理模块，包括滤波、进一步频率变换、空／时分集处理、波束成形及相关功能；多模式处理模块，产生多个空中接口波形，波形在调制解调器模块确定；信息安全模块，主要实现传输安全、身份认证及保护隐私等功能；编码调制模块，输出的编码信道比特流称为黑色(密文)比特流，经由信息安全模块变换为红色(明文)比特流，然后通过协议栈加以处理，产生网络比特或源比特流。

统一平台利用数字信号处理技术，采用嵌入式处理模块与通用 PC 相结合的方式来搭建。通过宽频段天线、宽带射频器件、宽带模数转换器(ADC)及数模转换器(DAC)，利用通用可编程处理器实现射频、基带及比特流处理。尽可能地采用可重新编程的软件代替了硬件电路，通过动态分配射频、中频、AD／DA、可编程处理器硬件和算法，将软件对象分配到硬件组件中，使得统一平台验证系统可以在线改变自己的特性，根据需要实时动态构建不同的通信体制，从而达到在不同的网络环境下，多种通信体制来回切换的目的。

其结构主要分为三层:软件无线电接口层、配置层和处理层。应用软件包含用于人机接口的高层图形界面，位于所有层次顶端。采用此种工作方式，可以比较容易地为软件和硬件建立“库函数”，系统级和底层硬件单元都具有可扩展性。同时具有一定的灵活性，可重构的硬件单元可以相对独立地进行升级工作。而且线性互连为处理单元间提供了较为简单的数据接口，减轻了对总线处理速度的要求。

3.1 平台的实现目标

当前，由于无线通信系统种类很多，各种通信体制由于自身的特点而应用于不同场合，基本结构相似但信号特征差异很大，例如工作频段、调制方式、波形结构、通信协议、数字信息编码方式、加密方式等都不同，这些差异极大地限制了不同通信体制之间的互连互通性。

新一代无线通信统一平台的目标是实现无所不在的、高质量的、高速率的、多媒体的信息传输。为了实现这一目标，需要克服技术上的巨大挑战。其中，为

新一代无线通信系统建立统一平台，是克服这些技术挑战的有效途径之一。本课题拟对于当前各种无线通信体制进行分析，结合新一代无线通信系统将要采用的核心技术需求，如 OFDM、MIMO、AMC 和 HARQ 等，研究统一平台在满足软硬件配置、可重构方法和工作模式切换等方面应该具备的性能及其指标。

通过理论研究和实验验证结合，得出适合新一代无线通信的软硬件可重构的统一平台结构、重构机理、硬件配置、软件加载和工作模式切换方法，验证所提出的软硬件可重构的新一代无线通信统一平台的可实现性。

3.2 平台的硬件结构

3.2.1 硬件构成及其功能

平台由高性能计算机、硬件处理模块和程控射频模块三个部分构成。如图 3.1 所示。高性能计算机给软件处理提供支撑，并提供人机交互界面，属于为软件处理提供支持的硬件模块。

硬件处理模块是直接参与信号处理的模块，主要由可重构的处理芯片、可重构控制器、波形组件存储器及其它外围电路构成。

射频模块要满足通信标准的频点、带宽、双工方式等指标，并可以在不同的标准之间可程控的切换频点、带宽等。

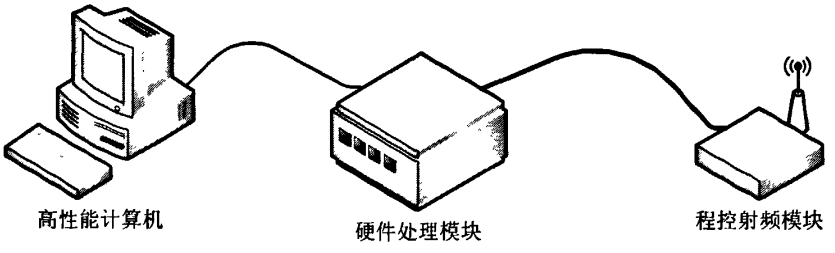

图 3.1 硬件平台示意图

Fig.3.1 Hardware platform schematic drawing

硬件平台，作为整个软件无线电平台的一部分，要实现完整的网络通信的功能，必须要做到多种功能的联合处理。

① 传输功能

硬件平台的作用之一是实现透明传输。在传输时，无论是提供软件支撑的硬件模块，还是承担信号处理的硬件模块，相对于软件模块，都是透明的。软件模块不关心硬件做了或者将要做什么样的处理，硬件模块也只需要传输软件模块发送来的数据，而不必关心其中承载的信息。

② 接入功能

在通信中，通信双方必须要建立连接，才能够传输数据。如果是网络通信，还必须划分出网络节点的类型，因为不同的节点在网络中的角色不同，接入网络的方式也不同。对于采用小区制的网络结构中，有中心站和移动台两类节点，接入功能就是在中心站和移动台之间完成建立连接的工作。

可重构的安全主要体现在对待加载的硬件波形组件库的鉴权。在可重构的控制器中保存了对每一个通信体制的硬件波形组建库的加载许可信息，当许可信息通过后，才可以从波形组件库存储器中取出相应的配置文件，进行重配置。

③ 可重构的功能

软硬件可重构的无线通信统一平台，要求软件与硬件必须协调工作。通过新的配置文件去生成电路，并调整射频的频段等。在可重构的时候，首先需要信号检测与模式识别。然后再根据识别出来的通信体制，加载相应的硬件配置文件，产生电路并调用相应的软件模块。

3.2.2 硬件平台的实现

平台计划实现 5 种通信体制，分别是：GSM，CDMA IS-95，WCDMA，TD-SCDMA，WiMAX。不同的标准要在同一个物理平台上得以实现，方法是对 FPGA 以及射频模块进行重构。对软件提供硬件支撑的模块是由一台高性能计算机完成的。硬件处理模块的核心的是两块 FPGA 。由一块 ARM 作可重构的控制模块。ARM 上移植操作系统，可以方便由上位机控制，进行重构。射频单元分为几个子单元，在双工方式上，既有 TDD 类型，也有 FDD 类型。模式和频段的选择，依照不同的通信体制，由 FPGA 控制切换。在 TDD 模式下，射频单元收发的切换也由 FPGA 通过排线来控制。天线采用了全频段的室内覆盖天线做验证。

其工作原理：（1）在发送端：通过 PC 的用户操作界面发送控制指令到 ARM 嵌入式处理模块，使之配置射频和中频处理单元达到需要的工作状态。同时启用核心框架的一系列自动配置机制，下载并安装所需要的应用层波形组件，构建当前需要的通信系统。（2）在接收端：前端射频单元采集各个频段数据，经过中频单元处理之后通过千兆网口传送到 PC。PC 通过模式识别确定当前的通信体制，并通过嵌入式处理模块控制前端射频模块，并配置可编程逻辑器件并使之正常工作。同时，PC 利用软件可重构机制动态构建当前的通信体制。 平台的工作原理如图 3.2 所示。

处理器模块的重构主要是 FPGA 的重加载。FPGA 的配置文件作为硬件波形库文件，存放在 FLASH 中。重加载时，首先通过多处理域的联合检测与识别确定通信体制，然后在安全模块鉴权通过后，ARM 调用 FLASH 中的配置文件，产生电平和时序，重新配置 FPGA。完成重配置后的 FPGA，通过排线，控制 RF 模

块的切换，完成全处理域的可重构。重构后的系统再完成接入，传输，安全等功能。整个硬件平台的重构在几秒之内完成，实现了灵活的快速可重构。

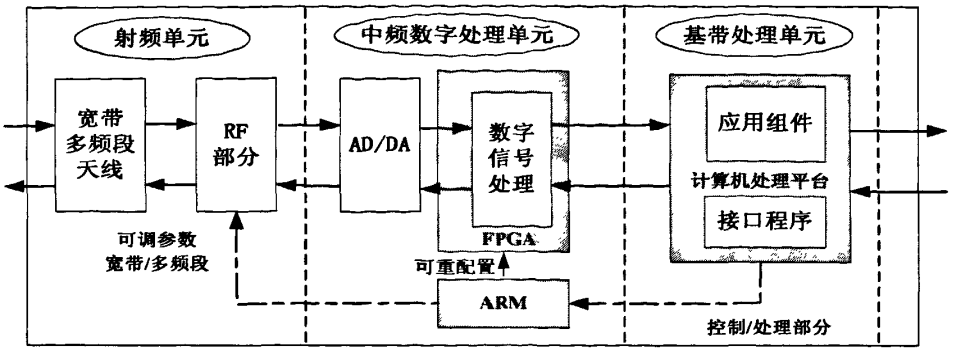

图 3.2 平台工作原理图

Fig.3.2 Platform work schematic diagram

统一平台硬件框架拟定义与硬件设备有关的属性，以及在运行时使用这些属性为硬件资源分配合适的软件资源的方法。属性是表征在不同使用环境中的硬件设施参数，不同的属性值满足不同的要求，可以在实现过程中进行选择。通过属性的注册过程实现对重配置的要求。其中，动态地装载软件波形是其基本的属性。

虽然不同使用环境对硬件的要求差异很大，难于用一个统一的硬件框架实现所有的无线通信功能，但是如果把面向对象的描述应用到硬件，那么可以把所有不同使用环境要求的硬件都放进一个框架中，该框架可以使用不同的属性（行为和接口）来表示不同硬件之间的实现差别，划分的重点在于把系统分成不同的物理单元以及把这些单元组成功能单元。

硬件设备具有相应的物理平台环境和设备性能要求的属性值。一些属性由设备描述给出，用于产生波形应用，核心框架可以解读设备描述文件。插槽类具有独立的物理结构、接口、供电和扩展环境属性，因为这些属性是最低层的，不同模块的插槽类属性不能共享。

3.3 平台的软件结构

这是一个层次型结构模型，它对软件开发的限制主要在体现在接口的定义和软件的体系结构上，而不是在具体功能的实现上。采用该软件体系结构的主要好处在于：(1)开放式的体系结构，可以最大化利用现成的商业产品并易于新技术的注入；(2)通过开放式的分层结构将核心和非核心应用与底层硬件分离开；(3)通过中间件技术提供分布式的处理环境，增强了软件的可移植性、可重用性和可扩展

性。软件结构如图 3.3 所示。

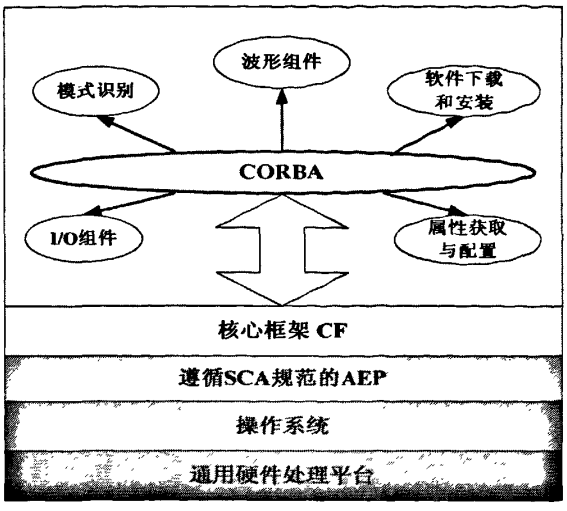

图 3.3 平台的软件体系结构图

Fig.2.2 SCA drawing of Platform

3.3.1 操作系统

Windows操作系统提供了强大的数据处理能力，保证了数字信号处理的高效性。系统提供的多进程、多线程以及高速网络接口等丰富的资源，在配置整个系统和动态构建应用层软件过程中发挥了巨大作用。操作系统层为上层应用软件提供标准的硬件访问接口（API）,使得上层应用软件具有设备无关性。

3.3.2 应用配置文件 AEP

AEP是遵循SCA规范定义的一系列的标准化接口，其目的是限制操作系统对上层波形组件的访问，从而提高软件的可重构性。

AEP对操作系统提供的服务进行强制性约束，操作系统对波形应用组件的控制，必须在AEP规定的接口标准之下进行。其优点是：① 在开发上层应用软件的过程中，进行资源分配的时候，不需要考虑底层硬件之间的复杂联系，极大的减小了上层软件开发的难度。② AEP降低了波形应用软件之间的耦合度，从而提高了应用软件的可移植性。

3.3.3 核心框架(Core Framework)

核心框架是软件体系结构的核心内容，定义了波形应用组件的配置、管理、互联和通信等一系列开放式接口。它为应用程序设计者提供对底层软件和硬件的抽象，包括四个部分：基本应用接口、框架控制接口、框架服务接口、域配置文件。通过这些接口和配置文件可实现对整个系统中各种波形应用的安装、卸载、

操作、配置和管理等，从而保持了波形应用组件实现的独立性，提高波形应用组件的可移植性。

3.3.4 实时中间件 CORBA

实时中间件 CORBA 位于上层应用组件和底层操作系统之间，采用对象请求代理(ORB)的方式为软件组件间的消息传递提供了统一的软总线，作为硬件平台间的通信机制。CORBA 技术可以解决软件模块需要不断更新和硬件平台保持稳定之间的矛盾，实现软件组件的即插即用、自动发现和动态部署等，从而满足新一代无线通信系统对硬件模块和软件组件的可重构性要求。

3.3.5 应用层软件

应用层组件包括所有的波形组件以及其它应用软件。波形组件是指动态构建通信系统所需的各种程序模块，包括：调制/解调模块、信道编解码模块、信源编解码模块。具体到程序就是指一个一个的“类”或者“函数”，比如说：GSM 系统类、Turbo 码编解码函数、PCM/ADPCM 语音编解码函数、ARM 语音编码函数等等。

总之，基于 SCA 的无线通信系统体系结构可以被广泛地应用于无线通信领域，成为设计和实现更便宜、更灵活、用途更广泛的无线通信系统的首选。

3.4 软件重构

3.4.1 软件重构定义

Ward Cunningham 和 Kent Beck 是最先认识到重构的重要性的两位软件专家，但最早对重构进行理论研究的是 Ralph Johnson 教授及其所带的学生。其中 Ralph Johnson 教授的博士研究生 Willian Opdyke 在其博士学位论文中首次对重构进行了系统的研究与描述[40]。

关于重构，很多人提出了自己的定义，但其实质是相同的。Ralph Johson 教授给出了如下的定义：重构是在一些原因（如提高效率与可维护性等）促使下，将一个面向对象的设计以不同的方式进行重新组织从而使设计更加灵活、重用性更强的过程。

Martin Flower 分名词与动词两种形式对重构进行定义。按照 Martin Flower 的观点，就是重构之前软件实现什么功能，重构之后软件同样实现什么功能。

3.4.2 软件重构的意义

软件设计与开发人员经过大量的实践与研究，认为通过重构，可以达到如下的作用：① 简化测试。② 使设计更加简单，更加容易理解。③ 提高软件设计的质量。④ 容易发现原有设计与代码中的臭虫。⑤ 使编码的效率提高。⑥ 提高软件的可维护性。⑦ 提高软件的扩展性[40]。

统一平台的软件模块是系统实现可重构功能的重要部件。为了实现多任务的

软硬件分配，操作系统可以按照计算量的要求和用户要求的服务质量把任务分配给硬件或软件。

平台最大程度地利用商用产品和协议，通过开放式分层结构将核心应用程序和非核心应用程序从底层硬件独立出来，通过通用 CORBA 提供一种分布式处理环境，从而达到软件模块的可移植性、重用性和伸缩性。

3.4.3 软件重构的特点

在软件开发的过程中， 要求统一平台的软件重构具有以下一些特点：

① 采用模块化结构

这样一来高级的通信应用可以利用已有的低层的程序模块，软件工程师不必将大量工作集中在系统设计的底层环节上，一些常用的程序模块都是现成的，避免了低水平的技术重复， 降低了开发中无谓的浪费。

② 软件的可重构性

即在不同开发环境下开发的软件程序可以应用于不同的硬件平台上。对硬件平台要屏蔽软件程序的开发环境， 所有的软件程序有统一的源代码; 对加载软件也要屏蔽由不同厂商开发硬件平台所带来的差异。

③ 多模式切换，互操作性

面向未来的移动通信终端，兼容多种通信模式的功能是必不可少的。从而，要求我们的软件可以灵活的检测无线网络环境，通过软件控制并调动各个模块组件，重新组建一个新型的通信系统，达到无缝的模式切换目的。而且要求软件是可以互操作的，从而有利于模式切换的智能化发展。

4 平台的软件可重构研究

本章是全文的核心章节。严格遵循 SCA 规范，建立一套完善的高效的软件重构框架是本文的重点，软件模块化是软件重构的基本前提。建立一套完善的软件重构机制包括几个方面的工作：以 SCA 为基础，通过创建一套高效可行的软件配置和管理模板，利用高度模块化的波形组件库，实时动态地构建满足用户需求的应用软件。

核心框架负责软件的配置与加载，它对整个系统的软件进行管理，并协调各个软件模块间的相互通信。从内容上来说，包括了一系列开放式接口和域配置文件。核心框架利用这些接口与配置文件，通过 CORBA“软总线”调用上层应用组件，完成资源的调配和应用程序的安装和卸载。

4.1 通信体制识别

4.1.1 概述

在软件重构的第一步就是要通过一些手段识别出当前的通信模式，从而启动重构软件的进程。

在本文中，通过识别调制方式来识别通信体制是一种简单但有效的方式。新一代无线通信统一平台所具有的多频段、多功能、多体制的特性，使其在进行通信信号解调之前就需要对信号调制模式及参数进行识别和估计，以便准确解调出调制信息。因此，对指定通信体制信号调制方式的识别是新一代无线通信统一平台实现多模通信功能所必需的。

目前，无线通信信号调制方式的识别有多种方法，如基于参数技术或决策树的方法，用于解决通信信号的自动识别问题，并且收到了不错的效果。

谱相关方法具有分辨率高、抗干扰能力强等特点,除了能识别和分离信号，还能通过检测信号的幅度、位移等特征来测定载波、信息速率及相位信息等。传统的信号分析模型都是以平稳随机过程为基础的，但是实际中的许多信号并非平稳信号，其平稳性总是与载频、基带信号及编码等周期信息有关，其一阶或高阶统计参数呈时间周期性，这类信号被称为循环平稳或周期平稳，谱相关理论就是建立在这种具有特殊性质的信号模型之上的[32]。谱相关方法处理的最大优势就在于它能够在展现信号原始傅里叶频谱的同时，还提供循环频率轴上包含信号所有信息的更为详尽的频谱，可以从中观察、检测出比较准确的信号频谱结构。采用谱相关法进行信号的识别，关键是特征参数的选取，其稳定性和可靠性都是识别过程中的难点和重点。而且采用谱相关方法的计算量相对较大，对硬件的实现是一

大考验。因此，如何简少运算量、优化识别流程将是研究的重点。

4.1.2 工作原理

我采用轮询的方式进行处理。从 A/D 中取出来的数值序列不断被送到调试识别模块，计算各种参数，最后通过算法进行识别。如果此次序列值不满足识别条件，表示识别失败。则进行下一轮的取值、计算和识别。工作原理如图 4.1 所示。

在加载到 FPGA 在运行， 则 CPLD 输出停止取值计算的信号，停止调制识别流程；反之则输出工作信号，使调制识别进入下一轮的工作状态，直至成功识别出调制方式。

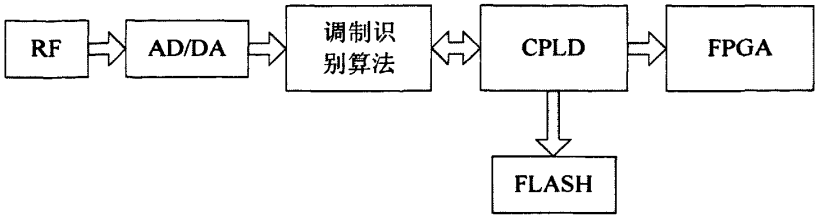

图 4.1 工作原理框图

Fig.4.1 Schematic diagram

4.1.3 调制识别算法

本文采用了一种常用通信信号的调制方式自动识别的算法，从信号的功率谱、二次功率谱和四次功率谱中提取一组特征参数，采用判决树方法，在不需要先验知识的情况下对常用通信信号调制样式进行自动识别。

参数的提取和计算是调制识别的核心。最大限度地利用不同调制样式的信号在时域、频域的区别，就可以实现信号调制方式的快速自动识别。信号频谱是对信号在频域的描述，不同调制方式的信号在频域上表现形式不同。信号功率谱及其高次谱可以反映多种调制方式的特性，可作为调制识别的特征参数。信号功率谱可以直接反映调制信号中各频率分量的功率分布。信号的平方功率谱是信号平方后的功率谱。对于载波有跳变的信号如BPSK，平方谱2倍载频处有很强的单频分量，而其他PSK信号无此特征。信号的四次方谱是信号四次方后的功率谱，可用来区分4PSK与8PSK,4PSK信号在4倍载频处有一单频分类，而8PSK信号无此特征[35]。由于QAM是幅度和相位联合调制的信号，因此幅度的波动较大，可以用这个参数将QAM信号识别出来。流程如图4.2所示。

① 信号平方谱

BPSK和GMSK信号平方谱上出现单频分量，平方后GMSK调制指数为1，平方谱上出现两个单频分量。

信号平方谱是信号平方后的功率谱，在忽略掉零频分量后，平方谱反映的是调制信号倍频后频谱特征。对于载波只有n跳变的信号如BPSK信号和DSB信号平方谱在二倍载频处有很强的单频分量，而其他的PSK信号和SSB信号则无此特征。因为调制指数为整数的FSK信号功率谱中传空号频率处有离散谱线，而GMSK信号在倍频后调制指数为1，因此检测平方谱单频分量还可以将GMSK信号检测出来。

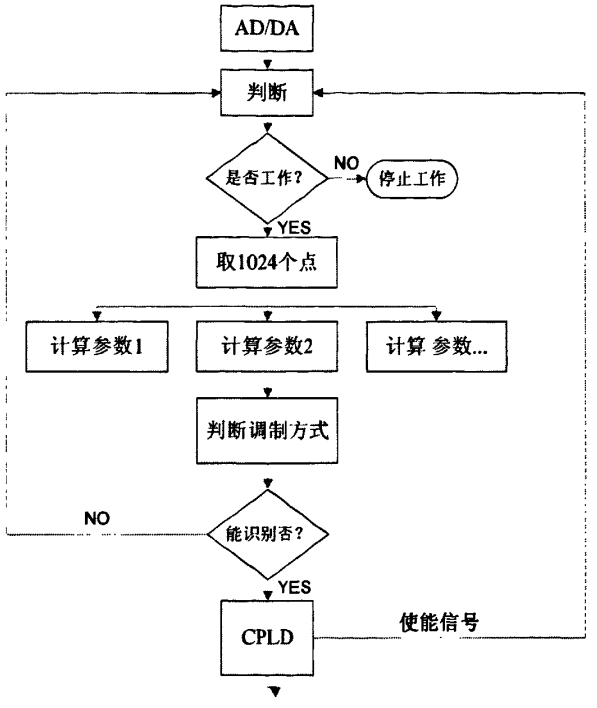

图 4.2 调制识别流程图

Fig.4.2 Modulation recognition flow chart

② 信号四次方谱

QPSK信号四次方谱上出现单频分量，而其他MPSK无此特性。信号四次方谱是信号四次方后的功率谱，在信号平方后进行截至频率为载波频率的高通滤波，并将处理后的信号向下搬移一倍频谱，然后再此求其平方谱。主要用来区别4PSK信号和8PSK信号。

③ 信号八次方谱

为确定信号是否是8PSK信号，应对信号的八次方谱进行分析。

④ R参数

R参数反映的是信号包络的变化程度。

$$R=\sigma/u^2$$

其中u和σ分别是信号包络的平方的均值和均方差。幅度调制信号可以利用信号包络的方差与均值平方之比来加以识别。在一定的信噪比前提下，利用统计参数R可以将MQAM,MPSK信号分开，其门限值问N。我们来看看识别GMSK、BPSK、QPSK、8PSK和QAM调制信号的算法图4.3所示。

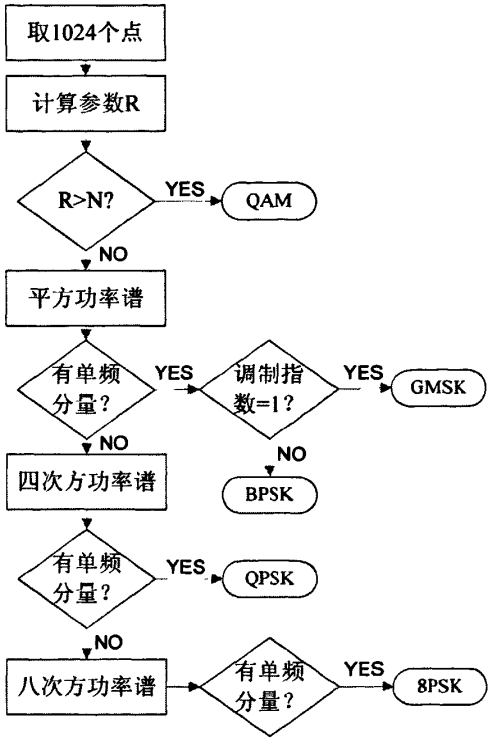

图4.3 调制识别算法

Fig.4.3 Modulation recognition algorithm chart

表4.1 调制识别算法的准确率（%）测试结果

Table4.1 Test results of the modulation recognition algorithm's accuracy rate

识别的调试方式 / 发射的调制方式	16QAM	BPSK	QPSK	GMSK	8PSK
16QAM	99.3	0.02	0.05	—	—
BPSK	—	99.9	0.01	—	—
QPSK	—	0.02	99.8	—	—
GMSK	—	—	—	100	—
8PSK	—	0.01	0.02	—	99.7

根据以上的算法，通过实际编程验证，可以发现效果比较理想。在接收信号为

10db的情况下，通过1000次的测试，识别的准确率都在99%以上，其具体的数据如表4.1所示。

4.2 PC 与 ARM 之间的接口实现

因为"统一平台"采用嵌入式处理模块与通用PC相结合的架构，所以存在PC的Windows系统与嵌入式处理平台的ucLinux系统之间的数据交互问题。

4.2.1 WinPcap

为了解决上述问题，开发了一个数据接口程序，专门用来实现不同平台之间的数据交互。这里的数据交互通过网口来实现，整个程序利用WinPcap来完成。

WinPcap是一个基于Win32平台的，用于捕获网络数据包并进行分析的开源库。通过WinPcap，Win32的应用程序可以直接访问没有被操作系统利用网络协议处理过的原始数据包。

WinPcap提供了以下功能:

① 捕获原始数据包，无论它是发往某台机器的，还是在其他设备（共享媒介）上进行交换的。

② 在数据包发送给某应用程序前，根据用户指定的规则过滤数据包。

③ 将原始数据包通过网络发送出去。

④ 收集并统计网络流量信息。

以上这些功能需要借助安装在 Win32 内核中的网络设备驱动程序才能实现，再加上几个动态链接库 DLL。所有这些功能都能通过一个强大的编程接口来表现出来，易于开发，并能在不同的操作系统上使用[26]。

4.2.2 接口程序实现

所有需要交换的数据在发送之前都需要打包。接收不同的数据包后均需要对接收任务队列进行写操作，但是写的类型有所不同。以太数据包定义：

080F：代表读写 IO 数据命令

080A：sram 数据传输命令

080B：接到读 IO 数据命令后返回的地址/数据信息包

在此只例举出 080A 类型的以太包帧格式：

① 前 N 包帧格式（包头 23bytes+数据）

Byte	0～5	6～11	12~13
含义	目的 MAC 地址 AA-AA-AA-AA-AA-MAC_5	源 MAC 地址 AA-AA-AA-AA-AA-xy,	协议类型 数据包为 080A

Byte	14～17	18	19~22	23~	最后 4 位

含义	localtime	Flag(帧标志)	本包接收 sram 写地址	数据部分	4 Bytes CRC 校验

② 最后一包帧格式（包头 36bytes+数据）

Byte	14~17	18	19~22	23~26
含义	localtime	Flag（帧标志）	本包接收 sram 写地址（低前）	接收时间

Byte	27	28~31	32~35
含义	最后一包有效数据长度（Byte）	初始地址（低前）	总长度（word，低前）

① Flag 标志

Flag（7 downto 2）= 000000。

Flag(1)：数据帧为第一帧时为 1，其他为 0。

Flag(0)：数据帧为最后一帧时为 1，其他为 0。

比如: flag=0x02 代表本帧为第一帧非最后一帧，flag=0x03 代表本帧为第一帧且最后一帧。

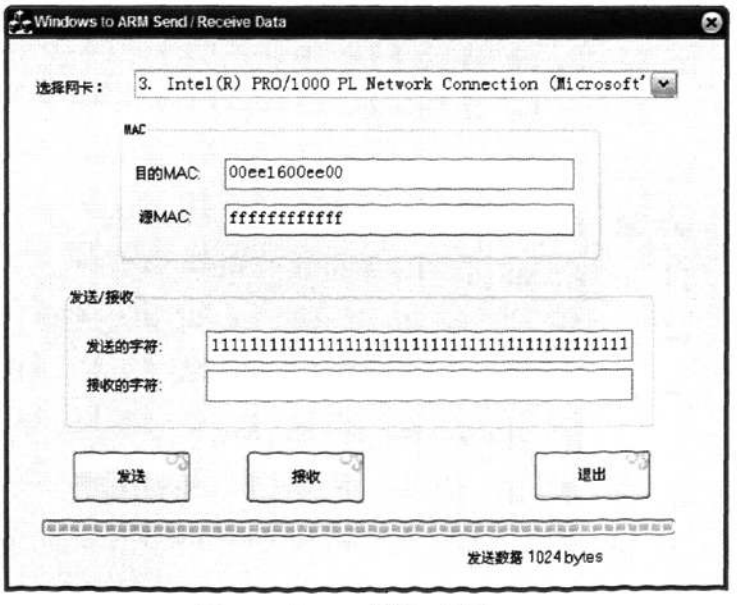

图4.4 PC与ARM的接口界面

Fig.4.4 Interface surface of PC and ARM routine contact

② 数据部分

按 byte 计数，即任务队列数据长度的 2 倍，每个数据包数据部分最多为 1024 个 bytes，数据长度超过 1024 个 bytes 时，分帧传送，且除最后一帧外数据长度都限制为 1024 个 bytes。

Flag=0x02 → 数据部分长度：1024 bytes，代表数据分帧，且为第一帧。

Flag=0x03 → 数据部分长度：34~1024 bytes，代表数据未分帧，且为最后一帧。

Flag=0x01 → 数据部分长度：34~1024 bytes，代表数据分帧，且为最后一帧。

Flag=0x00 → 数据部分长度：1024bytes，数据分帧，非第一帧，非最后一帧

③ 接收 sram 写地址

这 4 个比特为该包的目的 sram 写地址。

④ 最后一包有效数据长度（Byte）

当且仅当最后一包数据长度小于 24 个 bytes 时，该部分为实际需要写入目的 sram 的长度(bytes)，其他情况均为 0。其接口程序的界面如图 4.4 所示。

4.3 核心框架的实现

4.3.1 核心框架定义

核心框架（CF）由一系列开放式软件接口和域配置文件构成，它定义了嵌入式通信系统中应用程序组件的配置信息、管理机制及其通信方式。对于所有符合软件通信结构的系统设计，其核心框架的定义都是固定的。所有的设计都必须遵守核心框架的规范定义，应用软件的编写应当遵循这些从底层硬件和软件层中抽象出来的接口和服务，不能随意添加或更改接口定义。

核心框架内的所有接口是由 IDL 语言进行定义的。按照不同接口类的功能差异可将其划为基本应用接口、核心控制接口和核心服务接口。核心框架使用 CORBA 中间件来完成两个实体之间的通信。

核心框架为应用程序的设计者们提供了底层软件和硬件的更高级抽象，并通过利用设计限制的方法提高应用程序的可移植性。

按照功能性划分，核心框架被分成三个部分:第一是基本应用接口。基本应用接口是构建核心框架中应用(Applications)和设备(Device)的最基本的模块要素，主要被用于开发软件应用程序组件和完成组件间的信息交换这两项任务。第二是框架控制接口。它提供了控制和管理硬件资源、应用程序和域(系统)的方法。根据管理对象的不同又可细化为设备接口、设备管理接口和域管理器接口。第三是文件服务接口。文件服务接口是由文件、文件系统和文件管理器构成的服务接口。这些接口均为可运行的软件实体，其主要功能是为分布式系统提供文件访问的方法或行为。

4.3.2 核心框架的接口

核心框架提供的各种接口，一部分接口由软件平台实现,一部分接口由波形应用软件实现,其余部分由硬件设备实现。 CF通过这些接口和配置文件可实现对整个系统中各种应用组件的安装、卸载、管理和配置等，从而保持了应用组件的独立性和可移植性。

① 基本应用接口

基本应用接口是构建核心框架中应用(Applications)和设备(Device)的最基本的模块要素，主要被用于开发软件应用程序组件和完成组件间的信息交换这两项任务。这些接口并不是真正可运行的软件实体，而是采用IDL语言表示的接口定义。包括：端口、端口供应、生命周期、对象测试、属性设置、资源和资源工厂等接口。在设计所有顶层应用程序组件时都必须遵循这些接口定义，按照标准的形式将各个软组件的功能模块封装在这一系列的标准化接口内，也就是说所有的应用程序都必须提供和实现这些接口。核心框架的控制接口通过它们来装配新的应用程序。

基本应用接口通常被波形应用程序使用，这些接口可根据核心框架的需求来定义，在应用程序开发过程中由软件设计人员予以实现。此接口被框架中的应用层和框架控制接口用于装配一个应用程序，它不用执行系统内应用组件的创建、实例化、配置、控制、测试和拆分等操作。

② 框架控制接口

该接口提供了一系列控制和管理硬件资源、应用程序和系统（域）的方法。根据管理对象的不同又可细化为基本设备接口、设备管理接口和域管理器接口。该接口提供一系列装配和控制无线电设备的系统功能。

设备接口(文中没有强调是硬件设备的情况下都是指抽象的逻辑设备)由基本设备接口、加载设备接口、执行设备接口和聚合设备接口构成;设备管理接口就是前面提到过的设备管理器;域管理接口则由应用、应用工厂和域管理器组成。

③ 服务控制接口

服务控制接口是一些可以运行的软件实体，其作用是为系统提供文件访问的方法。包括：文件接口、文件系统接口和文件管理接口。应用层组件可以通过这些接口完成对系统内所有文件的访问。

1）文件接口

文件接口为系统提供分布式文件服务，在同一文件系统内通过核心框架的CORBA文件接口，所有文件的位置对于应用程序都是透明可见的，因此应用程序就可以使用该接口提供的读写文件操作，对处在软件通信结构兼容分布式系统内的所有文件进行访问。在访问核心框架各元件的配置文件属性、安装应用程序以

及装载和执行文件时应用程序组件都需要用到文件接口。

2）文件系统接口

文件系统接口提供了分布式文件系统服务的能力，它定义了对一个物理的文件系统实施远程访问的CORBA操作，当访问核心框架元件的配置文件属性、装载和执行文件以及安装卸载应用程序时都会使用到该接口。文件系统接口使得底层物理文件系统对用户而言变得透明，用户可以通过该接口对文件执行创建、删除和复制的操作。

3）文件管理系统接口

文件管理器接口提供了分布式文件管理服务，使用该接口可以实现对多个分布式文件系统的访问。尽管实际文件的存储可能是跨平台的，但仍然可以把文件管理器接口看作是单一的文件系统来管理文件。文件管理器继承文件系统接口，并且通过加载和卸载操作对文件系统接口进行扩展，因此文件管理器在投有加载其它文件系统的情况下，可以被当作是文件系统来实施访问。

4.3.3 域配置文件的实现

这里所说的“域”就是指系统的整个工作环境，它是一个由节点、事件通道、文件管理、软件应用和软件实例组成的集合，这个集合提供了对象聚合的实例。正如下图所示，域内包含 0—N 个节点、软件应用实例、软件运用，还包含了两个以上的事件通道和一个文件管理器。

硬件设备和软件组件共同构成一个系统域，域内硬件设备和软件组件的标识、能力、属性、相互依赖关系及位置由一组配置文件来描述，这组文件就称为域配置文件。软件通信结构中的域配置文件则是一系列由基于 CORBA 组件规范的可扩展标记语言(XML)编写的描述性文件。它对域内软件组件和硬件设备的特征进行了详细的描述，并对接口的功能、逻辑位置、连接关系和其他的一些相关参数也进行了相应地说明:如应用程序组件的属性描述、设备的启动需求等。它清楚地定义了硬件设备和应用组件的信息和通信的格式。

域管理器就是通过配置文件的组件信息来实现启动、初始化和维护动态构建的应用程序。域配置文件包括八种类型：软件包描述文件、属性文件、软件组件描述文件、软件装配描述文件、设备配置描述文件、设备管理描述文件、设备包描述文件、配置文件的描述文件。

SCA 关于配置文件的格式遵循 CORBA 的组件模型。CORBA 组件规范，详细定义了软件模块间的通信和配置的过程。域配置文件采用文本格式，其后缀名为“.dtd”。

位于核心框架内的域管理器主要提供了以下几项服务：

① 应用工厂安装和卸载域文件管理器上的应用软件。

② 设备管理器找到域内的节点、软件应用和软件应用实例。

③ 创建、终止和控制软件应用实例。

④ 设备管理器注册或注销节点以及节点内的服务和设备。

⑤ 注册和注销事件通道。域内有两个事件通道是缺省的，一个是流出域事件通道，通过这个通道客户程序可以感知到域的变化另一个是流入域事件通道，域管理器使用这个通道感知到域内发生的变化。域服务被映射到框架控制接口的各个部位，相应的接口完成域问题空间内所需的服务。

⑥ 域管理器必须能够找到域内的各个元件，并能对这些元件完成安装、卸载、注册和注销的操作。

⑦ 使用核心框架域内的应用接口可以控制和终止软件应用程序的实例。

⑧ 应用工厂提供创建应用的服务。对于域内的每个软件应用来说，被创建的应用工厂对象被用来部署域内的应用程序，因此应用工厂接口也是一个应用程序的实例化接口。

⑨ 设备管理器提供管理节点的服务，是一个部署或移出节点的接口。

⑩ 基本设备接口（设备、装载设备、执行设备和聚合设备）负责管理硬件设备。

域配置文件包括：

① 软件包描述文件 SPD。确定一个软件组件组件或非组件的实现情况，一个软件包的名字、作者、属性文件、处理器类型、操作系统、执行代码和软件属性等信息都包含在它的 SPD 文档中。应用程序的每个组件都有自己的 SPD 文件。

② 特性属性文件 PRF（Property File）。PRF 描述组件软件包或硬件包的各种属性特征。组件的配置、测试、执行和分配类型的信息都包含在文件的说明范围之内。

③ 软件组件描述文件 SCD（Software Component Descriptor）。SCD 用于描述软件通信结构中资源、资源工厂、设备的个性特征，反映了这些组件将要提供或使用的接口。

④ 软件装配描述文件 SAD（Software Assembly Descriptor）。SAD 描述了应用程序的应用个性包括软件的配置特性、组件的连接特性，反映了应用程序的各个组件是如何构建应用的信息。例如用哪些组件构建应用程序、如何找到组件并完成配置任务、怎样实现组件的连接等。应用工厂在创建应用时就必须要使用到这些信息。

⑤ 设备配置描述文件 DCD（Device Configuration Descriptor）。DCD 描述设备管理器的配置个性。在每个节点启动时，提供了一种将固有组件逻辑设备和服务部署到核心框架域的一般性方法，并指明组件如何在最初时启动一个设备并找

到域管理器。还使得系统管理员可以灵活地决定系统提供多少服务，部署什么样的节点。

⑥ 设备管理器配置描述文件 DMD（DomainManager Configuration Descriptor）。DMD 描述了域管理器的配置个性。

⑦ 设备包描述文件 DPD（Device Package Descriptor）。DPD 文件是基于软件通信结构硬件框架定义来实现的，主要用于确定硬件设备的类型以及特征。人机接口应用程序可根据系统内的 DPD 显示各硬件设备的相关信息，如设备的容量等。

⑧ 配置文件的描述文件 PRD（Profile Descriptor）为 SAD、SPD 或 DCD 实例提供完整的文档名，通过核心框架接口的"Profile"属性进行访问。

4.4 CORBA 的应用开发

4.4.1 CORBA 的应用

核心框架和波形组件开发，需要参照CORBA提供的接口标准。非CORBA的应用组件接口（如：非CORBA调制/解调、非CORBA输入/输出、非CORBA安全组件）需要开发适配器程序来转换，以满足非CORBA接口组件与CORBA资源之间通信的需求。

① 使用CORBA实现接口

在SCA 系统中规定使用CORBA 作为各个软件组件间的软总线，CORBA的使用可以大大提高系统的可扩展性和可移植性，使软件组件具有很好的可替换性。CORBA 为SCA 提供操作环境。SCA 架构中的CF 是一系列标准的CORBA接口，为了实现CF的功能和优点，屏蔽各种异构系统间的区别，CF中的各种接口都必须实现成CORBA形式的接口。另外，域内各种组件间的通信，以及整个域的控制、配置与管理也是通过CORBA 来完成的。

② 为SCA 提供可插拔传输机制

CORBA 支持可插拔传输机制，CORBA 可拔插协议框架可以灵活有效地支持高速的协议和网络、实时嵌入式系统的互联以及标准的TCP/IP 协议。能解决实时嵌入式应用程序对QoS 的严格要求。CORBA 可拔插协议框架的整体结构如图4所示。

SCA规范的要求，系统底层的传输可能基于某种总线，也可能基于不同的网络接口，CORBA对应用层来说虽然统一采用标准的消息协议，但是还需要CORBA能支持对底层传输协议的配置和插拔，以管理底层的通讯机制。由于CORBA 的互操作协议模型分离了ORB 消息协议层和传输协议层，所以允许自定义ORB的消息和传输协议，这正好满足了SCA对底层传输机制的要求。

③ CORBA 支持可裁剪、小印记需求

SCA 是典型的嵌入式CORBA 应用，一方面要限制系统资源开销，另一方面它在设计时就能确定资源分配、对象的位置和对象的创建，它的系统环境是可预测的，不必在运行时动态创建和动态使用对象。基于这些特点，SCA 系统只要遵循Mini CORBA 规范即可，对于CORBA 的大部分动态创建和动态使用对象的机制以及一些不需要的功能，如动态调用机制DII 和DSI、动态数据类型Any 和DynAny、接口仓库和实现仓库以及DCE、COM 和CORBA 的互操作等，可以将其从CORBA 中裁剪掉；另外还可以对静态桩和框架进行优化，去除类型安全检查和异常等。

4.4.2 CORBA 的开发步骤

使用CORBA开发分布式应用，以前很单纯的客户程序和服务器程序分别被分割成了三个部分：一部分是应用开发者写的应用程序；一部分是代理程序，即接口定义语言(Interface Definition Language，IDL)桩和框架；最后一个部分是运行时库[8]。在CORBA 中，这些运行时库是真正完成数据的打包和解包，以及发送网络请求等操作。而代理则仅仅是调用库中的相关操作来完成处理任务。

使用CORBA开发分布式应用，包括三部分：一部分是应用开发者写的应用程序；一部分是代理程序，即接口定义语言(IDL)节点和框架；最后一个部分是运行时库[8]。在CORBA中，这些运行时库是真正完成数据的打包和解包，以及发送网络请求等操作。而代理则仅仅是调用库中的相关操作来完成处理任务[17]。

从体系结构来看，虽然CORBA 应用的开发分为3 部分，但实际需要完成的工作却不多，因为与平台和网络相关的工作都由ORB 库完成了，我们需要做的是集中精力开发自己需要的应用逻辑，如哪个对象应该跟哪个对象通信，整个对象系统如何合作完成任务等，从而使得应用的开发就像是在本地完成一样，而不是分布式应用开发，这大大地提高了开发效率，缩短了开发周期。具体来说，应用开发者需要完成的工作主要包括以下步骤：

步骤1 用IDL 语言定义对象的接口；

步骤2 使用IDL编译器编译第一步定义的接口，生成桩和框架；

步骤3 按照写本地程序一样的方式编写分布式应用程序，实现相应的伺服程序；

步骤4 编译链接客户方应用程序、客户桩和CORBA 运行时库，产生客户方可执行程序；

步骤5 编译链接服务方应用程序、服务框架和CORBA运行时库，产生服务方可执行程序。

4.5 波形组件库的开发

4.5.1 波形组件库的定义

“波形”是为实现信息的无线传输而对信息所采取的一系列变换，一般包括无线通信双方为实现信息传输而采用的所有协议[7]。在本文中，“波形组件”就是实现某种通信体制所需的功能模块，这些模块是独立于硬件的、可重用的、可执行的软件应用程序。波形组件主要包括：调制解调、信道编译码和信源编码等，其内部有多个可选的模块实现不同通信体制的功能。模块之间具有明确的接口关系，功能相对独立。所有这些模块及其接口就构成了波形组件库。

多体制无线通信波形组件库的设计采用模块化设计思路，其中最重要的工作是进行模块划分，并且定义出相应模块的功能以及连接各模块之间的接口。通过模块化设计思路对现有的各种应用波形进行模块化开发，并设计出联系这些功能独立模块的接口，并根据 SCA 的要求进行标准化，使不同的开发人员通过使用或继承的方式，方便的把这些模块加载到不同的系统上并组合在一起，实现不同的通信功能。就是由于模块化设计思路的这种方式，充分体现了 SCA 的可移植性和可重用性的优点。模块划分如图 4.5 所示。

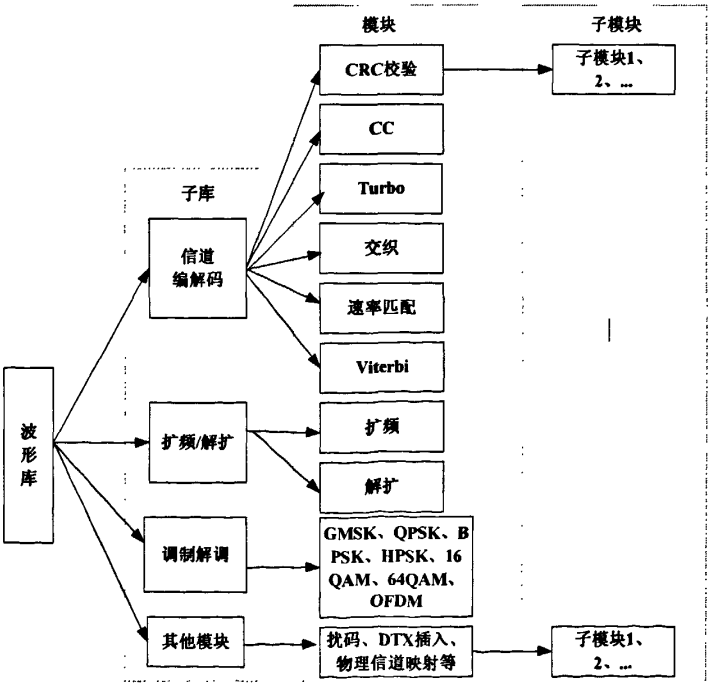

图 4.5 波形组件库功能模块的划分

Fig.4.5 Functional module division of waveform componets library

4.5.2 波形组件库的实现

建立波形库是为了便于设计人员的管理和使用，避免重复劳动，保证研究工作的顺利进行。波形组件库是软件可重构技术的基本内容，它制约着软件重构所生成的通信系统运行效率的高低。波形组件库的每个部分包括多个子程序模块，建立波形组件库的任务，就是要完成所需要的所有子程序模块的C语言代码实现。需要实现的部分波形组件的模块清单用一个表的形式给出，如表4.2所示。

表4.2 波形组件库

Table4.2 waveform componets library

通信体制	调制方式	信道编解码	信源编解码
GSM	GMSK	1)、分组编码（外编码）； 2)、(2,1,5)卷积编码（内编码） 3)、矩阵交织编码； 4)、Viterbi 译码；	1)、RPE-LTP； 包括：LPC、LTP 滤波 2)、ADPCM 语音编解码；
WCDMA	BPSK（上行） QPSK（下行）	1)、1/2 卷积编码(所有信道可用)； 2)、1/3 卷积编码（DCH、FACH 等信道）； 3)、1/3Turbo 码（DCH、FACH 等信道）； 4)、RS 编解码； 5)、上/下行扰码产生模块； 6)、MAP 译码	1)、ARM 语音编码； 2)、CRC16, $g(D)=D^{16}+D^{12}+D^{5}+1$；
WiMAX	1)、QPSK 2)、16QAM 3)、64QAM	1)、1/2 码率的卷积码 2)、Turbo 编码 3)、LDPC 编码 4)、Viterbi 译码	

波形组件的功能模块划分和设计完成后，如何调用波形组件，使波形库发挥作用，需要通过CORBA来实现。实际上，这些模块就是一些功能函数，经过IDL进行接口定义，将各个模块的实现和定义分离，并映射到编程语言上（如Java、C++）。定义了的接口的输入输出和原模块相同，具体实现需要调用模块里的函数。

需要访问某个功能模块时，就通过CORBA找到这个模块对应的接口，并赋给它输入参数，这个模块会调用功能函数并返回一个输出值给CORBA，这样就

完成了模块的访问，其实现如框图 4.6 所示。

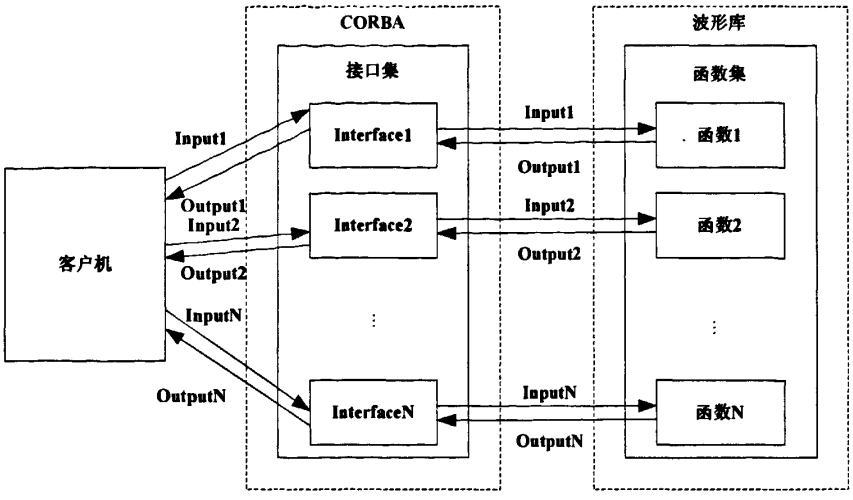

图 4.6 波形库的实现框图

Fig.4.6 The realization diagram of waveform componets library

4.6 应用软件的构建

4.6.1 系统节点的启动

系统节点的启动需要三个步骤首先根据客户程序提出的创建请求启动系统内的设备管理器，而后创建节点上的固有设备和相关服务，随后把固有的设备和服务部署到核心框架域中，以便为域内的应用实例提供相应的服务。具体的启动过程如下：

① 启动设备管理器

客户端在发出创建设备管理器的请求之后，设备管理器执行创建文件管理器的操作，并根据设备配置描述文件 DCD 提供的信息完成对节点内文件系统的加载操作。之后将 DCD 中那些默认缺省的设备注册到设备管理器上，然后创建的解析器，完成对设备配置描述文件和相关文件的解释，紧接着设备管理器创建日志端口实现设备管理器和日志服务的连接。在该步骤执行完毕之后，设备管理器便开始启动节点内的各个组件，按照 DCD 文件的说明创建相关的设备和服务，并将其部署到核心框架域中，最后设备管理器将自己注册到域管理器上。如图 4.7 所示：

② 创建节点组件

在设备管理器完成自己的注册操作之前，设备管理器还必须完成节点内所有组件的初始化配置任务，直到节点上所有的设备与服务完成创建和初始化任务之

后，设备管理器才能注册自己。其节点组件的创建过程如下图所示。需要注意的是，节点上的设备和提供的服务必须将自己注册到设备管理器上，而后再由设备管理器去完成各个组件在域管理器上的注册任务。

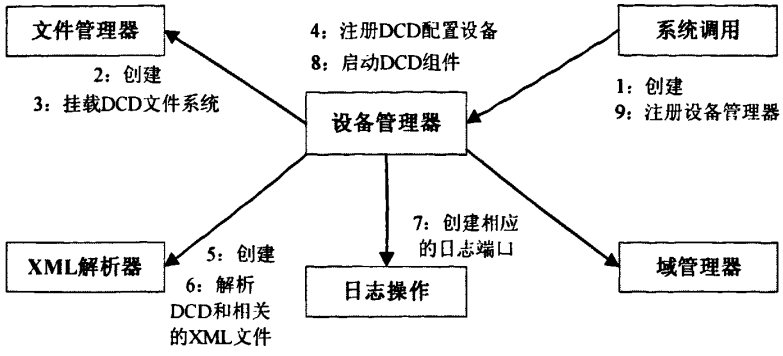

图 4.7 设备管理的启动步骤

Fig.4.7 Start step of device management

③ 部署节点组件

设备管理器完成节点内组件的创建和初始化任务后，便继续实施节点启动的第第三个步骤，部署节点组件至核心框架域中。设备管理器完成自身的注册后，域管理器即获得了节点内各注册组件的相关信息，在域管理器执行了加载操作之后，继而将设备管理器的文档说明添加到域配置文件中。

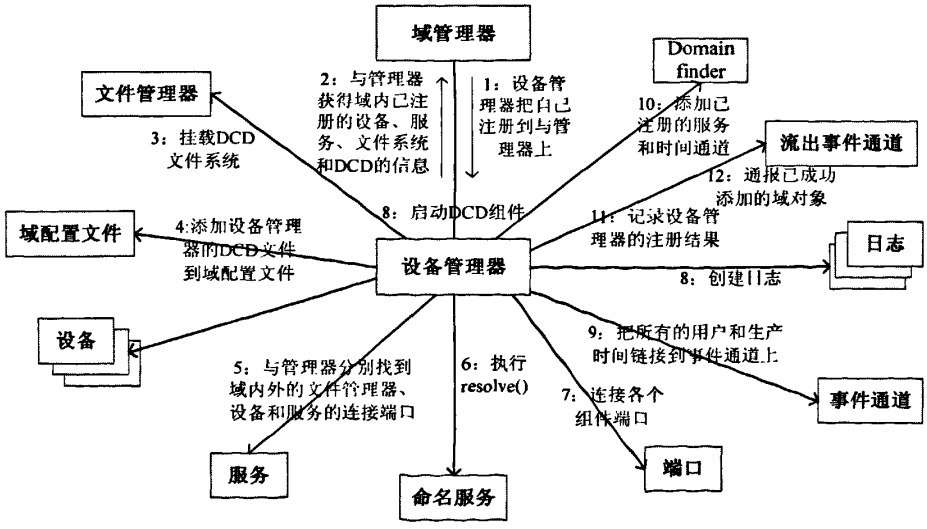

图 4.8 设备管理器部署节点步骤图

Fig.4.8 Step chart of equipment supervisor deployment node

完成了以上操作，域管理器接下来的任务便是寻找域内以及其他域的设备、服务、文件管理器和设备管理器与该节点的连接端口，并完成连接任务。而后，域管理器执行创建日志的操作，将所有生产和消费事件连接到域事件通道上，并记录设备管理器的注册结果，最后把已成功添加的域对象通知到流出事件通道中，使得用户可以知道节点启动的任务是否被完成。具体过程参照图 4.8 所示。

4.6.2 波形组件的安装

安装应用程序时，其操作步骤如图 4.9 所示。客户端首先向域管理器发出安装应用程序的安装请求，然后由域管理器将应用程序的软件配置文件添加到域配置文件中，并按照软件配置文件提供的相关信息构建该应用程序的应用工厂。应用工厂的创建任务完成后，域管理器运行 push()操作，将安装应用程序的信息事件发送到流出事件通道，而后域管理器记录应用程序安装的结果完成最后安装的步骤。

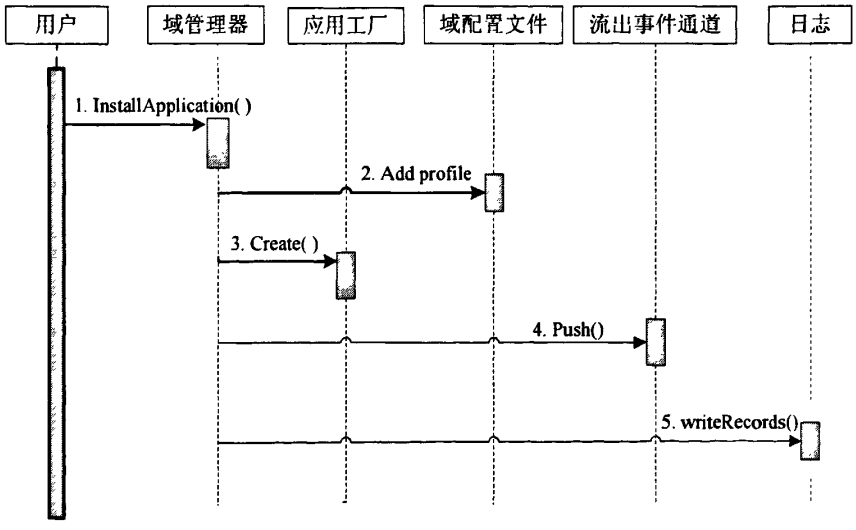

图 4.9 波形组件的安装

Fig.4.9 Installation diagram of waveform componets

在用户运行应用程序之前，必须知道系统中有哪些可用的应用程序，这样才能执行创建应用实例的任务。客户端获得应用工厂列表清单的示意图如下所示。客户端必须向域管理器发出一个请求，这个请求标志着客户端需要得到系统中已经安装的应用程序的列表清单，换句话说就是要让客户端知道有哪些类型的应用程序是有用的，是可以被用于创建应用实例的。域管理器接到请求之后，随即将域内的已经创建的应用工厂列表清单返回给客户端，客户端便可以根据自己的需要选择相应的应用工厂来完成创建应用实例的任务了。

4.6.3 应用软件的创建和拆分

软件的下载和安装是通过核心框架来实现。应用软件的构建流程如图 4.10 所示。其中“应用工厂”、“应用实例”、“域管理器”都是已经实现的 CORBA 对象。

① 构建“应用工厂”。根据用户的请求，通过 InstallApplication()接口启动并注册系统的“设备管理器”，并把设备管理器的说明文档添加到域配置文件中。然后，域管理器将需要构建的软件的配置文件添加到域配置文件中，并按照配置文件的相关信息构建该应用程序的“应用工厂”。

② 获取“应用工厂”清单。客户端通过 ApplicationFactories()获得当前创建好的“应用工厂”清单。因为，在运行应用程序之前，客户必须知道系统中有哪些可用的应用程序，这样才能执行创建应用实例的任务。

③ 创建“应用实例”。客户端在得到“应用工厂”清单后，就可以根据自己的需求创建专用类型的应用实例了。

实例化的具体过程：“应用工厂”得到客户请求—>对软件的配置文件进行审核—>审核成功后到“波形组件库”中下载应用组件—>创建所需要的资源—>初始化应用组件—>按照应用程序的软件配置文件实现各个资源的连接—>逐个配置资源组件—>初始化所有配置—>返回创建成功消息。

④ 获得创建的应用程序列表。所有创建成功的应用程序的结果都被记录到域配置文件，客户端可以通过 GetApplications( )获取到实例化的应用程序列表。

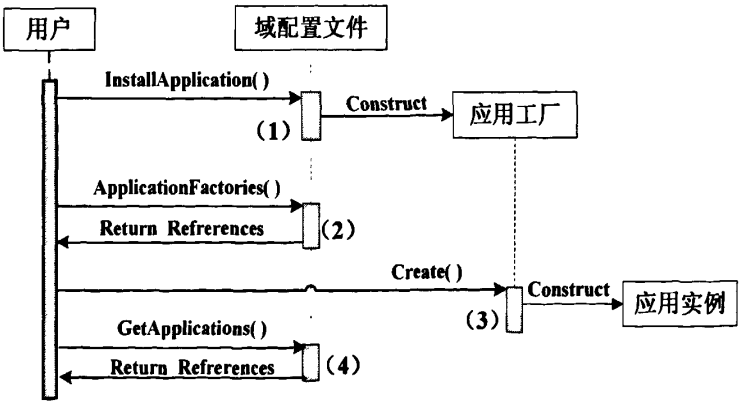

图 4.10 应用软件的构建流程

Fig.4.10 Application software's construction flow chart

当希望停止某种类型的应用实例在核心域的运行时，就需要及时地拆分这个应用实例，以返回给系统更多的容量来处理其他的波形或事件，软件通信结构同

样给出了拆分应用实例的规定，在进行系统设计时可以参照这些步骤执行。

当需要拆分应用实例时，客户端向核心域的应用发出释放对象的请求，而后由位于核心域的应用接口完成用户需要拆分的特定实例。应用接口首先拆分资源组件的各个连接端口，资源工厂释放其创建的各个资源组件，关闭该应用实例属下的资源工厂，而后应用接口释放自己创建的资源组件，关闭执行设备，卸载逻辑设备和装载设备上的各个软件组件，之后对执行和装载设备执行返还容量的操作，将这些设备的容量返回到系统中，在完成相应的日至操作后及时地将拆分的事件推入流出事件通道中，使外设及时地明确域内应用实例的拆分情况。应用程序的拆分如图 4.11 所示。

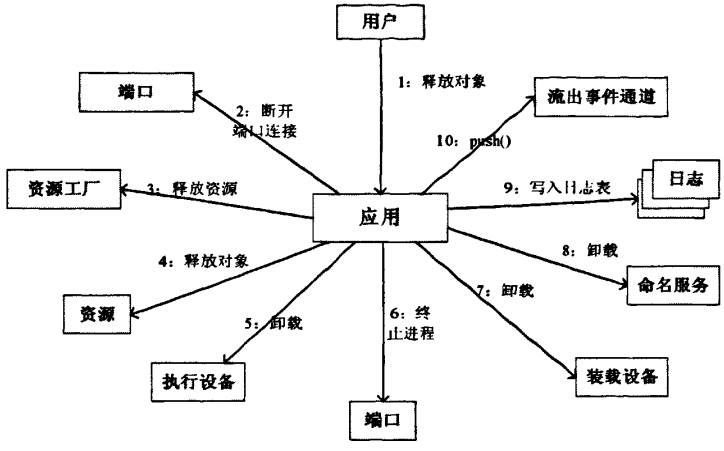

图 4.11 应用程序的拆分步骤

Fig.4.11 Resolution step of application procedure

4.7 应用实例

应用软件的动态构建是一个复杂的过程，图4.12展示了软件的主要工作流程。平台的接收端和发射端软件都运行在 PC 的 windows 环境，其中 windows 程序与嵌入式处理平台的之间的通信接口通过 Winpcap 来实现，这在本章第二节已经提到。

在 windows 环境下的应用软件采用 Visual C6.0 来开发，开发过程中大量运用了面向对象的思想，各个模块程序以类的形式进行划分，其中各个小的功能模块用函数来实现，既满足了各个模块之间的相互调用，又充分保证了各个模块之间的独立性。

在发射端，进行通信之前可以手动选择发射信号需要使用的通信体制。当发射端启动呼叫后，程序通过网口给 ARM 发送命令，ARM 控制前段的多频段射频天线。同时，在本地调用配置文件和波形组件库，并装载波形组件。当所有准备

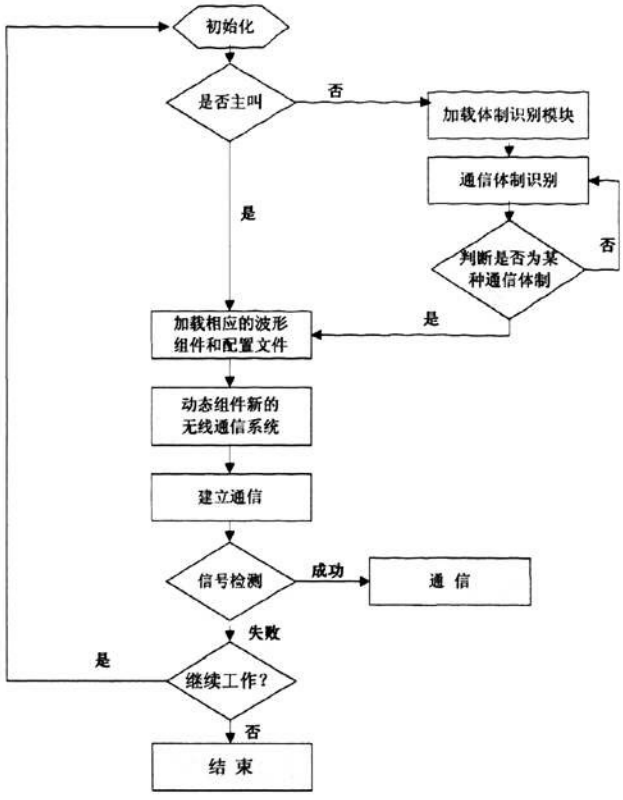

图 4.12 平台的软件框架

Fig.4.12 Software frame of platform

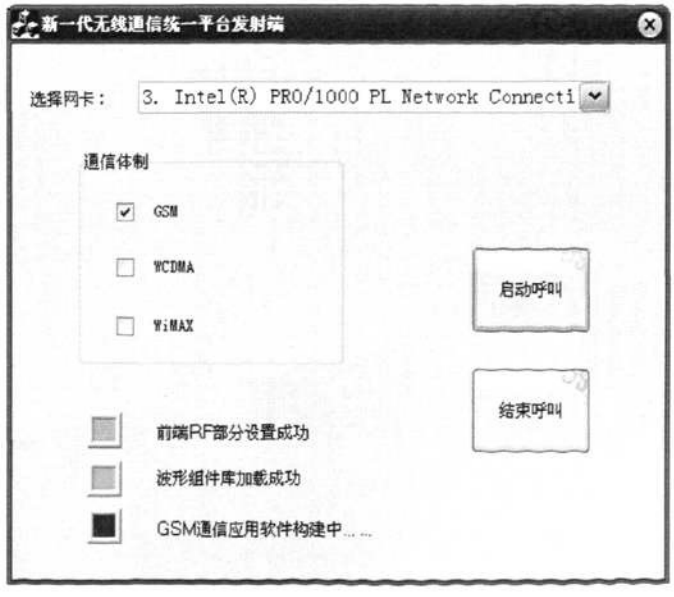

图 4.13 发射终端界面

Fig.4.13 Surface of transmitting terminal

工作完毕后就可以开始构建所需的应用软件了。发射端界面如图 4.13 所示。其中绿色标志表示该功能模块的工作成功完成，红色表示还未开始工作或者运行结果失败。其部分程序代码见附录 2 所示。

在接收端，相对工作更为复杂。当在轮询的频段中发现通信信号后，首先启动体制识别模块。当体制识别成功后把识别的结果写进配置文件，并触发加载配置文件和波形组件的工作。当构建应用软件成功后提示操作人员人工启动与发射方之间建立通信。接收端界面如图 4.14 所示。

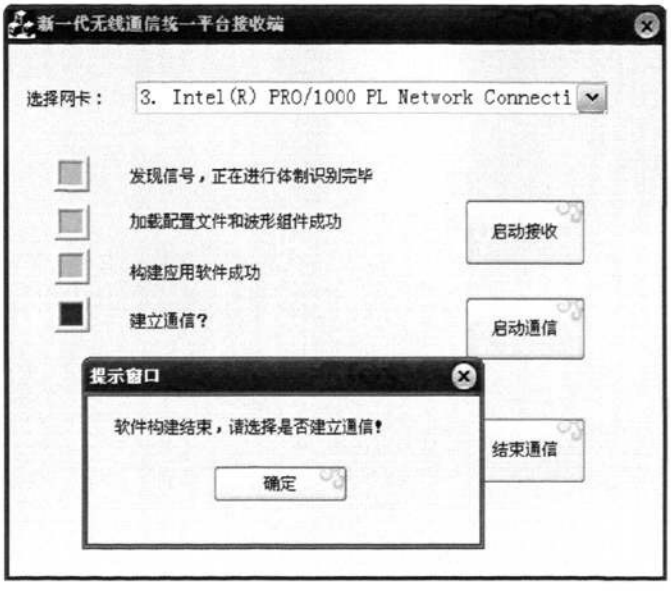

图 4.14 接收终端界面

Fig.4.14 Surface of Receiving terminal

5 结论与展望

纵观整个无线电系统的发展过程，数字化、模块化的系统设计理念已成为当代无线电系统发展的主流趋势，自从上个世纪80年代以后，无线电系统的频段信道逐步向多频段、多模式迈进，无线电系统服务则朝着多元化数据语音、数据、图像和视频等方向发展，而无线电系统的体系结构也正处在从硬件化的设计理念向基于公共结构的软件平台方向发展。在数字域实现对无线电系统的设计、验证、完善及其应用，对于无线电系统在军事领域和民用领域更快、更好、更深入地发展带来了巨大的灵活性，这也使得人们构建"经得起考验的无线电系统"的愿望成为了可能。

作为军用软件无线电系统开发的一部分，SCA在设计中大量地应用了软件无线电体系结构两种新的概念，围绕一种开放式体系结构来构建框架，以及利用面向对象技术来构建其他部分。力求创建一种开放式的、便于系统重构的、扩展的以及模块化的一个顶层设计规范，以通过采用强制性地定义一系列接口、运作规则及其系统需求的方法，来促进系统的可移植性、互操作性和组件的可交换性，实现软件重用和框架结构可扩展的目标。

遵循SCA规范，利用可重构软件无线电技术，针对新一代无线通信系统要求，对无线通信统一平台的软件可重构进行研究。完成该课题的研究，不仅对软件无线电领域的研究有着重要的参考和指导价值，而且能够促进无线通信领域的基础理论和技术的发展。

基于SCA的新一代无线通信平台是一种开放的、可互操作的通信系统。本文利用嵌入式处理模块与通用PC相结合的方式，对平台的软件重构性进行研究，其目的是为了在SCA框架内，对软件重构的一些关键技术的实现进行研究，并对多种通信体制下软件可重构的实现机制进行探讨。

虽然在整个项目的研究过程中，为了提高研究的效率，一部份工作是在通用PC的环境中来完成的，但是模块化和面向对象的思想得到了充分的体现，而且SCA中的各个关键技术及其软件重构的机制也得到了合理的验证，整个系统的工作流程基本实现。可见，基于SCA的新一代无线通信系统的软件可重构是可行的。

在以后的研究工作中，以下几个方面还需要做进一步的研究：整个系统需要全部移植到嵌入式处理平台上，在嵌入式操作系统的环境下进行研究；前端的通信体制识别的算法除了进行调试识别外，还可以根据各种通信信号的特点进行鉴别（比如工作频率、信道编码等），以提高体制识别的准确性；在整个实践过程中还不够智能化，在细节上的处理方法还需要做进一步的探讨。

放眼未来，可以预见 SCA 必将对 SDR 以后的发展产生巨大影响，它为新一代无线通信系统的发展指明了方向。未来的无线通信设备，应该是集多种通信体制于一体，通过系统的软件重构或者在线升级就能实现不断更新的通信制式。当然，可重构的新一代无线通信系统的商用化还有很长的路要走，还需要全球各个领域的一起努力。

致 谢

本文的研究工作是在我的导师葛利嘉老师和清华大学的粟欣老师的精心指导和悉心关怀下完成的，在我的学业和论文的研究工作中无不倾注着老师们辛勤的汗水和心血。两位老师的严谨治学态度、渊博的知识、无私的奉献精神使我深受启迪。从尊敬的葛老师和粟老师身上，我不仅学到了扎实、宽广的专业知识，也学到了做人的道理。在此我要向两位老师致以最衷心的感谢和深深的敬意。

本课题内容隶属清华大学承担的 863 项目，课题的研究工作大部分是在清华大学完成。在课题的研究过程中，得到了清华大学无线中心的老师和同学们的大力帮助。在清华的一年时间里，粟欣老师无论从学习和生活上都无私地给予了我很多的帮助，徐翼和刘莉莉同学在论文的撰写过程中也提出了很多宝贵意见，在此表示感谢。

在此，向所有关心和帮助过我的领导、老师、同学和朋友表示由衷的谢意！

衷心地感谢在百忙之中评阅论文和参加答辩的各位专家、教授！

李明全

二 OO 九年四月 于重庆

参 考 文 献

[1] 李承恕. 复合可重构—欧洲走向 4G 的研发之路[J]. 中兴通讯技术. 2003,6: 28～31

[2] 陈杰. 可重构 R F 芯片相继问世[J]，电子设计应用，2006,4:30-48.

[3] Xiang Ju QIN, Ming Cheng ZHU, Tai Yi ZHANG, Zhong Yi WEI. Analysis of the Fundamental and Implementatio Method about Dynamic Recofigurable FPGA. 电子器件. 2004,6:277-282.

[4] 韩亮,陈杰,陈晓东. 嵌入式可重构 DSP 处理器的指令译码器设计[J]. 微电子学与计算机. 2004,7:91-94.

[5] 罗振璧,于学军,刘阶萍等. 可重构性和可重构设计理论[J]. 清华大学学报(自然科学版). 2004,44(5):577-580.

[6] 吴晓渊. 分布式系统重构算法的初探[J]. 广西科学院学报. 2002,12.

[7] Piotr Rykaczewski, Dariusz Pienkowski, Radu Circa, Bernd Steinke. Signal Path Optimization in Software-Defined Radio Systems. IEEE TRANSACTIONS ON MICROWAVE THEORY AND TECHNIQUES. VOL. 2005,53(3): 1056-1064.

[8] Takashi Shono, Yushi Shirato, Hiroyuki Shiba, Kazuhiro Uehara, Katsuhiko Araki, and Masahiro Umehira. IEEE 802.11 Wireless LAN Implemented on Software Defined Radio With Hybrid Programmable Architecture. IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS. 2005,4(5):2299-2308.

[9] 王际兵,赵明,许希斌,姚彦. 软件无线电技术[J]. 电子科技导报. 1998:23-27。

[10] 王永明. 软件通信结构研究[D]. 国防科学技术大学硕士学位论文. 2006,10:7-43

[11] 邱永红,朱勤. 基于软件通信体系结构的无线通信系统研究[J]. 系统工程与电子技术.，2004,5(6):621-624.

[12] 「美」Jeffrey H.Reed 等著,陈强等翻. 软件无线电—无线电工程的现代方法[M]. 人民邮电出版社 2004 年 7 月出版.

[13] 洪锡军,刘献科,张激. 基于SCA的无线通信技术研究[J]. 计算机工程. 2005,31(8):120-122.

[14] 陈雪莲. 基于软件通信体系结构的波形应用接口设计与实现[D]. 国防科学技术大学研究生院. 2004.11.

[15] 李雅萍，杨尚森，李阳. CORBA技术在SCA系统中的应用[J]. 计算机工程与设计. 2008,29(16):4200-4206.

[16] 孙佩刚，赵海，张文波，赵明，基于软件通信体系结构的波形实现及其研究[J]. 计算机工程. 2006,32(17):170-174.

[17] 杨燕. 基于CORBA的"自适应软总线"设计技术[D]. 西北工业大学硕士学位论文.

2002.3:13-15.

[18] 王树凤，基于CORBA构件模型的动态配置平台的研究与实现[D]，国防科技大学硕士学位论文，2004.11:40-52.

[19] Michael Barth, Jonghun Yoo, Saehwa Kim, and Seongsoo Hong. "Design Patterns for Releasing Applications in C++ Implementations of JTRS Software Communications Architecture"[C]. Proceedings of the Ninth *IEEE* International Symposium on Object and Component-Oriented Real-Time Distributed Computing. 0-7695-2561-X/06,2006.

[20] Saehwa Kim, Jamison Masse, Seongsoo Hong, and Naehyuck Chang. "SCA-based Component Framework for Software Defined Radio"[C], Proceedings of the *IEEE* Workshop on Software Technologies for Future Embedded Systems (WSTFES.03),0-7695-1937-7/03. 2003.

[21] Support and Rationale Document for the Software Communieations ArChiteCture SpeeifiCation(v2.2)MSRC—S000SRDV2.2. 2001.12.

[22] Gamma E.Design Patterns:Elements of Reusable Object-oriented Software.Addison-Wesley. Reading，Massachusetts. 1995.

[23] The WinPcap Team. WinPcap Documentation[S]. http://www.winpcap.org

[24] Object ManagementGrouP."Real—TimeCORBA".OMG Document[J]. orbos/99—02—12ed. 1999.3.

[25] Object ManagementGrouP."CORBA Messaging SPecification".OMG Document orbos/98—05—05ed. 1998,5.

[26] Object ManagementGrouP."The Common Object Request Broker:Architecture and SPecification"，OMG Document，Version2.6，2001,12.

[27] A.Mehra，A.Indiresan，and K.G.shin."Structuring Communication Software for Quality-of-Service GuaranteeS".IEEE Transanctions On Software Engineering[J]，1997，23(10):616—634.

[28] Michi Henning, steve Vinoski.. 基于C++ CORBA高级编程[M]. 清华大学出版社. 2000,7.

[29] 骆骋,粟欣,赵明,姚彦. 基于通用计算机和网络的软件无线电结构[J]. 清华大学学报（自然科学版）. 2001.7

[30] 李振东,李文臣,任之良. 软件无线电分层体系结构[J]. 天津通信技术. 000.9.

[31] 杨雪松,王秉中. 可重构天线的研究进展[J]. 系统工程与电子技术. 2003,25(4).

[32] 李扬,李国通,杨根庆. 通信信号数字调制方式自动识别算法研究[J]. 电子与信息学报.2005,27(2)

[33] 刘献科,栋岭,陈涵生. 软件定义无线电及软件通信体系结构规范[J]. 计算机工程. 2004,30(1):95-96,131.

[34] 李庭胜. 基于DSP+FPGA系统在线重构技术[J]. 信息安全与通信保密. 2005,11.

[35] 张剑锋,马叶锋. 软件无线电及SCA简介[J]. 通信与广播电视. 2006,1.

[36] 葛洪慧,葛衡,葛洪芳. 面向对象中间件CORBA技术介绍[J]. 行业应用. 2007.

[37] 苏琨. 基于Ioc模式的软件开发框架重构[J]. 陕西电子技术. 2007,2.

[38] 朱新峰. 三层架构应用可重构性研究与实王[D]. 西北工大硕士毕业论文集. 2006.

[39] 华奇兵,许文波,费娜. 面向对象软件重构[J]. 重庆邮电学院学报. 2004,16(2).

[40] 杨芙清,梅宏,李克勤. 软件复用与软件构件技术[J]. 电子学报.1999,21.

[41] 张禄林,郎晓虹. 软件通信结构的目标及其构成[J]. 通信世界.2001,9(82).

[42] 李明全,粟欣,葛利嘉. 基于SCA的新一代无线通信系统的软件可重构研究[C]. 2008全国通信新理论与新技术学术大会（CTW2008）论文集. 2008,12.

附 录

A. 作者在攻读硕士学位期间发表的论文：

[1] 李明全,粟欣,葛利嘉. 基于SCA的新一代无线通信系统的软件可重构研究. 2008全国通信新理论与新技术学术大会（CTW2008），2008,12.

[2] 李明全,粟欣,葛利嘉. 新一代无线通信系统的软件可重构研究. 计算机工程与设计. 已录用

B. 部分程序实例：

```
void CSCADemoDlg::InitLAN()
{
    CString strTemp;
    if (pcap_findalldevs(&alldevs, errbuf) == -1) {
        strTemp.Format("Error in pcap_findalldevs: %s\n", errbuf);
        AfxMessageBox(strTemp);
        return; }
    /* 找到要选择的网卡结构 */
    UINT nNicNumSel = m_comblan.GetCurSel() + 1;
    UINT i;
    for(d=alldevs, i=0; i< nNicNumSel-1 ;d=d->next, i++);
    /* 打开选择的网卡 */
    if ( (pcapt= pcap_open_live(d->name, // 设备名称
                65536, // portion of the packet to capture.
                    // 65536 grants that the whole packet will be captured on all the MACs.
                1,     // 混杂模式
                1000, // 读超时为1秒
                Errbuf // error buffer) ) == NULL)
    {
        strTemp.Format("Unable to open the adapter %s.", d->description);
        AfxMessageBox(strTemp);
        /* Free the device list */
        pcap_freealldevs(alldevs);
        return ;
    }
    pcap_freealldevs(alldevs);   // Free the device list
}

void CSCADemoDlg::InitLanPacket(double *dataIn/*传输数据*/ ,int ndataLen/*数据的长度*/)
{   /*********************  header begin ************************/
    ReadMAC(m_chSendPacket, szDesMAC);
```

```
ReadMAC(m_chSendPacket+6, szSrcMAC);
m_chSendPacket[12] = 0x08;          // 协议类型: 080A表示本帧为数据传输
m_chSendPacket[13] = 0x0A;
for(int i = 14; i < 18; i++)   m_chSendPacket[i] = 0;
m_chSendPacket[18] = 0x00;          // 19~22 timestamp
m_chSendPacket[19] = 0x0b;          //
m_chSendPacket[20] = 0x00;
m_chSendPacket[21] = 0x00;
char tempbuf[2];
//sprintf(tempbuf, "%x", nDataLen);
tempbuf[0] = (unsigned char)( ((ndataLen*2)>>8)&0x00FF ); //一个int字节分成2个UChar来
传所以乘以2。
tempbuf[1] = (unsigned char)((ndataLen*2)&0x00FF);
m_chSendPacket[22] = tempbuf[0];          // 23~24表示数据包长，发不同数据时注意更改。
m_chSendPacket[23] = tempbuf[1];          // 固定只发8个字节数据
// 除去CRC之外的整个包的长度(头加数据的长度)
nPacketLen = 24+ndataLen*2;
/*********************  header end ***************************/
LoadData(m_chSendPacket,dataIn,ndataLen); // 装载发送数据
/*********************  CRC BEGIN  ***************************/
// 把unsigned char 装换成const char
char tempPacketChar[MAX_LENGTH];
const char *p;
p = tempPacketChar;
memcpy(tempPacketChar,m_chSendPacket,MAX_LENGTH);
// 对传输之前的数据包做CRC校验,并转化成16进制
unsigned long nCrcOut = RunCRC32(tempPacketChar, nPacketLen, CRC32_DEFAULT);
char crcbuf[10],crcTempchar[10];
sprintf(crcbuf, "%x", nCrcOut);
// 把校验值加到传输前的buf中
char chTemp[2], chRd;
for (i = 0; i<4; i++)
{
    chTemp[0] = crcbuf[2*i];
```

```
            chTemp[1] = crcbuf[2*i+1];
            sscanf(chTemp, "%x", &chRd);
            crcTempchar[i] = chRd;
        }
        m_chSendPacket[nPacketLen]   = crcTempchar[0];
        m_chSendPacket[nPacketLen+1] = crcTempchar[1];
        m_chSendPacket[nPacketLen+2] = crcTempchar[2];
        m_chSendPacket[nPacketLen+3] = crcTempchar[3];
        /*********************  CRC END  ***************************/
}

void CSCADemoDlg::OnSend()
{
        GetMAC();           // 获取MAC地址。
        InitLAN();          // 初始化网卡，获得打开的网卡指针。
        InitLanPacket();    // 装载发送的数据包。
        if (pcap_sendpacket(pcapt, m_chSendPacket, nPacketLen ) != 0) // 发送数据包
        {
            fprintf(stderr,"\nError sending the packet: \n", pcap_geterr(pcapt));
            return;
        }
        m_progress.SetRange(1,nPacketLen-1);
        m_progress.SetStep(1);
        for(int i=0;i<nPacketLen;i++)
            m_progress.SetPos(i);
}

void CSCADemoDlg::OnReceive()
{
        int res;
        struct pcap_pkthdr *header;
        const u_char *buf_rev;
        time_t local_tv_sec;
        struct tm *ltime;
```

```
    char timestr[16];
    CString strLen;
    //获取数据包
    while((res = pcap_next_ex( pcapt, &header, &buf_rev)) >= 0){
        if(res == 0) break;
        // 将时间戳转换成可识别的格式
        local_tv_sec = header->ts.tv_sec;
        ltime=localtime(&local_tv_sec);
        strftime( timestr, sizeof timestr, "%H:%M:%S", ltime);
        strLen.Format("发送数据 %d bytes\n", header->len);
        const u_char *revChar;
        revChar = buf_rev+24;  // 取出字符
        m_szReceiveChar.Format("%s" , revChar);
        m_revStr = strTime;
        UpdateData(false); }
    if(res == -1)
        printf("Error reading the packets: %s\n", pcap_geterr(pcapt));
}
```