电 子 科 技 大 学

UNIVERSITY OF ELECTRONIC SCIENCE AND TECHNOLOGY OF CHINA

专业学位硕士学位论文

MASTER THESIS FOR PROFESSIONAL DEGREE

论文题目 基于 VxWorks 系统的 VoIP 网关软件设计与实现

专业学位类别 工程硕士

学 号 201392010108

作者姓名 郑鑫

指导教师 蒋体钢 副教授

分类号 _____ 密级 _____

UDC [注1] _____

学 位 论 文

基于 VxWorks 系统的 VoIP 网关软件设计与实现

（题名和副题名）

郑鑫

（作者姓名）

指导教师 蒋体钢 副教授

电子科技大学 成 都

郑方森 高 工

万欣测控技术公司 绵 阳

（姓名、职称、单位名称）

申请学位级别 硕士 专业学位类别 工程硕士

工程领域名称 软件工程

提交论文日期 **2016 年 5 月** 论文答辩日期 2016.5.29

学位授予单位和日期 电子科技大学 2016 年 06 月

答辩委员会主席 _____

评阅人 _____

注 1：注明《国际十进分类法 UDC》的类号。

DESIGN AND IMPLEMENTATION OF VOIP GATEWAY SOFTWARE BASED ON VXWORKS SYSTEM

A Master Thesis Submitted to

University of Electronic Science and Technology of China

Major: **Master of Engineering**

Author: **Zheng Xin**

Advisor: **Associate Professor Jiang TiGang**

School: **School of Communication & Information Engineering**

独创性声明

本人声明所呈交的学位论文是本人在导师指导下进行的研究工作及取得的研究成果。据我所知，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，也不包含为获得电子科技大学或其它教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示谢意。

作者签名：_____ 日期： 年 月 日

论文使用授权

本学位论文作者完全了解电子科技大学有关保留、使用学位论文的规定，有权保留并向国家有关部门或机构送交论文的复印件和磁盘，允许论文被查阅和借阅。本人授权电子科技大学可以将学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存、汇编学位论文。

（保密的学位论文在解密后应遵守此规定）

作者签名：_____ 导师签名：_____

日期： 年 月 日

摘　要

近二十年，在技术不断发展的前提下，世界互联网得到飞速的发展，使得我们几乎能在世界上任何角落连接互联网。与此同时，我国的互联网也在不断拓展。而实时通信技术也因为互联网的普及而得到了广泛的应用，其中，VoIP 技术是一项新兴的实时通信技术，通过在互联网上传输话音数据，使人们能够更加便捷的进行语音通信。

在网络广泛应用之前，人们大多使用电路交换方式的 PSTN。在多年的积累下，PSTN 网络遍布全球，并且在今后长时间内 PSTN 会继续在通信方式上占据很大的比例。在这个实际情况下，研究并开发一种连接 PSTN 和 IP 网络的设备就成为了一种必然。

论文基于以上观点，提出一种网关设备的设计方案，其能够有效沟通 PSTN 网络和 IP 网络，使两种网络用户能够可靠的进行语音通信。

论文对网关设备进行了硬件、软件设计。其中硬件部分包含有微处理器、话音 AD/DA 转换芯片、话音编解码芯片等。软件系统设计采用模块化的方案，主要包含话音处理模块、信令转换模块和以太网处理模块等。软件系统的设计与实现是本文工作研究的重点。

软件系统以具有强大功能的 IXP425 处理器芯片为核心，运用嵌入式 VxWorks 作为底层操作系统，开发内容包括语音编码和解码、信令转换的设计和实现、以太网传输实现等。

经过开发和设计，该网关设备实现了以下功能：模拟语音的采集、编码及其逆过程；话音编码类型的转换；PSTN 侧与 IP 侧信令的转换；信令、语音数据在网络上的传输。

最后根据设计目标对网关设备进行了性能测试。测试结果验证了本文提出的基于 VxWorks 系统的 VoIP 网关设计方案和开发过程的正确性。

关键词：VoIP，SIP，PSTN，网关。

ABSTRACT

In recent twenty years, under the premise of the continuous development of technology, Internet world get rapid development, making us can connected to the Internet in any corner of the world. At the same time, China's Internet is constantly expanding.And real time communication technology got a wide range of applications because of the popularity of the Internet , Thereinto,VoIP technology is an emerging technology of the real time communication, that people can making voice communication more conveniently by transmiting voice data over the Internet.

Before the widely used of network，most people use PSTN which is circuit switched. Under the accumulation of many years, PSTN network worldwide, and the PSTN will continue to occupy a large proportion in the means of communication for a long time. In the actual situation, the research and development of a device whice can connect the PSTN and IP network has become inevitable.

Based on the above viewpoint, this thesis puts forward a design scheme of a gateway equipment, which can effectively communicate the PSTN network and IP network, so that the two network users can communicate with voice.

The hardware and software of the gateway equipment is designed in this thesis. The hardware part includes microprocessor, voice AD/DA conversion chip, voice codec chip, etc.. Software system design uses the modular program, mainly contains the voice processing module, the signal conversion module and the Ethernet processing module and so on. The design and implementation of software system is the emphasis of this thesis.

Software system with powerful function of IXP425 processor chip as the core,using embedded VxWorks as the underlying operating system, the development of content including voice coding and decoding, design and implementation of Signaling conversion, Ethernet transmission realization.

After the development and design, the gateway device realized the following functions: analog voice acquisition, coding and inverse process; voice coding type conversion; conversion of PSTN and IP side signaling; signaling and voice data in the network transmission.

At last, the performance of the gateway is tested according to the design goal. The

test results verify the correctness of the design scheme and the development process of VoIP gateway based on VxWorks system.

Keywords: VoIP，SIP，PSTN，Gateway.

目 录

图表目录

第一章 绪 论

现如今，遍布全世界的，开放而对等、灵活可扩展的网络平台已经被建立起来，其已经全面渗透到社会的各个领域，成为生产建设、经济贸易、科技创新、文化传播、生活娱乐的新型平台。在此基础上，许多传统的信息传输方式和服务业务类型正在发生巨变，基于 IP 网络的各种通信方式和服务业务正在兴起，其必然会替代传统的通信方式。在各种网络通信方式中，最具前景的 VoIP 正在蓬勃发展，其具有通话质量好、效率高、呼通率高、成本低等独特的优点，越来越被人们所接受。

近年来我国互联网发展迅速，已成为全球互联网发展的重要组成部分。以此为基础，加上方便快捷和便宜的成本，国内对 VoIP 的需求特别强劲。因此 VoIP 设备的研制对于 VoIP 系统建设和发展具有重大意义。

1.1 研究背景与意义

本课题来源于成都某公司所需——铁路区间通信系统项目，本单位与成都某公司共同开发在铁路区间进行话音和数据业务的传输。VoIP 网关的设计与实现是项目之一。

项目要求 VoIP 网关作为一个集成的软件环境，在 VxWorks 嵌入式操作系统平台的基础上，实现模拟侧电话或 PSTN 网络与 IP 网络之间语音的互通，并且实现两种系统之间信令的转换；并通过外部硬件所提供的电话接口，将模拟的声音信号进行压缩与封包，以数据封包的形式，依靠网关所提供的以太网接口，在 IP 网络的环境进行语音信号的传输。

VoIP 设备的研制对于通信网络建设具有重要的实际意义：

VoIP 正在逐步代替传统通信业务成为全球通信领域的主导者。VoIP 在 2010 年后的发展逐年成爆炸式增长，同时期我国的 VoIP 业务也得到了蓬勃发展。由于我国人口基数大，人们更加倾向于话费低廉的 VoIP 电话，从而使得市场规模逐年扩大。到目前为止，VoIP 电话与传统电话已经各占通信市场的半壁江山，而前者还在高速的增长中。

PSTN 是以电路交换为基础的技术，其历经了数十年的发展，已在全球形成了绝对的规模上的优势。因此 VoIP 的话务量绝大部分是与 PSTN 客户相互通话的业务量，所以在这种情况下，对于 IP 网络和 PSTN 网络之间互联、互通的有效性提出了更深层次的技术要求。

为融合现阶段的两种网络，实现 VoIP 与 PSTN 互通，就需要语音网关设备来解决在通话过程中存在的信令互通和语音传输这两大问题。因此，设计一个有效地连接 PSTN 网络和 IP 网络的小型通信网关设备，为家庭、小型公司及单位提供方便、快捷的通信方式，就有了意义。

1.2 国内外研究现状

在 VoIP 的实现中，最重要的是信令的处理。一套完备的信令流程是呼叫能够完成的可靠保证，并且也是通话质量的保证。在现阶段，比较优秀的信令体系是 IETF 的 SIP。SIP 协议是一种应用层协议，其规范并初始化了呼叫的过程，在连接了通话双方之后，通过 UDP 或 TCP 进行语音的传输，网络开销非常小，时效性很高，在当前大背景前提下，SIP 的光明前途勿庸置疑。

SIP 协议起源于 IETF 的会议，在第 35 届会议上，明确提出了会话初始协议 SIP。从此 SIP 组织逐渐获得同行业的高度关注，同时 IETF 在之后的一段时间内单独成立了 SIP 工作小组。2002 年 7 月，SIP 独立工作小组修改了 RFC2543 中的一些不可靠、不正确内容，发布了新标准 RFC3261，对 SIP 进行了增强，并对其定义进行了扩展。

在核心协议之外，SIP 的工作组还制定了一些补充协议，包括了消息头、安全、方法扩展和 QoS 的 30 个新文档作为 SIP 协议的补充协议，其中包括有 RFC3311 文档、RFC3327 文档、RFC3262 文档等。

到目前为止，发展速度快且应用范围较广的 IP 电话信息系统是无线语音、多媒体、网络 P2P 及远程教育等研究项目，目前具有较大影响的研究项目包括如下：积极致力于 SIP 应用与开发的 SER(SIP Express Router)，隶属于 Intel 工作组；在 SIP 的基础上来实现关于 VoIP 的应用操作的包括 Cisco 公司和 DataConnection 公司在内的其他公司；隶属于 SIPFoundry (ne Open Source SIP PBX for Linux) 的 sipX 项目；隶属于 Equivalence 公司的 OPAL(Open PhoneAbstraction Library)等。与此同时，国内的 SIP 研究工作也在进行之中，并获得不凡的成就。

国内科研机构或有实力的公司都基于 SIP 科研成果开展了对嵌入式终端的研究、生产工作，其逐渐大规模开始了以 SIP 为基础的嵌入式终端项目研究工作。

1.3 本论文的主要内容和结构安排

如前文所述，随着网络技术的发展，在结合嵌入式技术的基础上，我们能够架构出符合要求的网关设备，本文致力于设计出一个以微处理器为中心的网关设备，其基于 SIP 信令协议和 VxWorks 操作系统，有效地连接 PSTN 网络和 IP 网络，

以满足设计的需要。

研究的主要内容包括：

1.设计网关的系统及结构模型，指明该模型的实现方法和功能。

2.对话音编解码进行分析，利用芯片实现模拟语音信号与数字语音信号的转换。

3.对 SIP 协议进行分析，了解其体系结构以及流程等机制，设计其在设备中的实现方式。

4.以研究嵌入式为主调，在综合挑选适当的软件、硬件环境的基础上，搭建相应的系统平台；研究 VxWorks 操作系统的工作机制和原理，并将 VxWorks 系统移植到硬件平台；在 VxWorks 操作系统的基础上，实现话音的转换和信令的转换功能。

5.搭建测试平台，模拟实际运用场景进行网关设备的全面测试。包括呼叫成功率、接通时间、以太网丢包率等测试，在测试基础上论证方案是否成功。

本论文有七个章节，以从总体到细节的方式阐述了网关设备的设计过程，并通过测试进行论证。

第一章 绪论。通过对当前网络通信背景的介绍，展现本设备研究的意义。并阐述了近期国内外一些科研和研究的进程。

第二章 需求分析。包括关键技术分析，设备的需求分析，VoIP 网关设备功能设计和软件开发环境分析。

第三章 VoIP 网关设备概要设计。总体上对系统设计进行描述，提出硬件总体设计和软件概要平台设计。

第四章 话音处理模块软件详细设计与实现。分析并设计了话音处理功能和软件。

第五章 信令转换及以太网处理模块软件详细设计与实现。分析并设计了信令转换功能和软件、以太网处理软件。

第六章 系统测试和结论。对设备的呼叫成功率、接通时间、以太网丢包率等功能性能指标进行测试，并给出结论。

第七章 总结与展望 对本论文的研究工作做了一个总结，提出了优点和缺点。

1.4 本章小结

本章主要对 VoIP 的研究背景与意义、国内外研究现状进行了阐述，并对论文的主要内容和结构安排做了说明。

第二章 需求分析

为了达到能够在 IP 网络上用语音进行通信的目的，我们首先要进行模拟语音信号的 AD/DA 转换，把模拟信号进行数字化，之后会通过编码方式的转换将语音数据变为适合 IP 网络的码率，最后通过协议传输转换后的 IP 数据包。IP 电话极大程度上降低了对费用的要求，同时也在一定程度上对网络带宽的利用率进行了改进，而且这种技术的运用对宽带多媒体的应用起到了增强和促进发展的作用。

VoIP 的传输平台是分组交换网络，使之可以采用 UDP 协议进行传输，这就要求对模拟语音信号进行特殊的处理，处理的顺序依次为压缩、编码、打包等。而传统方式的电话网传输语音是根据电路交换的方式来进行。

2.1 关键技术分析

由于 VoIP 建立在分组交换上，因此分组交换具有的缺点都会体现在 VoIP 上，比如丢包、时延等。因此其技术主要致力于解决此问题。

1.信令技术

如果要实现电话的呼叫，必须有信令来支持，主流的网络电话信令有 IETF 组里包含的 SIP 和 ITU-T 组里包含的 H.323。H.323 系列制定出了关于多媒体通信的标准，此标准是建立在无质量保证和服务保证的网络分组上。而为了解决遇到的很多问题，IETF 开发了协议 SIP。两种信令相比来说，SIP 只用于初始化呼叫，协议简单可靠，而且资源占用很小。

2.话音处理技术

模拟信号要实现在 IP 网络上长距离的传输，必须经过压缩编码的过程。现在主流的编码技术包括 PCM 编码和 G.729 编码方式。前者采用 64kbit/s 的带宽，后者可以达到 8kbit/s 的带宽。两者语音质量都完全能满足通话的需求。

3.QoS 保障技术

VoIP 需要保障技术来确保通话质量和防止拥塞。

4.计算机电话集成技术

也就是说我们要将常规用的电话移植到个人电脑上去，使得使用更加方便。

2.2 设备的需求分析

本文设计的 VoIP 设备应用在铁路区间通信系统，用户的需求整理如下：

1.网关能够达到系统的总体要求。

本铁路区间通信系统项目总体框图如图 2-1 所示。

系统由一个铁路中心站点和多个铁路区间站点组成。各个区间站点通过光纤同中心站点相连。中心站接入设备通过光纤为每一个铁路区间站点提供一路以太网接口。

每个铁路区间站点安置一台 VoIP 网关，其一方通过以太网口接入区间设备，另一方可以接电话若干。

中心接入设备通过以太网与 VoIP 网关连接。VoIP 网关通过交换机连接 PSTN 网络。

要保证系统内所有电话都能互通，并能拨打 PSTN 侧电话。

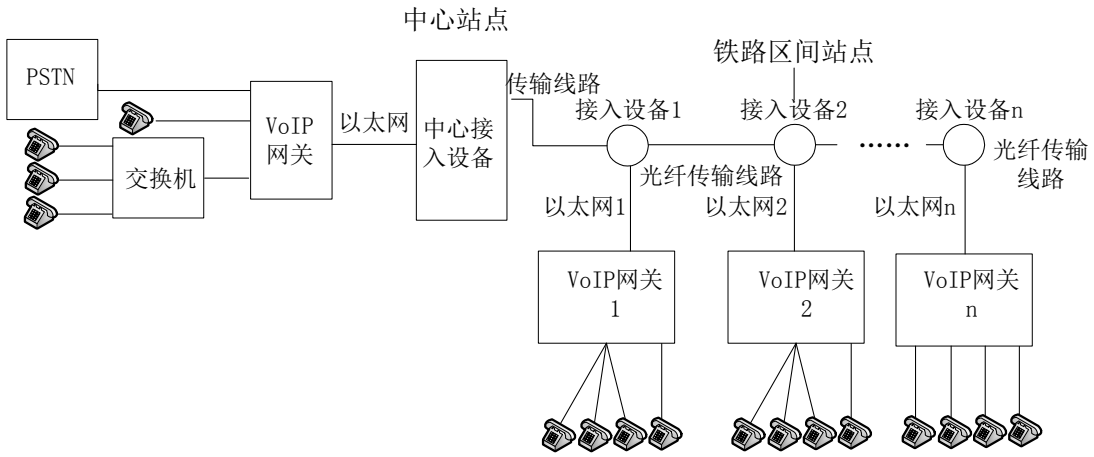

图 2-1 通信系统总体框图

2.VoIP 网关设备连接 PSTN 网络与 IP 网络，使两种网络能够互相通信。

VoIP 网关设备一端通过外线接口与 PSTN 交换机或电话连接，另一端通过以太网口连接网络交换机。使得 PSTN 和以太网这两种不同协议的网络有效的结合在一起。

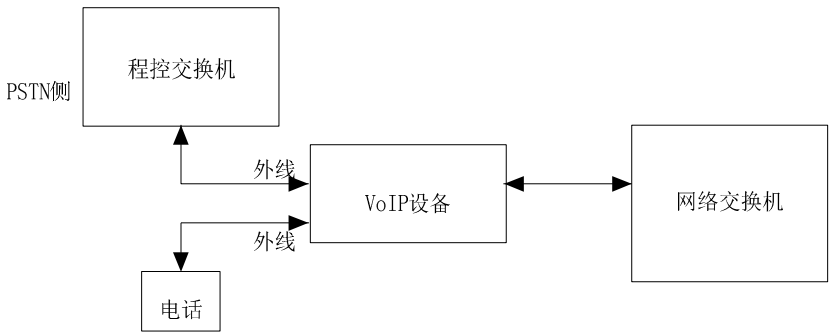

图 2-2 VoIP 设备连接图

3.VoIP 网关具有信令转换的功能，能将电话信令通过 IP 网络传输。

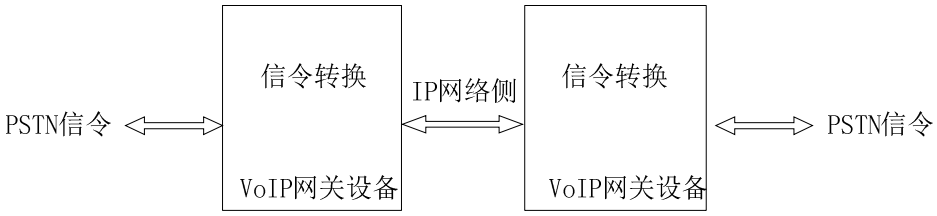

图 2-3 VoIP 设备信令转换示意图

4.有效的提供多种电话接口功能。

能够提供 FXO、FXS 电话接口，分别接入 PSTN 交换机或者直连模拟电话机。

5.运行稳定可靠。

2.3 开发环境分析

网关设备中的微处理器选用低功耗 32 位单片机 IXP425BD，开发软件为 Wind River 公司的系列软件，主要有 Tornado2.2、下载软件 FTP Server、超级终端、串口调试助手 V1.0.0.1、设置软件、TCP&UDP 测试工具 1.02。

Tornado2.2 软件主要用于程序开发，该软件是集编辑、编译、调试下载、仿真于一体的开发软件。该软件支持在线仿真与调试，方便程序的开发和后续调测。软件界面如图 2-4 所示。

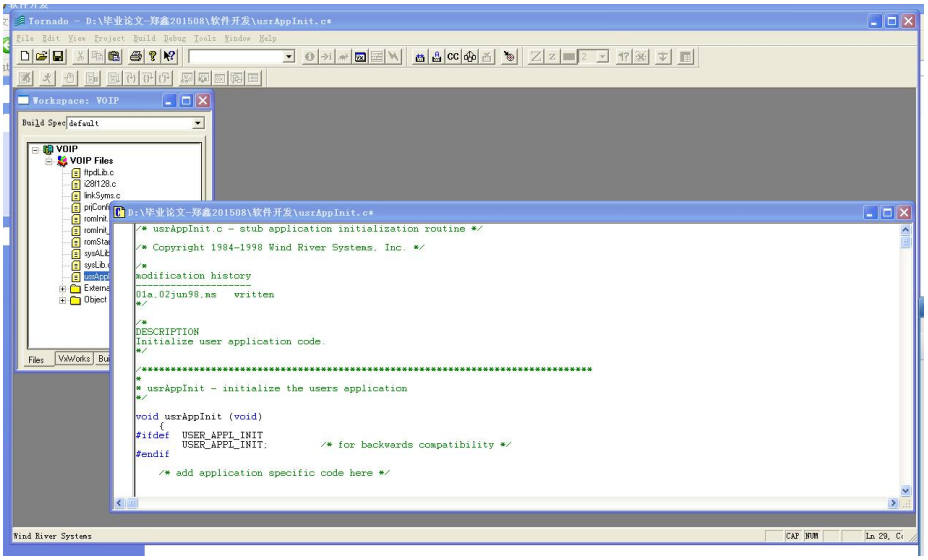

图 2-4 开发软件图形界面

FTP Server 下载软件主要是用于对 Tornado2.2 软件编译后的可执行文件下载

到 IXP425 的在片程序存储器。同时支持在系统调试，可以对设计的软件进行细节调试与修改。

串口调试助手 V1.0.0.1 和 TCP&UDP 测试工具是用于对通信模块软件进行调试、测试，以确保数据通信的可靠安全运行。

软件设计开发语言：C 语言。各应用程序用 C 语言编制。

2.4 本章小结

本章主要对 VoIP 网关设计进行了需求分析。基于用户的需求，分析了网关设备的关键技术和需求。然后对网关设备功能进行了设计，阐述了设备的性能和工作环境。最后分析了开发网关设备软件系统所需的工具软件。

第三章 VoIP 网关设备概要设计

本章对面向 SIP 信令的 VoIP 网关设备进行概要设计，内容包括硬件资源的配置、软件的方案设计、软件的总体思路、软件主流程以及软件开发的环境搭建。

3.1 总体设计

论文的设计目标为实现一个 VoIP 网关设备，其内部软件建立在 VxWorks 嵌入式操作系统平台上。

该网关设备可进行模拟声音的数字化、语音数据的以太网封包、PSTN 侧信令接收、信令相互转换、SIP 信令的传输等功能，也可以是作为一种独立的以太网结点存在的 IP 设备。

功能框图如图 3-1 所示。

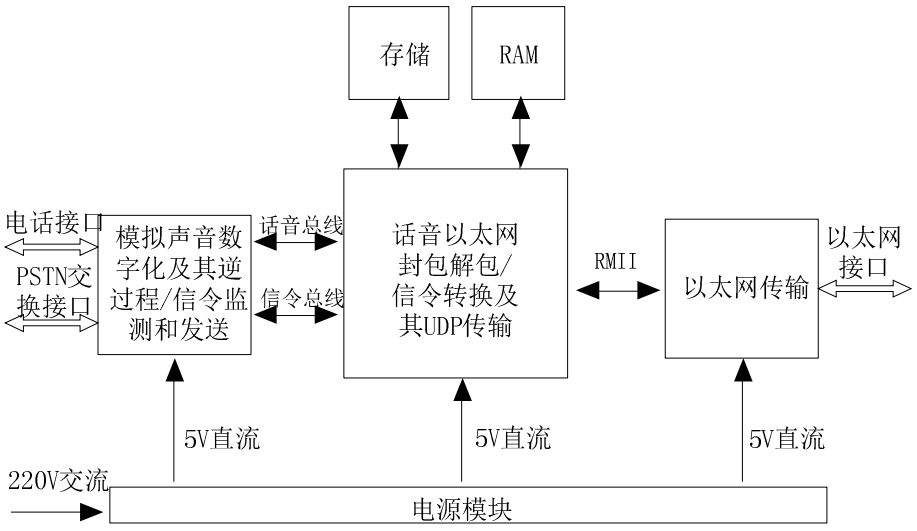

图 3-1 设备功能框图

其包括以下功能：

1.模拟语音数据的 A/D 与 D/A 转换：能够提供电话接口电路，模拟语音信号的压缩编解码（G.729）、处理电话机和 PSTN 侧信令等功能。支持与 PSTN 的接口。

2.语音数据的以太网封包解包：具有对数字语音信号进行 UDP 封包解包等功能。支持 TCP/IP，UDP 等协议。

3.PSTN 侧信令接收功能：具有接收和处理 PSTN 侧信令、对 PSTN 侧发出信令等功能。

4.信令转换功能：实现模拟侧电话机或 PSTN 侧信令与 SIP 信令之间的相互转换。

5.SIP 信令的传输功能：具有将 SIP 信令通过网络传输的功能。

6.系统的结构扩展性强，且脉络非常清晰，能够方便的对其完成新功能开发。

7.CPU 运算能力强，接口丰富，系统模块化设计，扩展性好；

网关设备的主要性能指标如下：

1.输入参数：电压 220V，频率 50Hz。

2.接线方式：提供 FXS 和 FXO 电话接口、网络接口。

3.通信接口：二路 RJ11 接口，一路 10/100Mbps 以太网接口；

4.通信协议：TCP/IP、UDP、SIP。

5.工作环境：额定工作温度范围为-10℃～+50℃；大气压为 63kPa～106kPa。

系统工作流程如下：模拟侧话音数据通过语音芯片转换为数据流，处理芯片收到数据流后将其封装为 UDP 包在 IP 网络上进行传输;信令数据同样经过话音芯片的获取后送入处理程序，再通过协议转换为 SIP 信令在 IP 网络上发往目的地。

以系统流程为基础，结合功能需求，本论文将系统分为四个部分，如图 3-2 所示：

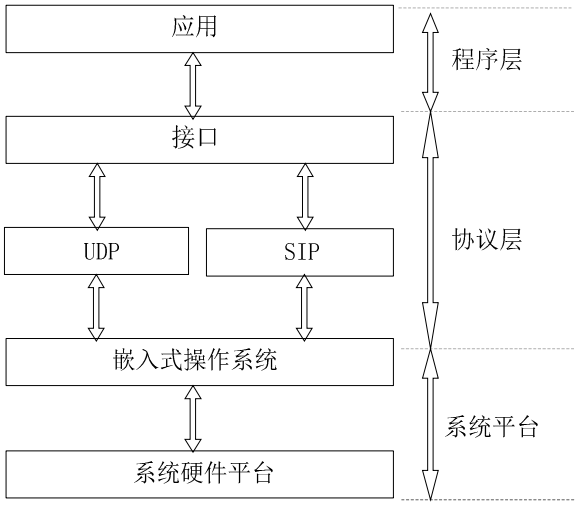

图 3-2 系统架构

第一层是系统平台，提供硬件支持；

第二层是操作系统，其具有两方面功能，一是管理硬件；二是提供系统的接

口；

第三层包含了 SIP 和 UDP 协议；

第四层是用户程序层，包括了话音处理模块等内容，其主要实现的是音频的编解码。

3.2 网关设备硬件设计

网关设备的硬件部分主要包括：微处理器电路（芯片 IXP425BD）、音频电路（芯片 IDT821054 和 AC483）、以太网电路（芯片 DM9161EP）。硬件平台的结构图如图 3-3 所示。

1.处理器电路

微处理器关系到系统运行的功能、性能、速度、稳定性、可靠性等非常重要的指标，由于本设计中的网关设备是语音处理系统，对实时性要求比较高，因此我们就要选择运行速度足够的处理器芯片。

本设计中的微处理器采用 INTEL 公司的 IXP425BD 芯片,其具有多种标准接口，能为各种设计提供完善的方案。

IXP425BD 微处理器包含有两个部分：高速片上 SRAM 工作区和低等待时间外部总线接口(HBI)。具有 32 位 SDRAM 内存控制器,芯片功耗低，533M 频率下典型功耗为 1.0W-1.5W。

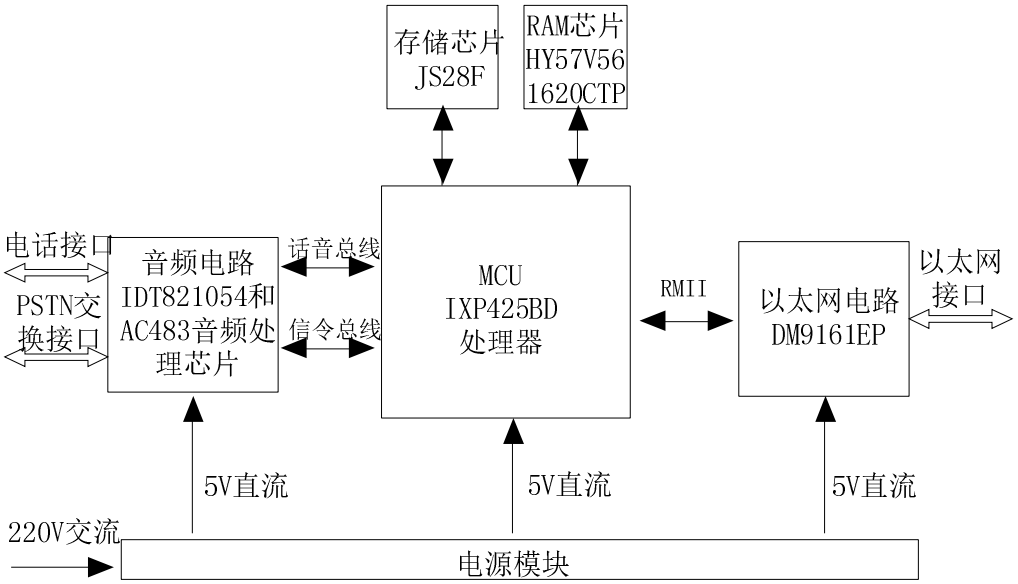

图 3-3 硬件平台架构框图

IXP425BD 微处理器具有集成的多种标准接口，包括 USB 2.0 及 10/100 Base-T 以太网媒体访问控制器(MAC)。其还提供一系列外设，可以使用在电信、音频及智能卡中。

存储器是硬件系统的最重要的组成部分，FLASH 存储器选用 HY57V561620CTP，用来储存操作系统程序和需加载的程序。

2.音频电路

音频电路由 IDT821054 和 AC483 芯片组成。

首先选用 821054 芯片，它具有 4 路输入输出通道，采用 A 律 13 折线将模拟话音信号进行抽样、量化、编码成 PCM 数字信号，抽样速率为 8k。其将模拟语音信号压缩编码后，通过 2M 话音总线送入 AC483 芯片。同时其提供信令寄存器，供 CPU 读取。

然后选用 AudioCodes 公司的 AC483 芯片，其与 IDT821054 的 2M 话音总线构成双向通道，将总线上的语音通过 G.729/G.711 压缩编码后，通过 A/D 总线送入 CPU，由其进行 UDP 封包。

另外还包括电话接口硬件部分电路，包含馈电、过压保护、振铃、等功能。

3.网络接口系统

模块包含 DM9161EP 芯片，其通过 RMII 接口与 IXP25BD 处理器连接，再由变压器 11FB-05NL 引出 10/100Mb/s 自适应的标准以太网接口，执行 IEEE802.3 标准局域网协议，提供 1 路 10Mb/s 和 100Mb/s 以太网接口。

3.3 网关设备软件概要设计

我们设计 VoIP 网关软件的目的，是将 PSTN 与 IP 网络之间不同的话音数据或信令方式连接在一起，提供两种网络之间的互联。软件设计框图见图 3-4。

网关软件设计要达到的主要功能：

1.PSTN 信令或电话信令经过信令处理模块后转换为 SIP 信令，并发送给目的主机；或者从网络上获取 SIP 信令，发送给模拟电话或 PSTN 侧；

2.网关能很好的完成模拟话机或 PSTN 侧与网络侧话音数据的转换和传输；

3.网关保持呼叫的正确性和连续性，保证通话的质量。

网关软件的工作流程：

1.通过 TCP/IP 接收网络信令后，利用 SIP 协议栈对其进行处理；如果协商成功，则解析收到的 UDP 协议语音流，生成模拟话音数据，发送给模拟话机或 PSTN 侧交换机；最后在信令控制下中断通话。

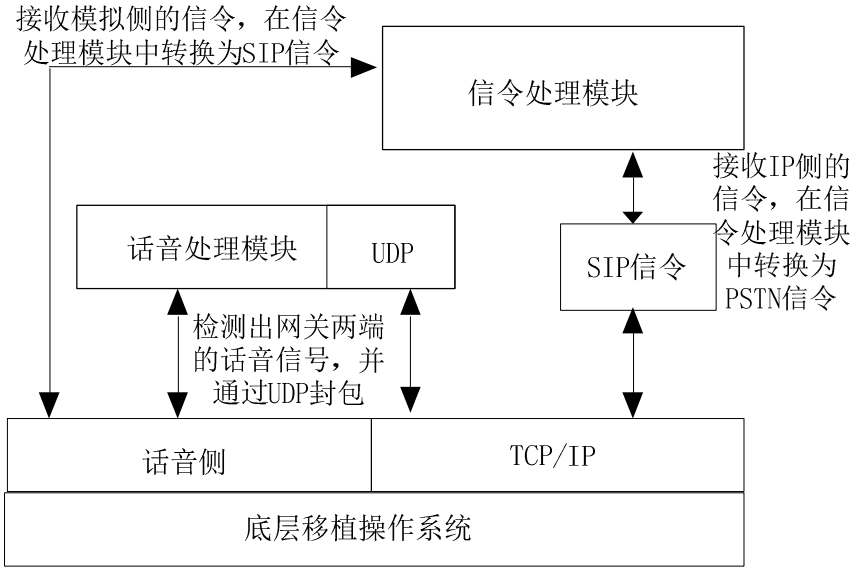

图 3-4 软件设计框图

2.接收到模拟话机或 PSTN 侧信令后，利用 SIP 协议栈产生网络 SIP 信令并传送至目的地址；如协商成功，则建立 UDP 语音流，组建网络侧和 PSTN 侧的语音通道；最后在信令控制下断开中断通话。

本设计采用 VxWorks 嵌入式实时操作系统进行多任务管理，便于系统维护和升级。软件实现框图如图 3-5 所示。

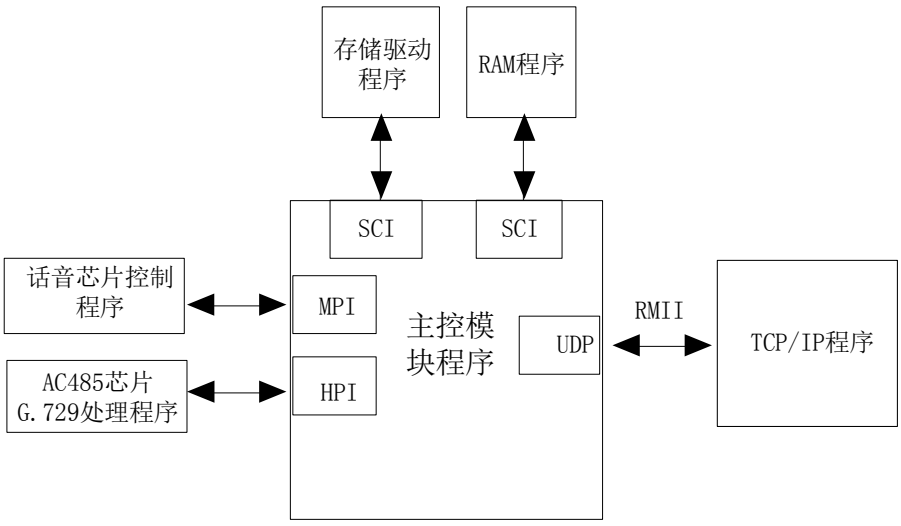

图 3-5 网关设备软件实现框图

本设备软件主要由运行于 32 位微处理器 IXP425 平台内的各模块软件构成，采用 C 语言编制，由 VxWorks 实时操作系统协同调度。

语音处理芯片软件任务：由 32 位微处理器 IXP425 控制 IDT821054 芯片对话音进行 AD/DA 转换，并控制接收和发送 PSTN 侧信令。两者通过 2M 总线进行数据交互。处理器还控制 AC483 芯片对话音进行 G.729 编码，通过 HPI 总线收发语音数据，以进行以太网传输。

信令转换软件任务：对 PSTN 侧信令和 SIP 信令进行判断、转换、发送。

以太网软件任务：对话音数据和 SIP 信令进行封包和解包，并通过 RMII 总线送入 DM9161EP 芯片进行物理层传输。

3.3.1 软件总体设计思路

按照前面描述的硬件资源配置和软件需求分析。为了便于系统升级维护，提高 CPU 利用率，且由于语音的实时性，决定采用 Vxworks 实时操作系统完成软件协调管理，以满足设备要求。

3.3.1.1 嵌入式操作系统 VxWorks

近年来，由于设备的小型化的便捷化，嵌入式系统越来越被广泛使用。且随着嵌入式系统技术的发展，其功能和性能越来越强大，已经成为了系统中非常重要的部分。

VxWorks 操作系统是美国风河公司推出的嵌入式操作系统。

VxWorks 操作系统具有如下的特点：

1.实时性

实时能是在有限的时间内响应一个外部的异步事件的能力。实时操作系统可以在规定时间内完成应该完成的功能，并可以在有限时间内对外部异步事件做出响应。

VxWorks 具有非常好的实时性，系统开销很小，系统功用程序，包括中断处理、进程间通信、进程调度等。

2.可靠性

VxWorks 以其良好的可靠性在软件界享有非常好的口碑，并且在中国也赢得了越来越多的用户。

3.兼容性

VxWorks 具有丰富的函数库，同时自带 TCP/IP 协议栈。

4.易用性

VxWorks 有 BSP（可以认为是一种低层驱动），可以减小驱动程序的编写过程。其具有强大的调试能力，可以在没有仿真器的情况下，通过串口调试。其也具有软件 DEBUG 功能，可以对软件部分进行模拟调试。

5.可裁减性

VxWorks 可精简到 8kB 的内核，空间占用率非常低。因此其具有非常好的灵活性。程序员按需选择其中部分来组成系统，这种操作方式保证了系统的安全和可靠。

VxWorks 操作系统基本构成模块如下：

1.VxSim-仿真器：能模拟目标机的运行；

2.I/O 系统：VxWorks 提供与 ANSl-C 兼容的 I/O 系统；

3.BSP-板级支持包：其提供了如下功能,硬件的初始化、内存映象和中断建立等；

4.网络特性：VxWorks 网络可以与其他协议进行通信，包括：SNMP、NFS、FTP、TCP/IP 等；

5.VxMP-共享的内存模块：其作用在多处理器上运行和执行的任务之间，管理了消息队列和共享信号量；

6.POSIX-兼容实时的标准：其提供支持实时标准 P.1003.1b；

7.VxVMI-虚拟的内存模块：它的作用是保护了指定的内存区；

8.文件系统：与 SCSI、RAM、MsDOS 等兼容。

3.3.1.2 集成开发环境 Tornado

嵌入式实时操作系统 VxWorks 的开发调试环境就是 Tornado。

Tornado 是交叉开发环境运行在主机上的部分，同时也是实现嵌入式程序的完整软件开发平台。其具有以下特点：

1.具有两种调试模式：系统级、任务级。

前者在调试的过程中应用系统须暂停；后者在调试过程中应用系统可保留在工作状态。

2.详细的帮助文档。

具有详尽的帮助文档以帮助开发人员进行设计工作。

3.开放的、可扩展的开发环境。

开放的开发环境有利于程序员利用各种工具帮助程序开发。

Tornado 核心工具介绍：

1.VxSim

VxSim 是一个原型仿真器，适用于在没有硬件支持下进行应用层程序的开发，它不适合开发设备驱动，但是支持任务调度，任务交互等内核支持的功能。

2. Browser

目标机系统浏览器，可以方便的监视目标机的状态，并可以动态捕捉以下信息:所有任务的堆栈使用情况、看门狗定时器、内存分区、目标机的 cpu 使用情况、消息队列、目标模块结构和符号、信号量、中断向量、详细的任务信息等。

3.WindView

WindView 能看到目标机的任务切换、信号量、消息队列、中断、看门狗等信息。

4.Debugger

最普通的调试行为，如设置断点、控制程序执行。

3.3.1.3 BSP 软件包的设计

Vxwbrks 的一个重要组成部分是板级支持包(BSP，Board Support Package)。

其在开发中具有的重要性不言而喻，BSP 没有正确的运行，那么系统就不会正常运行。

BSP 指的是部分设备驱动程序和用户编写的启动代码的集合。它实现了驱动部分设备、初始化等功能。BSP 在 VxWorks 系统中被描述为一个软件接口。开发过程如下：

1.获得一个参考板级的支持包和相关的代码模板；

2.准备开发环境；

3.编写预内核初始化的代码；

4.编写轮询方式来访问串口的系统驱动程序；

5.一旦启动内核，则必须使连接系统立即中断；

6.然后启动系统的时钟；

7.初始化驱动程序，进而进一步的完善支持包开发过程；

8.测试板级支持包。

介绍一下初始化模块的两个文件:

1.mmImt.s

由汇编语言编写，是个引导入口。

2.sysALib.B

其包含了系统相关的汇编级别的子程序，比如常字、说字、端口字节读写，系统描述符的访问，CPU 类型检测等。它也是由汇编语言编写的。

3.3.2 系统构成

系统的软件主要内容由三部分构成：驱动程序、应用程序（任务级代码）、操作系统移植及资源配置。

1.驱动程序

驱动程序主要涉及 32 位微处理器 IXP425 与各芯片之间的接口。

处理器 IXP425 通过 SPI 总线驱动话音芯片 IDT821054，完成各寄存器的设置和读写。

处理器 IXP425 通过 HPI 总线驱动 AC483 芯片，完成话音数据的 G.729 封包。

处理器 IXP425 通过 RMII 总线，驱动 DM9161EP 芯片，完成以太网包在物理层的收发。

上述驱动程序是基于处理器 IXP425 的外设驱动，各驱动程序的通用设计规则流程如图 3-6 所示。

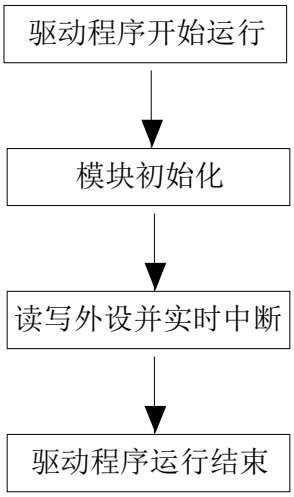

图 3-6 驱动程序算法流程图

2.基于 VxWorks 操作系统的任务软件

话音控制任务：完成话音处理芯片控制和话音数据的处理。

信令转换任务：完成信令的收发和转换处理。

以太网数据处理任务：完成数据的封包解包。

3.VxWorks 操作系统软件配置

本课题中软件运行是基于 VxWorks 操作系统，操作系统层面主要涉及移植、任务代码设计、资源分配设置和优先级设置。

3.3.3 软件主流程图

系统上电运行，处理器 IXP425 通过软件引导 VxWorks 运行；进行一系列的初始化操作；然后创建启动任务 TaskStart()，启动任务一方面对微控制器外设进行初始化以保证各外设模块可靠运行，另一方面创建各模块软件任务，最后由操作系统根据任务需求情况实时调度高优先级任务运行。流程图所图 3-7 所示。

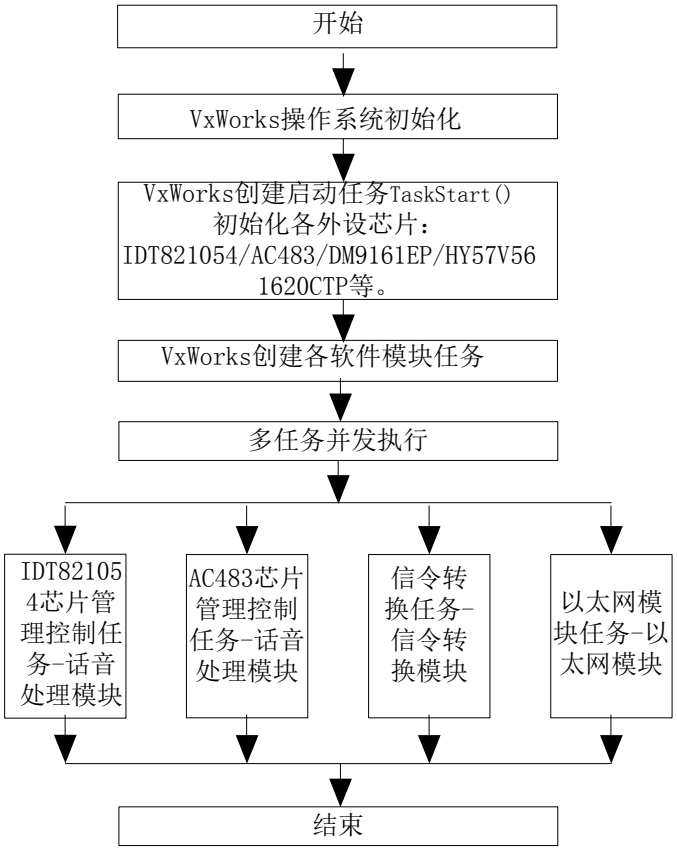

图 3-7 软件主流程图

3.3.4 VxWorks 操作系统环境构建

我们在 Tornado2.2 软件平台下搭建 VxWorks 开发环境。在进行测试并通过后，才能进行应用任务的设计和试验。

第一步，建立项目工程。建立相应的工程文件夹，在工程文件夹中添加开发 32 位微处理器 IXP425 必须的库文件，编写一个 main()函数并保正编译与调试。

第二步，将 VxWorks 资源文件复制到工程文件夹中，VxWorks 文件分别存放于 3 个文件夹中：BSP、VxWorks_CPU、VxWorks_LIB。

第三步，上述工程文件建立好之后，编译检查有无错误，并修改，至到编译

通过，表明软件开发环境搭建成功，可进行后续的软件开发。

3.4 本章小结

本章根据网关的功能需求设计了网关的硬件平台和软件平台，重点分析了微处理器 IXP425 的功能和特点、VxWorks 开发平台的构建。

第四章 话音处理模块软件详细设计与实现

话音信号的编解码是网关设备中的重要一环。在这一部分软件中，我们要实现话音从模拟信号到数字信号的转换及其逆过程。本章将对话音编解码软件进行详细设计。

4.1 话音处理分析

话音处理的流程如图 4-1 所示：

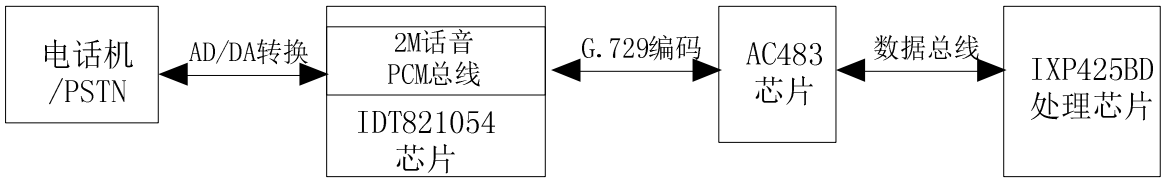

图 4-1 话音处理模块流程图

1.利用 IDT821054 芯片，实现模拟信号和数字信号的 AD/DA 转换；

2.同样，利用 AC483 芯片按照 G.729 压缩编码方式对话音信号进行编解码。

4.2 话音 AD/DA 转换分析及软件实现

4.2.1 话音 AD/DA 转换及编码方式分析

模拟信号的数字化包含三个过程：抽样、量化、编码。

抽样：将模拟信号变为在时间轴上离散的信号。经抽样后的样值信号就成为一种脉冲幅度调制(PAM)的信号。

量化：把幅度连续变化的模拟量变成用有限位二进制数字表示的数字量的过程。

编码：脉冲幅度调制信号并不是数字信号，需再将其量化和编码后，才变为成二进码数字信号。

在 IDT821054 芯片中采用的是 PCM-30/32 路通信系统，其为采用码组复接的时分复用系统，帧结构如图 4-2：

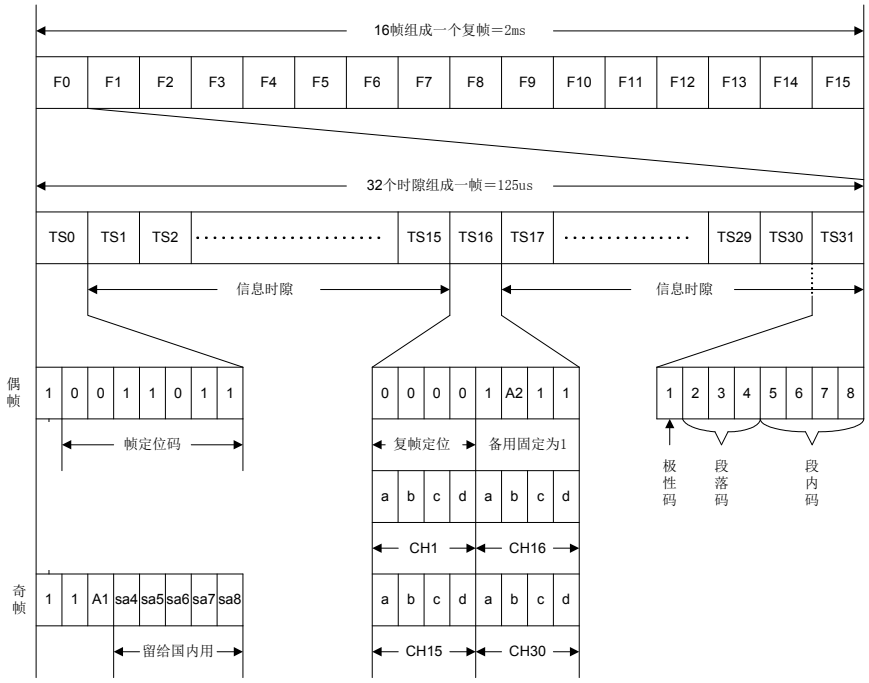

图 4-2 PCM 基群帧结构

图中，帧周期 T=1/8000 秒=125us，将其平均(即均匀)分为 32 个时隙，每个时隙的时间间隔为：125/32=3.91us。每一时隙传送 8bit 编码，则每个码的时间间隔为：3.91/8=488ns，每帧共传送 32×8=256bit 码字。

在 30/32 路系统中，帧结构的第一个时隙(TS0)用于传送帧同步信号，TS16 用于传送话路信令。故只有 30 个时隙用来传送话音信号，所以只能传送 30 个话路。

当采用共路信令传送方式时，必须将 16 个帧构成一个更大的帧，称为复帧。复帧的重复频率为 500Hz，周期为 2ms。

在本设计中，话音的 AD/DA 转换由 IDT821054 芯片完成。其将话音口的模拟信号转换为 PCM-30/32 复用方式的复帧结构，速率为 2M，并进行其逆过程。

4.2.2 MPI 总线驱动程序以及 IDT821054 芯片配置控制程序设计

本节主要设计 IXP425 处理芯片与 IDT821054 芯片之间的 MPI 总线通信实现，并通过 MPI 总线接口对 IDT821054 芯片进行配置和控制。

4.2.2.1 软件组成

程序由如下 4 个文件构成，完成了芯片之间 MPI 总线协议的实现和 IDT821054 芯片的配置和控制：

IDT821054_MPI.h：处理器 IXP425 的 MPI 总线头文件。

IDT821054_MPI.C：处理器 IXP425 的 MPI 总线控制程序。

IDT821054.h：IDT821054 配置和控制头文件。

IDT821054.C：IDT821054 配置和控制源程序。

4.2.2.2 MPI 总线驱动程序设计

MPI 总线驱动程序主要是根据 IDT821054 的 MPI 通信协议实现。

IDT821054 的 MPI 通信数据帧格式包括三个字节：第一个字节是命令字节，第二和第三个字节是数据字节。

其字节的辨识由片选信号 CS 指示，时钟最高为 8.192MHz。读写时序如图 4-3 和 4-4 所示。

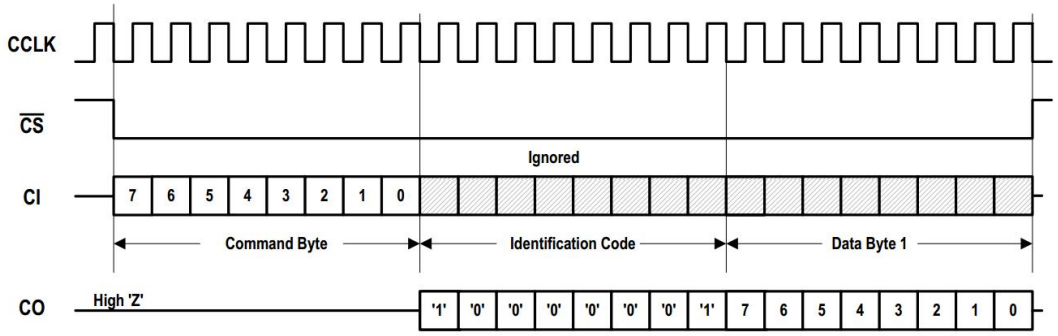

图 4-3 IDT821054 的 MPI 读操作时序

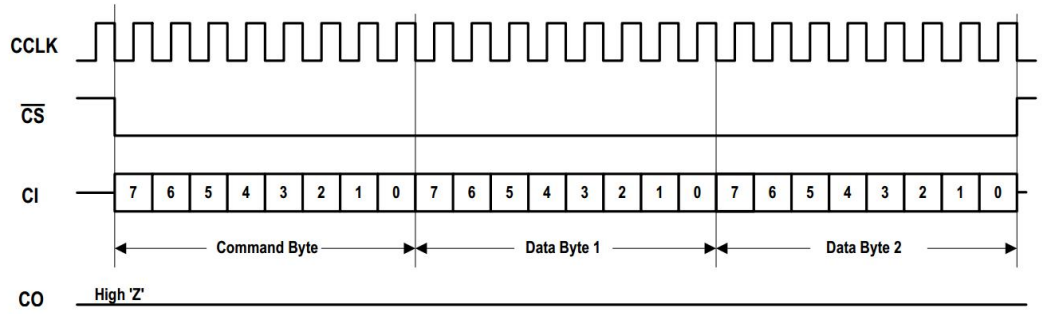

图 4-4 IDT821054 的 MPI 写操作时序

MPI 总线程序流程图如图 4-5 所示。

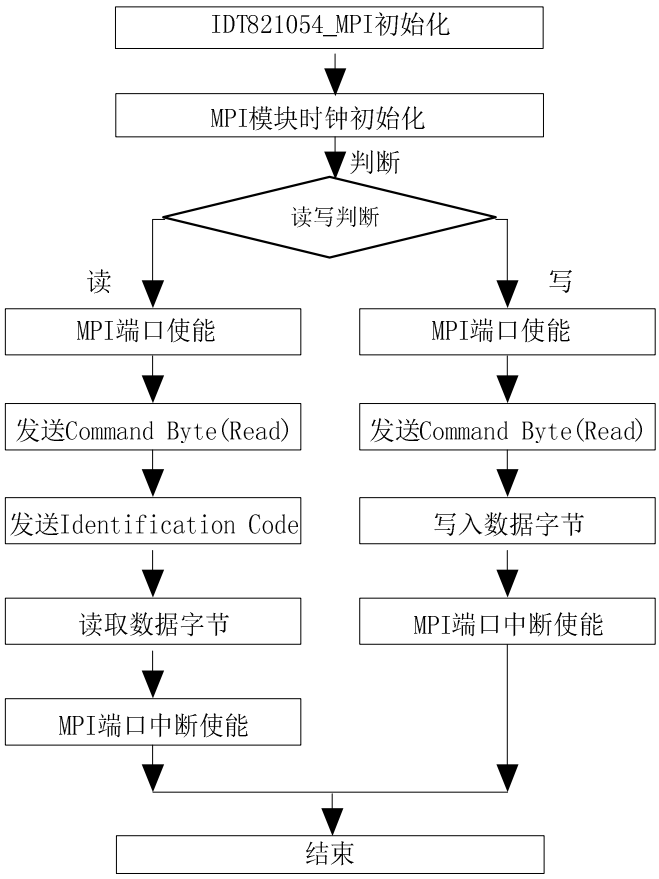

图 4-5 MPI 程序流程图

程序流程如下所述：

1.首先对 IDT821054 芯片的 MPI 接口进行初始化，定义其读写函数、控制管脚，并使能时钟接口；

2.通过标志位判断对 MPI 接口是读操作还是写操作；

3.如为读，则按照通信协议使能 MPI 端口，发送命令字节，读取数据字节，关闭使能；

4.如为写，则按照通信协议使能 MPI 端口，发送命令字节，发送数据字节，关闭使能。

管脚定义上，我们为以下几个通信接口分配管脚：

CS：IDT821054 话音芯片端口使能（处理器至 IDT821054 话音芯片），使用 IXP425 管脚 RTS0_N。

CI：串行控制接口数据输入（处理器至 IDT821054 话音芯片），使用 IXP425 管脚 TXDATA0。

CO：串行控制接口数据输出（IDT821054 话音芯片至处理器），使用 IXP425

管脚 RXDATA0。

CCLK：串行控制接口时钟输入（处理器至 IDT821054 话音芯片），使用 IXP425 管脚 HSS_TXCLK1。

我们设计一系列函数完成上述任务，函数名及其定义见表 4-1。

表 4-1 MPI 总线驱动函数及其意义

名称	意义、函数名
Read_BitOp()	读一个数据位，被 Read_DataOp()调用
Write_BitOp()	写一个数据位，被 Write_DataOp()调用
Read_DataOp()	读一个字节的数据位，被 Read_IDT821054()调用
Write_DataOp()	写一个字节的数据位，被 Write_IDT821054()调用
IDT821054_Chip_Select()	端口使能，被读操作和写操作函数调用
Delay_3Pulse()	端口中断使能，被读操作和写操作函数调用
Read_IDT821054()	对 IDT821054 进行一次读操作
Write_IDT821054()	对 IDT821054 进行一次写操作

在分析了 MPI 接口通信协议、定义了管脚、规划了软件流程图和各函数之后，对 IDT821054_MPI.C 文件的具体程序进行设计。

IDT821054_MPI.C 文件由 IXP425 处理芯片的 MPI 总线程序构成，包括 MPI 的初始化、端口使能、MPI 读字节和 MPI 写字节等程序。

1. MPI 初始化函数：

- 命令位：8 位，配置成 unsigned char Command_Code
- 数据位：8 位，配置成 unsigned char bdata Bdata_1054_Tmp
- CO：配置 IDT821054 的 MPI 输出口

```
#define Read_BitOp(ReturnBit) {CCLK=1;CCLK=0;_nop_();ReturnBit=CO;}
```

- CI：配置 IDT821054 的 MPI 输入口

```
#define Write_BitOp(Write_Bit) {CI=Write_Bit;CCLK=0;CCLK=1;}
```

- Write_DataOp( )：发送 8bit 一个字节数据的函数，调用可写一个数据位的数据

```
void Write_DataOp(unsigned char Write_Data)
{
Bdata_1054_Tmp=Write_Data;                    //将一个字节数据写入缓存
Write_BitOp(Bdata_1054_Tmp_7);                //将缓存的第 8 位写入总线
Write_BitOp(Bdata_1054_Tmp_6);                //将缓存的第 7 位写入总线
```

```
Write_BitOp(Bdata_1054_Tmp_5);
Write_BitOp(Bdata_1054_Tmp_4);
Write_BitOp(Bdata_1054_Tmp_3);
Write_BitOp(Bdata_1054_Tmp_2);
Write_BitOp(Bdata_1054_Tmp_1);
Write_BitOp(Bdata_1054_Tmp_0);                    //将缓存的第 1 位写入芯片
}
```

- Read_DataOp( )：接收 8bit 一个字节数据的函数，调用可读一个数据位的数据

```
unsigned char Read_DataOp(void)
{
Read_BitOp(Bdata_1054_Tmp_7);                     //将总线上的数据写入缓存的第 8 位
Read_BitOp(Bdata_1054_Tmp_6);                     //将总线上的数据写入缓存的第 7 位
Read_BitOp(Bdata_1054_Tmp_5);
Read_BitOp(Bdata_1054_Tmp_4);
Read_BitOp(Bdata_1054_Tmp_3);
Read_BitOp(Bdata_1054_Tmp_2);
Read_BitOp(Bdata_1054_Tmp_1);
Read_BitOp(Bdata_1054_Tmp_0);                     //将总线上的数据写入缓存的第 1 位
return(Bdata_1054_Tmp);                           //返回缓存中存储的 8 位数据
}
```

2.端口使能函数：IDT821054_Chip_Select

MPI 开始读写操作之前，判断发送标志位是否就绪，如果就绪则开始使能端口。

```
void IDT821054_Chip_Select(unsigned char flag)
{
if(flag){
Chip_Select=0;}                          //将 CS 管脚拉低，使能 IDT821054 芯片
else{
Chip_Select=1;}
```

3.端口中断使能函数：Delay_3Pulse()

在此函数调用 3 个 CCLK 周期后关闭 MPI 端口使能。

```
#define Delay_3Pulse()
{ Disable_All_IDT821054();CCLK=1;CCLK=0;CCLK=1;CCLK=0;CCLK=1;CCLK=
```

0; }

4.MPI 读操作函数：Read_IDT821054()

MPI 读字节的程序方法：端口使能之后，先发送 8bit 的命令代码，然后从 1054 的 CO 口接收数据并返回接收值，最后调用端口中断使能函数结束操作。

```
unsigned char Read_IDT821054(unsigned char Command_Code)
{
unsigned char DataValue;
Command_Code&=0x7F;                    //定义命令字节，0x7F 命令表示为读操作
Write_DataOp(Command_Code);            //将命令字节写入 IDT821054 芯片
Read_DataOp();                         //从 IDT821054 芯片中读数据
DataValue=Read_DataOp();               //将读出数据写入寄存器
Delay_3Pulse();                        //调用中断使能函数，结束读操作流程
return(DataValue);                     //返回读出寄存器数据
}
```

5.MPI 写操作函数：Write_IDT821054 ()

MPI 读字节的程序方法：端口使能之后，先发送 8bit 的命令代码，然后向 1054 的 CI 口写入数据，最后调用端口中断使能函数结束操作。

```
void Write_IDT821054(unsigned char Command_Code,unsigned char DataValue)
{
Command_Code=0x80;                     //定义命令字节，0x80 命令表示为写操作
Write_DataOp(Command_Code);            //将命令字节写入 IDT821054 芯片
Write_DataOp(DataValue);               //将数据字节写入 IDT821054 芯片
 Delay_3Pulse();                       //调用中断使能函数，结束写操作流程
}
```

通过上述程序，完成了 MPI 总线驱动，使得能够通过 MPI 接口对 IDT821054 芯片进行操作。

4.2.2.3 IDT821054 芯片配置控制程序设计

通过 MPI 总线驱动程序对 IDT821054 芯片进行具体配置，以达到提供话音接口以及模拟话音数据与 PCM-30/32 数字 2M 话音总线数据之间的转换。

需要配置的话音芯片参数包括：话音接口类型、话音接口增益、话音通道开关、话音通道在 PCM-30/32 帧结构中所占时隙等。上述信息都保存在 FLASH 中，在初始化时，写入 IDT821054 芯片中。

我们设计一系列函数完成上述任务，函数名及其定义见表 4-2。

表 4-2 IDT821054 芯片配置控制函数及其意义

名称	意义、函数名
Init_IDT821054()	IDT821054 芯片配置函数，调用下面各函数
Config_PCM_Card_Coefficient_RAM()	从 RAM 中读取配置信息
Config_PCM_Card_Coefficient_Select()	配置话音口类型和话音口增益
Config_PCM_Card_Loop()	话音通道环回控制
Config_PCM_SB1_8054()	1054 的 SB1 端口操作使能
Config_PCM_SB2_8054()	1054 的 SB2 端口操作使能
Config_PCM_SB3_8054()	1054 的 SB3 端口操作使能
Config_Channel_IO_Data_8054()	1054 芯片话音口使能
Config_PCM_Card_Receive_TimeSlot()	配置各话音通道接收时隙
Config_PCM_Card_Transmit_TimeSlot()	配置各话音通道发送时隙
Config_PCM_Card_8054_Power_On()	1054 的芯片开始处理话音数据

在分析了需配置的话音芯片参数、规划了各函数之后，对 IDT821054.C 文件的具体程序进行设计。

1.IDT821054 配置函数：Init_IDT821054( )

```
void Init_IDT821054(void)
{
Config_PCM_Card_Coefficient_RAM();
Config_PCM_Card_Coefficient_Select();
Config_PCM_Card_Loop();
Config_PCM_Card_Receive_TimeSlot();
Config_PCM_SB1_8054();
Config_PCM_SB2_8054();
Config_PCM_SB3_8054();
Config_PCM_Card_8054_Power_On();
Config_Channel_IO_Data_8054();
Config_PCM_Card_Transmit_TimeSlot();}
```

2.话音通道环回控制函数：Config_PCM_Card_Loop( )

```
void Config_PCM_Card_Loop()
```

```
{
IDT821054_Chip_Select();                    //MPI 口使能
Write_IDT821054(Channel_Program_Enable,0xF2);      //写入通道配置使能命令
IDT821054_Chip_Select();                    //MPI 口使能
Write_IDT821054(Loop_Status_Control_Other,0x00);   //写入控制命令，设置不环回
}
```

3. 1054 芯片开始处理话音数据：Config_PCM_Card_8054_Power_On()

```
void Config_PCM_Card_8054_Power_On()
{
IDT821054_Chip_Select();                    //MPI 口使能
Write_IDT821054(Channel_Program_Enable,0xF2);      //写入通道配置使能命令
IDT821054_Chip_Select();                    //MPI 口使能
Write_IDT821054(Teletax_Ramp_Start_8054,0x00);     //写入控制命令，1054 芯片开始处理话音数据
}
```

4. 配置各话音通道接收时隙：Config_PCM_Card_Receive_TimeSlot()

```
void Config_PCM_Card_Receive_TimeSlot(unsigned char Index_Cards)
{
IDT821054_Chip_Select();                    //MPI 口使能
Write_IDT821054(Channel_Program_Enable,j);         //写入通道配置使能命令
IDT821054_Chip_Select();                    //MPI 口使能
if(Index_Cards==1&&i>2)                     //按照顺序依次写话音接口在 2M 总线中占用的时隙
{Write_IDT821054(Receive_Timeslot_Select_8054,0x8F);}
else
{Write_IDT821054(Receive_Timeslot_Select_8054,Table_TimeSlot[Index_Cards*2+i]);} }}
```

以上列举了 4 个主要函数的代码，其余函数代码不再一一阐述。在程序运行完成之后，对 IDT821054 配置完毕。

此时，IDT821054 提供 2 路电话接口，其中 1 路设置为 FXS 接口，直接连接电话机；另外 1 路设置为 FXO 接口，可通过交换机连入 PSTN。

此 2 路语音数据通过数字话音总线接口与 AC483 芯片交换话音数据信息，其数据格式为脉冲编码调制的 PCM-30/32 编码 2M 速率编码。

4.3 话音编码方式转换分析及软件实现

现在我们已经完成了语音信号的 AD/DA 转换，得到了每一路带宽为 64kbps 的 PCM 编码语音信号。可以说，这种编码方式的语音，其质量是相当好的，完全满足电话的需求。

随着网络飞速发展，各种多媒体业务在网络上不断增长，使得带宽资源越来越宝贵，而 PCM 话音编码方式带宽过高，如果网络有延迟，则会很容易产生语音不流畅的情况。为了能够在较差的网络环境或有限的带宽中完成更多的话音数据信息的传输，就必须对话音信号进行适当的压缩。

话音的压缩算法、声音检测等通过芯片 AC483 实现，软件通过管理接口配置 AC483 压缩算法类型和 2M 时隙分配，并通过 AD 总线读取压缩后的话音编码。AC483 芯片中内含 G．729 和 G．711 算法。经过这两种算法压缩后的音频带宽为 8k 和 64k。从目前实际情况看来，G．729 算法压缩率高，声音质量好，能够充分的满足我们设计的要求。

4.3.1 G.729 编码方式分析

G.729 是国际电信联盟于 1996 年推出的一种编码方式，具有 8kbit/s 码率的语音编码算法建议，能够将话音数据压缩到一个比较低的码率，G.729 编码器原理框图如图 4-6 所示。

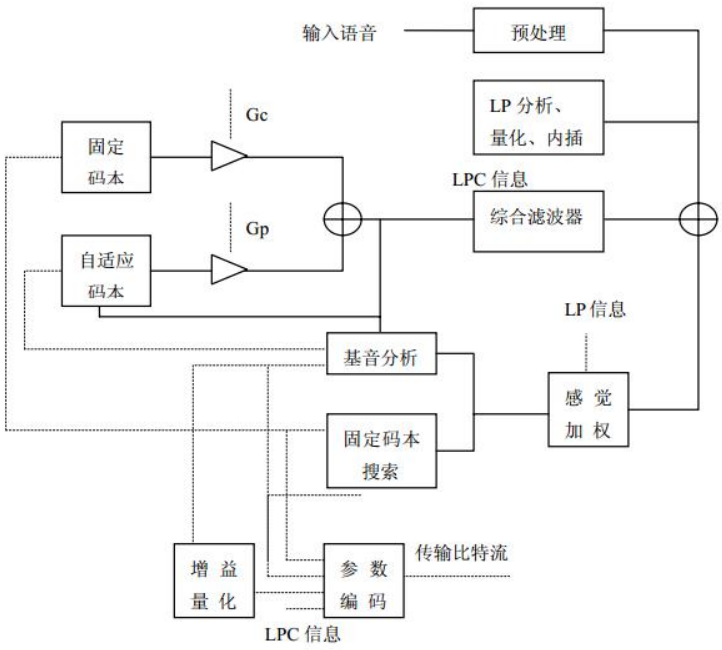

图 4-6 G.729 编码器原理框图

在本设计中，AC483 芯片提供了多种外部接口，包含有 PCM 接口、HPI 总线接口、存储器接口等，另外还包含了时钟接口等以接收外部晶体振荡器的时钟输入。

其结构如图 4-7 所示。

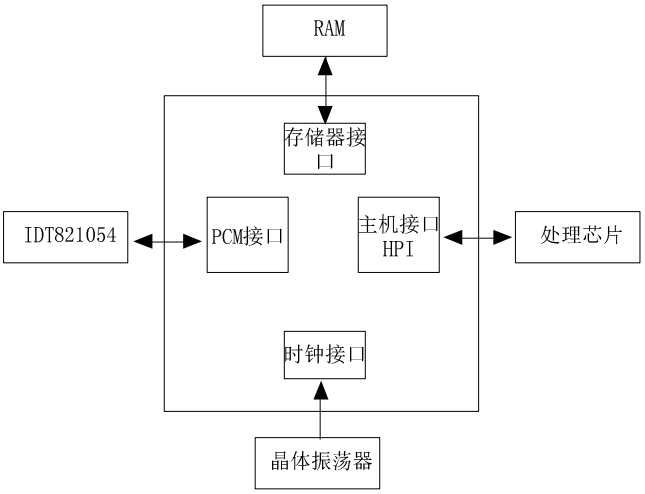

图 4-7 AC483 接口图

1. AC483 的 PCM 接口输入端外接 IDT821054 芯片的 2M 话音总线；

2.AC483 通过标准主机 HPI 接口与处理器连接，包括 4 位地址总线和 8 位数据总线。经压缩后的 8kbit/s 码率的语音编码由此与处理器芯片交互；

3.通过时钟接口接收晶体振荡器的时钟信号；

4.AC483 通过存储器接口与片外 RAM 连接。

4.3.2 AC483 芯片配置控制程序设计

为使 AC483 芯片正常工作，芯片厂商提供了必须的程序，包括有“48304 CK.312”、“48304 ce1.312.09” 和“48304 ce2.312.09”等软件，我们在处理器芯片主程序中调用这些接口函数。其提供了芯片初始化的内核程序和工作程序。软件流程框图如图 4-8 所示。

首先进行程序的加载；完成以后芯片已经开始工作并进入初始化模式；此时微处理器利用程序对 AC483 进行初始化配置，内容包括芯片话音通道的设置、码率的选择等内容；系统在初始化话音通道和配置话音编码等工作方式，并配置语音转换时隙后启动 AC483 内核。之后 CPU 即可通过读取数据总线的方式读取到 AC483 采集并压缩的音频编码数据，该系统创建 PollingProcess_tsk 任务每 10ms

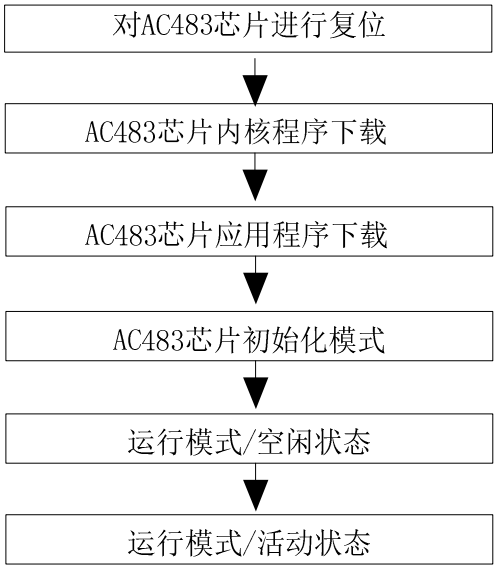

图 4-8 AC483 芯片软件流程框图

读取音频编码数据，并根据当前事务状态回调 UDP 等相关封包处理函数。函数定义见表 4-3。

表 4-3 话音编码方式转换函数及其意义

名称	意义、函数名
NumActiveDevices()	设置被初始化的 DSP 设备数量。
ModifyDefaultAC48xMemoryMap()	应用新的 AC483 物理地址，将寄存器设置为 1
NewAC48xMemoryMap ()	设置新的 AC483 芯片物理地址向量
unsigned long LocalIPAddress()	设置变量寄存器为 0
TUserPCMSetupInfo UserPCMInfo()	设置 TDM 总线
TUserChannelSetupInfo *UserChInfo()	指向一个数组的通道配置结构
UserCallProgressTonesNum()	设置使用的呼叫程序通道数量
acTCallProgressTone* UserCallProgressTones()	呼叫进程的增益定义。
TJitterBufStatusReportParams* JitterBufStatusReportParams()	设置抖动缓存状态报告
TExtendedVoiceParams()	设置话音参数

初始化程序函数调用如下。

int vpOpenStack(

```
int NumActiveDevices;
int ModifyDefaultAC48xMemoryMap;
unsigned long *NewAC48xMemoryMap;
unsigned long LocalIPAddress;
struct TUserPCMSetupInfo UserPCMInfo;
TUserChannelSetupInfo *UserChInfo;
int UserCallProgressTonesNum;
struct acTCallProgressTone *UserCallProgressTones;
TJitterBufStatusReportParams *JitterBufStatusReportParams;
struct TExtendedVoiceParams ExtVoiceParams
)
```

初始化之后，设备进入运行模式，开始话音的处理，用户随时可以对芯片的配置进行改变。

AC483 芯片会周期性的读取语音信息，然后将其压缩为数据包。包内携带了所有的信息和数据，这些包通过 HPI 接口送入处理器。

对 AC483 芯片进行控制程序如下。（这个是配置接口函数的具体参数，以便初始化时调用 vpOpenStack 这个函数以达到配置目的）

```
int AC48304(void)
{
int ret,evtret;
int i;
char type;
char temp_ipaddr[20];
struct in_addr IPaddr;
unsigned long rc1=0,rc2=0;
//设置 AGC 信息参数
struct TUserAGCSetupInfo UserAGCInfo={20,15,3};
//设置抖动缓存状态报告参数
struct TJitterBufStatusReportParams    JitterBufStatusReportParams ={1, 100 };
//设置 TDM 总线信息
struct TUserPCMSetupInfo UserPCMInfo;
//初始化提示音方式
UserCallProgressTonesNum=UserCallProgressSetup(CallProgressToneFile);
```

```
Init_Call_Progress_Tone();
//增加对 UserChInfo 的初始化
ConfigureUserPCMSetupInfo(&UserPCMInfo);
SetupUserChInfo();
//获取编程代码版本号
AC48xGetProgramVersion(0,&AC48304ProgramDate,&AC48304ProgramRev,AC483
04ProgramName);
//打印版本号和日期
printf("AC48304ProgramDate=%d 年%d 月%d
日,AC48304ProgramRev=%04d.%02d,AC48304ProgramName=%s!\r\n",
AC48304ProgramDate&0xFFFF,(AC48304ProgramDate>>16)&0xFF,(AC48304Progra
mDate>>24)&0xFF,
AC48304ProgramRev/100,AC48304ProgramRev%100,AC48304ProgramName);
printf("vpOpenStack()返回值=%d!\r\n",VPStackVersion);
//创建 10ms 定时查询任务
ret=taskSpawn("t 查询 AC483", 155 ,0, 81920,
(FUNCPTR)PollingProcess_tsk,0,0,0,0,0,0,0,0,0,0);
if(ret==ERROR){ printf("PollingProcess_tsk 创建错误!\r\n"); }
//创建事件处理任务 EventHandler
evtret=taskSpawn("t 事件处理 ", 201 ,0, 8192,
(FUNCPTR)EventHandler_tsk,0,0,0,0,0,0,0,0,0,0);
if(evtret==ERROR)
{ printf("EventHandler_tsk 创建错误!\r\n");
return evtret; }
。
。
。
}
```

程序运行后，控制 AC483 芯片完成模拟话音与话音数据包之间的转换。

4.4 本章小结

本章对话音处理提出了软件设计的分析及实现方法，话音处理主要是利用处理器芯片 IXP425 对 AD/DA 转换芯片 IDT821054 和话音数据转换芯片 AC483 的相

关寄存器进行数据读写操作和控制，使其完成语音处理流程，建立了模拟话音信号至话音数据包的转换流程及其逆过程。

第五章 信令转换及以太网处理模块软件详细设计与实现

信令处理是网关设备中比较复杂和重要的环节。在这一部分软件中，我们要实现模拟电话侧或 PSTN 侧的信令与网络侧 SIP 信令的相互转换。并使信令和话音数据在网络上进行传输。本章将对信令处理软件进行设计。

5.1 信令转换总体流程

信令转换的流程如图 5-1 所示：

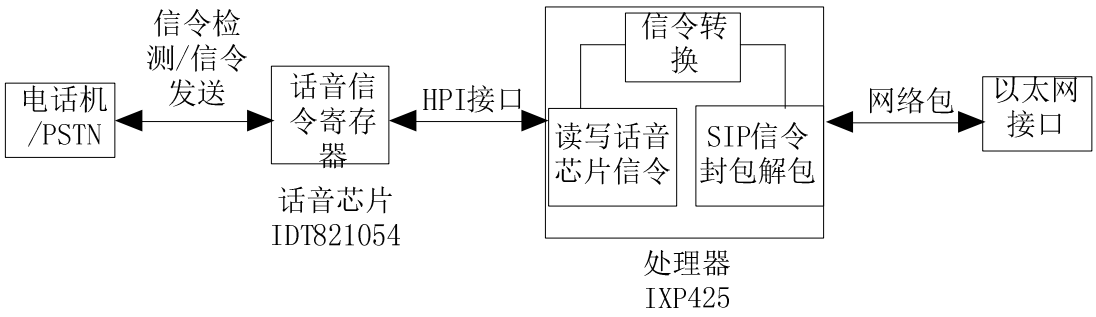

图 5-1 信令转换模块流程图

流程分析：

1.利用 IDT821054 芯片，通过对芯片提供的信令 IO 口 SLIC 的检测与控制，实现模拟电话侧或 PSTN 侧信令的收集与发送；软件上所有信令操作都以对 IDT821054 信令寄存器的读写来实现；

2.处理器 IXP425 通过与 IDT821054 相连的 HPI 控制总线实现对 IDT821054 信令寄存器的读写；并将信令数据保存在处理器的缓存中；

3.在处理器 IXP425 内部实现模拟侧话音信令和 SIP 信令的转换；

4.在以太网侧，将 SIP 信令和话音数据进行封包解包操作。

5.2 模拟接口/PSTN 侧信令处理分析及软件实现

如第四章所述，我们已经通过软件实现了处理器 IXP425 与话音芯片 IDT821054 的通信。现在我们要做的，就是对 IDT821054 芯片的信令寄存器进行读写操作控制。

5.2.1 IDT821054 信令处理方式分析

IDT821054 芯片共有 4 路模拟语音通道，它为每一路通道提供了 5 个信令控制管脚，包括了模拟电话机控制和 PSTN 侧信令控制，管脚描述如下。

SI：摘挂机检测；

SB1：话路通断控制；

SB2：负载通断控制（PSTN 侧专用）；

SO1：铃流通断控制；

SO2：恒流源通断控制。

我们通过对这五个管脚的控制，就能从模拟侧获取信令或发送信令至模拟侧。现将其寄存器表整理如表 5-1 所示。

表 5-1 IDT821054 信令管脚寄存器列表

GREG9	b7	b6	b5	b4	b3	b2	b1	b0
Command	0	0	1	0	1	0	0	0
I/O data	SIB[3]	SIB[2]	SIB[1]	SIB[0]	SIA[3]	SIA[2]	SIA[1]	SIA[0]
The SIA[3:0]比特用于获取 SI1 信号的状态，分别代表通道 4 至 1。 The SIB[3:0]比特用于获取 SI2 信号的状态，分别代表通道 4 至 1。								
GREG10	b7	b6	b5	b4	b3	b2	b1	b0
Command	R/W	0	1	0	1	0	0	1
I/O data	SB1C[3]	SB1C[2]	SB1C[1]	SB1[0]	SB1[3]	SB1[2]	SB1[1]	SB1[0]
The SB1C[3:0]比特用于设置 I/O 口 SB1 的方向，如为 0，则为输入；如为 1，则为输出。 如为输入，则 The SB1[3:0]比特用于获取 SB1 信号的状态，分别代表通道 4 至 1。 如为输出，则 The SB1[3:0]比特用于设置 SB1 信号的状态，分别代表通道 4 至 1。								
GREG11	b7	b6	b5	b4	b3	b2	b1	b0
Command	R/W	0	1	0	1	0	1	0
I/O data	SB2C[3]	SB2C[2]	SB2C[1]	SB2[0]	SB2[3]	SB2[2]	SB2[1]	SB2[0]
The SB2C[3:0]比特用于设置 I/O 口 SB2 的方向，如为 0，则为输入；如为 1，则为输出。 如为输入，则 The SB2[3:0]比特用于获取 SB2 信号的状态，分别代表通道 4 至 1。 如为输出，则 The SB2[3:0]比特用于设置 SB2 信号的状态，分别代表通道 4 至 1。								
LREG4	b7	b6	b5	b4	b3	b2	b1	b0
Command	R/W	0	0	0	0	0	1	1
I/O data	S02[3]	S02[2]	S02[1]	S02[0]	S01[3]	S01[2]	S01[1]	S01[0]
The SO2[3:0]比特用于设置 SO2 信号的状态，分别代表通道 4 至 1。 The SO1[3:0]比特用于获取 SO1 信号的状态，分别代表通道 4 至 1。								

5.2.2 IDT821054 信令处理方式程序设计

IDT821054 芯片信令处理模块流程图见图 5-2。

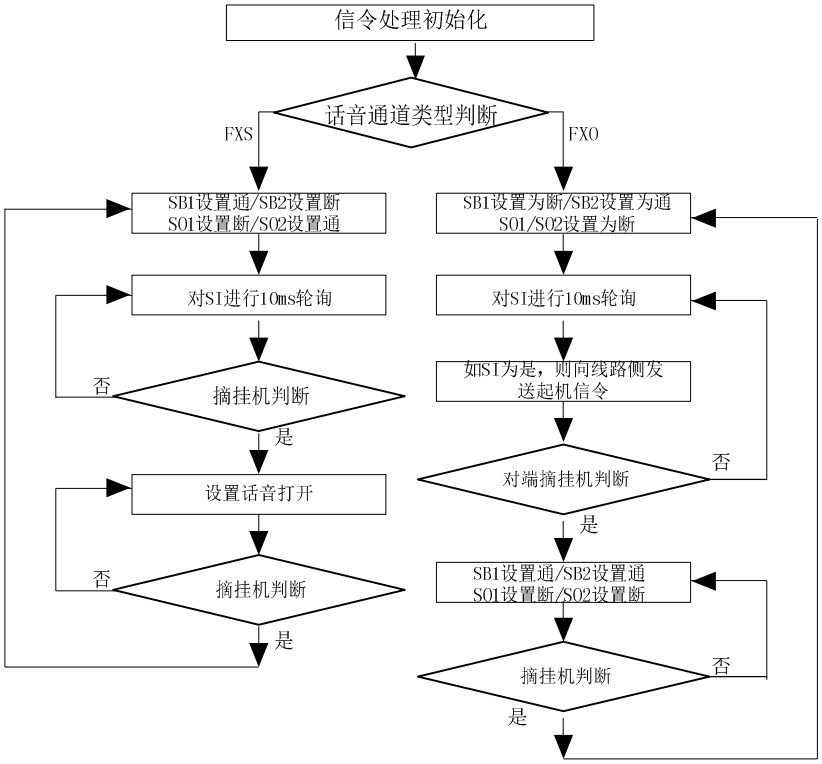

图 5-2 IDT821054 芯片信令处理模块流程图

程序流程如下所述：

1.首先对信令处理进行初始化；

2.通过标志位判断需操作话音通道的类型；

3.判断话音通道类型为 FXS（连接模拟电话机）还是 FXO（连接 PSTN 交换机）后，调用第四章设计的 IDT821054 读写寄存器函数（在从芯片读取信令时，使用 Read_IDT821054()，在向芯片写信令时，使用 Write_IDT821054()），对上述五个信令控制管脚根据类型参数进行初始化配置；

4.对 SI 寄存器进行 10ms 间隔的轮询，以此来检测摘挂机信号；

5.检测到信号后，根据通道类型对 SO 和 SB 参数进行设置，以保证通话顺利进行；

6.继续检测摘挂机信号，以维持通话或切断通话回到初始状态。

我们设计一系列函数完成上述任务，函数名及其定义见表 5-2。

表 5-2 信令处理函数及其意义

名称	意义、函数名
Deal_PCM_Card()	信令处理初始化
Deal_FXS_Tel()	处理 FXS 接口信令
Deal_FXO_Tel()	处理 FXO 接口信令
Interrupt_Deal_PCM_Card()	中断处理接口信令
Read_a_Bit()	读取 SB 参数
Read_M_Bit()	读取 SI 参数
Write_a_Bit()	设置 SB 参数
Write_M_Bit()	设置 SO 参数

在定义了各话音接口、规划了软件流程图和各函数之后，对 IDT821054 话音芯片信令处理函数进行具体设计，包括信令处理初始化、FXS 接口处理、FXO 接口处理、各信令管脚处理等程序。

1.信令处理初始化函数：Deal_PCM_Card()

```
void Deal_PCM_Card(void)
{
。
。
。
switch(Type_PCM [Channel])                    //根据话音通道标志位判断接口类型
{
case Type_MX_FXS:      //如为 FXS 接口，调用 Deal_FXS_Tel()函数对其进行处理
Deal_FXS_Tel(Channel+j);break;
case Type_MX_FXO:      //如为 FXS 接口，调用 Deal_FXS_Tel()函数对其进行处理
Deal_FXO_Tel(Channel+j);break;
default:break;
}}
```

2. 处理 FXS 接口信令：Deal_FXS_Tel()

```
void Deal_FXS_Tel(unsigned char Channel)
{
if(Get4BytesBit(Vaule_M_Bit_Changed,Channel))
```

```
{
if((Status_Hook[Channel]&0x03)==0x03)                    //检测本地起机
{
DataTemp=Get4BytesBit(Vaule_M_Bit,Channel);              //获取 SI 接口状态
Write_a_Bit(Channel,DataTemp);//Vaule_M_Bit[Channel]    //设置 SB 接口状态
}
if(Get4BytesBit(Vaule_M_Bit,Channel))// Vaule_M_Bit[Channel]
{                                                        //检测本地挂机
if(Flag_Value_M_Bit_is_1_Begin[Channel])                 //获取 SI 接口状态
{
Flag_Value_M_Bit_is_1_Begin[Channel]=0;                  //对 SI 接口状态标志位清零
Flag_Value_M_Bit_is_0_Begin[Channel]=1;                  //在本起的情况下才判断本挂
Times_Local_Hook_Down[Channel]=Times_2ms;                //开始计时
}
if((Status_Hook[Channel]&0x03)==0x03&&Delta_Times_2ms(Times_Local_Hook_D
own[Channel], Times_2ms)>=Times_Local_Hook_Down_Standard)
{                                                        //本挂计时时间溢出
Flag_Value_M_Bit_is_1_Begin[Channel]=1;
Clr4BytesBit(Vaule_M_Bit_Changed,Channel);               //SI 接口状态标识位清零
Status_Hook[Channel]&=Local_Hook_Down;                   //重置为本挂状态
Write_a_Bit(Channel,1);                                  //写 SI 接口状态
Set4BytesBit(SB1_OFF_Channel,Channel);
}}
else
{
if(Flag_Value_M_Bit_is_0_Begin[Channel])                 //本起
{                                                        //第 1 次本起检测
Flag_Value_M_Bit_is_0_Begin[Channel]=0;                  //标志清零
Flag_Value_M_Bit_is_1_Begin[Channel]=1;
Times_Local_Hook_Up[Channel]=Times_2ms;
}
。
。
```

```
}}}}
```

3.处理 FXO 接口信令：Deal_FXO_Tel()

```
void Deal_FXO_Tel(unsigned char Channel)
{
if(Get4BytesBit(Vaule_a_Bit_Changed,Channel))                    //检测到 PSTN 侧呼叫
{
Clr4BytesBit(Vaule_a_Bit_Changed,Channel);
if(Get4BytesBit(Vaule_a_Bit,Channel))
{
Set4BytesBit(WEON_FXOMXChannel,Channel);                 //向对端写呼叫开始信令
}else
{
Set4BytesBit(WEOFF_FXOMXChannel,Channel);                //向对端写呼叫结束信令
}}
if(Get4BytesBit(Vaule_M_Bit_Changed,Channel))            //收到对端摘机信令
{
Clr4BytesBit(Vaule_M_Bit_Changed,Channel);
if(Get4BytesBit(Vaule_M_Bit,Channel))
{
Write_a_Bit(Channel,1);                          //设置 SB 接口，负载接通，开始通话
}else
{
if(Get4BytesBit(Vaule_a_Bit,Channel))            //检测 PSTN 挂机信号
{
Write_a_Bit(Channel,0);                          //设置 SB 接口，结束通话
。
。
。
}}}}
```

以上列举了 3 个主要函数的代码，其余函数代码不再一一阐述。此时，2 路电话接口设置完毕，其信令状态的获取和设置由呼叫流程控制，实现了通过话音芯片 IDT821054 对模拟话机侧 FXS 接口和 PSTN 侧 FXO 接口的信令处理。所有信令信息存储在处理器缓存。

5.3 网络侧信令分析

如上节所述，我们已经通过软件实现了 IDT821054 芯片对模拟话机侧 FXS 接口和 PSTN 侧 FXO 接口的信令处理，现在我们要做的，就是将 IDT821054 芯片的信令与 SIP 信令进行转换，此由处理器 IXP425 在其内部完成。

5.3.1 SIP 信令简介

5.3.1.1 SIP 的主要功能

SIP-初始会话协议是一种应用层控制协议，支持通信中以下几个方面的功能：

1.用户可实施性：确认被呼叫的用户方是否愿意参加相互之间的通信会话；

2.用户物理位置的确定：通过 IP 地址来明确用户具体位置；

3.建立会话：连接通话双方；

4.管理会话：其中包含新业务之间的调换使用、变更会话参数、和中止及转移会话等内容。

5.3.1.2 SIP 协议的体系结构

1. SIP 协议的层次结构

SIP 层次分为语法及编码层，传输层、事务层和事务用户层。

2.SIP 消息

SIP 是协议的一种基本文本，使用的字符集为 UTF-8。消息可分为应答(Response)以及请求(Request) 两大类型。

1）请求消息-SIP

SIP 基本请求消息方法描述见表 5-3。

表 5-3 SIP 基本请求消息方法

方法名	描述
OPTIONS	查询该服务器与之相关联的功能和信息
REGISTER	关于客户机的相关资料注册于定位服务器
CANCEL	采用 CANCEL 方法来取消已经连接的呼叫
BYE	客户端运用 BYE 的方法及时向服务器发送消息来结束呼叫
ACK	相应信息，用来回复 INVITE
INVITE	邀请信息，由此来建立会话

2）消息的应答

第一个数字的响应类别定义是状态码，详细内容见表 5-4 所示。

表 5-4 响应消息状态码

状态码	类型	含义
1xx	临时应答	表示请求被接受并在被处理
2xx	成功处理	标识请求处理正确
3xx	重定向	转发到其他的服务器上处理，该请求的完成需要添加附加操作
4xx	客户端错误	这个不能在服务器上完成或者错误的格式被请求
5xx	服务器错误	这个明显合法的请求方式不能被这个服务器以正确的方式处理
6xx	全局错误	无法处理

3．SIP 消息头

消息头是 SIP 协议的主体，其携带了所有会话信息，所带字段值的意义及格式在不同的字段间表示的都不尽相同。

5.3.2 oSIP 信令分析

oSIP 是由 SIP 和 SDP 协议演变过来的。本论文将利用 oSIP 协议进行描述和设计。

5.3.2.1 oSIP 结构

oSIP 由三个模块构成：

1.状态机模块；

2.解析器模块；

3.工具模块。

我们详细介绍前两个模块，oSIP 的模块结构图如下：

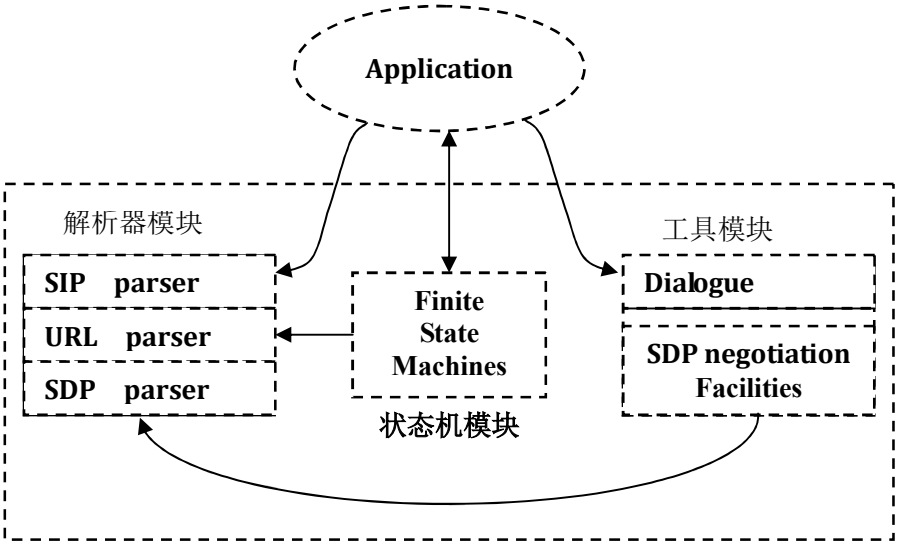

图 5-3 oSIP 模块结构图

5.3.2.2 解析器模块

1.oSIP Parser

表 5-5 oSIP 头域及其意义

名称	意义、函数名
Call—ID	唯一表示一个特定的请求 set(),get(),parse(),2char(),free(),clone(),getnumber()
Authorization	重新测试请求 Init(),set(),parse(),get(),getauth_type(),setauth_type()
Accept	指出的格式进行规格化 set(),get()
Call-INFO	呼叫信息 set(),get(),init(),parse(),2char(),free(),clone()
Contact	提供访问后续请求特定 UA 实例的联系方法 set(),get(),init(),parse(),2char(),free(),clone()
Content—Encoding	它的值指示适用于实体消息体的其他的内容编码 set(),get()
Content—Length	指示消息体的长度 set(),get(),init(),parse(),2char()
Content—Type	实体头域指定发送给接收者的消息体媒体类型 set(),get(),init(),parse(),2char(),free(),clone()
Route	路由信息 set(),get(),init(),parse(),2char(),free()
Via	set(),append(),get(),init(),free(),parse(),2char()

2.oSDP Parser

在 oSIP 中操作功能如表 5-6 所示。

表 5-6 oSIP 操作功能列表

type(类型)	Functions(函数名称—简写)
v	version_set(),version_get()
o	ORIGIN_SET(),USERNAME_GET(),SESS_ID_GET(), SESS_VERSION_GET(),NETTYPE_GET(),ADDRTYPE_GET()
s	name_set(),name_get()
i	info_set(),info_get()
u	uri_set(),uri_get()
e	email_add(),email_get()
p	phone_add(),phone_get()
c	connection_add(),connection_get(),nettype_get(), addrtype_get(),addr_get(),addr_multicast_ttl_get()
b	bandwidth_add(),bwtype_get(),bandwidth_get()
t	time_descr_add(),start_time_get(),stop_time_get()
r	repeat_add(),repeat_get()
z	adjustments_set(),adjustments_get()
k	key_set(),keytype_get(),keydata_get()
a	attribute_add(),att_field_get(),att_value_get()
m	media_add(),media_get(),port_get(),number_of_port_get()

3.URI Parser

URI 主要包含了参数格式，这里不一一列举。

5.3.2.3 状态机模块

1.概述

状态机模块：完成对事务的状态记录。

总共有四种 oSIP 状态机，包括：ICT（带 invit 事件的 out 处理）；IST（带 invit 事件的 in 处理）；NICT（不带 invit 事件的 out 处理）；NIST（不带 invit 事件的 in 处理）。

2.ICT 状态机

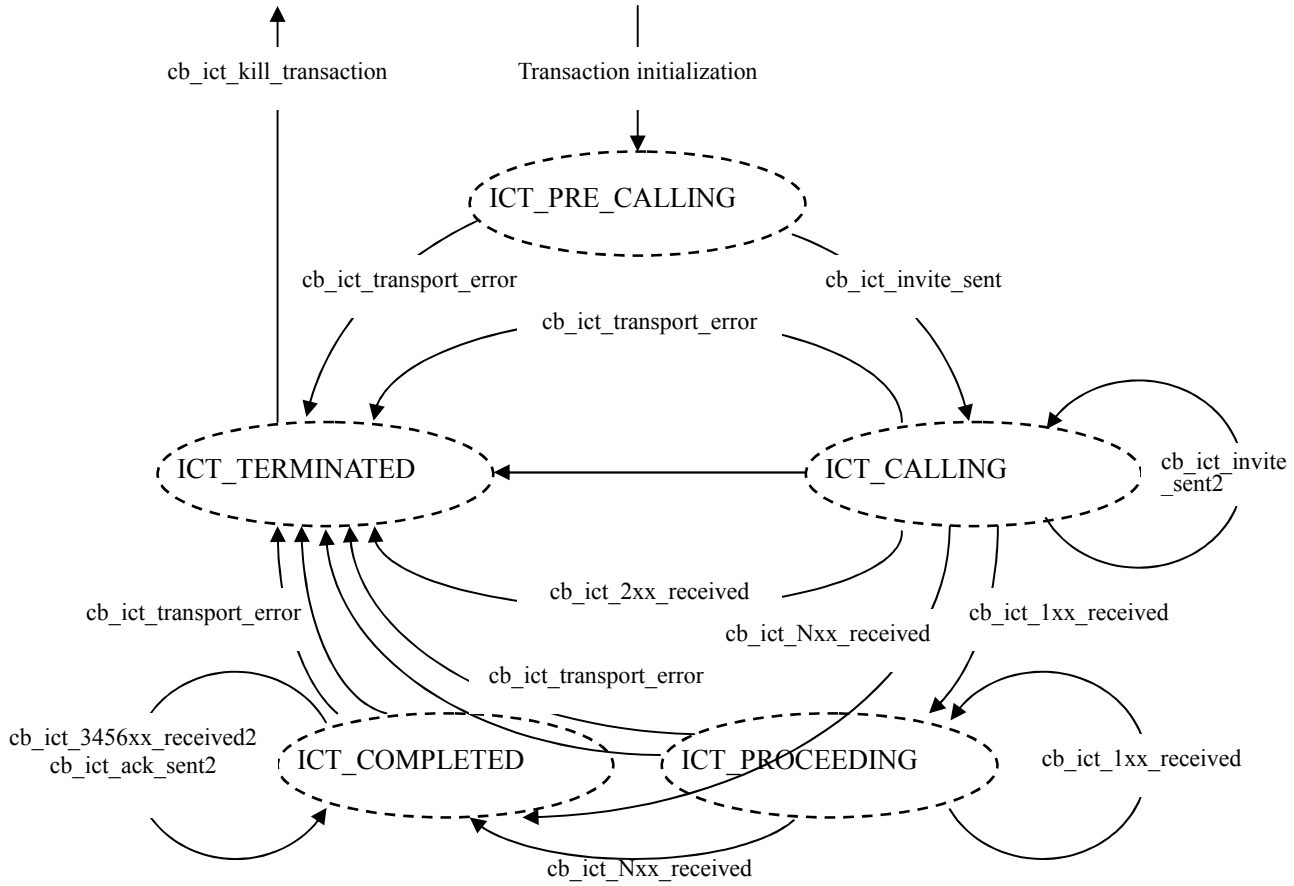

图 5-4 ICT State Machine

注：
cb_ict_Nxx_received：其中 N 表示以下几个值
3 -- cb_ict_3xx_received
4 -- cb_ict_4xx_received
5 -- cb_ict_5xx_received
6 -- cb_ict_6xx_received

3. NICT 状态机

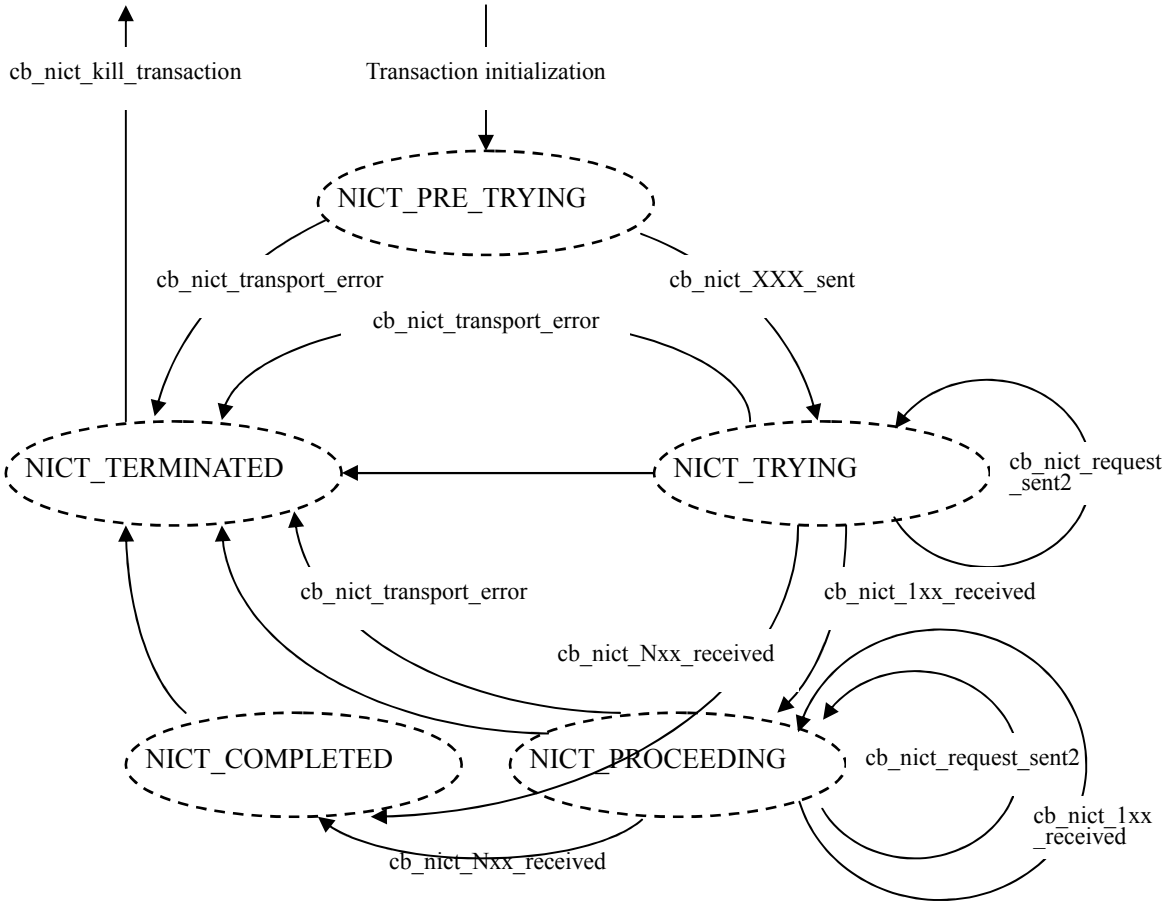

图 5-5 NICT State Machine

注：

cb_nict_XXX_sent：其中 XXX 表示一下几种消息类型，

register	--	cb_nict_register_sent
bye	--	cb_nict_bye_sent
options	--	cb_nict_options_sent
info	--	cb_nict_info_sent
cancel	--	cb_nict_cancel_sent
notify	--	cb_nict_notify_sent
subscribe	--	cb_nict_subscribe_sent
unknown	--	cb_nict_unknown_sent

cb_nict_Nxx_received：其中 N 表示一下几个值

2	--	cb_nict_2xx_received
3	--	cb_nict_3xx_received
4	--	cb_nict_4xx_received
5	--	cb_nict_5xx_received
6	--	cb_nict_6xx_received

4.IST 状态机

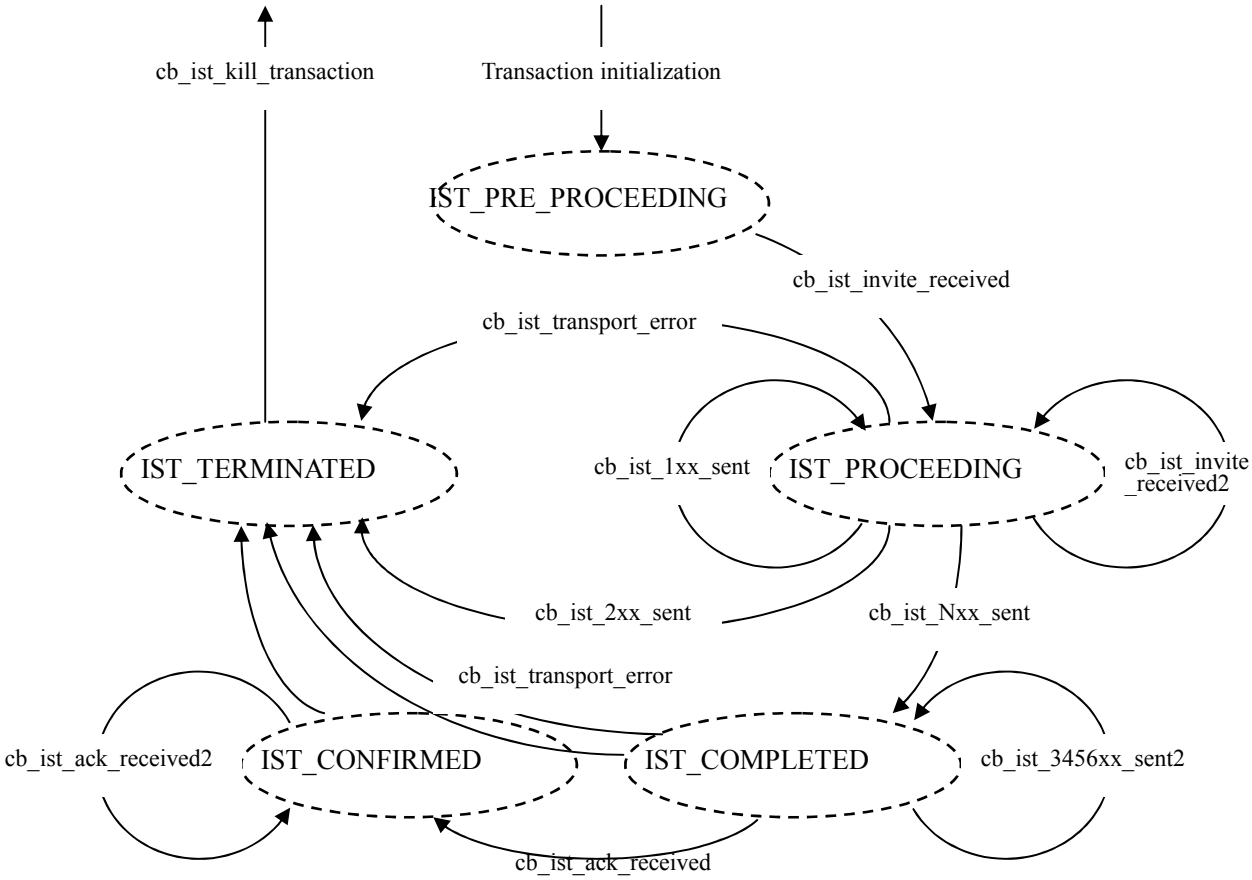

图 5-6 IST State Machine

注：
cb_ist_Nxx_sent：其中 N 表示一下几个值，
3 -- cb_ist_3xx_sent
4 -- cb_ist_4xx_sent
5 -- cb_ist_5xx_sent
6 -- cb_ist_6xx_sent

5.NIST 状态机

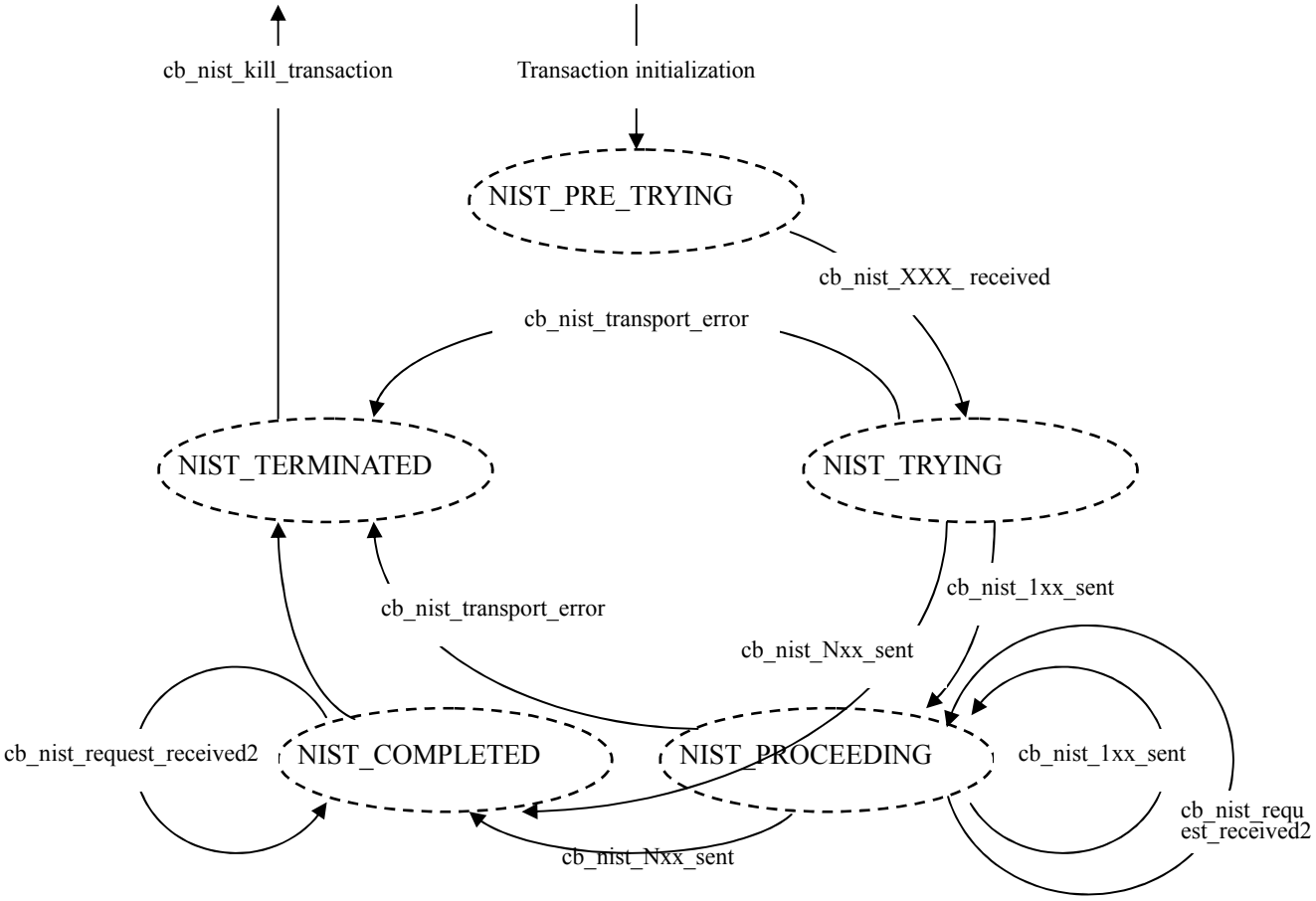

图 5-7 NIST State Machine

注：

cb_nist_XXX_ received：其中 XXX 表示一下几种消息类型，

register	--	cb_nist_register_received
bye	--	cb_nist_bye_received
options	--	cb_nist_options_received
info	--	cb_nist_info_received
cancel	--	cb_nist_cancel_received
notify	--	cb_nist_notify_received
subscribe	--	cb_nist_subscribe_received
unknown	--	cb_nist_unknown_received

cb_nist_Nxx_ sent：其中 N 表示一下几个值

2	--	cb_nist_2xx_sent
3	--	cb_nist_3xx_sent
4	--	cb_nist_4xx_sent
5	--	cb_nist_5xx_sent
6	--	cb_nist_6xx_sent

5.4 信令转换软件设计

5.4.1 程序主流程设计

程序主流程图如图 5-8 所示：

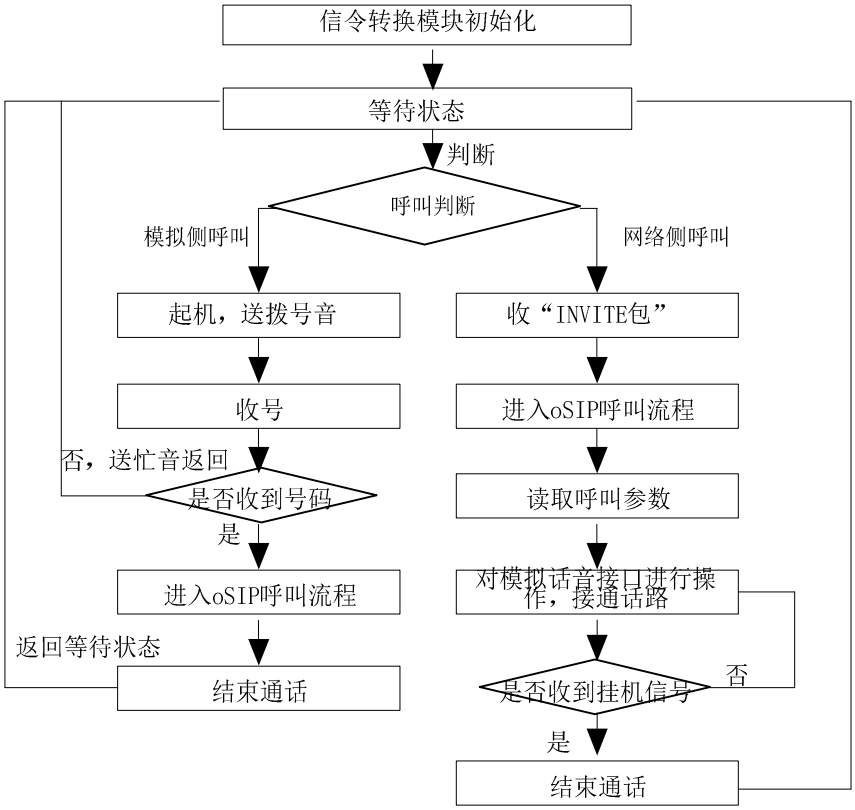

图 5-8 信令处理流程图

首先进行的是对信令转换模块的初始化，初始化完毕之后进入等待状态。这时对标志位进行判断，如果有检测到模拟电话/PSTN 侧的呼叫摘机，则向其送出拨号音并开始收号操作，收到号码后，向网络侧发起 oSIP 呼叫流程，通话完毕后返回等待状态。

如果检测到有网络侧的呼叫，则进入呼叫流程，读取信令后配置模拟话机/PSTN 侧相应寄存器，并接通话路，在收到挂机信号后结束通话返回等待状态。

5.4.2 信令转换模块初始化软件设计

对信令转换模块进行初始化程序如下：

```
int
eXosip_init (void)                                    //初始化函数
```

```
{osip_t *osip;
int i;
memset (&eXosip, 0, sizeof (eXosip));
snprintf (eXosip.ipv4_for_gateway, 256, "%s", GateWay);              //初始化网关
#ifndef MINISIZE
snprintf (eXosip.event_package, 256, "%s", "dialog");
#endif
eXosip.user_agent = osip_strdup ("eXosip/" EXOSIP_VERSION);   //初始化 osip 版本
if (eXosip.user_agent == NULL)
return OSIP_NOMEM;
eXosip.j_calls = NULL;                                                  //状态机开关
eXosip.j_stop_ua = 0;
eXosip.keep_alive = 17000;                                               //设置保持时间
eXosip_set_callbacks (osip);                                             //设置 oSIP 回叫
配置
osip_fifo_init (eXosip.j_events);                                        //oSIP 缓存的初
始化
osip_free (eXosip.user_agent);                                           //设置用户代理
eXtl_udp.proto_port=SIP_PORT;                                            //设置端口参数
。
。
。
return OSIP_SUCCESS;}
```

其包含的底层函数主要利用成熟的开源 API 函数。

按照上述软件设计，完成了信令转换模块的初始化，其对模拟侧和网络侧都进行了初始配置，并进入等待状态。

5.4.3 oSIP 呼叫流程设计

5.4.3.1 呼叫流程框图

1.正常呼叫流程

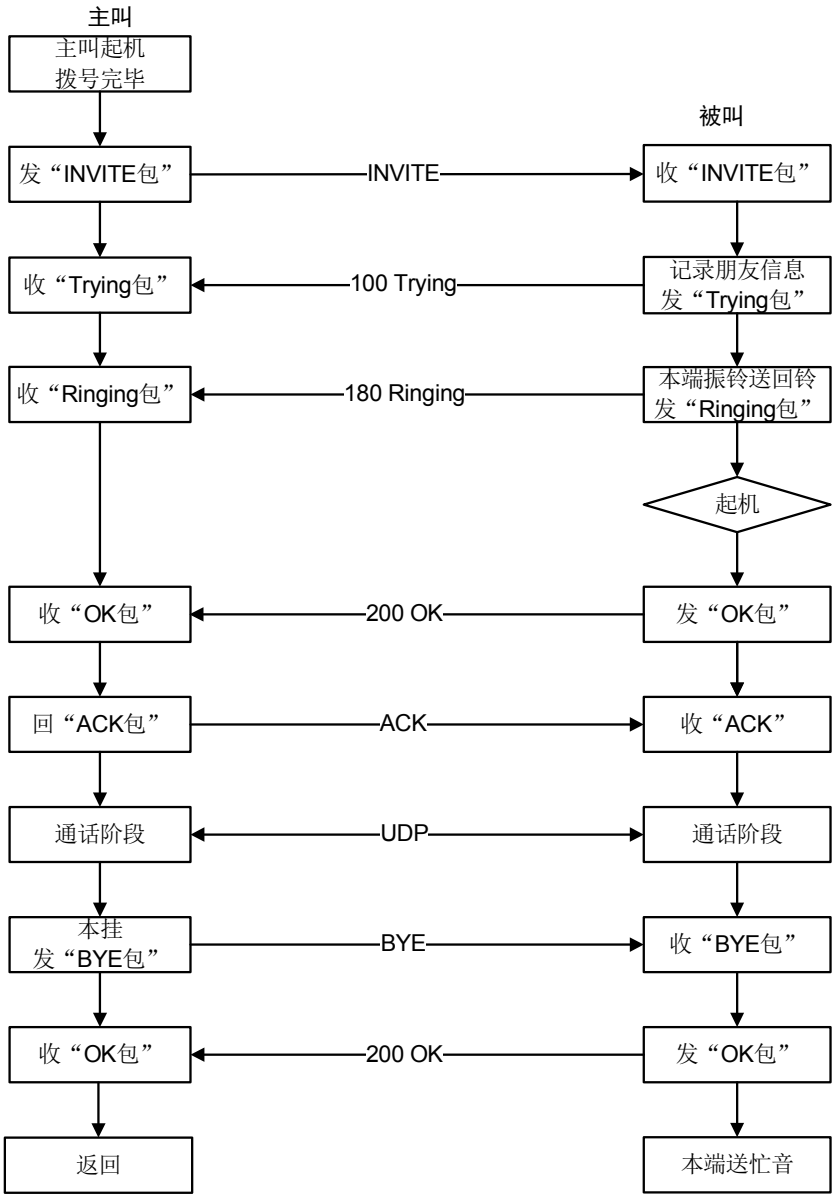

图 5-9 正常呼叫流程图

1)首先主叫检测起机拨号；

2)检测到起机后，生成一个 INVITE 包，并将此包送至被叫。被叫记录信息后发回“100 Trying”包；

3)如果确认了地址信息，就会发 180 Ringing 包；

4)被叫端检测起机信号；

5)被叫端回一个 200 OK 包；

6)收到 200 OK 后，主叫端发送 ACK 包；

7)建立通话，协议 UDP；

8)检测双方挂机信号，并发送 BYE 包；

9)确认 OK 包，结束通话。

2.呼叫遇忙流程

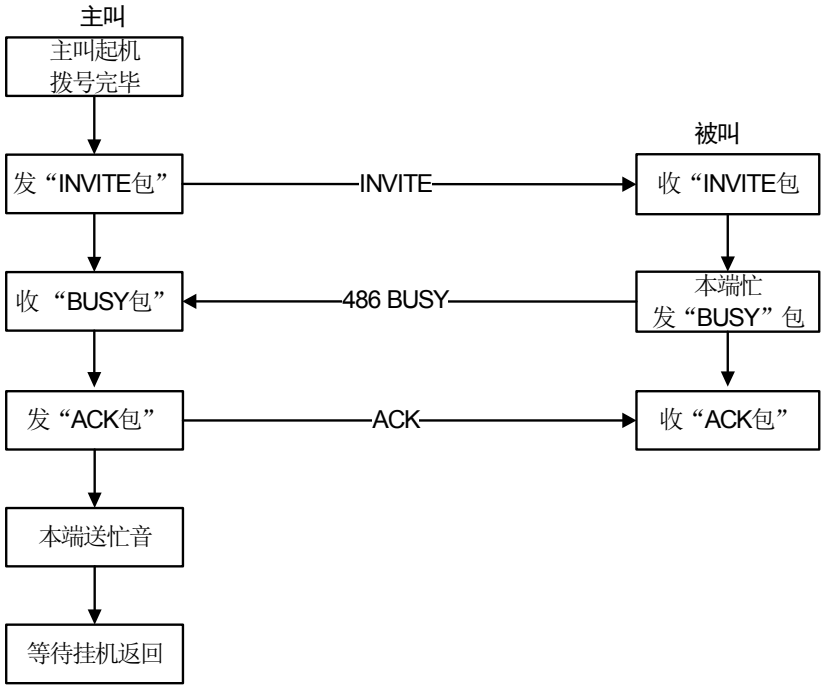

图 5-10 遇忙呼叫流程图

1)首先主叫检测起机拨号；

2)检测到起机后，生成一个 INVITE 包，并将此包送至被叫；

3)被叫发回“100 Trying”包；

4)如果对端检测为忙碌状态，则回应 486 BUSY 包；

5)本端确认后送 ACK 包给被叫端；

6)本端送忙音；

7)本端挂机后返回

3.被叫无应答(主叫取消)流程

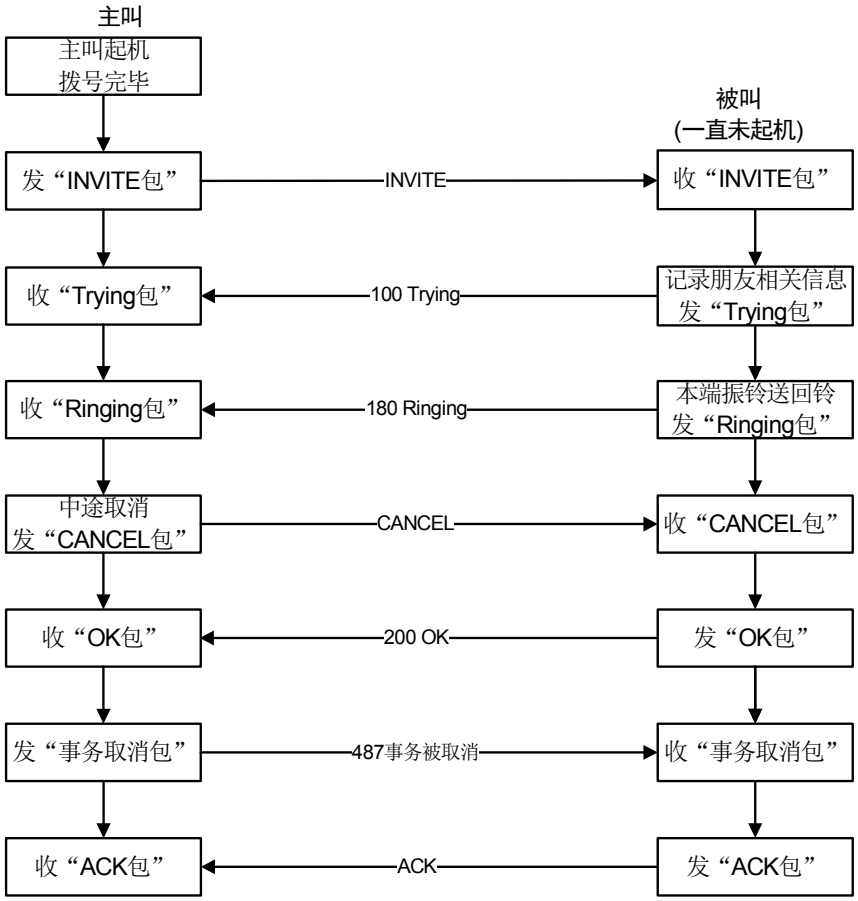

图 5-11 被叫无应答（主动取消）流程图

1)首先主叫检测起机拨号；

2)检测到起机后，生成一个 INVITE 包，并将此包送至被叫；

3)被叫发回“100 Trying”包；

4)被叫发回 180 Ringing 包；

5)如果主叫端中途取消，则发送 CANCEL 包；

6)被叫端收 CANCEL 包；

7)被叫端发送 OK 包；

8)主叫端收 OK 包，发送事务取消包；

9)被叫端收事务取消包；

10)被叫端发 ACK 包，结束呼叫流程。

4.被叫无应答(被叫取消)流程

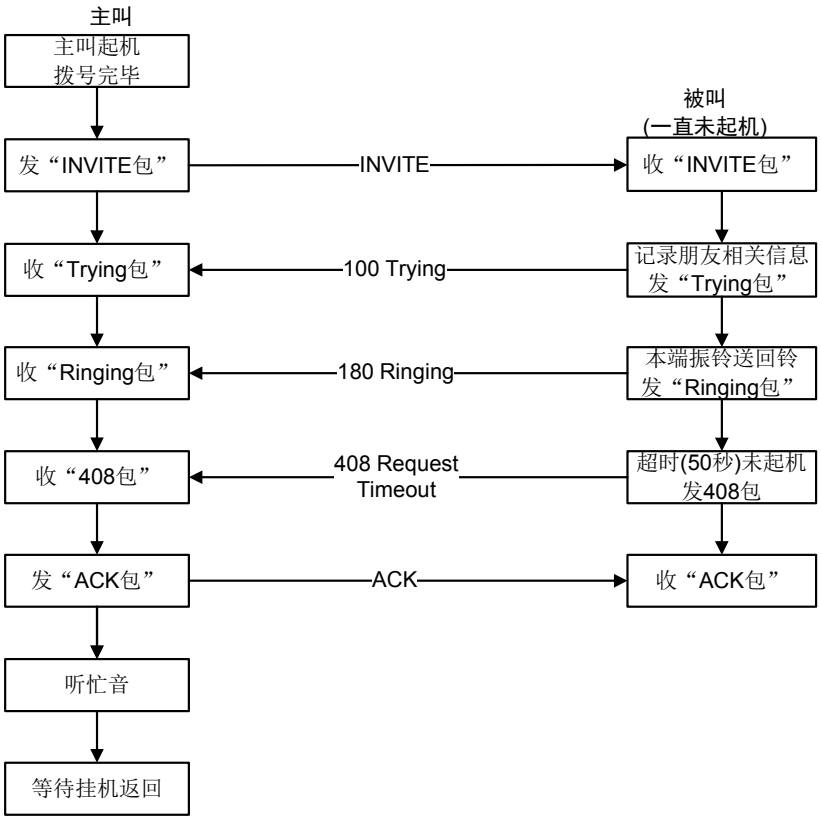

图 5-12 被叫无应答（超时）流程图

1)前序步骤与正常呼叫流程一致；

2)发送 180 Ringing 包后，如果对端无应答，则被叫端发送 Timeout 包；

3)主叫端收 Timeout 包后，发送 ACK 包；

4)被叫端收 ACK 包，结束呼叫流程。

5.4.3.2 呼叫流程软件设计

1.发起呼叫

在接收到呼叫任务后，程序发送 INVITE 信令，并进入状态机模式，开始处理呼叫流程。发送 INVITE 信令程序如下：

a) 建立一个呼叫函数（呼叫初始化参数设置）：UAC_Build_Call()

```
int UAC_Build_Call (char *source_num,char *sourceIP,char *dest_num,char
*destIP,char CID,char *subject)
{
```

```
int call_id, dialog_id;
char from[50];
char to[50];
char audio_port[10];                                    //接收和发送音频的
UDP端口号
from[0]='\0';to[0]='\0';
Make_sip_num(source_num,sourceIP,NULL,from); //创建SIP源端IP地址
Make_sip_num(dest_num,destIP,SIP_PORT_str,to); //创建SIP目的端IP地址
snprintf(audio_port,10,"%d",LocalUDPPort+CID*10); //创建UDP接收端口
call_id=_josua_start_call(from,to,subject,NULL, audio_port,NULL);      //开始一个
UA会话
return (OK);
}
```

程序设置了呼叫的各种参数，包括源端目的端 IP 地址、UDP 端口号等。

b) 开始一个INVITE呼叫函数：UAC_start_call()

```
int UAC_start_call (char *subject, char *route,char *port)
{
osip_message_t *invite;
int i;
if (0 != _check_url (from)){                                    //分析from和
to的url地址
return -1;}
if (0 != _check_url (to)){
return -1;}
i = eXosip_call_build_initial_invite (&invite, to, from, route, subject);     //创建invite
会话
if (i != 0)
{return -1;}
osip_message_set_header(invite, (const char *)"Allow","INVITE, ACK, CANCEL,
BYE");                                                          //为SIP会话
增加一个头
{
char tmp[4096];
```

```
osip_snprintf  (tmp,  4096,                                    //根据解析器模块
oSDP Parser 中参数进行配置
"v=0\r\n"
。
。
。
localip, localip,port);
osip_message_set_body (invite, tmp, strlen (tmp));             //填充会话内容
}
i = eXosip_call_send_initial_invite (invite);                  //发送invite
if (i > 0){
eXosip_call_set_reference (i, reference);}
OSIP_TRACE (osip_trace("CALL STARTED FOR CID: %i\n", i));
eXosip_unlock ();
return i;}
```

此函数在进行了参数的解析和配置后，开始并发送一个 INVITE 呼叫消息，并根据相应的返回消息类型，利用状态机对后续呼叫流程进行处理。

程序中还包括发送Trying包、Ringing包、OK包、ACK包等程序，这里不一一列举。

2.接收 oSIP 消息的处理

对于接收oSIP消息的处理，流程图如图5-13所述。

a) 起始行的解析

解析起始行，首先判断是否是 oSIP 请求，如果是，则解析请求方法、请求 URI 以及 oSIP 的版本；如果不是，则解析 oSIP 版本、状态码及原因短语。最后都会转跳至解析 oSIP 的头域。

b) 头域的解析

如果一个成员有多个参数，可以直接将成员赋值；

如果一个成员只包含一个参数，直接调用函数。

c) 消息体的解析

解析的过程是判断标题头的值。

按照上述软件设计，实现了信令转换模块的初始化、发送 oSIP 消息、处理 oSIP 消息、转换流程等，可以完成信令转换的要求。

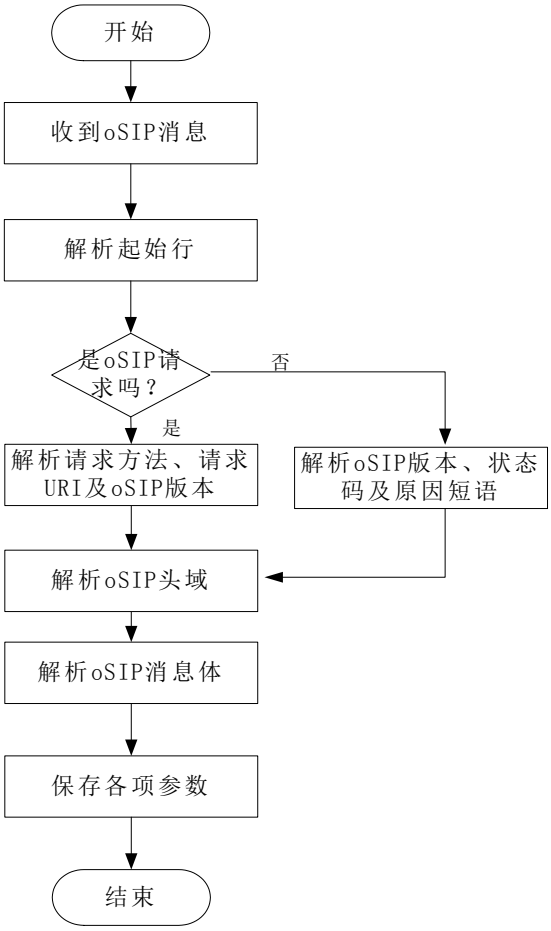

图 5-13 oSIP 消息解析流程

5.5 以太网模块处理实现

我们使用 UDP 协议来进行以太网封包解包。

5.5.1 UDP 分析

用户数据报协议 UDP 是一个位于传输层的协议，其为无连接的不可靠传送服务。其工作流程为将要在网络上传输的数据进行数据包报文封装，报文头部信息有 8 个字节，报文的其余字节为需要传输的正常数据。

UDP 报文格式如下：

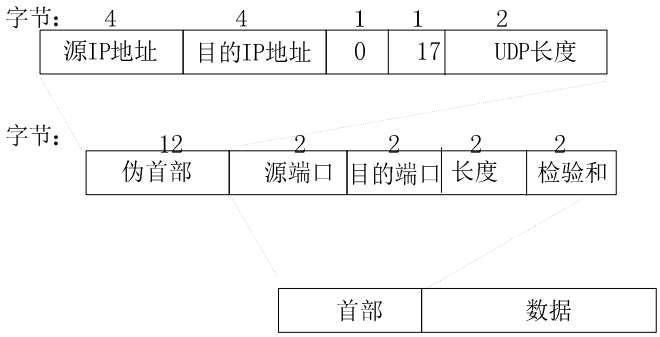

图 5-14 UDP 数据包格式

5.5.2 以太网处理软件实现

报文封包流程：

1.从处理器缓存中获取需要传送的数据与地址；

2.形成包结构，将地址信息填入包头；

3.将数据信息填入包体，其长度可由用户定义；

4.利用底层函数将其发送。

报文拆包流程：

1.将接受的包数据放入缓存；

2.以 UDP 协议格式分解包头，并判断包体长度；

3.获取包体数据，放入处理器缓存中。

我们创建一个 socket，来进行 UDP 报文的处理。程序函数及定义见表 5-7：

表 5-7 UDP 处理函数及定义

名称	意义、函数名
bind( )	为 socket 绑定一个名称
listen( )	对 socket 连接使能
accept( )	接受一个从 socket 来的连接
connect( )	初始化一个到 socket 的连接
sendto( )	向 socket 发送一个信息
send( )	向 socket 发送数据
recvfrom( )	从 socket 接收一个信息
recv( )	从 socket 接收数据
setsockopt( )	设置 socket 参数

创建 socket 的程序如下：

```
STATUS Build_Single_Port(void)
{
struct sockaddr_in serveraddr;
semCpuPort = semBCreate(SEM_Q_FIFO,SEM_FULL);          //创建一个二进制信号量，用以保护 socket 收发
bzero((char *)&ToIPaddr,sizeof(ToIPaddr));
ToIPaddr.sin_family=AF_INET;
//inet_aton(localip,&ToIPaddr.sin_addr);
ToIPaddr.sin_addr.s_addr = htonl(devConfig.serverip);          //初始化目的 IP
ToIPaddr.sin_port=htons(CPUCOMMPORT);                          //初始化目的端口
sockfd=socket(AF_INET,SOCK_DGRAM,0);                           //创建一个 socket
if(bind(sockfd,(struct sockaddr*)&serveraddr,sizeof(serveraddr))<0) //绑定该 socket 的 IP 和端口*/
{
printf("bind error!\r\n");
return ERROR;
}
ioctl(sockfd,FIONBIO,(int)&rc);//设为阻塞型应用!
。
。
。
}
```

创建后就可以直接调用sendto()发送UDP包，recvfrom()接收UDP包，例如：

```
sendto(sockfd,tmpframe,Num,0, sockaddr,sizeof(struct sockaddr_in));
FrameLen=recvfrom(sockfd,Frame,1500,0, sockaddr)。
```

5.6 本章小结

本章主要内容是网关信令转换软件系统的设计与实现，并包含以太网封包解包部分。

第六章 系统测试和结论

6.1 系统测试

6.1.1 测试平台

本次系统测试平台包括本次设计的话音网关、基于 PC 的 SIP 软终端、SIP 电话、SIP 注册服务器、PSTN 程控交换机，测试平台的连接关系如图 6-1 所示。

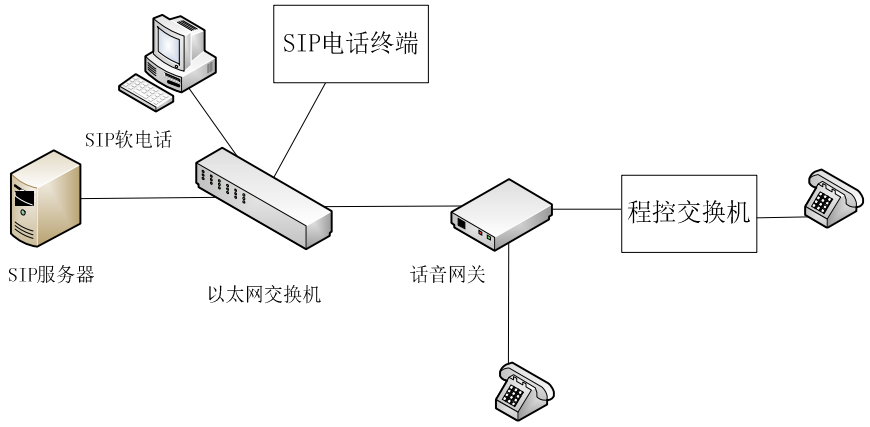

图 6-1 测试平台连接图

6.1.2 测试方案

本测试方案是通过在网络环境下搭建一个测试环境，分别使用嵌入式终端和一个运行于 PC 机上的 SIP 软电话，使用耳麦做呼叫测试。主要从通话流程抓包、呼叫成功率、呼叫保持能力、丢包率来测试终端的功能和性能。

连接好相关设备后，按照表 6-1 配置各个设备。

表 6-1 测试配置表

设备	IP 地址	SIP URI/call number
SIP 服务器	192.168.1.107	无
SIP 软电话	192.168.1.93	1001@192.168.1.93
SIP 电话终端	192.168.1.118	1002@192.168.1.118
话音网关	192.168.1.228	1003@192.168.1.228 1004@192.168.1.228
程控交换机上电话机		803

话音网关配置为VoIP侧1003接入网关本身电话，1004转接至PSTN 803，PSTN

侧设置收四位号码后响应，判断转发号码。

通话测试多次 SIP 到 PSTN 和 SIP 到 PSTN 的响铃时间，通话质量和挂断情况。

6.1.3 测试结果

6.1.3.1 SIP 和 UDP 以太网包分析

在测试过程，由于 PSTN 侧无法监视信令，因此通过监控 SIP 软终端所在计算机，利用抓包软件 CommView，捕获包截图如下。在有 SIP 服务器情况下，以太网包通过服务器转发。

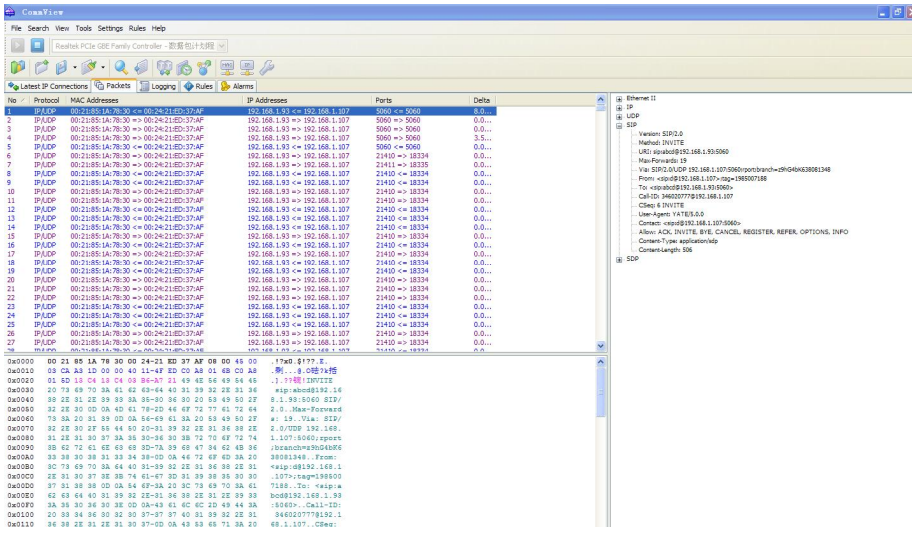

图 6-2 抓包软件总体抓包图

一．从模拟话机侧呼叫 SIP 软终端流程，与前文对应，通过抓包内容描述。

具体流程如下：送出 INVITE 消息—返回一个 100 Trying 消息—产生一个 180 应答—回应 200 OK 消息—产生 ACK 消息—模拟话机侧挂机。

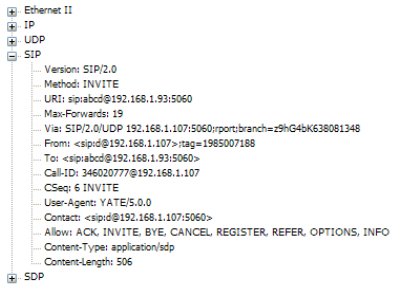

图 6-3 SIP 软电话接收到 INVITE 消息

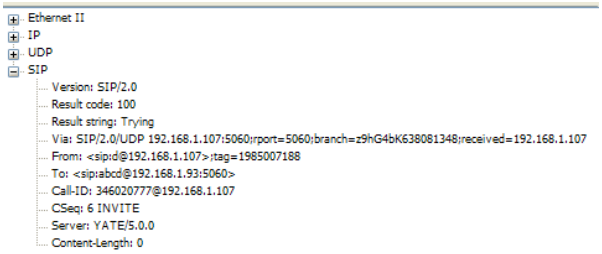

图 6-4 SIP 软电话向网关侧发送 100 回复

从包信息中可以看到发送端发出的 INVITE 信息和接收端回应的 100 Trying 信息。

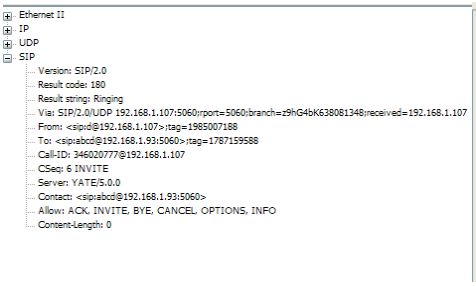

图 6-5 SIP 软电话向网关侧发送 108 回复

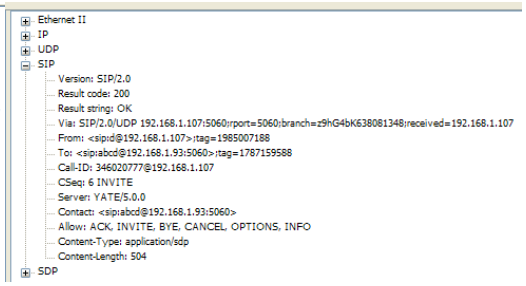

图 6-6 软电话起机后回复 OK

从包信息中可以看到接收端继续回应的 180 Ringing 和 200 OK 信息。

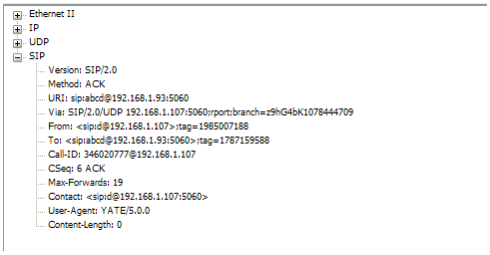

图 6-7 接收到 ACK 表示通话建立

从包信息中可以看到发送端发出的 ACK 的信息，双方由此进入通话状态。

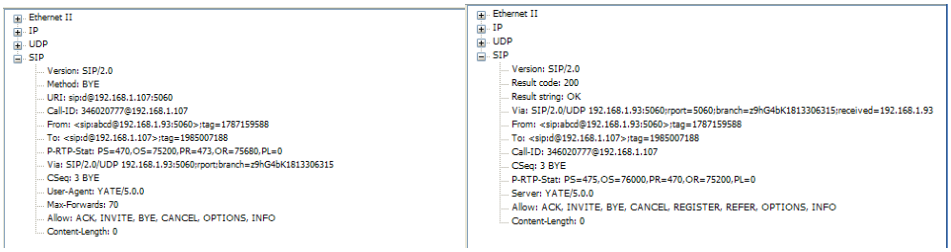

图 6-8 模拟话机侧挂机结束

从包信息中可以看到发送端发出的 BYE 信息和接收端回应的 200 OK 信息，就此双方通话结束。

呼叫通话成功，通过抓包显示，符合从模拟话机侧呼叫 SIP 软终端的流程设计。

二．从 SIP 软终端呼叫 PSTN 侧流程，与前文对应，通过抓包内容描述。

具体流程如下：SIP 软终端送出 INVITE 请求—返回一个“100 Trying”消息—产生一个 180 应答—回应 200 OK 消息—发送 ACK 消息—SIP 软终端侧挂机。

图 6-9 SIP 软电话发出 INVITE 消息图

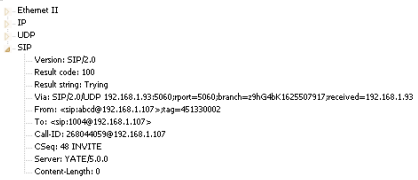

图 6-10 网关返回 100 回复

从包信息中可以看到发送端发出的 INVITE 信息和接收端回应的 100 Trying 信息。

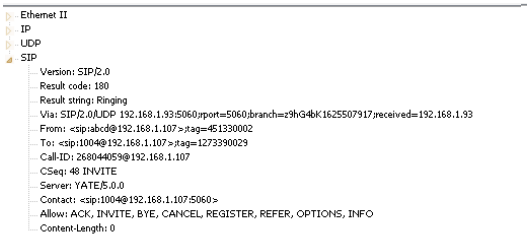

图 6-11 网关发送 180 应答

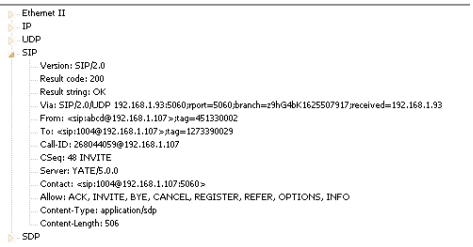

图 6-12 网关回复 OK

从包信息中可以看到接收端继续回应的 180 Ringing 和 200 OK 信息。

图 6-13 接收到 ACK 表示通话建立

从包信息中可以看到发送端发出的 ACK 的信息，双方由此进入通话状态。

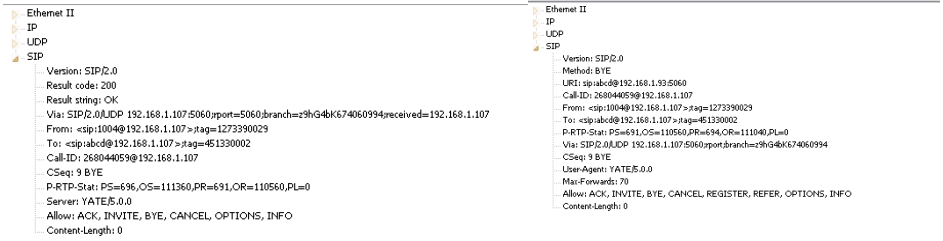

图 6-14 SIP 软终端侧挂机结束

从包信息中可以看到发送端发出的 BYE 信息和接收端回应的 200 OK 信息，就此双方通话结束。

呼叫通话成功，通过抓包显示，符合从 SIP 软终端呼叫 PSTN 侧的流程设计。

6.1.3.2 呼叫成功率

我们设计了 4 种呼叫测试方向：

1.模拟电话通过 FXS 接口呼叫网络侧 SIP 电话终端；

2.PSTN 侧通过 FXO 接口呼叫网络侧 SIP 电话终端；

3.PC 侧 SIP 软电话呼叫模拟电话；

4.PC 侧 SIP 电话终端呼叫 PSTN 侧 FXO 接口；

5.网络侧 SIP 电话终端呼叫模拟电话。

表 6-2 呼叫成功率

方向	测试次数	成功次数	成功率
1	100	100	100%
2	100	96	96%
3	100	98	98%
4	100	99	99%
5	100	97	97%

根据测试结果，从网络侧发起呼叫成功率要小于从交换机侧发起的呼叫。说明在网关信令转换能力上还有改善的空间。但是满足电话使用需要。

6.1.3.3 呼叫保持能力

掉线是人们谈到 VoIP 应用时常提到的一个问题，呼叫保持能力的测试是希望验证系统的稳定性和健壮性。在呼叫建立之后，长时间保持接通状态，以验证被测设备的工作稳定性。

表 6-3 呼叫保持时间测试表

方向	测试次数	保持时间
模拟电话-SIP 电话终端	1	5h
PSTN 侧-SIP 电话终端	1	6h
SIP 软电话-PSTN 侧	1	4h

测试结果显示，在网络稳定的情况下，设备运行非常稳定，完全能够达到通话要求。

6.1.3.4 丢包率

我们编写了测试软件来验证网关设备的网络处理能力。

1、在同一网段内运行。

Bandwidth usage：download=l7．325959 kbits／sec，upload=22．474432 kbits／sec

udp-message-udp-stats：

Global statistics：

udp-message—number of udp packet sent=590
udp-message-number of udp bytes sent=-14889 bytes
udp-message-number of udp packet received=589
udp-message-number ofudp bytes received=7797 bytes
udp-message-number of udp packet lost=O
udp-message·number of udp packets received too late=O
udp-message-number of bad formaRed udp packets=0
udp-message-number ofpacket discarded because of queue overflow—O
Bandwidth
usage：download=1 6．786639 kbits／sec，upload=2 1．983863 kbits/sec
udp-message-udp-stats：
Global statistics：
udp-message-number of udp packet sent=640
udp-message-number ofudp bytes sent=16162 bytes
udp-message-number ofudp packet received=638
udp-message-number of udp bytes received=8444 bytes
udp-message-number of udp packet lost=O
udp-message-number of udp packets received too late=O
udp-message-number of bad formatted udp packets=0
udp-message-number of packet discarded because of queue overflow=O
2、在不同的网段运行
Bandwidth usage： download=18．656193 kbits／sec kbits／sec，upload=18．229572kbits／see
udp-message-udp—stats：
Global statistics：
udp-message-number of udp packet sent=3107
udp-message-number of udp bytes sent=56083 bytes
udp-message-number of udp packet received=3072
udp-message-number of rip bytes received=55457 bytes
udp-message-number of udp packet lost=O
udp-message-number of udp packets received too late=4
udp-message-number ofbad formatted udp packets=0
udp—message-number of packet discarded because of queue overflow=O

Bandwidth usage：download=18．917293 kbits／sec，upload=19．000682 kbits／sec

udp-message-udp-stats：

udp-message-number of udp packet sent=3107

udp-message-number of udp bytes sent=56083 bytes

udp-message-number ofudp packet received=3122

udp-message-number of udp bytes received=56357 bytes

udp-message-number of udp packet lost=6

udp-message-number of rip packets received too late=4

udp-message-number of bad formatted udp packets=0

udp-message-numberofpacket discardedbecause ofqucue overflow=0

3、参数分析

在同一网段内：发送包 3072，丢失包 0，坏包 0，未到终端包 4，丢包率 p=(0+0+4)／3072=0．001；

在不同网段内：发送包 3122，丢失包 1，坏包 0，未到终端包 4，丢包率 p=(6+0+4)／3122=0．003。

测试结果显示，丢包率非常低，满足通话需求。

6.1.3.5 呼叫压力测试

论文设计了一种呼叫压力测试方法：

1.PSTN 侧通过网关设备的 FXO 接口呼叫网络侧 SIP 电话终端，并保持通话状态；

2.模拟电话机通过网关设备的 FXS 接口呼叫 PC 侧 SIP 软电话，在对方挂断后重新呼叫，并反复重复上述过程。

测试结果显示，在第 2 项操作反复进行时，第 1 项操作中的话音通道维持稳定，通话质量不下降，未出现断线情况。说明网关设备能够满足多种话音连接方式同时存在的情况，并能维持通话质量不变。

综合上述数据，本设计能够满足通信要求。

6.2 本章小结

根据各项指标，本话音网关能满足以下要求：

1.VoIP 网关设备能够有效的沟通两种采用不同协议的网络，为两种网络业务的互通建立一条有效的途径；

2.VoIP 网关具有信令转换的功能，能将本地信令传送到公共电话交换网；

3.能够提供 FXO、FXS 电话接口，分别接入 PSTN 交换机或者直连电话。

4.运行稳定可靠。

本设计圆满完成。

第七章 总结与展望

作为当今发展比较迅速的两种技术，嵌入式技术和 VoIP 技术都具有宽广的发展前景。将两种技术进行结合也是发展的必然，因此本文设计在嵌入式技术和 VoIP 技术的基础上，将两者进行融合，设计并实现 VoIP 网关设备，很好的沟通了现有的 PSTN 网络和 IP 网络。

本文做了如下工作：

1.首先对 VoIP 网关设备的实现进行了理论分析。包括了话音 AD/DA 转换、SIP 协议、G.729 压缩编码等关键技术。

2.进行了总体设计，包括硬件搭建和软件体系架构的确定。明确了使用的芯片、协议和软件操作系统等。

3.分模块对网关设备的软件进行了设计。包括话音处理模块、信令转换模块和以太网模块等。

4.给出各个模块的软件流程和具体代码。

5.搭建测试平台，对设备进行验证，经过测试，网关设备达到了系统设计要求。

7.1 本文的主要贡献

本文的设计方案实现了设计目标和需求目标，为成都某铁路区间项目提供了满足要求的设备。并且模块化，集成度高，能满足生产化的要求。

7.2 下一步工作的展望

测试结果表明，本 VoIP 网关设备实现了主要功能，但性能还可以进一步的提高，比如在接通率方面。针对这些问题，下一步的工作就是提高系统稳定性并且增加无线功能，如增加 IP 网与 GSM 网的通信。

致 谢

本论文的顺利完成离不开导师蒋体钢副教授的指导，再次向蒋体钢副教授表示衷心的感谢。同时也对提供过帮助和资料的同学、朋友表示感谢。

参考文献

[1]J.Rosenberg,H.Schulzrinne,G.Camarillo,et al.SIP:Session Initiation Protocol[J].IETF RFC3261, June 2001,318-356

[2]H.Sengar,H.Wang,D.Wijesekra,et al.Detecting VoIP Floods Using the Hellinger Distance[J],IEEE Transactions on Parallel and Distributed systems, Vol. 19, No. 6, June 2008.,794-805

[3]李钦德.多终端融合的 VoIP 智能切换优化的研究与实现[D].北京邮电大学,2011 年

[4]陈丽珊;许克辉.VoIP 流量识别[J].软件,2011 年 06 期,13-14

[5]S.Rubini,J.Corbet, 魏永明译.设备驱动程序(第二版) [M].北京:中国电力出版社,2003-01 ,465-474

[6]于天宇,潘志安,胡克.IP 电话多方通话的设计与实现[J].电脑开发与应用,2013 年 04 期,11-12

[7]S.Anderson, S.Niccolini,D. Hogrefe.SIPFIX: AScheme for Distributed SIP Monitoring[J].IEEE IM2009,2009.

[8]万兵,杨阳.VoIP 流量监测技术的研究与应用[J].电信快报,2010 年 02 期,7-8

[9]H.Son,Y.Lee.An Anomaly Traffic Detection Method for VoIP Applications using Flow Data[J].PAM2009 Student Workshop, 2009

[10]贺阳.P2P 流媒体业务流量分析与识别[D].北京邮电大学,2012 年

[11]Y.Ding,G.Su.Intrusion detection system for signal based SIP attacks through timed HCPN[J].Second International Conference on Availability, Reliability and Security(ARES), 2007

[12]张登银等.VoIP 技术分析与系统设计[M].人民邮电出版社,2003

[13]黄永峰,李建庆.下一代网络核心控制协议:SIP 及其应用[M].人民邮电出版社,2009

[14]葛伦跃.IEEE 802.11 无线局域网中 VoIP 技术研究[D].合肥工业大学,2011

[15]B.Douskalis.IP 电话技术稳定的 VoIP 服务集成[M].机械工业出版社,2000

[16]王瑞刚.IP 电话终端设备-原理、电路及应用[M].西安电子科技大学出版社,2003-11,140-145

[17]沈鑫剡.多媒体传输网络与 VoIP 系统设计[M].人民邮电出版社,2005,108-288

[18]罗斯青.H.323 和 SIP 的比较[J].通讯世界,2002.5,16-17

[19]商思娇.Windows 下基于 SIP 的 VoIP 视频电话及实验平台的设计与实现[D].吉林大学,2014

[20]沈鑫剡.IP 交换网原理、技术及实现[M].人民邮电出版社,2003,251-289

[21]张翠平,王艳平.VoIP 语音质量及评测方法[J].电信网技术,2011 年 12 期,22-24

[22]李琳,柴乔林.H.323 与 SIP 在 VoIP 应用中的实现及比较[J].计算机应用 2002,22(9),21-22

[23]黄永峰.IP 网络多媒体通信技术[M].人民邮电出版社,2003

[24]吴其林.WLAN 中 VoIP 容量与准入控制研究[D].合肥工业大学,2011 年

[25]张登银.VoIP 技术分析与系统设计[M]. 北京:人民邮电出版社,2003-05,153-166

[26]J.Rosenberg,J.Weinberger,C.Huitema,et a1.STUN-Simple Traversal of User Datagram Protocol(UDP)Through Network Address Translators(NATs),RFC 3489 Network Working Group[S],March 2003

[27]秦基伟.语音质量客观评价系统的研究及实现[D].重庆大学,2013 年

[28]张磊,王阿禅.VoIP 语音技术及应用[M].机械工业出版社,2000,189-234

[29]J.Rosenberg Computer Telephony:The Session Initiation Protocol(SIP)：A Keycomponent for Interact Telephony[J].June 2000

[30]王立众.移动 VoIP 中语音质量增强关键技术研究[D].北京邮电大学,2011 年

[31]A.Johnston,S.Donovan,C.Cunningham Session Initiation Protocol(SIP)Basic Call Row Examples[J].IETF RFC 3665,2003

[32] D.Collins,舒华英译.VoIP 技术与应用人民[M]. 北京:邮电出版社,2003-06,3-14

[33] J.Davidson,J.Peters,韩柯译.IP 话音基础[M]. 北京:电子工业出版社,2001

[34]J.Hagenauer:Iterative Decoding of Binary Block and Convolutional Codes[J]. IEEE Trans.on Information Theory.1996, 42(2): 429-445

[35]黄亮.基于软交换的 VoIP 通信系统研究与应用[D].武汉科技大学,2011 年

[36] 占亿民,林宝成.VoIP 技术及其在宽带 IP 网的应用[J].有线电视技术,2002,9(15),11-13

[37]E.Aire,T.Maharaj, L.Lide Implementation Considerations in a SIP Based Secure Voice over IP network[J],AFRICON,2004