摘要

本文对嵌入式网络管理系统进行了需求分析，完成了硬件系统和软件系统的设计和基于 RABBIT 系统的实现。系统实现时重点实现嵌入式 Web 服务器，在嵌入式 Web 服务器实现时进行了 TCP/IP 协议裁减，通过访问 Web 服务器用户可实现网元的管理。相应的 CGI 程序通过启动相关的系统功能实现相应的网管项目，从而通过 IP 网实现了对通信设备的网络管理。

本文在完成嵌入式通信设备网络管理系统的需求分析后，用 UML 完成了系统的静态和动态建模，根据系统功能要求选择相应的硬件部件 RABBIT2000 处理器、PCA9564 IIC 控制器等实现了系统的硬件设计，在软件设计中分析比较了 Dynamic C 中的任务调度方法以及 μC/OS-Ⅱ操作系统，确定了系统运行的操作系统环境为 μC/OS-Ⅱ，分析了 Web 服务器实现的关键技术和相关协议 HTTP、ICMP、TCP、IP 等，分析设计了文件系统 FS2、看门狗、通信命令、IIC 通信、分级安全管理，按照高内聚、低耦合的原则完成了系统任务的划分并依任务的重要程度完成了优先级的设定，根据系统通信的需要完成了相关状态机的设计，依据电路板的参数和告警信息设计了交互命令字及对应数据结构。基于 RABBIT2000 处理器系统和 Dynamic C 环境实现了系统编码，并依据 ITU-T 的 M.30XX 建议对系统进行了测试评估。

关键词：嵌入式系统，Web 服务器，网络管理，μC/OS-Ⅱ，HTTP 协议

Abstract

The thesis disserts the requirement analysis of embedded network management system,hardware system and software system design,and realization based RABBIT system.The important part is the realization of embedded Web server and reduction in TCP/IP protocol. The user can manage the network element through accessing the Web server. When the server is accessed correspoding CGI program can be called to implement network management work,then the user can manage the communication equipment through IP network.

After completed the requirement analysis of embedded network management system,the thesis disserts the realization of static and dynamic modeling in UML, hardware system design with RABBIT2000 processor,PCA9564 IIC controller etc accoding to system's function requirement. About software design,the thesis disserts the reason of selecting the μC/OS-II as the operation system after analysed and compared the task scheduling method in Dynamic C and μC/OS-II. Furthermore,some key technique such as HTTP, ICMP, TCP, IP protocols are analysed, file system FS2,watchdog,communication command,IIC communication, rating security management are designed. Task's creation is on the principle of high cohesion and low coupling ,and its priority is set according to the importance. All state machine's design are comply with the necessity of system, mutual command and data structure are created according to the circuit board's parameter and alarm information.The system is realized based on RABBIT2000 processor and dynamic C compiling environment. At last, the system is evaluated comply with ITU-T M.30xx regulation.

Key Words: Embedded System, Web Server, Network Management, μC/OS-II, HTTP Protocol

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和取得的研究成果，除了文中特别加以标注和致谢之处外，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 天津工业大学 或其他教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示了谢意。

学位论文作者签名：张会波 签字日期：06年 1月16日

学位论文版权使用授权书

本学位论文作者完全了解 天津工业大学 有关保留、使用学位论文的规定。特授权 天津工业大学 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，并采用影印、缩印或扫描等复制手段保存、汇编以供查阅和借阅。同意学校向国家有关部门或机构送交论文的复印件和磁盘。

（保密的学位论文在解密后适用本授权说明）

学位论文作者签名：张会波 导师签名：

签字日期：06 年 1月16日 签字日期：06年 1月16日

学位论文的主要创新点

一、用UML实现了嵌入式实时通信系统的静态建模和动态建模。UML是一种通用的强大的建模语言，特别适于面向对象的分析与建模，一般嵌入式实时系统因其自身资源有限，为了节约资源并提高系统的运行效率，开发语言一般均为非面向对象语言，无法实现从类到生成代码等功能，但其强大的建模功能可以很好的描述实时系统的动态特性以及时序特性，本文用UML/Realtime的建模思想和方法实现了系统的静态及动态建模。

二、根据系统的功能需求实现了TCP/IP的协议裁减，实现了必须的相关协议，无关协议全部裁减掉，从而节省了系统资源，保证了Web的运行效率。

三、在嵌入式系统中采用文件系统来管理配置信息等数据。

四、根据网元的特性以及以后的发展需要，建立了一套主从设备交互通信的命令，可满足网管系统与网元的交互通信功能。

五、实现了基于Web的网管系统，利用通用的浏览器即可实现系统访问，从而简化了访问过程，同时利用企业内部数据通信网（DCN）易于实现集中网管，而且提高了系统的安全性。

第一章 引言

1.1 课题目的及意义

随着通信技术的发展和通信网规模的不断扩大，各类通信设备和通信系统在通信网中的应用越来越广，相应通信设备的维护量也愈来愈大，开发方便、易用的网管系统对于快速维护设备和降低维护工作量就愈发重要。而嵌入式 Web 提供通用 Web 的基本访问功能，由于 IP 网的通用性，嵌入式 Web 不仅易于使用，同时也方便利用企业内部网络（Intranet）实现集中监控，实现远程登入维护，不仅缩短了故障处理时间，而且节约了运行维护成本。

通信网络管理系统可提供对多个网络元素的有效管理，按照ITU-T M.30XX建议中对网管系统的管理功能的规定，网管系统应提供性能管理、故障管理、配置管理、计费管理、安全管理等在内的各项管理功能。性能管理通过对网络中的设备进行测试和监视统计获取关于网络运行状态的各种性能参数值，进而根据参数值进行性能分析和控制。故障管理通过对网络运行状态进行监控检测出故障信息，根据故障信息进行定位和诊断，然后可采用相应的操作恢复故障。配置管理实现对网元设备的参数变更管理，配置相应的网元运行参数。计费管理收集用户使用网络资源的信息，并保存到相关文件中，可以据此向用户收取资源使用费用。安全管理可保护网络资源，使网络资源处于安全运行状态。

嵌入式Web是相对于通用的Web 服务器来讲，由于嵌入式系统的资源有限，它所实现的功能比通用的Web 服务器要少，对有些不需要的功能和协议可根据系统的需求进行裁减，实现瘦Web服务器。

北京天诺泰利通信有限公司开发了系列协议转换器，Ethernet—E1转换器，V.35-E1转换器，此类协议器可单独应用，也可集中应用，集中应用时将各类协议转换器插入5U机箧，5U机箧可插入机架中，不仅节省资源，而且方便管理，为了方便集中网管，缩短障碍处理时间，降低运营维护成本，开发网络管理系统对本类设备进行统一管理和配置是非常必要的。为此天诺泰利公司启动了本系统的开发计划，作为5U机箧的网管系统。

1.2 现状

目前协议转换设备较多，各厂家也有类似的网络管理系统，但大多自成体系，需专用软件才能实现网管功能，这样网管系统的开发成本会大幅增加，从而使整个系统的开发周期延长，用户使用这些网管系统还需安装他们的专用软件，甚至通过专用硬件连接才能实现网管功能，非常不方便。由于天诺泰利公司开发了一系列协议转换设备，要想提高设备的整体性能和功能，必须开发相应的网管系统，因通信设备的差异性，无法采用通用网管系统实现网络管理，所以必须开发针对此公司设备的网管系统，实现各网元的有效管理。

第二章 需求分析

2.1 系统功能简介

本系统是作为一个5U协议转换设备机箧的网络管理系统，系统要求实现网络管理的主要功能，包括性能管理、安全管理、配置管理、故障管理，对应有相应的功能模块实现。机箧内插有电源板、E1－Ethernet板，E1－V.35板等电路板，机箧共有16个槽位，为了硬件设计的方便和降低生产成本，其中1号、16号槽位固定插电源板，9号槽位固定插网管板，即本文设计实现的网络管理系统，其它槽位可插任何协议转换电路板。下图为5U机箧槽位图，蓝色标记为网管板默认位置。其中电源板可外接220V交流市电和48V直流电源，提供±5V，±12V直流电压，额定输出功率300W，满足机箧内其它电路板的功率消耗。E1－Ethernet板，E1－V.35板分别实现E1到Ethernet，E1到V.35的协议转换，板上CPU为P8966X，支持IIC协议，图2-1为5U机箧槽位图。

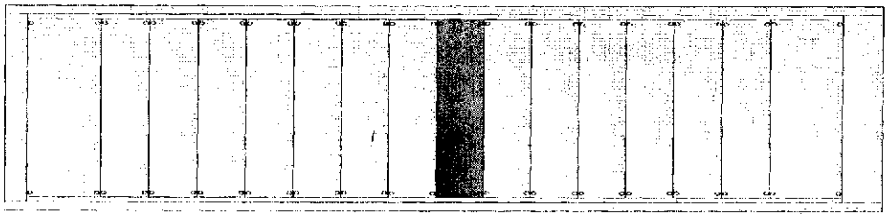

图2-1 5U机箧槽位图

通过本网管系统实现对其它协议电路板的网络管理，要求系统能够及时查询他们的告警信息、状态信息，并对其进行参数配置、复位等，同时要求能够支持远程登录，安全管理好，易维护，界面友好。

2.2 需求分析

2.2.1 系统总体功能需求

根据网管系统的功能要求，结合被网管设备的特点，系统总体功能需求可确

定如图2-2，系统划分为用户登录、设备监视、设备管理、系统管理、系统帮助等几个功能模块,如图2-2所示。

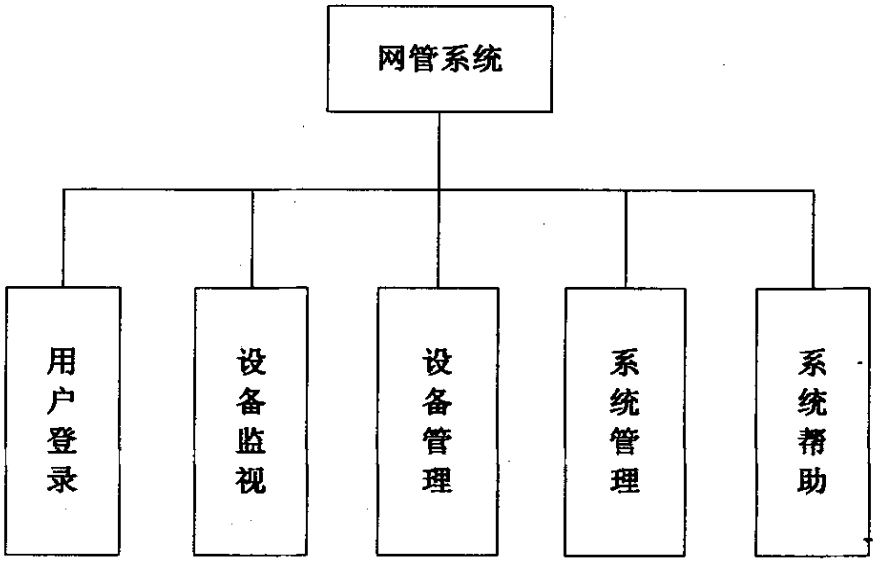

图2-2 系统总体功能需求框图

2.2.2 用户登录

本系统支持 HTTP 协议登录，用户可通过通用浏览器输入 IP 地址，以不同权限登录本系统：一般维护人员或高级维护人员，不同权限登录将被系统授予不同的访问权限，这样可以提高系统的安全性，避免无关人员访问不在其权限范围内的信息，如图 2-3 所示。

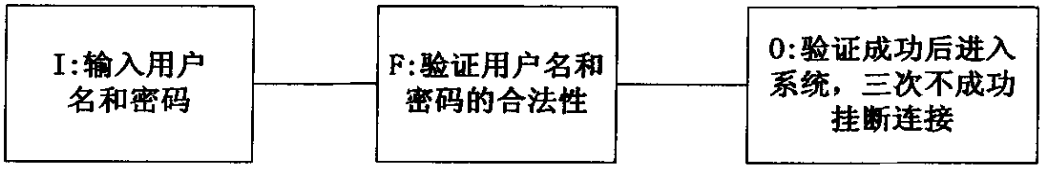

图2-3 用户登录模块功能

2.2.3 设备监视

成功登录系统后即进入此模块，此模块根据轮询结果动态显示机筐内所有槽位的设备配置及运行情况，并提供按钮超级链接，点击按钮即可对相应槽位的

设备执行相关操作，如图 2-4 所示。

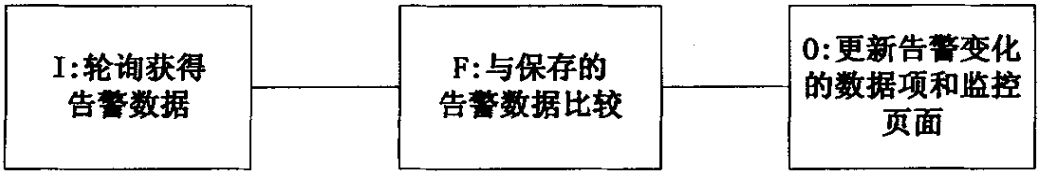

图 2-4 设备监视模块功能

2.2.4 设备管理

本模块是系统功能的主要实现模块，通过与任一槽位设备的交互通信实现告警查询、状态查询、参数配置、设备复位等功能，如图 2-5 所示。

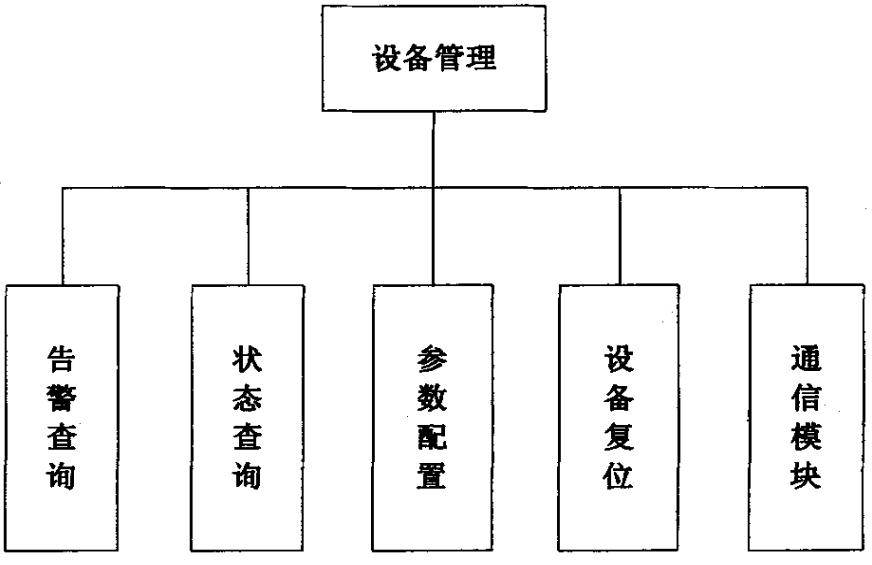

图 2-5 设备管理功能需求框图

2.2.4.1 告警查询

提供告警参数选择界面，根据所选参数选项封装告警查询命令，然后调用通信模块发送告警查询命令给被查询设备，通过被查询设备的应答信息解析有无告警及告警类别和级别，用新的告警查询结果更新相关告警记录，若有告警则保存告警记录，以友好的表单方式输出解析结果，如图 2-6 所示。

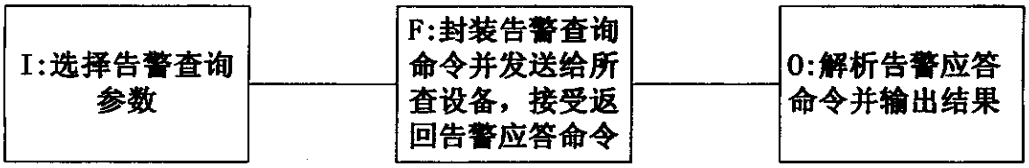

图 2-6 告警查询子模块功能

2.2.4.2 状态查询

调用通信模块发送状态查询命令给被查询设备，通过被查询设备的应答信息解析被查设备的配置情况，并核对与保存的配置数据是否一致，若不一致则更新保存的配置数据，然后以友好的表单方式输出结果，如图 2-7 所示。

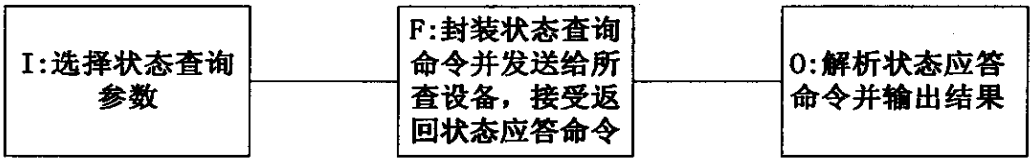

图 2-7 状态查询子模块功能

2.2.4.3 参数配置

提供详细的参数选择页面，能根据所选项使相关选项生效失效，有利于维护人员正确配置数据，能读取保存的上次配置数据。根据所选参数选项封装参数配置命令，然后调用通信模块发送给要配置设备，并根据返回命令解析配置是否成功以及不成功的项目，将配置结果以友好的表单方式输出，，如图2-8所示。

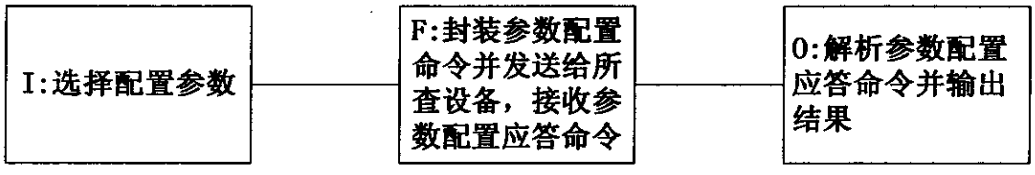

图2-8 参数配置子模块功能

2.2.4.4 设备复位

因复位影响系统的正常运行，当选择复位时应让操作人员再次证实，以免误操作影响设备正常运行。因被复位设备收到复位命令后立即执行复位动作，无法发送复位的应答信息，当复位命令发送成功后即可提示复位成功，如图2-9所示。

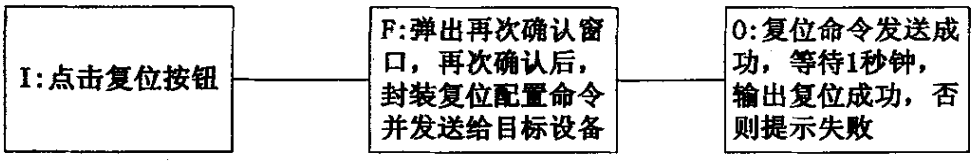

图2-9 复位子模块功能

2.2.4.5 通信模块

通信模块包括网管板与其它被管理电路板的通信和IE浏览器与本系统的通信两部分功能。网管板与其它被管理线卡的通信通过IIC总线实现，通过发送给目标设备各类命令然后接收应答命令，实现网管板和其它各类设备板的通信。IE浏览器与本系统的通信通过TCP/IP协议族实现，应用层主要通过HTTP协议实现，如图2-10所示。

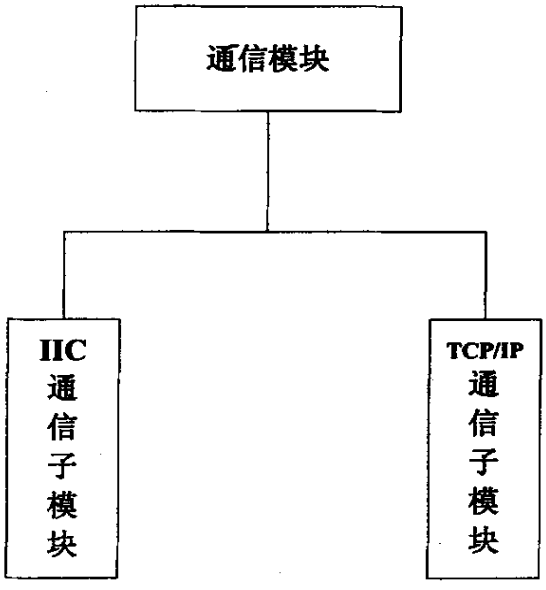

图2-10 通信模块功能

(1) IIC通信子模块

本子模块实现网管板与其它各类线卡的通信。当执行各类命令及状态轮询时，调用本模块发送命令包数据，并接收应答的命令包数据，发送和接收均按字节操作，并有出错等待重发机制保证数据能够准确发送和接收，如图2-11所示。

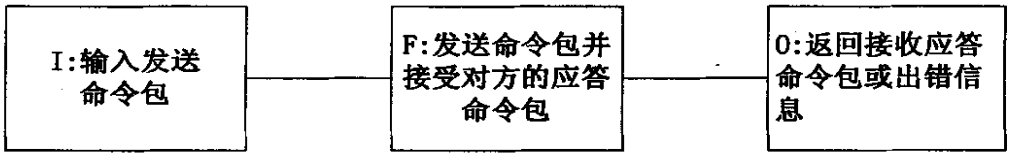

图2-11 IIC通信子模块功能

(2) TCP/IP通信子模块

本模块实现TCP/IP族的协议栈。因为嵌入式系统的资源宝贵，为了节约内存和FLASH空间，对协议族实行了裁减，主要实现了ARP、TCP、IP、ICMP、HTTP协议。ARP协议实现IP地址和数据链路层使用的MAC地址之间的转换，IP协议用于实现数据包的发送，ICMP协议实现数据差错的控制和PING命令，TCP协议实现传输层的功能，因为系统基于BS架构，HTTP协议用于实现HTML文本的解释执行，如图2-12所示。

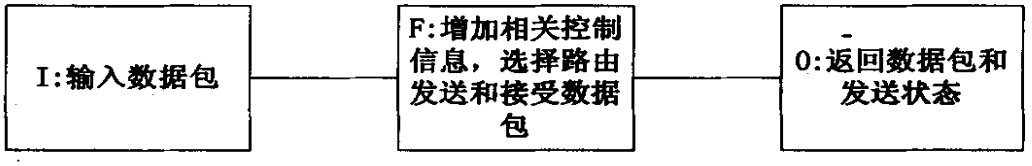

图2-12 TCP/IP通信子模块功能

2.2.5 系统管理模块

包括用户名和密码管理、网络参数配置、系统时钟管理三项功能。用户名和密码管理实现登录系统的用户名和密码的增删和修改，网络参数配置可动态修改本设备的IP地址、网关、DNS等参数，系统时钟管理可修改系统的日期和时间，如图2-13所示。

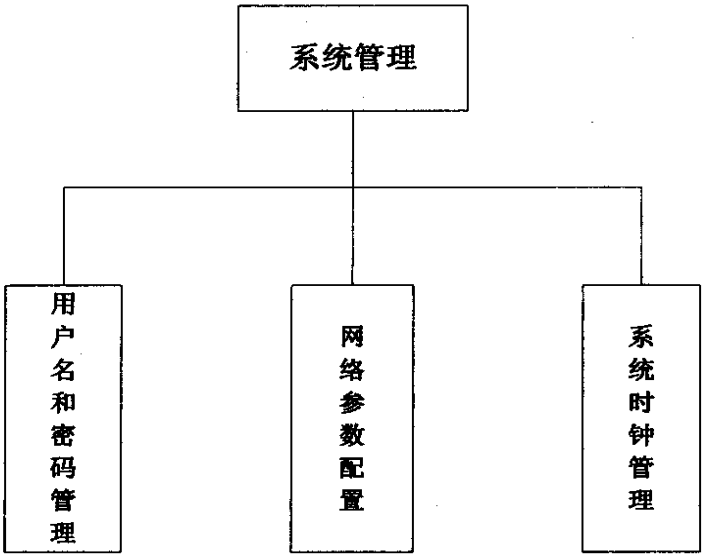

图2-13 系统管理功能需求框图

(1) 用户名和密码管理

用户可修改当前用户的用户名和密码，修改前需进行安全验证，修改时提供密码的正确性验证。可新建用户名和密码，也可查询已存在用户的密码，如图2-14所示。

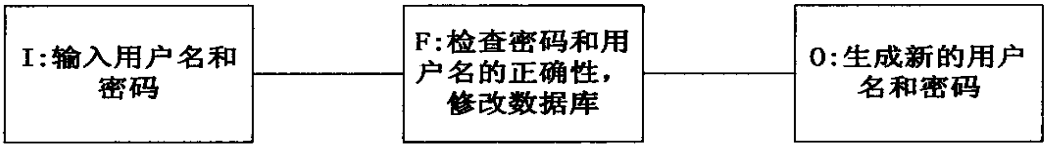

图2-14 用户名和密码管理子模块功能

(2) 网络参数配置

可动态配置本设备的IP地址(IP4)，子网掩码，DNS等参数，并对输入数据进行正确性验证，如图2-15所示。

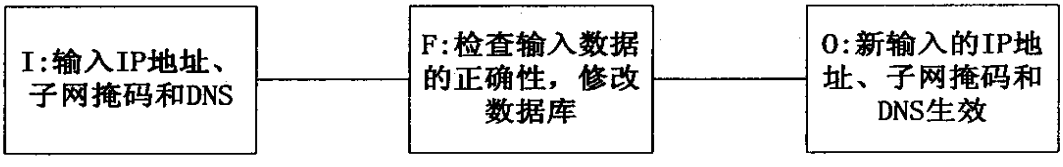

图2-15 网络参数配置子模块功能

(3) 系统时钟管理

可查询系统当前的日期和时间，并可修改系统的日期和时间，如图2-16所示。

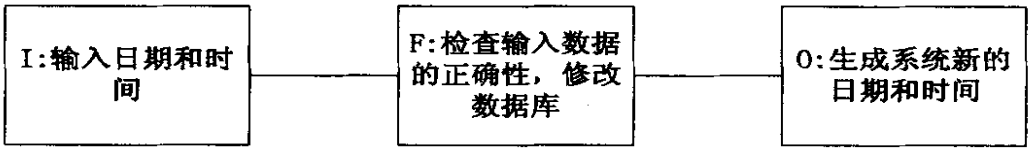

图2-16 系统时钟管理子模块功能

2.2.6 系统帮助

对系统的使用人员提供在线帮助，帮助系统结构与系统功能结构相对应，如图2-17所示。

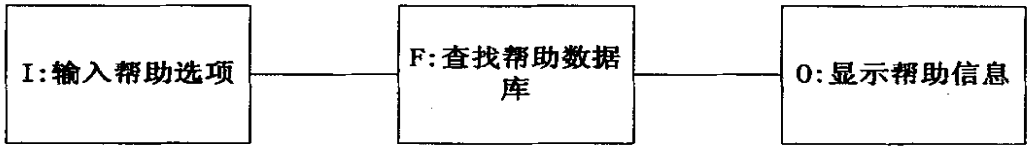

图2-17 系统帮助模块功能

2.3 模型建立

UML是一种标准、基于构件的建模工具语言，通过9种图以不同的设计角度提供系统建模的完整细节，同样适用于嵌入式系统的分析设计。采用UML建模方法分析系统，运用时序图、状态图等可很好的描述系统的实时性，从而完成系统的静态建模和动态建模。虽然大多数嵌入式系统不会采用面向对象语言实现代码，但面向对象的方法同样适用于系统建模[6]。

2.3.1 角色确定

由于本系统为网络管理系统，对系统实行分级安全管理，不同权限的人可使用不同的用例。主要角色为General Operator(一般机务员)，System Administrator(系统管理员)。

2.3.1.1 General Operator(一般机务员)

一般机务员完成日常的一般监视和维护工作，可以使用系统的除影响系统安全外的所有功能。

2.3.1.2 System Administrator（系统管理员）

系统管理员除使用一般机务员的用例外，还可使用系统管理、参数配置等所有用例。

2.3.2 用例分析

根据系统的功能需求，得出如下用例：

- User Login(用户登录)
- Check Slot(设备监视)
- Alarm Query(告警查询)
- Status Query(状态查询)
- Device Reset(设备复位)
- IIC Communication(IIC通信)
- TCP/IP Communication(TCP/IP通信)
- Disconnect TCP/IP Connection(断开TCP/IP连接)
- File System R/W(文件系统读写)
- Parameter Config(参数配置)
- Username and Password Management(用户名和密码管理)
- Network Config(网络参数配置)
- RTC Management(系统时钟管理)
- System Help(系统帮助)

General Operator（一般机务员）的用例关系如图2-18，System Administrator(系统管理员)用例关系如图2-19。

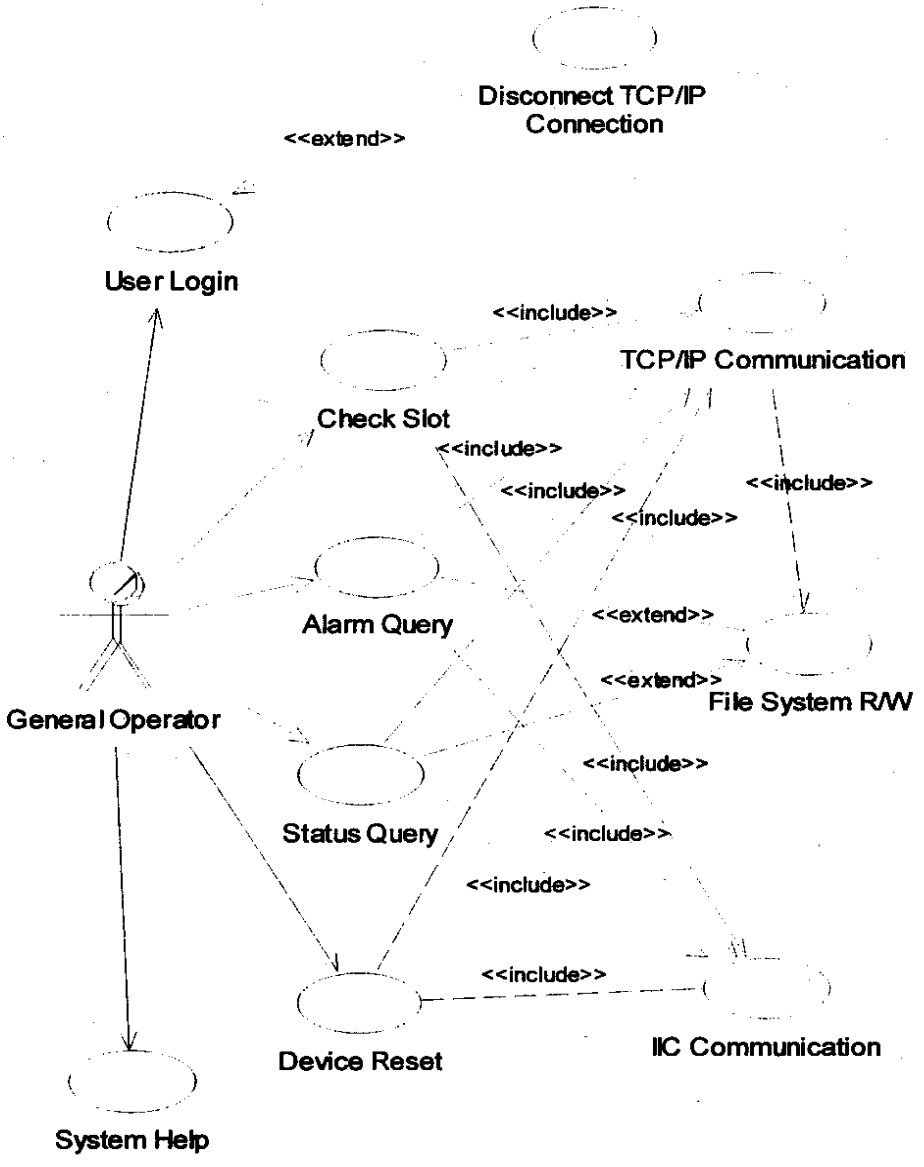

图2-18 General Operator(一般机务员)用例关系图

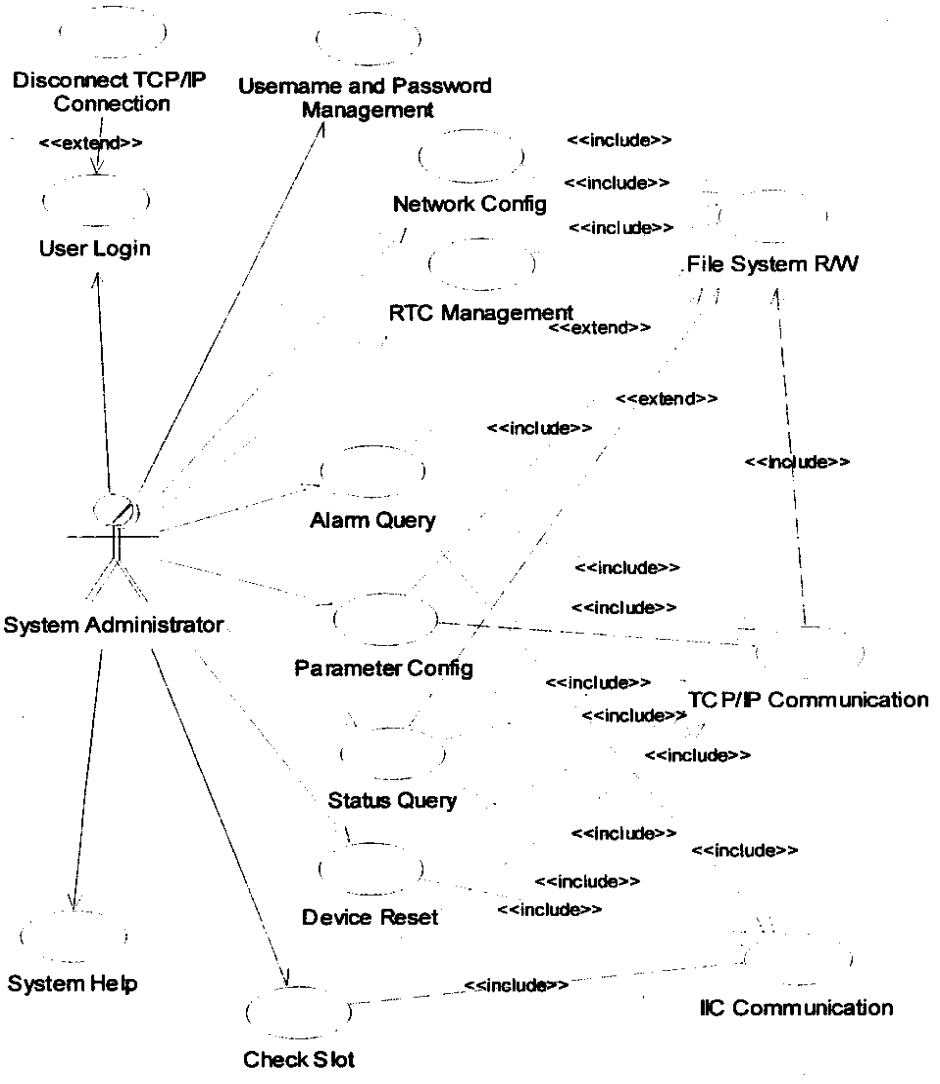

图2-19 System Administrator(系统管理员)用例关系图

2.3.4 系统动态建模

通过时序图、协作图、状态图、活动图更好地描述系统。

2.3.4.1 时序图

(1) General Operator(一般机务员)时序图

General Operator(一般机务员)时序图见图2-20。

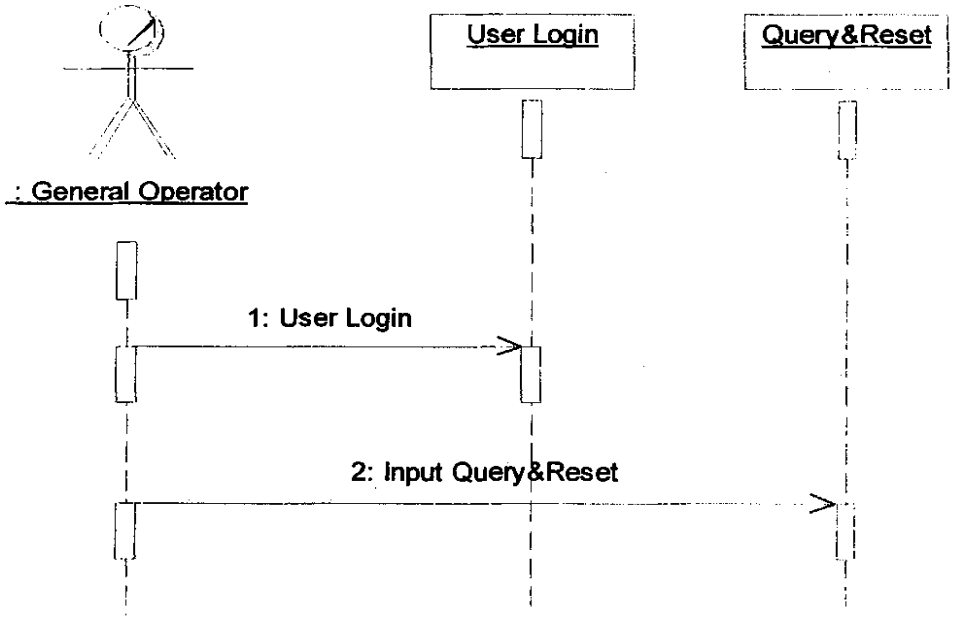

图2-20 General Operator(一般机务员)时序图

General Operator：一般机务员，使用系统最多的人员，完成最基本的系统操作,包括各类查询和复位功能.。

User Login:用户登录,需输入必要的验证信息。

Query&Reset:查询和复位，输入查询和复位的参数。

(2) System Administrator（系统管理员）时序图

System Administrator(系统管理员)时序图见图2-21。

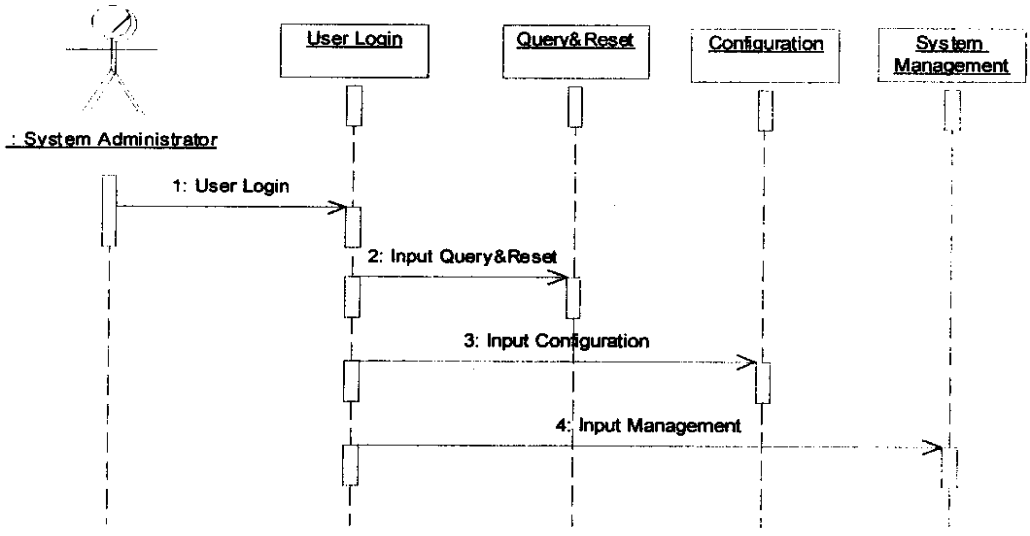

图2-21 System Administrator(系统管理员)时序图

General Operator：一般机务员，使用系统最多的人员，完成最基本的系统操作,包括各类查询和复位功能.。

User Login:用户登录,需输入必要的验证信息。

Query&Reset:查询和复位，输入查询和复位的参数。

Configuration：配置，输入配置参数。

System Management：系统管理，输入相应的参数。

2.3.4.2 协作图

(1) General Operator(一般机务员) 协作图

General Operator(一般机务员)协作图见图2-22。

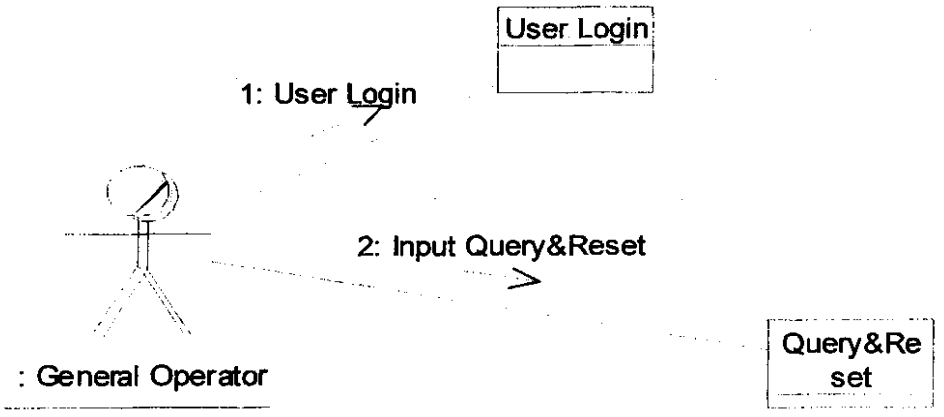

图2-22 General Operator(一般机务员)协作图

(2) System Administrator（系统管理员）协作图

System Administrator（系统管理员）协作图见图2-23。

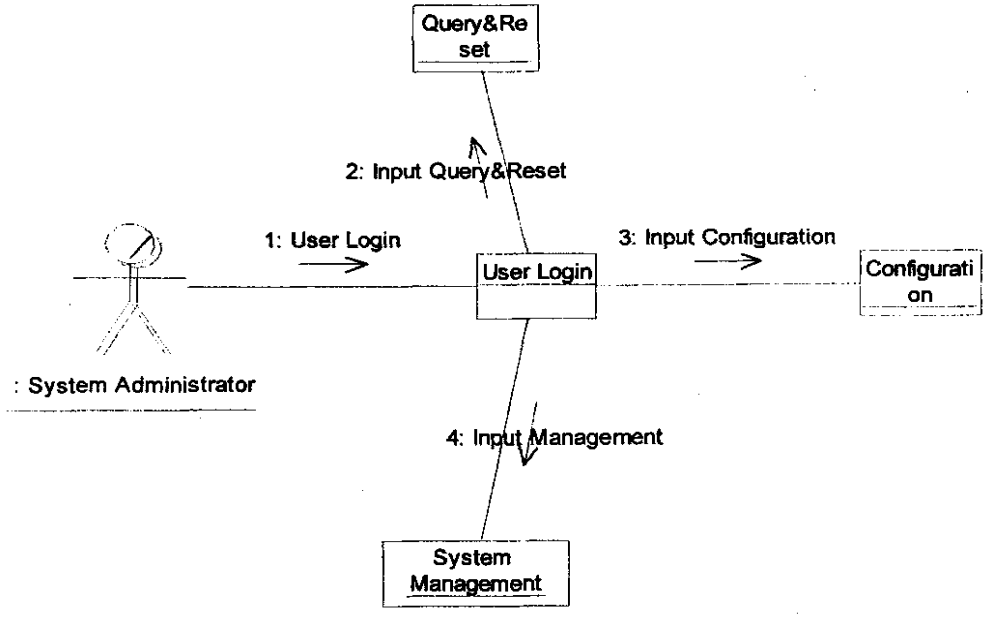

图2-23 System Administrator（系统管理员）协作图

2.3.4.3 状态图

(1) General Operator(一般机务员) 状态图

General Operator(一般机务员)操作状态图见图2-24。

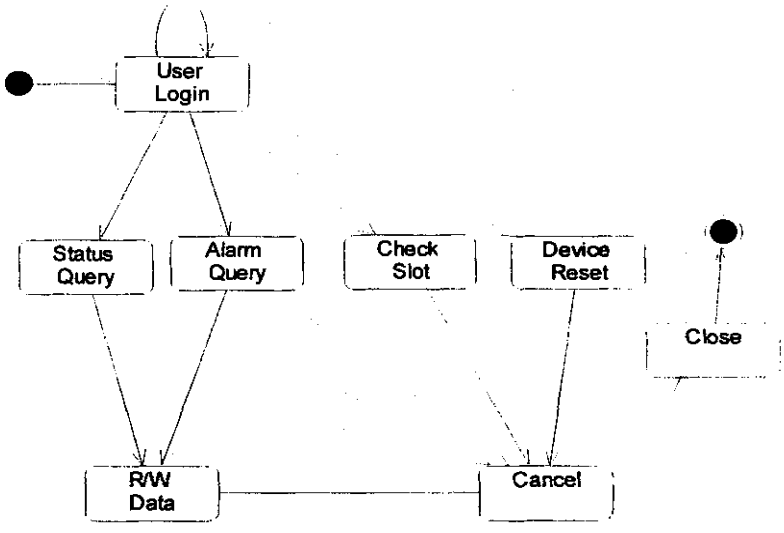

图2-24 General Operator(一般机务员)操作状态图

User Login:登录状态

Status Query：设备状态查询状态

Alarm Query：告警状态查询状态

Check Slot：设备监控状态

Device Reset：设备复位状态

R/W Data：读写文件系统数据状态

Cancel：取消操作状态

Close：关闭连接状态

(2) System Administrator（系统管理员）状态图

System Administrator（系统管理员）操作状态图见图2-25。

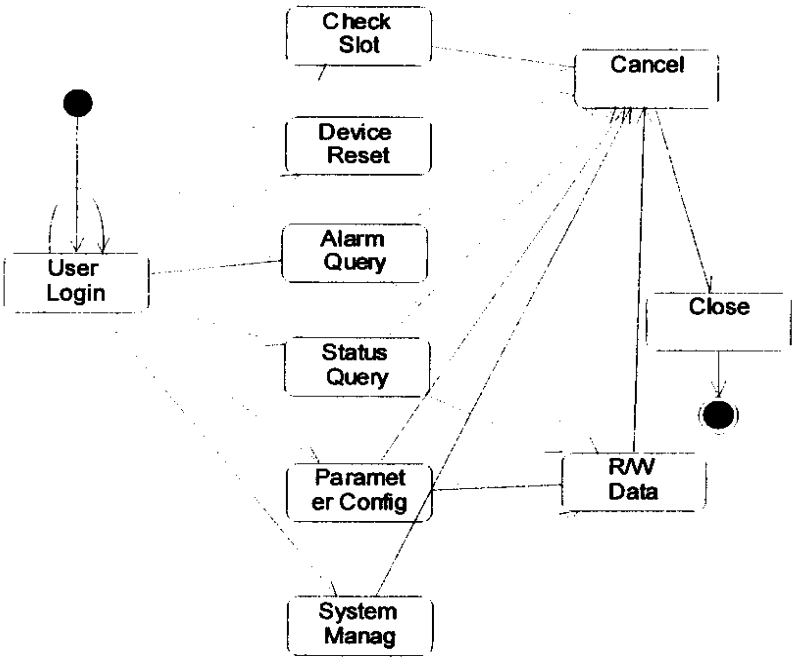

图2-25 System Administrator（系统管理员）操作状态图

User Login:登录状态

Status Query：设备状态查询状态

Alarm Query：告警状态查询状态

Check Slot：设备监控状态

Device Reset：设备复位状态

Parameter Config：参数配置状态

System Manag：系统管理状态

R/W Data：读写文件系统数据状态

Cancel：取消操作状态

Close：关闭连接状态

2.3.4.5 活动图

操作活动图见图2-26。

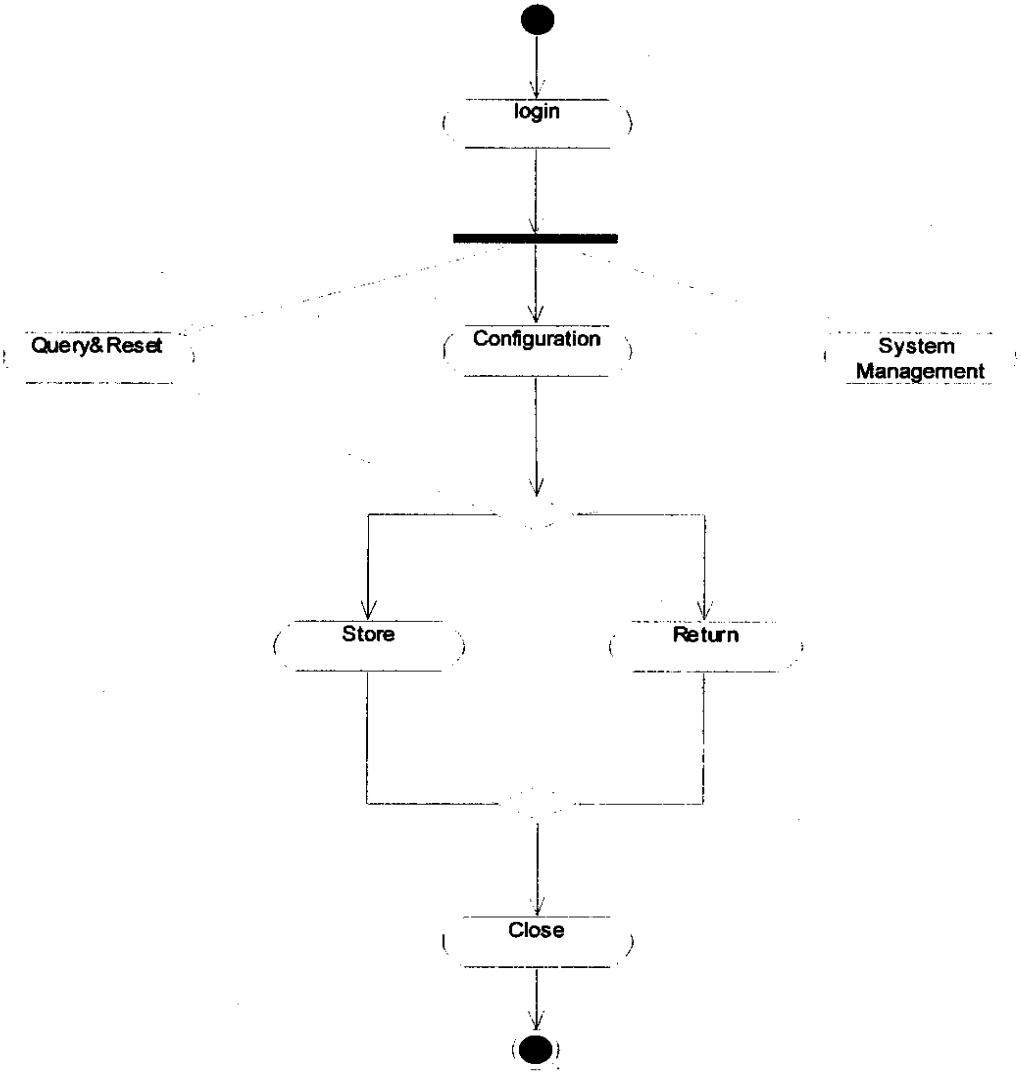

图2-26 操作活动图

login:登录状态活动

Query&Reset:查询和复位状态活动

Configuration:配置活动状态

System Management:系统管理活动状态

Store:存储活动状态

Return:取消操作活动,回到操作前状态

Close:关闭活动状态

第三章 硬件设计

3.1 器件选择

上一章详细分析了系统的需求，系统要实现的功能已经确定，相应实现系统所需的硬件也就定下来了。主要器件为MCU、Ethernet网协议控制器、IIC通信器件、SRAM、FLASH、UART通信器件。

因为通信系统对可靠性要求高，并能够在恶劣环境（温度，防尘）中运行，具有灵活的供电电压，所以选用可靠性非常高的Rabbit2000处理器，实际选用由Rabbitsemiconductor公司生产的RCM2200开发板，此板标准设计为SRAM 128k，FLASH存储器256k，但考虑到以后的发展并给后续开发留有余量，将SRAM 和FLASH容量都提高为512KB，提供10M以太接口，通信总线选用IIC总线，虽然RABBIT2000 CPU提供IIC接口，但考虑到IIC总线上所接设备较多，而RABBIT2000 CPU IIC接口的驱动能力有限，所以IIC总线控制器选用Philips公司的PCA9564，同时系统也提供串口登录方式，串行通信控制芯片为Maxim公司的RS232 ice。

3.1.1 Rabbit2000处理器

Rabbit2000是美国Rabbit semiconductor公司生产的8位处理器，100管脚封装，其中8根数据线，20根地址线，4个串行输入输出口，通信速率可达7,375,000bps，5个并行8位I／O口，1个从属口，具有TimerA、TimerB两类定时器，主频最高可达30MHz，工作电压2.7～5.0V，低功耗。Rabbit2000有8位外部数据总线和8位内部数据总线，由于充分利用了8位外部数据总线，同时又具有紧凑指令集和进行16位运算，其性能完全不亚于16位和32位处理器[7]，图3-1为RABBIT2000处理器系统结构。
RABBIT2000处理器系统结构。

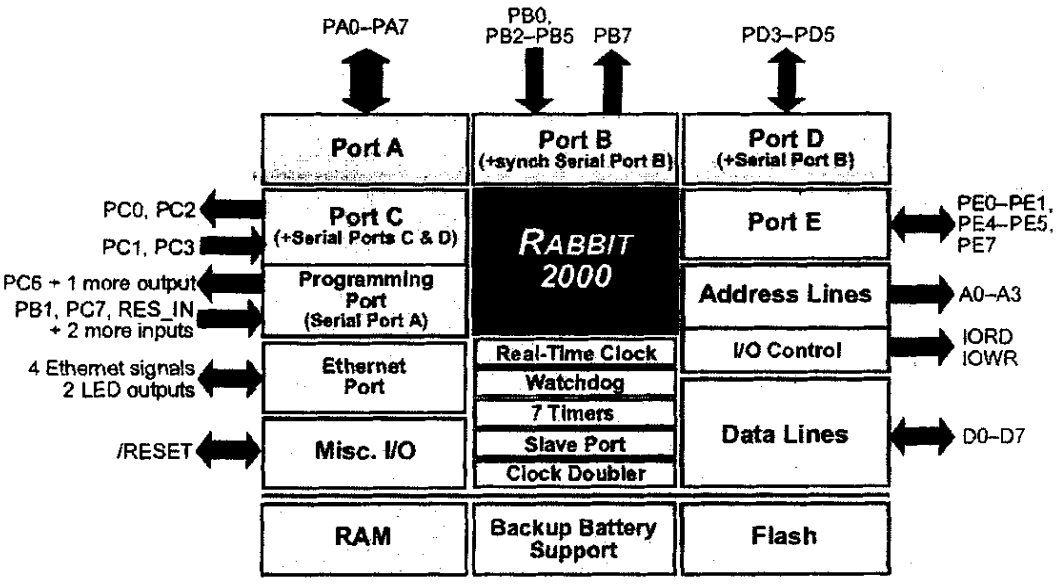

图3-1 RABBIT2000处理器系统结构图[2]

3.1.2 RAM、FLASH存储器和Ethernet网协议控制器

Rabbit2000可寻址1Mega以上地址，根据系统规模并考虑到以后发展的需要，我们选择SRAM和FLASH存储器各512KB。Ethernet网协议控制器选用RTL8019AS。

3.1.2.1 SRAM

SRAM选用SAMSUNG公司的K6T4008C1B-VF70，512K×8bit。该芯片具有如下特点：[3]

- 采用存储技术TFT；
- 运行电压4.5～5.5V，低数据保持电压2V；
- 三态输出，兼容TTL电平；
- 读写访问周期：70ns。

3.1.2.2 FLASH

Flash 选用SST29VF512，512K×8bit。该芯片具有如下特点：[3]

- 统一读写电压，4.5～5.5V；
- 高可靠性；

- 低功耗；
- 统一扇区大小128字节；
- 快速读取周期 70ns。

3.1.2.3 RTL8019AS

RTL8019AS是由台湾睿昱(Realtek)公司生产的一种高度集成的以太网控制器。具有以下特点：[4]

- 适应于EthernetII、IEEE802.3 、10Base5、10Base2、10BaseT；
- 与NE2000 兼容，支持8 位、16 位数据总线；
- 全双工，收发可同时达到10Mbps 的速率，具有睡眠模式，以降低功耗；
- 内置16KB 的SRAM，用于收发缓冲，降低对主处理器的速度要求；
- 可连接同轴电缆和双绞线，并可自动检测所连接的介质；
- 100 脚的TQFP 封装，缩小PCB 尺寸。

3.1.3 IIC总线控制器PCA9564

PCA9564是philips公司的IIC总线控制器。具有如下特点：[5]

- 工作电压2.3 V - 3.6 V，可兼容5V；
- 具有360K的快速通信方式；
- 并行接口速率高达50Mhz；
- 内部时钟源 。

3.1.4 MAX232—ce

MAX232—ce是maxim公司的芯片。具有如下特点：[6]

- 适用于EIA/TIA-232E ，满足 V.28/V.24标准；
- 2路rs-232驱动和接收；
- +5V工作电压。

3.1.5 系统组成

器件选定以后，系统的组成见图3-2。

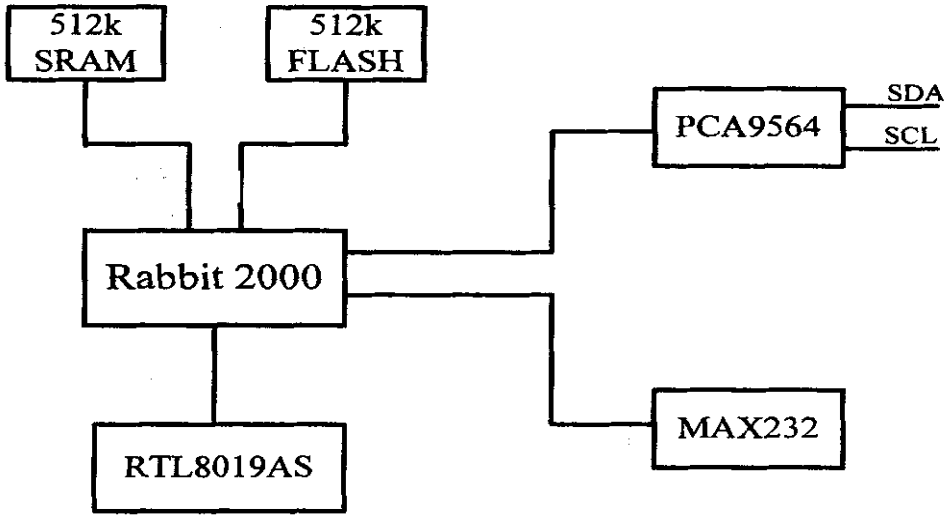

图3-2 硬件组成结构框图

3.2 电路设计

3.2.1 RCM2200接口电路设计

因为网管板是插在机筐中运行，除Ethernet口直接从网管板的前面板引出外，其它输入输出接口均与5U机筐的背板相连。信号线为SCL、SDA两条IIC信号线，还有电源输入和地线，图3-3是网管板与背板的接口电路图[7]。

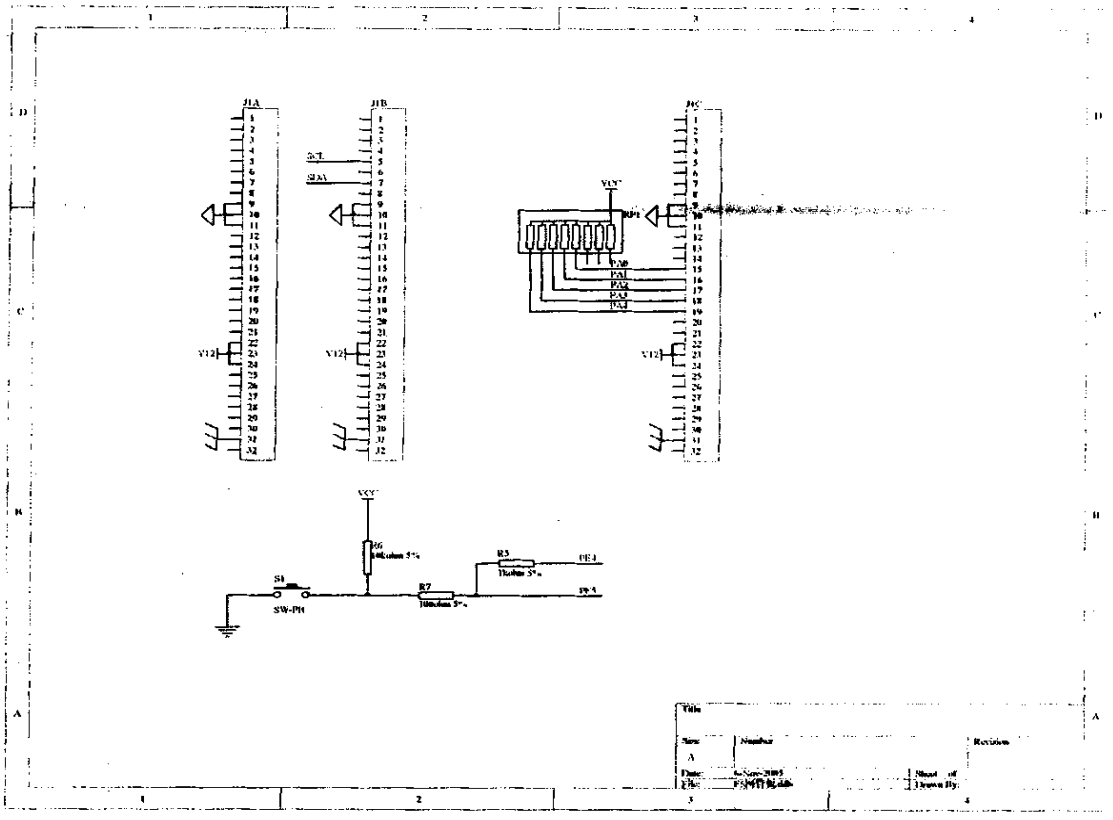

图3-3 网管板与背板接口图

3.2.2 RCM2200接口电路设计

RCM2200上的接口和信号线需要引出，包括Ethernet接口、网管板运行指示指示灯、Ethernet传输指示灯、告警灯的引出电路和RS－232的接口电路。具体信号定义及连接如图3-4。

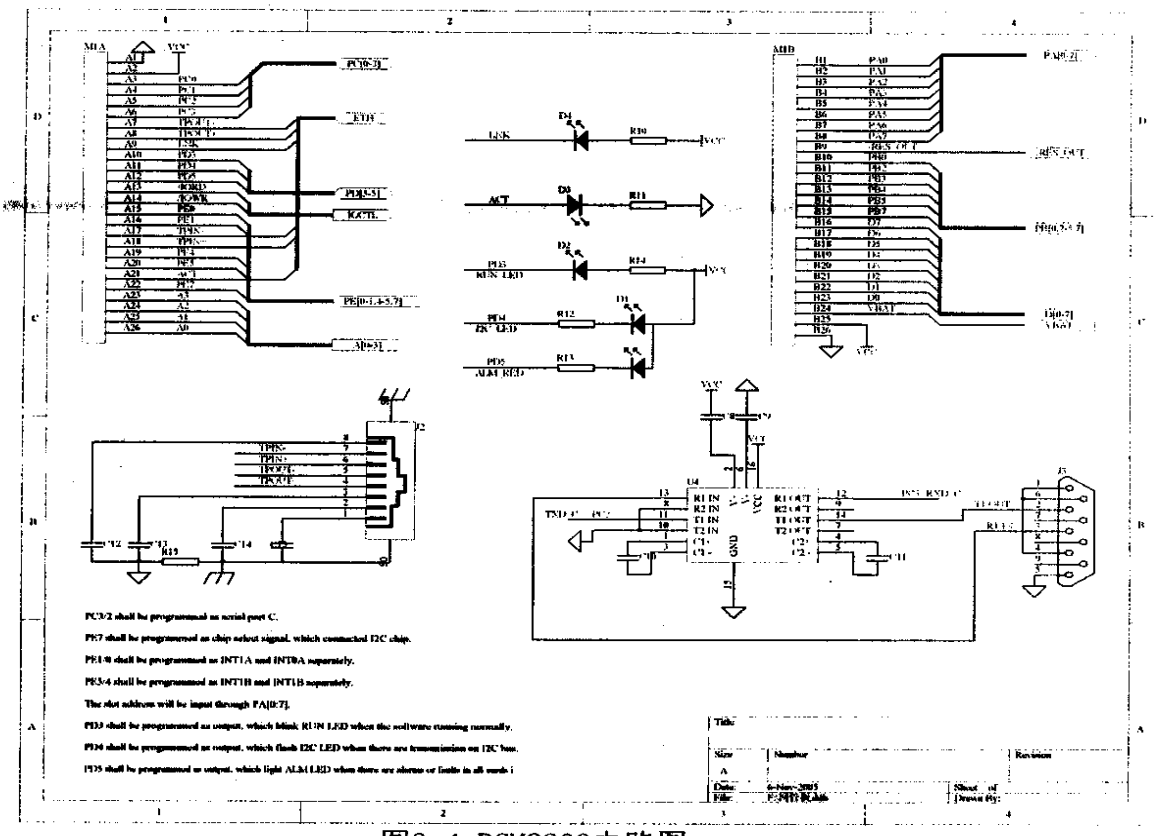

图3-4 RCM2200电路图

3.2.3 IIC电路图

PCA9564通过8根数据总线与RABBIT2000相连，同时通过SCL和SDA与其它电路板通信。PCA9564的各引脚连接如图3-5。

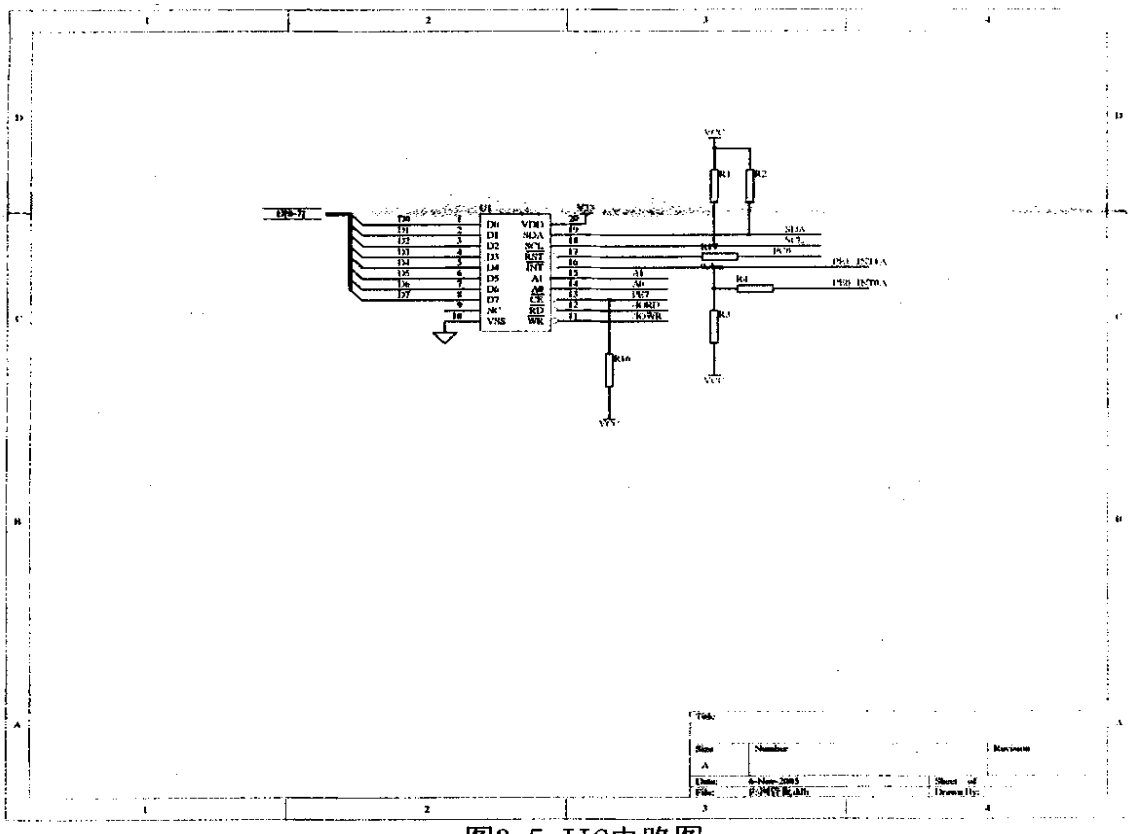

图3-5 IIC电路图

3.2.4 电源电路图

电源电路完成网管板的电源引入和分布，同时相关电路实现整流和滤波，从而提供稳定可靠的电源，如图3-6。

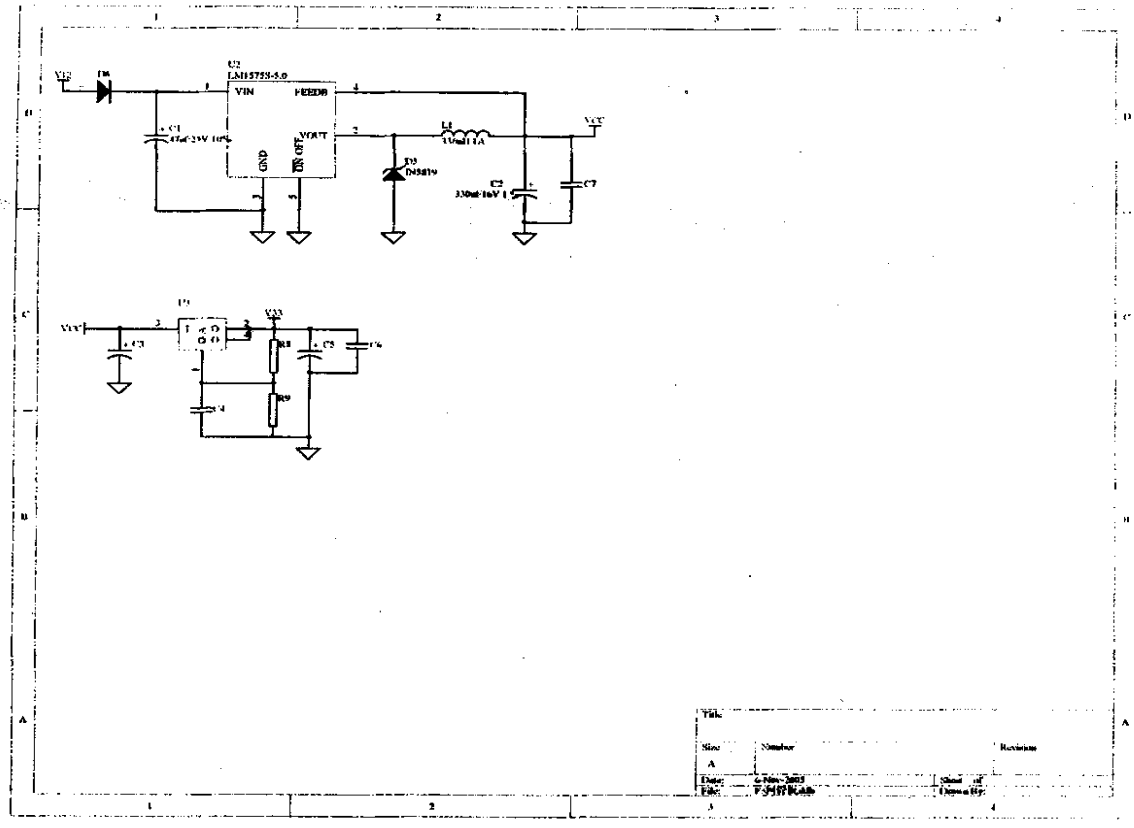
VIN
FEEDB
GND
VOUT
ON/OFF
VCC
Title
Size
Number
Revision
A
Date:
Sheet of

图3-6 电源电路图

第四章 软件设计

根据系统的需求分析，本系统的软件需要实现一个Web服务器，并能与其它电路板通信，获取电路板的实时工作状态以及配置电路板的相关参数，同时管理线卡的告警信息。系统采用多任务调度，实现任务调度要采用操作系统。软件开发平台采用Rabbit semiconductor 公司提供的Dynamic C开发环境，前台脚本程序采用HTML、SHTML语言和Javascript实现。

4.1 Dynamic C开发环境

RABBIT提供基于Windows 操作系统下的开发环境Dynamic C及相关配套开发工具，我们开发是选用Dynamic C 7.20版本。Dynamic C提供一套完整的程序编辑、编译、调试以及下载功能，其中的C语法与ANSI C基本相同，但也有其增加的强大功能，通过专用的诊断（diag）连接器以高达115Kb的速度下载编译后的bin文件，不用专用的FLASH烧写工具，调测极为方便[x]。

Dynamic C的开发环境的主要特点如下：

- 函数链（function chaining）功能，允许一段代码插入到一个和多个函数，函数执行时，相关程序段则被执行。
- 合作状态（costatements）功能，在一个主程序中模拟并发并行进程的执行的情况。
- 合作函数（cofunction）功能，在一个主程序中提供合作进程执行的情况。
- 时间片（slice）功能，在一个程序中提供抢占式时间片功能。
- 支持嵌入到C语言中的汇编代码和独立的汇编代码。
- 提供共享（shared）和保护（protected）关键字，可以实现变量的更好保护与安全访问。
- 通过MMU（Memory Management Unit）充分使用扩展内存。

从上述特征来看，Dynamic C提供的功能是强大的，非常适合开发嵌入式系统。典型的调测连接如图4-1。

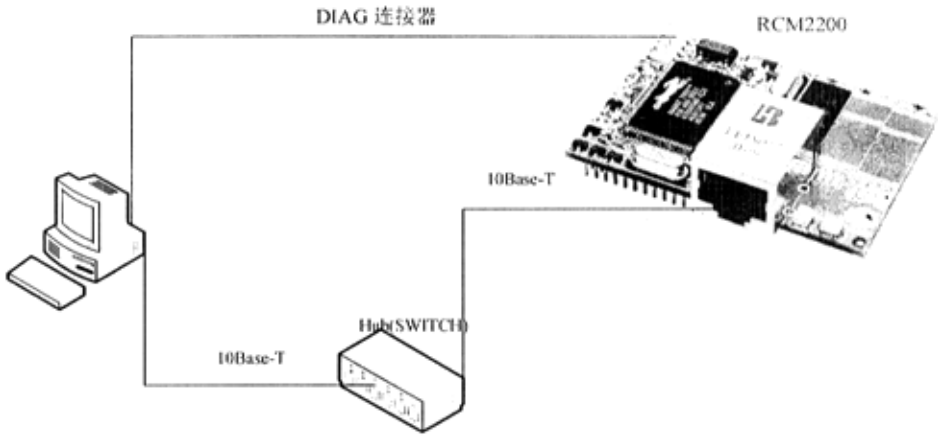

图4-1 Dynamic C调试环境连接图

4.2 操作系统的选择

4.2.1 合作状态（costatements）

合作状态是Dynamic C对C语言的扩展，可以实现状态机的简单执行，合作状态是合作式的，执行的任务自愿被挂起和唤醒，合作状态的主体是一组有序操作（任务）表。每一个合作状态都有自己的指针指到当前需要执行的单元。系统初始化时，每个合作状态的指针指到其第一个单元[M]。典型的调用情况如下：

```
while(1){
    costate{ ...} // task 1

    costate{        // task 2
            waitfor( buttonpushed() );
            turnondevice1();
            waitfor( DelaySec(60L) );
            turnondevice2();
            waitfor( DelaySec(60L) );
            turnoffdevice1();
            turnoffdevice2();
    }
```

```
costate{ ...} // task n
    }
```

4.2.2 合作函数（cofunction）

和合作状态一样，合作函数用于实现合作多任务执行，不同的是有一种类似于函数的形式，可以传给它的变量数值，也可以返回一个数值，但返回值不能为结构体[8]。

4.2.3 时间片（slice）

按定义的时间片各costate语句中的函数轮流执行，每一时间片有自己的栈空间和执行时间，定时器中断每次发生时检测是否有时间片到达，如到达对应时间片的函数则让出CPU，下一时间片获得CPU运行，时间片之间可实现合作式或抢占式调度[8]。

4.2.4 μC/OS-Ⅱ操作系统

虽然Dynamic C提供了合作式和抢占式的调度方式，但本系统的多个任务之间的优先级是不同的且任务相对较多，若要实现多任务之间的优先级调度，保证各任务之间正常切换，及时响应优先级高的任务，还应采用功能更强大的实时操作系统完成任务调度。

由于本系统采用的是RABBIT2000处理器，系统的资源较少，所以选用比较小的操作系统比较适合。μC/OS-Ⅱ是内核较小的操作系统。其特点如下：

- 公开源代码 源代码清晰易读，组织合理，注释清楚。
- 可移植性强 绝大部分代码使用ANSI C，与处理器相关的极少部分用汇编编写，移植工作量相对较小。
- 可裁剪 根据使用的功能不同，可裁剪系统的相关功能。
- 抢占式 保证优先级最高的任务优先执行。
- 多任务 系统可以最多调度64个任务，但系统保留了8个任务，应用程序最多只有56个优先级不同的任务。
- 可确定性 函数的执行时间是可确定的。
- 任务栈 每个任务都有自己的栈空间。

- 中断管理 最多可有255级中断嵌套，足够满足中小系统的需求。
- 稳定性与可靠性 从μC/OS到μC/OS-II，其成功的商业应用足以说明该系统的稳定和可靠。

下面具体分析系统的内核。

4.2.4.1 临界区

操作系统中称一次仅允许一个进程访问的资源为临界资源，进程中访问临界资源的那段代码称为临界区。显然，为了实现进程互斥访问临界资源，诸进程不能同时进入自己的临界区。

μC/OS-II中定义了两个宏用于实现进入和退出临界区：[9]

OS_ENTER_CRITICAL()：进入临界区

OS_EXIT_CRITICAL()：退出临界区

其实OS_ENTER_CRITICAL()完成关中断的作用，OS_EXIT_CRITICAL()完成开中断的作用。典型的调用如下：

```
INT32U Val;
void TaskX(void *pdata)
{
  for (;;) {
  OS_ENTER_CRITICAL(); /* 关中断 */
  .
  ./* 访问临界区 */
  .
  OS_EXIT_CRITICAL(); /* 开中断 */


  }
}
```

4.2.4.2 任务（task）和任务控制块（TCB）

在操作系统之上执行的一个应用程序称为一个任务，对应每一个任务有一唯一的TCB标志任务的存在，记录其执行情况和相关资源的分配及使用情况。

(1) 任务（task）

嵌入式系统中任务和通用操作系统的不同之处是任务一经启动，永远处于执

行和等待执行中，没有通用操作系统中常见的返回值，典型格式如下：

```
void Task(void)
{
  for (;;) {
  ./* 等待某一信号发生 */
  ./* 执行本任务相关的操作 */
  .
  ./* 清除某一信号，让出CPU */
  }
 }
```

(2) μC/OS-Ⅱ中任务相关的操作：[9]

OSTaskCreate()：创建一个任务，指定任务执行的函数、栈指针及优先级。

OSTaskCreateExt()：完成和OSTaskCreate()一样的功能，但带有附加信息。

OSTaskSuspend()：无条件挂起（或阻塞）当前任务。

OSTaskResume()：唤醒被阻塞的任务，可指定任务优先级。

OSTimeDlyResume()：唤醒执行了OSTimeDly()或OSTimeDlyHMSM()后被挂起的任务。

OSTaskDel()：删除一指定优先级的任务，也可删除自己，任务一经删除必须OSTaskCreate()或OSTaskCreateExt()创建任务才能再次执行。

OSStart()：用于启动多个任务开始执行。

OSIntExit()：告知μC/OS-Ⅱ一个ISR执行完毕。

OS_TASK_SW()：一个宏，调用后可从一低优先级的任务切换到高优先级的任务。

OSMBoxPost()：通过邮箱发送消息到任务。当任务的邮箱已有一个消息时，发送失败。

OSQPost()：通过队列发送消息到任务，队列为先进先出（FIFO），当队列满时，发送失败。

OSQPostFront()：通过队列发送消息到任务，队列为后进先出（LIFO），当队列满时，发送失败。

OSSemPost()：一个信号量被发送信号。

OSTimeTick()：用于处理一个时钟节拍，操作系统检查所有的任务是否有超时事件发生。

OSMBoxPend()：表示任务希望获得一个消息，当邮箱中有消息时，消息被获

得后清除，否则任务进入挂起状态。

OSQPend():表示任务希望从一个队列中获得一个消息，当队列中有消息时消息被返回，否则任务进入挂起状态。

OSSemPend():表示任务为独享一个资源时，等待信号量，若semaphore>0，则对信号量进行减操作并返回任务，否则进入挂起状态。

OSTimeDly()：任务延时一定的时钟节拍(0~65535)。

OSTimeDlyHMSM()：任务延时一定的时间(时，分，秒，毫秒)。

系统中的任务可根据调度情况处于不同的状态，从而保证任务之间的协调执行，任一时刻只有一个任务处于运行状态，如图4-2。

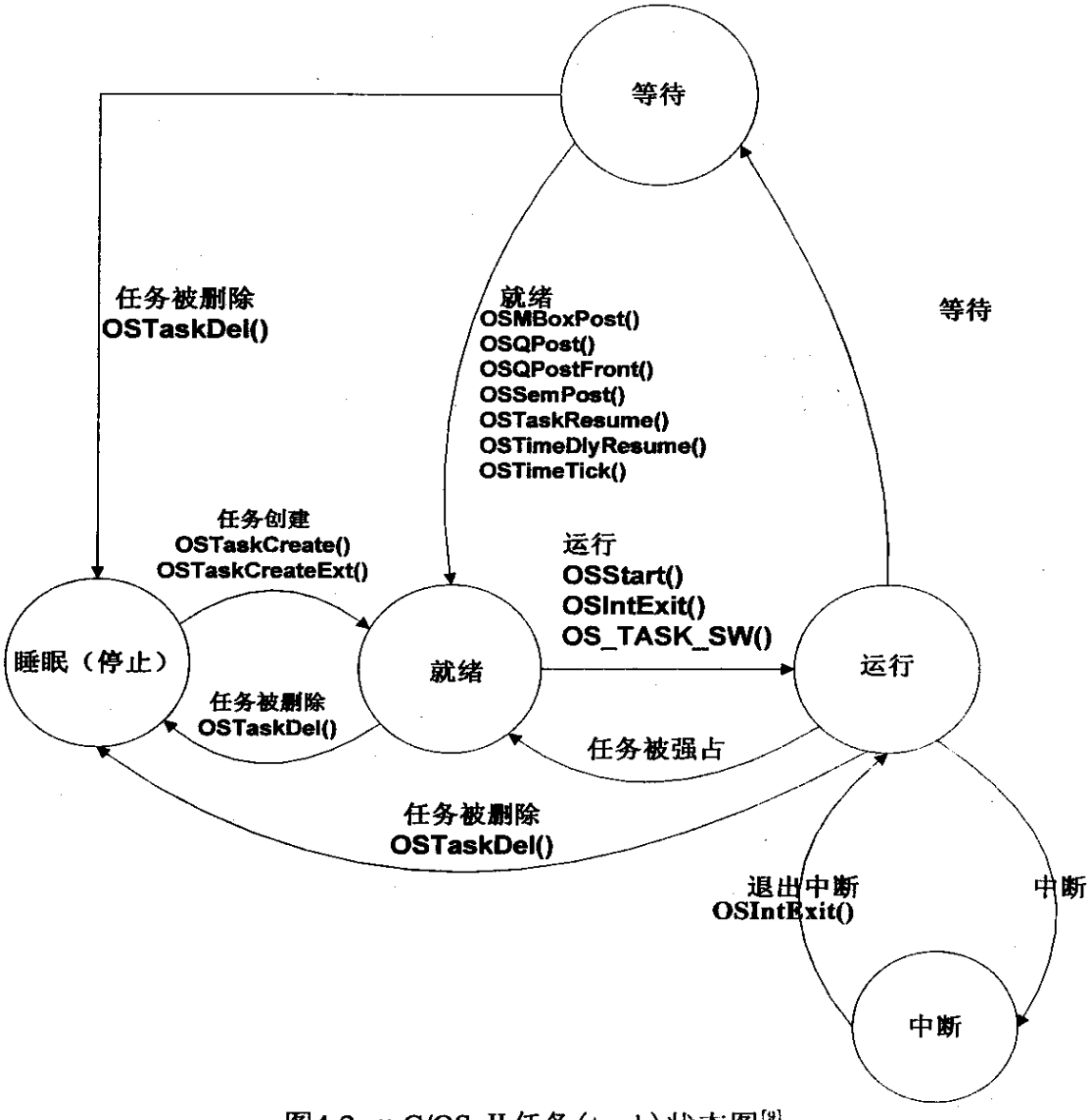

图4-2 μC/OS-Ⅱ任务(task)状态图[9]

(3) 任务控制块（TCB）

任务被创建时，即被分配一任务控制块（Task Control Block）。一个任务控制块就是一个结构体，当一个任务被强占时μC/OS-II用它保持任务的执行信息，当该任务重新获得CPU后，该任务可从其被中断的地方精确执行[5]。任务控制块的定义如下：

```
typedef struct os_tcb {
    OS_STK *OSTCBStkPtr;      //任务的栈顶指针
    #if OS_TASK_CREATE_EXT_EN
        void *OSTCBExtPtr;        //附加用户信息指针
        OS_STK *OSTCBStkBottom; //栈底指针
        INT32U OSTCBStkSize;      //栈的字节单位长度
        INT16U OSTCBOpt; //保存用于传递给OSTaskCreateExt()变量
        INT16U OSTCBId;     //任务的标识
    #endif
    struct os_tcb *OSTCBNext;//指向后一个任务
    struct os_tcb *OSTCBPrev; //指向前一个任务
    #if (OS_Q_EN && (OS_MAX_QS >= 2)) || OS_MBOX_EN || OS_SEM_EN
        OS_EVENT *OSTCBEventPtr;  //指向事件控制块
    #endif
    #if (OS_Q_EN && (OS_MAX_QS >= 2)) || OS_MBOX_EN
        void *OSTCBMsg;     //指向发给本任务的消息
    #endif
    INT16U OSTCBDly;    //该任务延时时钟节拍数
    INT8U OSTCBStat;    //任务的状态
    INT8U OSTCBPrio;    //任务的优先级
    INT8U OSTCBX;       //与下面三项共同用于加速任务的执行或等待
    INT8U OSTCBY;        //某一事件发生
    INT8U OSTCBBitX;
    INT8U OSTCBBitY;
    #if OS_TASK_DEL_EN
        BOOLEAN OSTCBDelReq; //标志该任务是否被请求删除
    #endif
```

```
} OS_TCB;
```

(4) 任务调度

μC/OS-Ⅱ总是运行进入就绪态任务中优先级最高的任务。任务的就绪标志放在就绪表(ready list)中，就绪表中有两个变量OSRedyGrp 和OSRdyTbl[256]，通过此二个变量和优先级判定表OSUnMapTbl[256]来确定执行哪个任务。任务的调度由OSSched()函数完成[9]。

```
void OSSched (void)
{
INT8U y;
OS_ENTER_CRITICAL();
if ((OSLockNesting | OSIntNesting) == 0)
{
  y = OSUnMapTbl[OSRdyGrp];
  OSPrioHighRdy = (INT8U)((y << 3) +OSUnMapTbl[OSRdyTbl[y]]);
   if (OSPrioHighRdy != OSPrioCur)
    {
       OSTCBHighRdy = OSTCBPrioTbl[OSPrioHighRdy];
       OSCtxSwCtr++;
       OS_TASK_SW();
     }
}
OS_EXIT_CRITICAL();
}
```

4.2.5 各类任务调度方法的比较

通过以上分析，可以看出时间片（slice）、合作函数（cofunction）、合作状态（costatements）等方法实现简单，控制容易，但其调度方法单一，无法满足实时系统的较高要求，任务增加时调度复杂。而μC/OS-Ⅱ操作系统不仅可实现最多64个任务的优先级调度，而且是抢占式内核，较好的满足我们系统的实时性要求，所以我们选定μC/OS-Ⅱ操作系统。

4.2.6 μC/OS-Ⅱ移植

移植μC/OS-Ⅱ主要改写与cpu硬件相关的三个文件：OS_CPU.H, OS_CPU_C.C,

OS_CPU_A.S。 OS_CPU.H中主要定义与处理器相关的常数、宏,类型声明。在OS_CPU_C.C中定义了六个简单的函数:

OSTaskStkInit()

OSTaskCreateHook()

OSTaskDelHook()

OSTaskSwHook()

OSTaskStatHook()

OSTimeTickHook()

除OSTaskStkInit()外,其它5个函数只是声明,不用实现。在OS_CPU_A.S中实现了四个函数:

OSStartHighRdy()

OSCtxSw()

OSIntCtxSw()

OSTickISR()

OSStartHighRdy()启动处于就绪态的最高优先级的任务的执行。OSCtxSw()实现任务的切换。OSIntCtxSw()被OSIntExit()调用完成中断现场的切换。OSTickISR()实现对应时钟tick的中断处理程序。

4.3 Web服务器的设计

通常实现一个Web服务器需要很多系统资源,由于我们设计的系统为嵌入式系统,资源有限,而我们只要求实现Web网页浏览网管系统的信息即可,所以我们实现一个瘦Web服务器,即实现TCP/IP底层及高层的相关协议即可[10]。自顶向下需实现的协议如下:

HTTP 网页的请求/应答

TCP 可靠的通信

IP 低层次的数据传输

ICMP 诊断(ping命令)

ARP 地址解析协议

协议栈结构如图4-3。

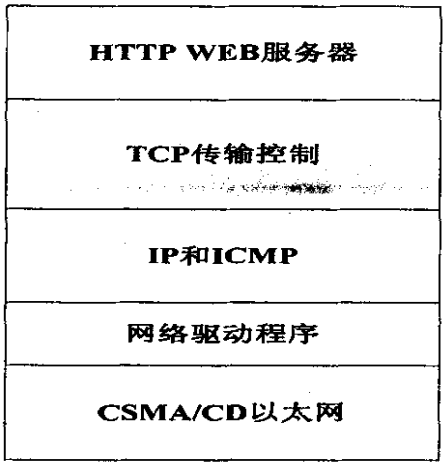

图4-3 Web协议栈

4.3.1 协议原理及实现

4.3.1.1 TCP/IP协议工作原理[11]

对接收到的TCP/IP数据帧进行解包，供应用程序使用，对欲发送的数据进行打包，然后发送出去。

(1) ARP协议

ARP的任务是把IP地址转化成物理地址。ARP解析一个IP地址时，首先搜索ARP cache和ARP表作匹配，若匹配成功，ARP就把物理地址返回给提供IP地址的应用，反之，则向网络上发布ARP请求消息，这个消息被广播到局域网上的每一个设备。ARP请求包括接收设备的IP地址。若一个设备认出此IP地址属于自己，就把包含自己物理地址的应答报文返回给产生ARP请求的机器，ARP请求机器会将此信息存放到ARP表和ARP cache中。

ARP请求和ARP应答报文的格式如图3-5所示，当一个ARP请求发出时，除了接收端硬件地址(请求机想知道的)之外所有域都被使用。ARP应答中，使用所有的域。

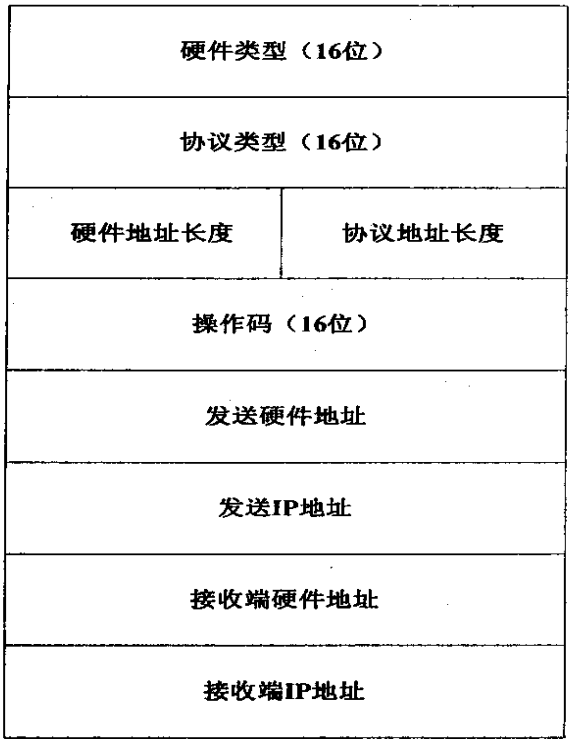

图4-4 ARP 请求和应答报文格式

- 硬件类型

  硬件类型识别硬件接口类型。

- 协议类型

  协议类型标识发送设备所使用的协议类型，TCP/IP中，这些协议通常是EtherType。

- 硬件地址长度

  数据报中硬件地址以字节为单位的长度。

- 协议地址长度

  数据报中所用协议地址以字节为单位的长度。

- 操作码

  操作码（Opcode）指明数据报是ARP请求还是ARP应答，若为ARP请求，此值为1；若为ARP应答，此值为2。

- 发送方硬件地址

  发送方设备的硬件地址。

- 发送方IP地址

  发送方设备的IP地址。

- 接收方硬件地址

接收方设备的硬件地址。

- 接收方IP地址

接收方设备的IP地址。

反向地址解析协议(RARP)以与ARP相反的方式工作，RARP发出要反向解析的物理地址并希望返回其IP地址。

(2) IP协议

IP协议是网际协议，是TCP/IP协议和网络驱动程序之间的低层接口。IP的功能由IP头结构中的数据定义。我们的系统中实现的是IPV4，但要实现IPV6只需修改相关代码即可。

① IP头格式

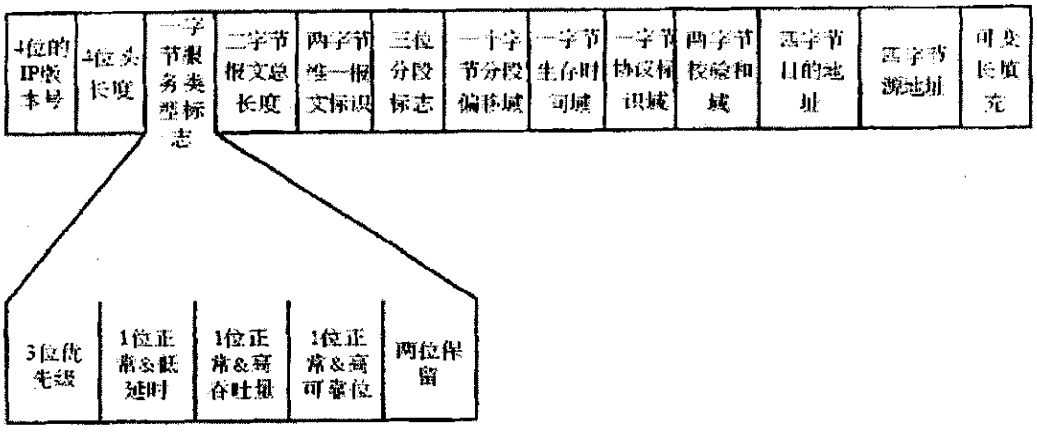

图4-5 IP头格式

- 版本

IP头中前四位标识了IP的操作版本。

- Internet头长度

头中下面4位包括头长度，以32位为单位表示。

- 服务类型

下面的一个字节包括一系列标志，这些标志能保证优先级、延时、吞吐量以及报文数的可靠性参数。。

- 总长度(Total Length)

报文总长度，16位域，长度以字节为单位。

- 标识(Identifier)

每个IP报文被赋予一个惟一的16位标识，用于标识数据报的分段。

- 分段标志(Fragmentation Flag)

  下一个域包括3个1位标志，标识报文是否允许被分段和是否使用了这些域。

- 分段偏移(Fragment Offset)

  指出分段报文相对于整个报文开始处的偏移。

- 生存时间(TTL)

  限制IP报文的转发次数。TTL在经过每一跳时加1。在到达它的最大值之后，报文就被认为是不可转发的。之后产生一个ICMP报文并发回源机器，不可转发的报文被丢弃。

- 协议

  指示IP头之后的协议。

- 校验和(checksum)

  校验和是16位的错误检测域。

- 源IP地址

  源计算机的IP地址。

- 目的IP地址

  目的计算机的IP地址。

- 填充

  为了保证IP头长度是32位的整数倍，要填充额外的0。

这些头域说明IPv4的网际层是无连接的：网络中的转发设备可以自由决定通过网络的报文的理想转发路径。它也不提供任何上层协议如TCP所提供的应答、流控、序化功能。IP也不能用于引导IP报文中的数据到正确的目的应用程序。这些功能留给上层协议实现。

② IP功能

- 寻址和路由

  IP的主要功能是使报文送到特定目的地。连接源和目的地网络中的路由器和交换机使用目的IP地址确定经过网络的最优路径。

- 分段和重组

  当应用数据的一段不能完全放在一个IP报文中时，它们必须分段成两个或更多的报文。当分段发生时，IP必须能重组报文。

- 损坏报文补偿

  检测和补偿在传输过程中遭到破坏或丢失的报文。

(3) ICMP协议

ICMP是网际控制消息协议，为控制、测试、管理功能而设计的多层协议。各种ICMP协议从主机到主机层延伸至进程/应用层，它是IP的一个附件，用于帮助网络上的所有节点实现简单的诊断并返回错误消息。图4-6为ICMP报文头。

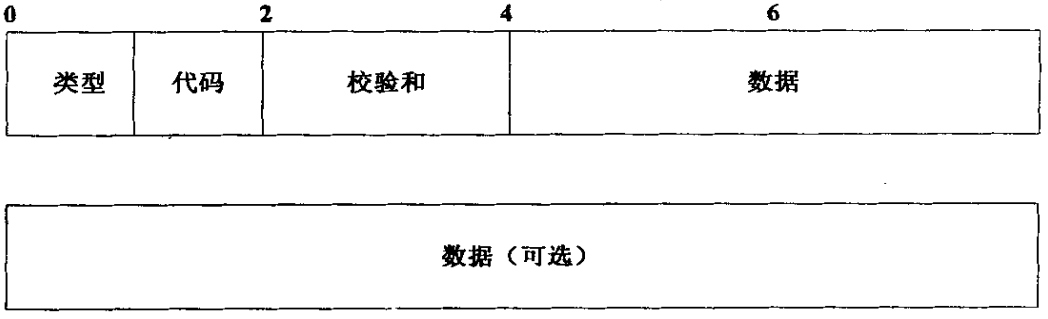

图4-6 ICMP报文头

消息长度因消息类型而异，但最小值为8字节。类型和代码字段表示需要的ICMP操作，最常用的值如下：

代码 0 网络不可达

类型 0 echo 应答

3 目标不可达

8 echo请求

1 主机不可达

2 协议不可用

4 端口不可用

5 需要进行分段但不允许

6 目标网络未知

7 目标主机未知

我们的系统中实现ICMP协议主要是为了实现ping命令，允许Web应答ping。

(4) TCP协议

TCP是传输层协议（ OSI参考模型中第四层），它使用IP，提供可靠的应用数据传输。TCP在两个或多个主机之间建立面向连接的通信。TCP支持多数据流操作，提供流控和错误控制，甚至完成对乱序到达报文的重新排序。

① TCP头结构

TCP的头结构如图4-7。

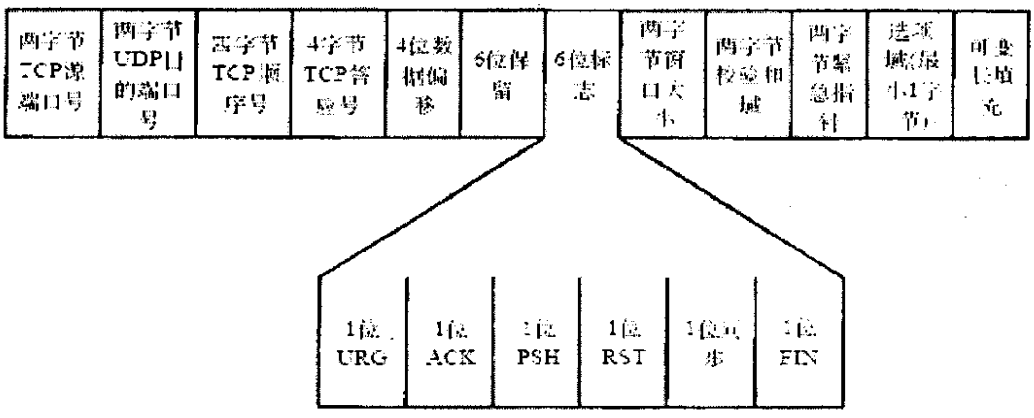

图4-7 TCP头格式

TCP协议头最少20个字节，包括以下各域：

- TCP 源端口

  初始化通信的端口号。

- TCP目的端口

  传输的目的，这个端口指明报文接收计算机上的应用程序地址接口。

- TCP序列号

  接收端计算机用于重组分段的报文成最初形式。

- TCP应答号

  标识下一个希望收到的报文的第一个字节。

- 数据偏移

  TCP头大小。

- 保留

  保留位。

- 标志

  对应紧急标志、有意义的应答标志、推、重置连接标志、同步序列号标志、完成发送数据标志的控制。

- 窗口大小

  目的机想收到的每个TCP数据段大小。

- 校验和

  源主机基于数据内容计算的校验和，用于接收时校验。

- 紧急

  为紧急时，要加快处理此数据段。

- 选项

标识哪个选项有效。

- 数据

应用数据段。

- 填充

保证TCP头是32位的整数倍。

② TCP作用

- 多路复用数据流

TCP是用户应用与许多网络通信协议之间的接口。TCP通过端口区分同时接收和发送的多个应用数据。套接字由驻留在主机上的特定应用端口号和机器IP地址联合构成。

- 测试数据的完整性

TCP通过数学计算产生校验和，数据到达目的地，对接收数据执行相同的数学计算，产生的结果应该和存储的结果相同，若二者相同，则认为传输正确，否则，要求重发数据。

- 重新排序

重新排序经过不同路径到达的先后顺序不正确的报文。

- 流控

流量控制使用的是TCP窗口大小。源和目的机的窗口大小通过TCP头进行通信。任何一台主机将被所收数据淹没时，会减小发送机的速率。

- 计时机制

传输数据段时，设置一个计时器，超时则重传。计时器可以间接地管理网络拥塞，其方法是当超时出现时减慢传输率。

- 应答接收

若ACK被设置，目的TCP机器必须要对接收到的数据作出应答。

(5) HTTP协议

HTTP协议是TCP/IP协议族的高层应用协议，它使Web服务器和浏览器可以通过Web交换数据。它是一种请求/响应协议，即服务器等待并响应客户方请求。HTTP使用可靠的TCP连接，通常采用80端口，当然也可以定义为其它端口号。客户/服务器传输过程可分为四个步骤：

1) 浏览器与服务器建立连接；
2) 浏览器向服务器请求文档；
3) 服务器响应浏览器请求；
4) 断开连接。

HTTP是一种无状态协议，它不维持连接的状态信息。

系统中拟实现HTTP/1.1版本。

① **HTTP/1.1规范**

- 客户请求

  客户请求包含以下信息：

  - 请求方法
  - 请求头
  - 请求数据

请求方法适用于特定URL或Web页面的程序，表4-1列出了请求方法。

表4-1 请求方法

方法	描述
GET	请求指定的文档
HEAD	仅请求文档头
POST	请求服务器接收指定文档作为可执行的信息
PUT	用从客户端传送的数据取代指定文档中的内容
DELETE	请求服务器删除指定页面
OPTIONS	允许客户端查看服务器的性能
TRACE	用于测试—允许客户端查看消息回收过程

头信息是可选项，它用于向服务器提供客户端的其他信息。请求头如表4-2。

表4-2 请求头

头	描述
Accept	客户端接收的数据类型
Authorization	认证消息，包括用户名和口令
User-agent	客户方软件类型
Referer	用户获取的Web 页面

•服务器响应

包括以下关键部分：

- 状态码
- 响应头
- 响应数据

HTTP定义了多组返回给浏览器的状态码。表4-3列出了这些状态码信息。

表4-3 HTTP返回给浏览器的状态码

状态类及代码	描述
客户方错误(1xx)	
100	继续
101	交换协议
成功（2xx）	
200	OK
201	已创建
202	接收
203	非认证信息
204	无内容
205	重置内容
206	部分内容
重定向（3xx）	
300	多路选择
301	永久转移
302	暂时转移
303	参见其它
304	未修改
305	使用代理
客户方错误（4xx）	
400	错误请求
401	未认证
402	需要付费
403	禁止
404	未找到
405	方法不允许
406	不接受
407	需要代理认证
408	请求超时
409	冲突
410	失败

411	需要长度
412	条件失败
413	请求实体太大
414	请求URI太长
415	不支持媒体类型
服务器错误（5xx）	
500	服务器内部错误
501	未实现
502	网关失败
504	网关超时
505	HTTP版本不支持

响应头向客户方提供服务器和/或请求文档的信息，所有的头均以空行结束。表4-4列出了所有头信息。

表4-4 HTTP响应头方法描述

方法	描述
Server	Web服务器信息
Date	当前日期/时间
Last Modified	请求文档最近修改时间
Expires	请求文档过期时间
Content-length	数据长度(字节)
Content-type	数据MIME类型
WWW-authenticate	用于通知客户方需要的认证信息

如果有客户方请求的数据，数据放在响应头之后，否则服务器断开连接。

② MIME与Web

多用途互联网邮件扩展（ MIME ）在Web中用于指定大量数据的类型。MIME使用户可发送多种格式的数据，而不单单是文本数据。由于使用了MIME，用户可以发送和接收包含非ASCII码数据如声音、视频、图像应用等的页面。

当Web浏览器与服务器建立连接时，它们协商MIME类型。浏览器向服务器发送它所能接收的MIME类型，这部分信息位于请求头标中。服务器通知客户方它

发送的数据包含的MIME类型。

表4-5列出了Web中常见的MIME类型。

表4-5 常见MIME类型

MIME类型	描述
text/plain	纯ASCII码文本
text/html	HTML文本
image/gif	GIF图像
image/jpeg	JPEG图像
application/msword	Microsoft Word
video/mpeg	MPEG视频
audio/wave	Wave 音频
Application/x-tar	Tar 压缩数据

4.3.1.2 协议栈实现

因为系统中只需实现相关的协议，相关协议的实现可放在相应的库文件中，ICMP协议的实现放在icmp.lib中，ARP协议的实现放在arp.lib中，TCP协议的实现放在tcp.lib中，ip协议的实现放在ip.lib中，http协议的实现则放在http.lib中，在主程序中引用相关的库即可实现相应的协议功能。例如应用HTTP协议时可如下引用：#use http.lib。

TCP/IP协议工作原理如图4-8，分析了相关协议的工作原理后，在编写CGI程序时也就相对简单，可以综合运用协议的各个工作方法，例如：可分不同情况使用HTTP协议的POST和GET请求方法实现环境变量传递，从而使代码的效率更高。

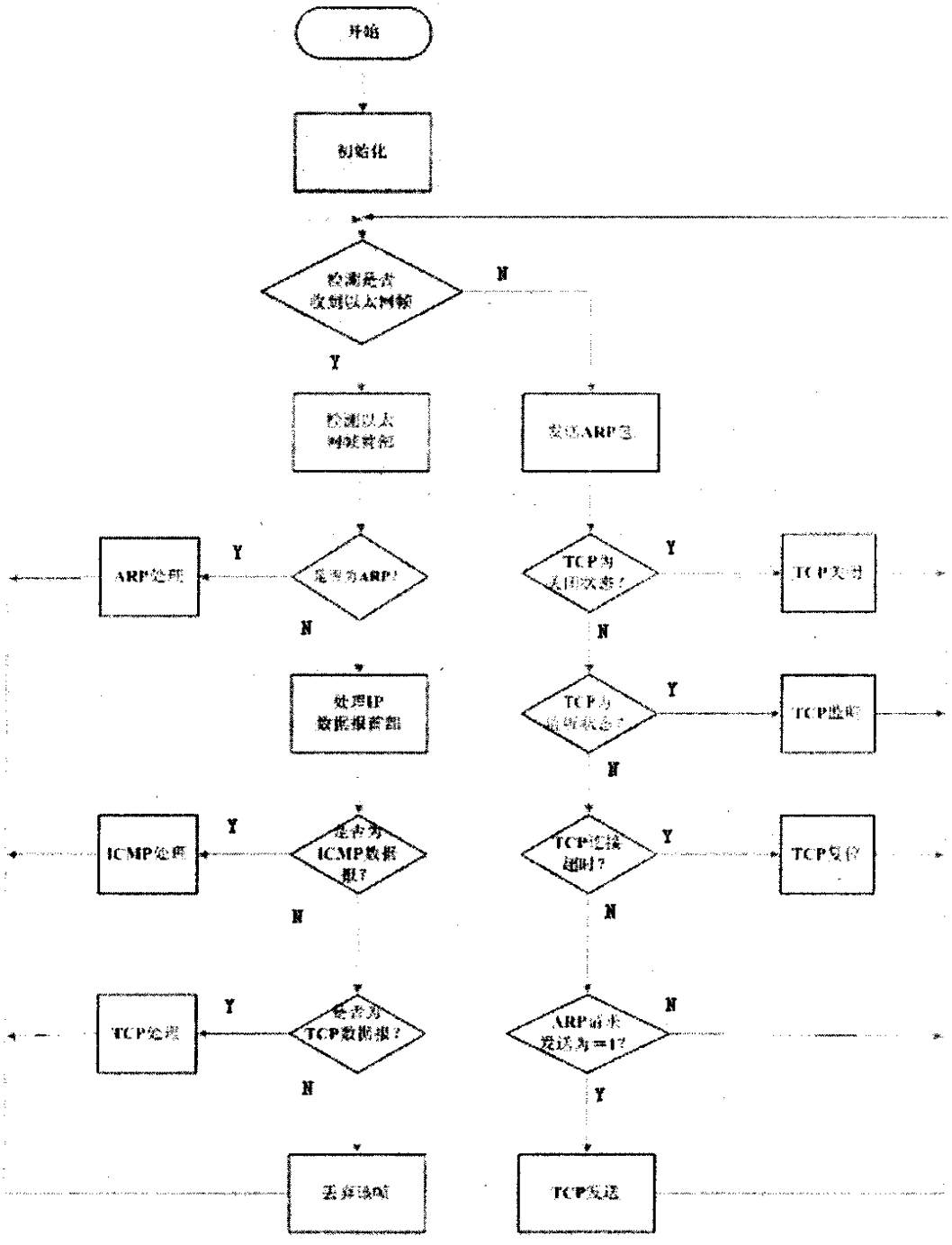

图4-8 TCP/IP协议栈流程图

4.3.2 SERVER设计

基于RABBIT2000的Web可设计如图4-9结构。

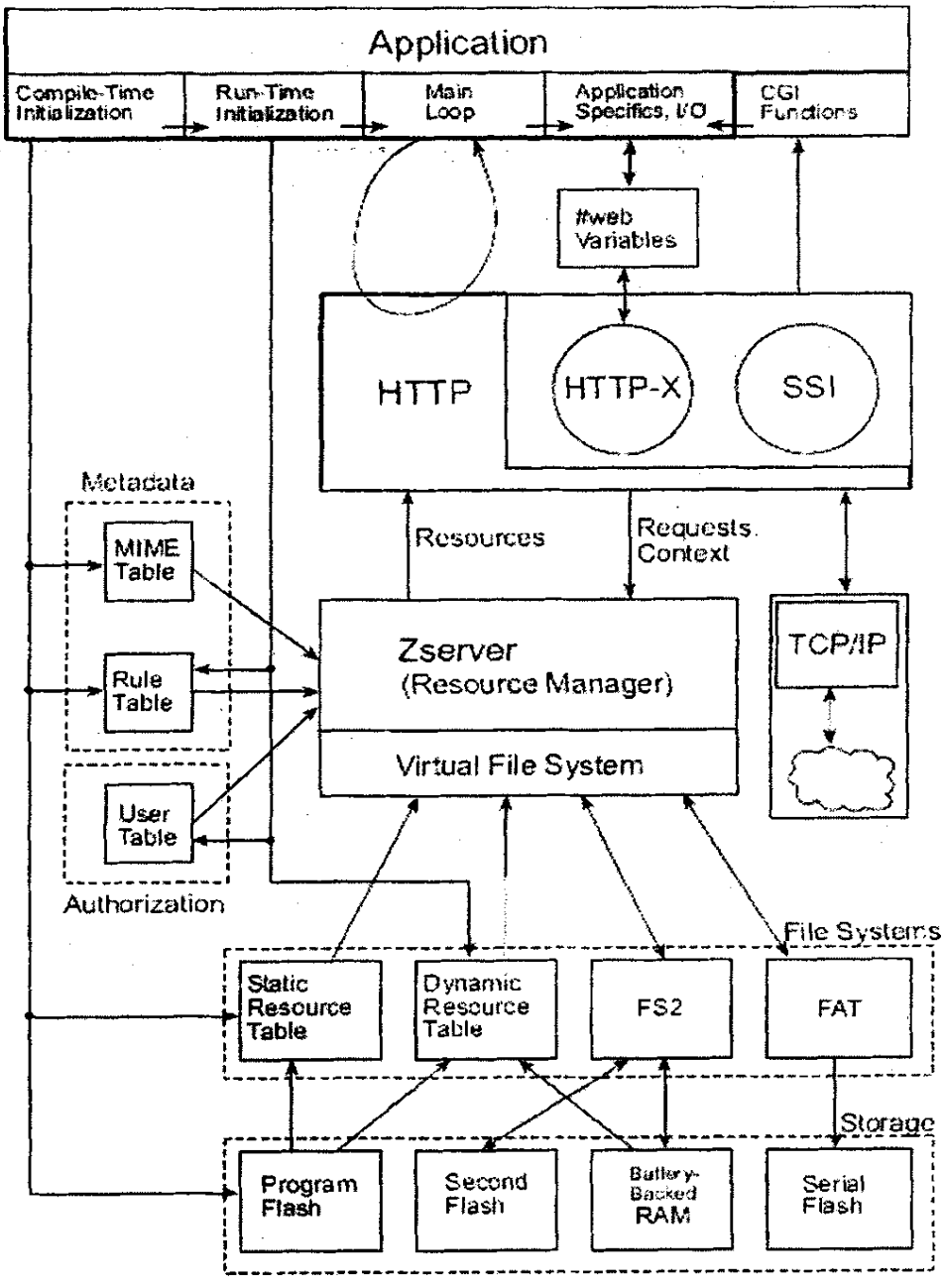

图4-9 Web组件结构图[12]

4.3.2.1 应用程序

(1) 编译时初始化参数

编译时即可选择的项目，如静态数据结构和表，网络参数。

① MIME（多用途互联网邮件扩展）类型映射表

此表是必须的，告诉浏览器所提供内容的类型，其结构定义如下：

```
typedef struct {
    char extension[10];
    char type[HTTP_MAXNAME];
    int (*fptr)(/* HttpState* */);
    } MIMETypeMap;
```

如下为通常Web所需定义的MIME表：

```
const HttpType http_types[] =
{
   { ".shtml", "text/html", shtml_handler }, // ssi
   { ".html", "text/html", NULL },
   { ".gif", "image/gif", NULL },
   { ".cgi", "", NULL }
};
```

② 规则（rule）表

在使用文件系统时用来规定文件的存取规则。

③ 静态（static）资源表

定义系统资源的一种方法，该方法是可选的。

④ Flash 编程

表示通过#ximport将资源文件存到Flash中。

(2) 运行时初始化参数

主程序运行时需首先调用某些库函数。

1. sock_init()：必须调用，初始化TCP/IP网络系统。
2. sspec_automount()：可选调用，初始化Zserver所用文件系统。
3. http_init():必须调用，初始化HTTP server。
4. 其它设置用户标识（user ID）和动态资源的函数，可选调用。

(3) 主循环（main loop）

http_handler()不停的被调用，使得HTTP server能够处理网络请求。

(4) 应用规范和接口（Application Specific and I/O）

应用程序，也可以称为后台程序。它们可以和HTTP server以诸多方式通信，可以直接调用资源管理器中的函数。

(5) CGI(公共网管接口)函数

CGI（Common Gateway Interface）的主要功能是在WWW环境下，实现从客户端传递信息给WWW Server，再由WWW Server去启动所指定的程序代码来完成特定的工作。CGI仅是在WWW Server上可执行的程序码，而它的工作就是控制信息要求而且产生并传回所需的文件。使用CGI，Server可以读取并显示在客户端无法读取的格式（如：SQL Data Base），在服务器端和客户端之间，产生客户端所需要的信息。基本上，在主/从（Client/Server）环境下，其IPC（Inter Process Communication）协议是利用信息传递及存储器共享（环境变量）的方式来完成。CGI有其特定的语法规则，必须遵守其语法规则，才能实现主从端信息交互的目的。

为了实现HTTP server的诸多后台处理，我们需要编写许多CGI函数。

4.3.2.2 HTTP功能块

HTTP server需要监视外部请求，分析其请求的资源是否合法以及是否允许其访问，若合法且允许其访问，则将资源返回给浏览器。SSI（服务器服务包含）是一种脚本语言，可实现动态包括资源返回给请求者。HTTP server通过TCP/IP管道与外部通信。

(1) 分析请求

获取包含在URL（同一资源定位器）中资源名称，URL就是要获取的资源的名称。

(2) 获取用户标识（user ID）

浏览器可以发送用户名和密码，否则则为匿名登录。收到用户名和密码后，HTTP server 核实是否正确，若正确则确定该用户的组（group），然后按照该组的访问权限确定用户请求的资源是否允许访问。

(3) 返回资源

HTTP server验证了组的访问权限后，即可返回资源。资源可以为HTML文件、脚本文件、CGI函数等。

4.3.2.3 Zserver功能块

Zserver即为一个资源管理器，是整个系统的交换中心，通过它可以访问系统的其它资源，它提供对设备和文件的统一接口。完成Zserver的功能需要相关数据，元数据（Metadata）和授权（Authorization）数据。元数据即MIME表和规则表，授权数据即用户标识表。

4.3.3 Web工作流程

系统采用html、shtml文件实现Web网页显示，用户在Web网页上的输入用Form的形式提交给http server，后台调用相关CGI程序实现Web服务，同时在html和shtml网页文件中使用Javascript脚本语言实现程序的一些控制和错误检查，在服务器端采用SSI技术实现动态网页。CGI程序和所有后台程序均采用Dynamic C编译开发环境，同时嵌入了一些汇编代码，以提高代码的效率。Dynamic C的语法与ANSI C基本相同。Web服务器—CGI原理图如图4-10。

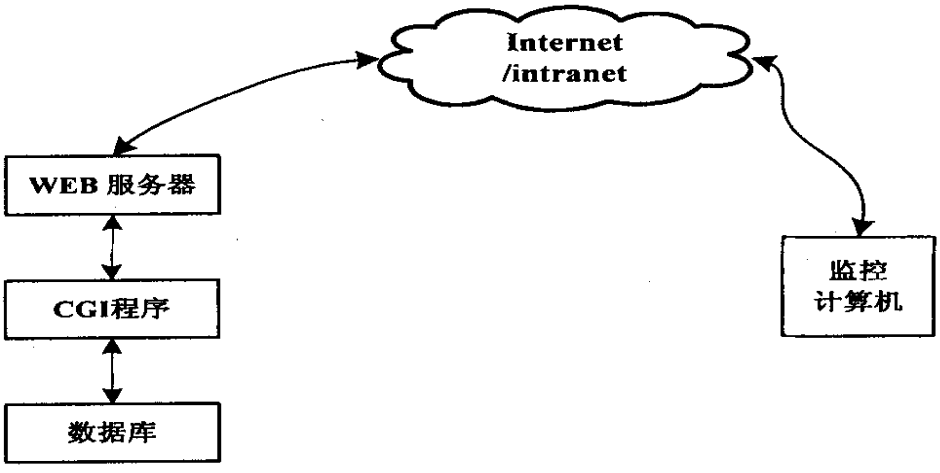

图4-10 Web服务器—CGI原理图

4.4 文件系统

作为嵌入式系统，其存储空间资源有限，但是为了保证系统的正常运行和浏览器访问Web时数据存取的及时性，必须采用数据管理系统来管理数据的读写，这样可以规范系统的数据管理，降低了管理数据的麻烦和不安全，从而优化整个Web Server的性能[13]。为此在系统中采用FS2文件系统来管理系统中的相关数据，该文件系统最大支持255个文件，每个文件要求至少两个扇区，一个用于存放文件数据，另一个用于存放索引控制信息，文件最大长度为200Kbytes，文件系统是不可重入的，多任务执行时，应使用临界区互斥操作进行同步操作。主要特点如下：

- 文件的部分读写
- 多种设备类型并发应用
- 支持分区设备

- 高效率支持字节可写设备
- 高性能调节
- 向后兼容性

在使用FS2时可定义相关的宏来实现限制文件的个数等功能。调用相关的函数完成文件操作，表4-6列出了常用的文件操作函数。

表4-6 常用的文件操作函数

函数名	功能描述
fcreate ()	创建一个文件
fopen_rd ()	打开一个文件用于读
fopen_wr ()	打开一个文件用于写（读）
fwrite ()	从当前指针开始写一个文件
fread ()	从当前指针开始读一个文件
fseek ()	移动文件读写指针

因为文件系统建立在Flash上，那么Flash的相应的参数必须定义在相关的数据结构中来实现操作，定义FlashIDBlock如下。

```
typedef struct {
   long flashID; // Flash 标识
   int flashType; // Flash 类型
   int flashSize; // Flash 大小
  int sectorSize; // Flash 扇区大小
   int numSectors; // 扇区总数
   int flashSpeed; // Flash访问速度
  long flash2ID; // 第二块Flash 标识
  int flash2Type; //第二块 Flash 类型
  int flash2Size; // 第二块Flash 大小
  int sector2Size; //第二块 Flash 扇区大小
  int num2Sectors; //第二块扇区总数
  int flash2Speed; //第二块Flash访问速度
  } FlashIDBlock;
```

系统访问Flash时要首先查找对应的参数，一致则进行操作，否则返回失败，更换Flash时，应更改对应的Flash参数。

4.5 IIC总线通信设计

本系统与线卡的通信采用IIC协议，通信时以主从方式实现网管系统与线卡的通信。各设备在IIC总线上的连接如图4-11。

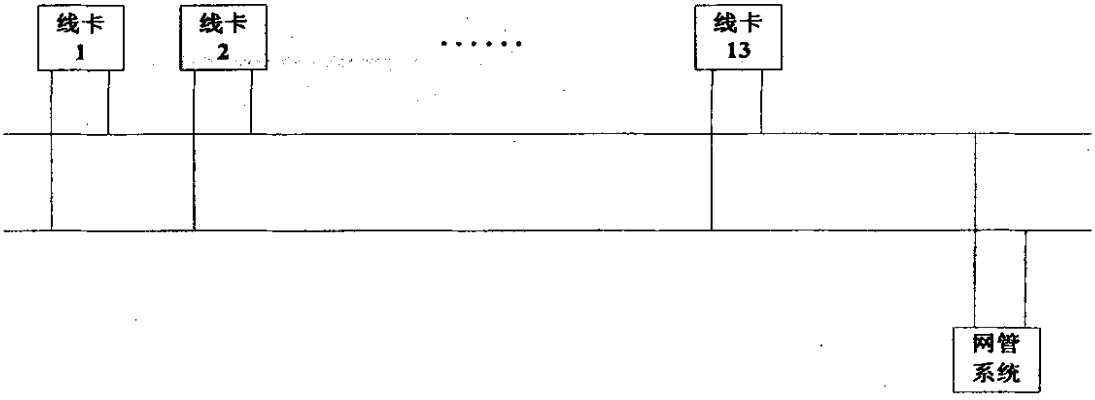

图4-11 IIC总线连接图

IIC通信主要是通过状态机实现的，由于本系统中IIC控制器采用的是Philips公司的PCA9564，网管系统作为管理其它设备的主设备，互相之间的通信以网管系统发起为主，所以PCA9564采用主从工作方式，网管系统为主，被管理电路板为从，参照PCA9564的规范设计状态机如图4-12和图4-13。又因为系统是采用轮询和发送命令的方式来获得被访问设备的状态告警信息，发送的查询和配置命令以及应答命令固定长度为18字节，所以状态机分为发送和接收两套状态机，但实际设计程序时可写在一个程序中。

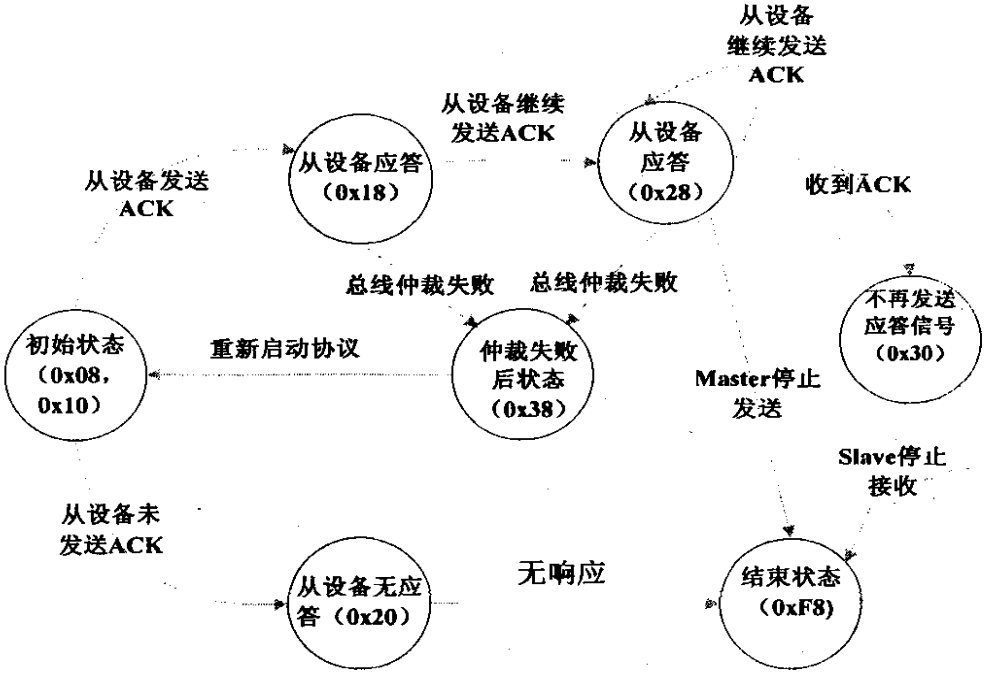

图4-12 PCA9564 IIC控制器发送状态机[5]

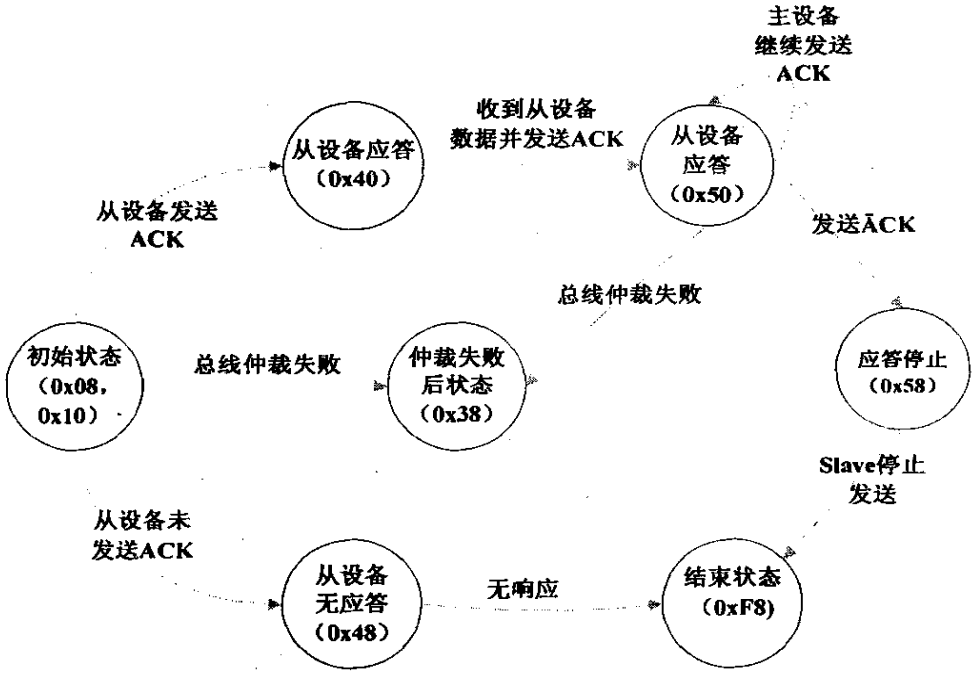

图4-13 PCA9564 IIC控制器接收状态机[6]

4.6 WATCHDOG 设计

作为嵌入式系统，WATCHDOG的作用是非常重要而且必需的，它能使程序进入死循环或进入某种不正常状态后通过WATCHDOG复位重新启动系统使系统恢复正常。

Rabbit2000处理器中有一个16位的WATCHDOG 定时器，可以通过编程来实现WATCHDOG功能。通过采用虚拟驱动技术可实现WATCHDOG从62.5MS到15.94S的复位定时。系统中我们选择5S作为WATCHDOG的复位定时[2]。

4.7 IIC命令格式及数据结构设计

由于本系统对从属设备的管理通过发送和接收命令的方式完成，所以通信双方必须遵循相同的命令格式，为完成主从通信定义了七条指令：CONFIG、CONFIG_ACK、ALARM、ALARM_ACK、RESET、STATUS、STATUS_ACK。

4.7.1 CONFIG命令

CONFIG命令格式如表4-7。

表4-7 CONFIG命令格式

字段	内容	长度	注释
1	板卡号	1	Byte1
2	板卡类型	1	Byte2
3	数据长度	1	Byte3
4	命令 0x20h	1	Config Byte4
5	Tx-N*64k	8	Byte5~8
	Rx-n*64k		Byte 9~12
6	Txclk	Bit7,6	10 master clk; 11 slave clk
7	range	Bit 5	1 long haul 0 short haul
8	target	Bit 4	0 local 1 remote
9	framer_format	Bit 3,2	00 transparent 01 framer_crc4 10 frmer_nocrc4 1
10	Line_coding	Bit 1	0 ami 1 hdb3
11	pad	Bit0	
12	loopback	1	Byte 14 high 有效 Bit0 frmloop Bit1 payloop Bit2 remloop Bit3 locloop 0000 为CLRLOOP
13	Xb2	2	Byte 15~16
14	Checksum error	1	byte 17 0 为正确 其余皆为错误
15	Checksum	1	Byte 18

根据命令格式定义如下结构：

```
typedef struct
{
  char slot;    //板卡号，即7位IIC地址
  char type;   //板卡类型
  char len;    //命令长度
  char cmd;   //命令字0x20h
  unsigned long tx_n64k; //发送方向选择的信道
  unsigned long rx_n64k; //接收方向选择的信道
  char el_set; //传输方式、码型、时钟等的参数选择
  char loopback; //环回测试
  int xb2; //桥接
  char chksum_err; //校验有错否
  char chksum; //奇偶校验和
} config_cmd_struct;
```

4.7.2 CONFIG_ACK命令

CONFIG_ACK命令格式如表4-8。

表4-8 CONFIG_ACK命令格式

字段	内容	长度	注释
1	板卡号	1	Byte1
2	板卡类型	1	Byte 2
3	数据长度	1	Byte3
4	命令 0x21h	1	Config_ack Byte4
5	RX_N*64k error	1	5 , 0 为ERROR
6	TX _N*64k error	1	6
7	Txclk error	1	7
8	Range error	1	8
9	Local error	1	9
10	framer_format error	1	10
11	Line_coding error	1	11
12	Loopback error	1	12

13	Xb2 error	1	13
14	Pad	3	14~16
15	Checksum_error	1	17 0 为正确 其余皆为错误
16	checksum	1	Byte18

根据命令格式定义如下结构：

```
typedef struct
{
   char slot; //板卡号，即 7 位 IIC 地址
   char type; //板卡类型
   char len; //命令长度
   char cmd; //命令字
   char tx_n64k_err; //发送方向配置信道有错
   char rx_n64k_err; //接收方向配置信道有错
   char tclk_err; //时钟配置出错
   char range_err; //中继范围配置出错
   char target_err; //目标配置出错
   char framer_format_err; //帧格式配置出错
   char line_coding_err; //线路编码出错
   char loopback_err; //环回配置出错
   char rsvd[3]; //预留字段
   char chksum_err; //校验和出错
   char chksum; //奇偶校验和
} config_ack_struct;
```

4.7.3 ALARM命令

ALARM命令格式如表4-9。

表4-9 ALARM命令格式

字段	内容	长度	注释
1	板卡号	1	Byte1
2	板卡类型	1	Byte2

2	数据长度	1	Byte3
3	命令0x24h	1	ALARM Byte4
4	Alarm_set	1	BYTE 5 Bit 2 Error-Counter Update Select (ECUS) 0 = update error counters once a second 1 = update error counters every 62.5ms (500 frames)
5	pad	11	0 Byte6～16
6	Checksum error	1	byte 17 0 为正确 其余皆为错误
7	Checksum	1	Byte 18

根据命令格式定义如下结构：

```
typedef struct
{
    char slot; //板卡号，即7位IIC地址
    char type; //板卡类型
    char len; //命令长度
    char cmd; //命令字0x24h
    char alarm_set; //告警信息更新时间
    char rsvd[11]; //预留字段
    char chksum_err; //校验和出错
    char chksum; //奇偶校验和
} alarm_cmd_struct;
```

4.7.4 ALARM_ACK 命令

ALARM_ACK命令格式如表4-10。

表4-10 ALARM_ACK命令格式

字段	内容	长度	注释
1	板卡号	1	Byte1
2	板卡类型	1	Byte2
3	数据长度	1	Byte3
4	命令 0x25h	1	ALARM_ACK Byte4
4	E1_ALARM	1	Byte5 AIS BIT0 RAI BIT1 OOF BIT2 LOS BIT3
5	LCVCR	2	Byte 6, 7 Line-Code Violation Count Register
6	PCVCR	2	Byte8, 9 Path Code Violation Count Register
7	FOSCR	2	Byte 10, 11 Frames Out-of-Sync Count Register
8	Xb2	2	Byte 12~13
9	PAD	3	Byte 14, 15, 16
10	Checksum error	1	byte 17 0 为正确 其余皆为错误
11	Checksum	1	Byte 18

根据命令格式定义如下结构：

```
typedef struct
{
    char slot; //板卡号，即7位IIC地址
```

```
    char type; //板卡类型
    char len; //命令长度
    char cmd; //命令字 0x25h
    char el_alarm; //el告警指示
    int lcvcr; //线路编码失效计数器值
    int pcvcr; //通路编码失效计数器值
    int foscr; //帧失步计数器值
    int xb2; //桥接状态
    char rsvd[3]; //预留字段
    char chksum_err; //校验和出错
    char chksum; //奇偶校验和
} alarm_ack_struct;
```

4.7.5 RESET 命令

RESET命令格式如表4-11。

表4-11 RESET命令格式

字段	内容	长度	注释
1	板卡号	1	Byte1
2	板卡类型	1	Byte2
3	数据长度	1	Byte3
4	命令0x26h	1	ALARM Byte4
5	pad	12	0
6	Checksum error	1	byte 17 0 为正确 其余皆为错误
7	Checksum	1	Byte 18

根据命令格式定义如下结构：

```
typedef struct
{
    char slot; //板卡号，即7位IIC地址
    char type; //板卡类型
    char len; //命令长度
```

```
    char cmd; //命令字0x26h
    char rsvd[12]; //预留字段
    char chksum_err; //校验和出错
    char chksum; //奇偶校验和
} reset_cmd_struct;
```

4.7.6 STATUS 命令

STATUS命令格式如表4-12。

表4-12 STATUS命令格式

字段	内容	长度	注释
1	板卡号	1	Byte1
2	板卡类型	1	Byte2
3	数据长度	1	Byte3
4	命令0x22h	1	Status Byte4
5	pad	12	0
6	Checksum error	1	byte 17 0 为正确 其余皆为错误
7	Checksum	1	Byte 18

根据命令格式定义如下结构：

```
typedef struct
{
    char slot; //板卡号，即7位IIC地址
    char type; //板卡类型
    char len; //命令长度
    char cmd; //命令字0x22h
    char rsvd[12]; //预留字段
    char chksum_err; //校验和出错
    char chksum; //奇偶校验和
} status_cmd_struct;
```

4.7.7 STATUS_ACK命令

STATUS_ACK命令格式如表4-13。

表4-13 STATUS_ACK命令格式

字段	内容	长度	注释
1	板卡号	1	Byte1
2	板卡类型	1	Byte2
3	数据长度	1	Byte3
4	命令 0x23h	1	Status_ack Byte4
5	Tx-N*64k	8	Byte5~8
	Rx-n*64k		Byte 9~12
6	Txclk	Bit7,6	10 master clk; 11 slave clk
7	range	Bit 5	1 long haul 0 short haul
8	target	Bit 4	0 local 1 remote
9	framer_format	Bit 3,2	00 transparent 01 framer_crc4 10 frmer_nocrc4 1
10	Line_coding	Bit 1	0 ami 1 hdb3
11	pad	Bit0	
12	loopback	1	Byte 14 high 有效Bit0 payloop Bit1 frmloop Bit2 locloop Bit3 remloop
13	Xb2	2	Byte 15~16
14	Checksum error	1	byte 17 0 为正确 其余皆为错误

15	Checksum	1	Byte 18

由于STATUS_ACK命令与CONFIG命令所有的字段及格式完全相同，所以这两个命令属同一数据结构。

4.7.8 联合体定义

因为上述7条命令长度统一定义为18字节，为了处理IIC命令的方便，定义联合体如下：

```
typedef union
{
    config_cmd_struct  ConfigCmd; //config命令
    config_ack_struct  ConfigAck;  //config_ack命令
    alarm_cmd_struct  AlarmCmd;//alarm命令
    alarm_ack_struct   AlarmAck; //alarm_ack命令
    status_cmd_struct  StatusCmd; //status命令
    config_cmd_struct  StatusAck; //status_ack命令
    reset_cmd_struct   ResetCmd; //reset命令
    unsigned char Byte[18]; //长度为18的字节数组
} iic_format_union;
```

4.8 任务设计

根据μC/OS-II对任务的要求，按照高内聚、低耦合的原则，保证任务的功能独立性最好，划分定义如下相关任务：http_server(httptask)、告警轮询（polltask）、定时器(timertask)、IIC（iictask）、fs2（fstask）。

4.8.1 任务功能

Httptask实现http server的功能，解析用户的数据请求，并将结果返回给用户。

Polltask实现机箧内各设备的定时轮询，每隔2秒钟轮询一次，一次只轮询一块电路板，机箧内共有13块电路板，完全轮询一次的时间如下：

$$\sum_{i=1}^{13} 2i = 182 \ s$$

$$182/13 = 14 \ s$$

每块线卡的告警信息平均14秒被轮询一次，能够满足实际需求。

Timertask实现系统中的定时器操作，主要用于完成定时器值的处理，检查是否有定时器事件发生。

IICtask实现IIC数据的发送和接收，发送和接收数据均设缓冲区，又由于IIC总线为临界资源，访问时首先进入临界区，成功进入则执行，否则等待，进入临界区后置临界区忙标志，使用完临界区后置相关标志。使用IIC前首先查询临界资源标志。

Efs2task实现文件系统的访问，包括读写相关文件的各类操作。

4.8.2 任务优先级设定

μC/OS-II中共有64个任务，64个任务的优先级为0～63，任务的优先级号越小，任务的优先级越高，考虑到系统的扩展需求，任务优先级从10开始分配，根据任务的重要程度设定优先级如表4-14。

表4-14 任务优先级分配表

任务	优先级
Polltask	10
Htpptask	12
Timertask	14
Iictask	16
Fs2task	18

4.8.3 任务实现

每个任务都应该定期去复位WACTHDOG计数器，而且任务执行完自己的操作后应该让出CPU或等待某个事件发生。一般任务的格式如下：

```
void xxxxtask(void* ptr)
{
 //执行初始化
```

```
    :
    :
  while(1) {
     // 处理本任务相关操作
        :
        :
     kick_watchdog(); //复位WATCHDOG
     OSTimeDlyHMSM(); //释放CPU 一段时间
  }
 }
```

httptask 任务的实现如下：

```
  void httptask(void* ptr)
  {
  http_init();

  while(1) {
     http_handler();  //处理http请求
     kick_watchdog(); //复位WATCHDOG
     OSTimeDlyHMSM(0,0,0,20); //释放CPU 20毫秒钟
  }
 }
```

polltask 任务的实现如下：

```
 void polltask(void* ptr)
 { char i;
   while(1)
   {
    for(i=1;i<8;i++)
    {
     alarm_poll(i); //告警轮询
     OSTimeDlyHMSM(0,0,2,0);  //释放CPU 2秒钟
    }
    for(i=9;i<15;i++)
```

```
    {
     alarm_poll(i); //告警轮询
     OSTimeDlyHMSM(0,0,2,0); //释放CPU 2秒钟
    }
    kick_watchdog();//复位WATCHDOG
  }
}
```

4.9 分级安全管理

根据系统功能的不同实行分级安全管理，实际系统中可分多个级别，我们根据系统的特点，分为两级，一级为一般机务员级，二级为系统管理员级，通过划分Server中的网页分属于哪个级别对应的域即可实现按用户名和密码存取相关的网页。

域的数据结构如下：

```
typedef struct {
 char username[HTTP_MAXNAME];
 char password[HTTP_MAXNAME];
 char realm[HTTP_MAXNAME];
} HttpRealm;
```

定义了不同的域之后在HTTPSPEC常量数组定义时声明该文件属于那个域，那么只有具有此域的访问权限的用户才可以访问对应的文件或资源，从而实现了资源的分级安全管理。一般定义格式如下：

```
{ HTTPSPEC_FILE,  "/afterlogin.html",  afterlogin_html, NULL, 0, NULL,&realmname }
```

4.10 系统帮助

一个好的系统必须提供详细而有效的帮助，除了依据相关的设计文档提供的用户使用手册外，还应提供实时在线帮助。本系统中的帮助主要是以HTML文件的形式提供的，在登录到系统进行相应的操作时可通过菜单栏上的相关链接访问当前相关操作的帮助文档。

第五章 编码实现

系统中的程序文件分前台和后台程序，后台程序主要包括实现用户请求的CGI程序、通信程序等，前台程序主要提供用户显示和输入界面以及差错控制等功能。后台程序主要用C和汇编语言来实现，前台程序用HTML和Javascript语言实现。

5.1 后台程序实现

相关协议栈的程序均放在相应的库中实现，第四章已经讲过。根据系统的需求分析，将系统应用软件划分存放在四个文件中，三个为库文件，一个为C文件，如表5-1。

表5-1 后台文件功能表

文件名	主要功能
main.c	完成包括TCP/IP协议等的初始化，创建任务并启动操作系统执行
cgi.lib	Zserver 所有网页请求的后台处理函数
cmd.lib	系统需于外界通信的所有命令函数的封装及收到应答命令的解析
iic. lib	与IIC读写有关的函数及状态机

5.2 前台程序实现

前台程序主要为*.shtml、*.html文件，实现数据的输入及结果输出。在POST通信时，使用SSI(Server Side Includes)技术实现Server与shtml文件的动态交互通信，从而能够在浏览器的网页上动态显示设备运行状态。主要文件及功能如表5-2。

表5-2 前台文件功能表

文件名	主要功能
index.html	登入系统的主界面
slot.shtml	成功登入系统后，线卡的动态监视页面

change.html	修改相关操作的主界面
changeIP.html	修改IP地址、子网掩码、DNS参数的界面
change_time.html	修改系统时间界面
afterchnIP.html	成功修改IP地址之后的界面
main.html	维护操作的主界面
configmain.shtml	配置维护操作的主界面
alarmquery.html	告警查询主界面
r_check.html	复位操作主界面
s_check.html	状态查询主界面
com_intro.html	帮助文件主界面

除上述文件外，一些较短的输出提示信息以常数字符串的形式定义在cgi.lib中，使用起来更方便。

5.3 其它相关资源

除以上文件外，为了使Web页面更易读，输出信息更直观，系统中还引入了部分*.gif文件，提高了系统的通用性和可操作性。

5.4 系统实现效果

系统实现后相关页面如下：

图5-1 系统登入页面

该页面实现系统的登入安全验证。

图5-2 参数配置页面

该页面实现参数配置参数选择。

其它相关页面不再列举。

5.5 其它附属程序

一个软硬件系统在出厂前应该写入必要的初始化参数，从而可使系统在初次运行时能够实现系统的基本功能，然后维护人员可修改相关参数适应系统所处的网络环境。在我们这个系统中需写入的初始化参数主要有：IP地址、子网掩码、网关、登入默认用户名和密码等初始化值。为此，提供初始化写入工具，通过程序Param_initialize.c实现，主要实现上述初始化值的写入。

5.6 程序写入方式

本系统提供512k Flash，Dynamic C编译产生bin文件，可通过两种方式写入Flash，一种以Dynamic C直接写入，另一种是用Rabbitsemiconductor公司提供的专用下载工具RTU将编译生成的bin文件和Bios一同写入，两种方式适于不同的情况，从而使bin文件写入较方便，编译调测时使用第一种方式的多，生产时一般使用第二种方式。

第六章　系统测试评估

作为一个高速率数据通信设备，应提供给客户安全、可靠的数据通信服务，因而对系统的网管系统的性能要求较高。按照电信管理网（TMN）的分层模型划分，本系统属于网络管理层并包括网元管理层的功能。

6.1 网管系统评价标准

ITU-T的M.30XX建议中规定了一个网管系统的五个方面的管理功能，依次是性能管理、故障管理、配置管理、计费管理、安全管理。由于整个系统提供的是通信协议转换功能，主要是面向高端客户的接入和点到点的专线服务，所以计费管理功能可不予实现，主要实现其它几方面的功能即可[14]。

6.1.1 性能管理

通过对网络中的设备进行测试和监视统计获取关于网络运行状态的各种性能参数值，本系统中可以监视统计线路误码次数、帧失步次数等，可以保存到文件中用于后台统计分析，从而进一步评价网络运行质量。

6.1.2 故障管理

故障管理可分为故障检测、故障诊断定位和故障恢复。故障检测可通过与被管理设备的通信状况以及应答信息判定设备是否处于故障状态，若无法与被管理设备完成正常通信则认为设备处于严重故障状态，否则通过被管理设备的应答信息进行故障定位，较严重故障时可启动设备复位功能，使设备恢复正常运行，复位后设备仍不能正常运行则为不可恢复故障，相应的告警指示标明当前告警的级别及类型，维护人员可据此进行故障定位及修复。在本系统中可查询到E1链路的AIS、RAI、OOF、LOS等告警类型和级别，提供远端环回、近端环回、静荷环回、帧环回等故障定位方法。各设备的告警信息可保存到文件中，用于统计查询。

6.1.3 配置管理

配置管理通过发送配置命令，对网络中的通信设备和设施的变化进行管理。提供友好的人机界面给用户进行参数选择，设备初次运行时可对其进行初始化配置，运行后可更改其相应的配置进行业务变更，各设备的最新配置信息存放在文

件系统中，以备数据恢复和配置参考用。

6.1.4 安全管理

安全管理的功能是保护网络资源，使网络资源处于安全运行状态。安全是多方面的，本系统中通过设置二级密码权限分级别管理访问相关管理功能，从而保护了网络资源，使系统资源处于安全状态。

6.2 测试结果

根据网管系统的评价标准，对本系统的各项功能和性能进行了测试。测试主要以黑盒测试法为主。

6.2.1 功能测试

主要测试系统是否实现了需求分析中的各项功能。经测试人员认真测试，证明需求中的功能基本实现。

6.2.2 性能测试

本系统为电信产品，应具有较高的可靠性及可维修性。衡量系统性能的指标主要有以下几项：

.平均故障间隔时间：MTBF(Mean Time Between Failures)，即在规定的条件下和规定的时间内，系统累计运行时间与故障次数之比。

.平均修复时间：MTTR(Mean Time To Repair)，即在规定的条件下和规定的时间内，产品在任一规定的维修级别上，修复性维修总时间与在该级别上被修复产品的故障总数之比。

.可用度：A（Availability），指可维修产品在规定的条件与时间内，维持其规定功能的能力，它综合反映可靠性和维修性。计算方法：产品能工作时间与能工作时间、不能工作时间的和之比。如：A=MTBF/(MTBF+MTTR)。

.年停机时间：DT（Downtime），在一年内，产品由于故障维修而处于不能工作的全部时间之和。停机时间跟可用度之间换算关系：年停机时间=(1-A)×8760×60（分钟）。

通常所指的产品可用性包括可靠性和可维修性两个方面。可靠性用MTBF来衡量，可维修性用MTTR来衡量，而可用性则用可用度A来衡量。

测试平均故障间隔时间时取20块网管板插入不同的机筐，连续运转20天（480小时）未发生故障，可基本认为平均故障间隔时间大于9000小时，满足信

息产业部发布的路由器设备技术规范，平均修复时间<0.5小时，满足前述标准，从而可计算出其它指标，可用度＝MTTR/(MTBF+MTTR)>99.99%，年停机时间<0.5小时[15]。

由于本系统的告警查询、设备复位、参数配置等功能是通过发送和接收命令的方式完成的，命令收发成功率将影响系统的正常通信和整个系统的通信，测试情况如表6-1：

表6-1 命令发送测试表

命令类别	总测试次数	成功次数	成功率(%)	备注
告警查询	1000	1000	100	
设备复位	1000	1000	100	
状态查询	1000	1000	100	
参数配置	1000	998	99.8	个别参数配置不成功，命令发送成功

由于本系统是整个5U数据通信机筐的网管系统，其电磁兼容性等测试将以整机的形式通过相关测试机构的测试，通信产品入网证也将以整机的形式获得，整机产品开发也基本完成，现该设备正在北京通信公司试用测试，取得相关证书还需要进一步测试及完善。

结束语

本系统实现了基于Web的嵌入式网络管理系统，提供了基本的网管系统的功能，可方便运营企业集中网管监控及远程维护,易于运行维护及组网，降低运营成本。通过本系统的设计我们可有如下结论。

.通信设备在提供可靠性传输及交换功能的情况下，提供方便快捷的网管手段是通信系统必不可少的一项功能。

.运用成熟的、通用的标准和技术开发通信系统可提高系统的开发速度，降低开发成本，可降低开发难度。

.基于嵌入式Web的网管系统提供一种方便的网络管理手段，在实现Web SERVER时具有一定的通用性。

.本系统根据功能需求，结合嵌入式系统节约资源的特性，实现了TCP/IP协议的裁减。

由于本人知识水平有限,在系统的设计及实现过程中难免存在各种错误和不完善之处，请各位老师指正。

参考文献

[1] Lutz Bichler, Ansgar Radermacher, Integrating Data Flow Equations with UML/Realtime, Real-time Systems, 2004, 26:107-125

[2] Rabbit2000 Microprocessor[EB/OL], http://www.rabbitsemiconductor.com, 2005

[3] Requires4008.pdf, http://www.datasheetarchive.com

[4] RTL8019AS 芯片资料, http://www.laogu.net

[5] Pca9564_x.pdf, http://www.philips.com

[6] Max232xxE.pdf, http://www.maxim-ic.com

[7] Rabbit 2000™ Microprocessor Designer's Handbook[EB/OL], http://www.rabbitsemiconductor.com, 2005

[8] Dynamic C User's Manual[EB/OL], http://www.rabbitsemiconductor.com, 2005

[9] Jean J. Labrosse, μC/OS, The Real-Time Kernel Revision, http://www.uCOS-II.com, 2005

[10] 黄天成，余智欣，袁学文，新型嵌入式Web服务器系统的设计与研究，计算机工程，2005，6：176-178

[11] M.Tim Jones，嵌入式系统TCP/IP应用层协议（路晓村等译），北京：电子工业出版社，2003，97-121

[12] Dynamic C TCP/IP User's Manual[EB/OL], http://www.rabbitsemiconductor.com, 2005

[13] Ching-Chin Hsu, Sheue-Ling Hwang, A Study of Interface Design Improvement in an Engineering Data Management System on the World Wide Web, Computers&Industrial Engineering, 2004, 47, No.1:31-43

[14] ITU-T, M.3010, Principles of Telecommunications Management Network (TMN), 1996

[15] 信息产业部，YD / T1096—2001，路由器设备技术规范——低端路由器，2001

发表论文和参加科研情况

[1] 第一作者 基于 rabbit2000 处理器的通信网络管理系统设计和实现 微处理机(国际刊号：ISSN1002-2279，国内刊号：CN21-1216/TP)录用，拟发表在 2007 第 2 期

感谢

首先感谢我的导师韩其睿老师在我学习和课题中的种种帮助，在每次遇到难题时都是他给我指点迷津，鼓励启发我独立解决问题，通过言传身教使我逐步形成了严谨的科学态度，在学识水平提高的同时也对科研工作有了更深刻的理解。

其次感谢我们计算机学院的各位老师，特别是德高望重的李兰友老师，不仅在专业课的讲授中孜孜以求，同时在我做课题的过程中也给了我一些有益的帮助，还有杨连贺老师给我的课外解答，还有叶华老师为我们所做的各种工作。总之，感谢计算机学院所有老师两年多来给我的帮助。同时也感谢通信学院的苗长云老师，以及其它学院给我们上过课，答过疑的各位老师，是你们的诲人不倦才使我们工大人才辈出。

再次感谢我的同学，特别是范方，在各方面都给了我很多帮助，还有童永木、袁加全、庞群友、薛英等，我们在互相帮助中共同提高，我很幸运自己能有这么好的同学共同完成学业。

又次感谢我做课题的北京天诺泰利公司的所有同事，特别是研发部的牛夫贤博士给我解答了诸多疑难，使我的课题得以顺利完成。

又次应该感谢我的家人，是我妻子的支持才使我年过而立再次走进校门，攻读硕士学位，实现了人生的更高追求，但同时我也对我的女儿深表歉意，因为远离家乡失去了很多陪她成长的时光，让小小的心灵过早的尝受了思念之苦。还有我的父母，年近花甲，我却不能经常在身边尽儿子的义务。

最后再次感谢所有在学习和生活中帮助支持过我的老师、同学、公司的同事，祝你们身体健康、万事如意。