摘　要

随着通讯技术、网络技术和半导体技术的飞速发展，嵌入式技术与网络技术结合的条件已经非常成熟。并且为了提高产品性能，实现家用电器、工业控制装置或仪器、安全监控系统、汽车电子等各种智能设备的远程控制、维护和故障诊断功能，网络化已经成为嵌入式系统发展的一个重要趋势。

本文提出一种 DSP 系统结合嵌入式以太网接入模块的方案，实现了计算机网络与以 DSP 为核心的嵌入式系统的互连。以太网接入系统模块主要由 8 位单片机和 RTL8019 通用网络接口控制器组成，DSP 系统只要和以太网接入系统模块中的单片机通过串口通信便可实现整个系统的网络接入功能。

由于目前嵌入式系统接入网络的主要困难在于 TCP/IP 协议栈的实现对于系统资源要求很高，嵌入式系统资源有限，因此在软件实现 TCP/IP 协议栈的过程中，需要对协议栈进行适当的裁减和优化。本文专门使用了一个章节对嵌入式 TCP/IP 协议栈进行了深入细致的研究和分析。

在硬件设计方面本文完成了以太网接口和 DSP 与单片机通过串口通信两部分硬件接口电路的设计，并详尽描述了各部分的相互关系及工作原理。

软件设计部分则分为 DSP 与单片机的串口通信程序、以太网接口驱动程序和 TCP/IP 协议栈程序三部分，本文对每一部分都分章节的做了详细的说明，并给出了程序实现的流程图。

最后，文章通过对实验测试结果的分析，得出了本课题研究的“嵌入式网络接入方法可以实现 DSP 系统与以太网实现互联”这一结论，从而使实现对 DSP 系统进行远程控制、维护等功能成为可能。

关键词：DSP，以太网，TCP/IP，嵌入式系统

Research and Realization of Ethernet Technology in the DSP System

Abstract

Along with development and popularization of communication technology,network technology and Semi-conductor technology,the case of embemded technology combining with network technology has become very mature.To realize long-distance control, maintenance and trouble diagnosis based on appliance wiring, industrial control equipment or the instrument, safety supervises and control system, car electronics etc,network has become a important trend of the development of embeded system.

A method ,which achieved the connection of DSP system with Ethernet interface and Internet,is given in this paper.The Ethernet interface module is made up of 8 bit MCU and network card chip RTL8019AS. DSP system connects with Ethernet by communicating with MCU in the Ethernet interface module

In the hardware designation, the circuit of hardware in communication between MCU and DSP and the Ethernet interface module are achieved in this paper.

System software designation is introduced in this paper,which is consisit of asynchronous serial communication between DSP and MCU program,the driver of network card and TCP/IP protocol program module.

Finally, by testing the system and the analysis of data, the method achieved in this paper can make DSP embedded system connect with Ethernet .So long-distance control and maitanace based on DSP embedded system also come true.

Key Words: DSP, Ethernet, TCP/IP, Embedded system

独 创 性 说 明

本人郑重声明：所呈交的论文是我个人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写的研究成果，也不包含为获得沈阳工业大学或其他教育机构的学位或证书所使用过的材料。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做了明确的说明并表示了谢意。

签名：莫云鹏 日期：2007.3.16

关于论文使用授权的说明

本人完全了解沈阳工业大学有关保留、使用学位论文的规定，即：学校有权保留送交论文的复印件，允许论文被查阅和借阅；学校可以公布论文的全部或部分内容，可以采用影印、缩印或其他复制手段保存论文。

（保密的论文在解密后应遵循此规定）

签名：莫云鹏 导师签名：范镇瑞 日期：2007.3.16

1 绪论

1.1 课题来源

本课题为自选课题，课题题目名称为“DSP 系统网络接入技术的研究与实现”。

1.2 课题研究目的

Internet 是人类历史发展中的一个伟大的里程碑，它的起源和发展引发了一场革命，改变了世界。Internet 充分发挥了计算机的效能，帮助人们跨越时间和空间的障碍。网络用户可以通过网络服务共享信息、协调工作，而不受地理范围的限制，也可以避免由于时区的限制。宽带网络的出现，使得语音图像的传输成为可能，由此产生了视频会议和 IP 电话，使分布在世界各地的公司职员可以同时参加公司的会议[1][2][3]。

Internet 的出现为人们提供了一个划时代的信息媒体。人们可以通过网络访问信息库、图书馆，便捷地查询各类信息资源，广泛地阅读各种书籍；可以通过 Internet 收发电子邮件、打电话；通过网络提供各种商业信息，进行网上营销；政府部门可以通过网络平台公开施政，提高政府工作的透明度和效率；越来越多的人已经享受到了在家办公的自由和乐趣[4][5][6]。

随着计算机技术的发展，嵌入式计算机系统广泛地深入到社会生产生活的各个领域。如果能够将这些系统接入 Internet，人们就可以远程控制它们工作，及时了解设备的工作状态，提高设备的信息化程度，推动信息社会的发展。为了解决上述问题，嵌入式 Internet 技术应运而生[7][8][9]。

综上所述，实现嵌入式系统与 Internet 的互联已经成为当前嵌入式系统发展的热点领域和重要方向，因此本课题以实现以 DSP 为核心的嵌入式系统与网络的互联为研究目的，并最终通过对系统软、硬件的合理设计实现了 DSP 系统通过网络的数据传输和信息共享。

1.3 课题研究意义

在信息时代的今天，Internet 技术将不会仅仅局限于人与人（或团体）之间信息的交流。例如 E－mail、 WEB 浏览和电子商务 e－business 等，Internet 技术都将会深入到人们日常生活和工作的电子设备中。并且随着 IA（信息电器）的出现，嵌入式网络

技术正逐步取代传统的以 PC 为中心的应用，成为未来 Internet 发展中的主力军，并广泛应用于智能家居系统、工业智能化从站系统、LED 网络控制显示屏系统、网络安全加密系统等各个领域的各个方面。各国信息界同仁都在密切关注并积极研究电子设备与 Internet 的连接问题[10][11]。

DSP 芯片是专门为实现各种数字信号处理算法而设计的、具有特殊结构的微处理器，其卓越的性能、不断上升的性价比、日渐完善的开发方式使其应用越来越广泛。将计算机网络技术引入以 DSP 为核心的嵌入式系统，使其成为数字化、网络化相结合，集通信、计算机和视听功能于一体的电子产品，必将大大提升 DSP 系统的应用价值和市场前景。本文的研究工作解决了以 DSP 为核心的嵌入式系统与 Internet 的连接问题，从而为 DSP 设备的应用开辟了更为广阔的前景[12][13][14]。

1.4 国内外研究现状

目前国外许多大公司如 Cygnal，Maxim，Microchip 等成立了嵌入式 Internet 联盟，来专门讨论和制定嵌入式 Internet 领域的标准和开发相关的技术[15]。这些公司都推出了内部固化了 TCP/IP 协议栈的嵌入式微处理器，这些产品使开发人员省去了编写并且移植庞大繁琐的 TCP/IP 协议栈软件的工作，从而大大提高了开发的效率。同时国内也有像周立功电子、武汉力源科技等很多有实力的公司在嵌入式网络接入技术的研究方面取得了相当的成就，它们所推出的产品通常是将 MCU 与网卡芯片的集成模块，其产品内部的结构组成以及原理与本文所开发的嵌入式网络接口系统模块类似。开发人员可以通过使用这些产品省去网络部分的设计，从而将主要精力放到应用系统的设计上去。

1.5 目前存在问题

嵌入式系统实现 Internet 接入的前提条件是系统的软件中要有 TCP/IP 协议支持。由于以 DSP 为核心的嵌入式系统的硬件资源有限以及成本的敏感性，实现如台式机中那样完整的 TCP/IP 协议是比较困难的[16]。因此根据实际系统的应用场合与特点，对 TCP/IP 协议栈进行合理地精简以及选择合理的网络接入方案将成为研究网络技术与嵌入式系统结合的一个重要研究方向。

1.6 本课题研究内容

本论文的研究基于 DSP 系统网络接入功能应用的开发，因此开发工作的主要内容包括：

(1)研究 TCP/IP 协议并结合嵌入式系统特点进行合理选择。课题分析了 TCP/IP 协议族中各种协议的基本功能，根据嵌入式系统存储容量小、功能单一的特点对协议族进行了选择和简化，使 TCP/IP 协议能够在嵌入式系统中方便实现而同时能实现以太网接入的基本功能。

(2)系统硬件的开发。使用 STC 公司的高性能 51 系列单片机 STC89C516RD＋与 Realtek 公司的网络控制器芯片直接接口组成嵌入式以太网接口系统。设计基于 TMS320VC5416 DSP 芯片的 DSP 最小系统，并将 DSP 芯片的多通道缓冲串口 MCBSP 与通用异步串行收发器芯片 MAX3111E 接口实现 DSP 系统的异步串行通信，最终将 DSP 系统与以太网接口系统通过串口连接，实现 DSP 系统接入以太网的功能。

(3)驱动程序开发。相关的驱动程序主要有以太网接口系统中单片机与网络控制器芯片的接口驱动程序、TMSVC5416 DSP 微处理器与 MAX3111E 芯片的接口驱动程序。

(4)协议软件开发。编写符合嵌入式以太网接口系统资源特点的高度精简的 TCP/IP 协议栈软件程序，及 DSP 系统与以太网接口系统的串口通信协议程序。

2 嵌入式 TCP/IP 协议研究

2.1 协议概述

网络协议通常分不同层次进行开发，每一层分别负责不同的通信功能。TCP/IP 协议是一组不同层次上的多个协议的组合。通常 TCP/IP 被认为是一个四层协议系统[17]。如图 2.1 所示：

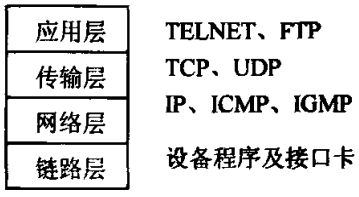


图 2.1 TCP/IP 协议分层

Fig. 2.1 TCP/IP protocol layer

尽管通常称该协议族为 TCP/IP，但 TCP 和 IP 只是其中的两种协议而已。在这个四层次中，每一层负责不同的功能。

（1） 链路层，有时也叫数据链路层和网络接口层，一般包括操作系统中的设备驱动程序和计算机中对应的网络接口卡[18]。它们一起处理与双绞线（或其他任何传输媒介）的物理接口细节。根据网络使用的硬件不同，TCP/IP 支持多种不同的链路层协议，如以太网、令牌环网、FDDI（光纤分布式数据接口）及 RS-232 串行线路等。本课题只讨论以太网网络链路层协议。以太网采用一种称作 CSMA/CD 的媒体接入方法，其意思是带冲突检测的载波监听多路访问。其规则是：

1） 若媒体空闲，则传输；否则转第二步。

2） 若媒体忙，一直监听直到信道空闲，然后立即传输。

3） 若在传输监听到冲突，则发出一个短小的人为干扰信号，让所有的站点都知道发生了冲突并停止传输。

4） 发完人为干扰信号等待一段随机的时间，再次试图传输（从第一步开始重复）。

在 TCP/IP 协议族中，链路层的协议主要有 ARP 和 RARP。ARP 就是地址解析协议，实现 IP 地址到 MAC 地址的转换。以太网数据链路有自己的寻址机制（48bit 地址）。网络接口有一个硬件地址，在硬件层次上进行的数据帧交换必须有正确的接口地址。当系统把以太网数据帧发送到位于同一局域网上的另一台主机时，是根据 48 bit 的以太网地址来确定目的接口的。设备驱动程序不检查 IP 数据报中的目的 IP 地址。TCP/IP 也有自己的地址——32 bit 的 IP 地址。如果目的主机在本地网络上（如以太网、令牌环网或点对点链接的另一端），那么 IP 数据报可以直接送到目的主机上。如果目的主机在一个远程网络上，那么就通过 IP 选路函数来确定位于本地网络上的下一站路由器地址，

并让它转发 IP 数据报。发送端主机必须把 32bit 的 IP 地址变换成 48bit 的以太网地址，从逻辑 Internet 地址到对应的物理硬件地址需要进行翻译。ARP 协议就是为 IP 地址到对应的硬件地址之间提供动态映射。

ARP 协议包括 ARP 请求和 ARP 响应两部分。主机需要发送 IP 数据报文时，首先查找是否有对应该 IP 地址的硬件以太网地址，如果没有则发送 ARP 请求。ARP 请求作为以太网广播数据包发送到同一局域网上的所有主机。ARP 请求数据帧中包含目的主机的 IP 地址，其意思是“如果你是这个 IP 地址的拥有者，请回答你的硬件地址”[19]。

目的主机收到这个广播报文后识别出这是发送端询问它的 IP 地址，于是发送一个 ARP 应答。这个 ARP 应答包含 IP 地址及对应的硬件地址。主机通过 ARP 请求和 ARP 应答建立起地址的映射。

RARP（逆地址解析协议）用于解决如何从 MAC 地址得到 IP 地址。具有本地磁盘的系统引导时，一般是从配置文件中读取 IP 地址，但无盘工作站则需要用 RARP 来获得 IP 地址。网络上的每个系统都具有唯一的硬件地址，它是由网络接口生产厂家配置的。无盘系统的 RARP 实现过程是从接口卡读取唯一的硬件地址，然后发送一份 RARP 请求（一帧在网络上广播的数据），请求某个主机响应无盘系统的 IP 地址(在 RARP 应答中)。

（2） 网络层，有时也称为互联网层，处理数据分组在网络中的活动。网络层提供的是一种不可靠的服务，它只是尽可能快地把分组从源节点送到目的节点，但是并不提供可靠性保证。网络层协议包括 IP（网际协议），ICMP（Internet 互联网控制报文协议），以及 IGMP（Internet 组管理协议）[20]。

IP 是 TCP/IP 协议族中最为核心的协议，所有网络层和传输层的网络数据分组都以 IP 数据报格式传输。IP 提供不可靠、无连接的数据报传送。不可靠的就是它不保证 IP 数据报能成功地到达目的地。IP 仅提供最好地传输服务。如果发生某种错误，IP 丢弃该数据报，然后发送 ICMP 消息到信源。可靠性由上层来提供。无连接就是 IP 并不维护任何关于后续数据报的状态信息，每个数据报的处理是相互独立的，也就是 IP 数据报可以不按发送顺序接收[21]。

ICMP 是 IP 的附属协议。IP 层用 ICMP 来与其它主机或路由器交换错误报文和其他重要信息。ICMP 报文是在 IP 数据报内部被传输的，根据类型字段和代码字段不同，ICMP 报文分为多种类型。

多播路由器需要知道多播数据应该向那些接口转发。IGMP 协议是 Internet 组管理协议，它让一个物理网络上的所有系统知道主机当前所在的多播组，向多播路由器提供必要的信息。IGMP 也被当作 IP 报文的一部分，IGMP 报文通过 IP 数据报进行传输。与其他协议报文不同，IGMP 有固定的报文长度。IGMP 有两种类型，类型 1 说明是由多播路由器发出的查询报文，类型 2 说明是主机发出的报告报文。

(3)传输层主要为两台主机上的应用程序提供端到端的通信。在 TCP/IP 协议族中，有两个互不相同的传输协议：TCP（传输控制协议）和 UDP（用户数据报协议）[22]。

TCP 协议为用户提供高可靠性的数据通信。TCP 提供一种面向连接的、可靠的字节流服务。面向连接就是指两个使用 TCP 的应用在彼此交换数据之前必须先建立一个 TCP 连接。TCP 通过接收确认、超时重发、检验和、流量控制等手段提供可靠性[23]。字节流服务是指两个应用程序通过 TCP 连接交换 8 bit 字节构成的字节流，TCP 不在字节流中插入记录标识符。

UDP 是一个简单的面向数据报的传输协议。它把应用层传送给 IP 层的数据发送出去，但是并不保证它们能够到达目的地。可靠性由应用层提供[24]。

4、应用层负责处理特定的应用程序。在常用的 TCP/IP 实现中主要有 telnet、FTP、SMTP、和 SNMP 等。根据不同的业务需求，应用层采用的协议也不同。

2.2 协议的选择

基于嵌入式系统实现 TCP/IP 协议，必然要受到嵌入式系统的约束和限制。嵌入式系统是为完成某种特定的功能而设计的专用系统。嵌入式以太网接入系统也是完成特定的功能，没必要也不可能实现所有的 TCP/IP 协议，而必须结合嵌入式系统的特点对 TCP/IP 协议进行选择。

2.2.1 链路层协议

链路层主要有三个目的：（1）为 IP 模块发送和接收 IP 数据报；（2）为 ARP 模块发送 ARP 请求和接收 ARP 应答；（3）为 RARP 模块发送 RARP 请求和接收 RARP 应答。以太网链路层依靠 CSMA/CD 协议、ARP 和 RARP 实现这几个目的。

CSMA / CD（载波监听多路访问/冲突检测）协议是在同一个局域网上的多台计算机共享同一物理传输介质的基础。嵌入式系统要接入以太网络同其它计算机进行通信就必须实现该协议[25]。

以太网上数据的传输是采用网络的 MAC 地址来进行识别的，这就要求系统有实现 IP 地址到 MAC 地址的转换的功能，即 ARP（地址解析协议）。系统要同其它计算机通信，需要根据 IP 地址获得 MAC 地址，就必须要支持 ARP 协议。

RARP（逆地址解析协议）用于解决如何从 MAC 地址得到 IP 地址，主要用于无盘工作站中。在嵌入式以太网接入系统中，系统执行特定的任务，可以把 IP 地址存储于本地存储器中，不必从其它服务器得到 IP 地址，这样就不需要实现 RARP 协议[26]。

2.2.2 网络层协议

网络层主要负责处理数据包在网络中的活动。在 TCP/IP 协议族中，网络层协议包括 IP，ICMP，以及 IGMP 等。

IP 协议是 TCP/IP 族的核心协议，它用来完成异构网络之间的通信。所有的 TCP、UDP、ICMP 数据都以 IP 数据报格式传输。因此系统数据跨越不同的网络进行传输就必须要实现 IP 协议[27]。

ICMP 协议主要用来传递差错报文以及其他需要注意的信息。ICMP 中规定了多种协议类型和代码，如果完全的实现也要耗费不少的系统资源。嵌入式接入系统中，在 ICMP 协议中能够测试网络的连通情况即可，因此只需支持 ICMP 中 Ping 协议和端口不可达协议。

网络层另外一个重要的协议是 IGMP 协议，它主要用于支持主机和路由器进行组播。嵌入式以太网接入系统作为一种专用系统接入网络的技术，为了降低处理协议的复杂程度，在需要向多个目标发送信息时可以直接采用广播方式，不必要采用组播的方式进行通信。因此在设计中不考虑实现 IGMP 协议。

2.2.3 传输层协议

传输层主要在两台主机之间提供端到端的通信。传输层有两种不相同的传输协议：TCP（传输控制协议）和 UDP（用户数据报协议）[28]。

TCP 是面向连接的，在不可靠的网络服务上提供端到端的可靠字节流。TCP 协议设计了严格的 3 次建立连接握手过程、4 次关闭连接握手过程以及捎带确认信息数据传输过程。因为通过 TCP 协议进行通信时，存在一个建立连接的过程，所以必然存在一定的延时。在实时性要求不太高的设备中采用 TCP 协议，保证传输的质量[29]。

UDP 协议是用来提供不面向连接的，它只是简单地把数据报从一台主机发送到另一台主机，但并不保证该数据报能到达另一端,可靠性必须由应用层来提供。在实时性要求高的设备中采用 UDP 协议，UDP 固有的传输可靠性低的缺陷可以通过应用层的协议进行弥补。

2.2.4 应用层协议

应用层协议主要是指用户进程，根据不同的业务采用不同的协议。因此本课题不考虑应用层协议。

2.3 系统嵌入式 TCP/IP 协议栈

根据前面的协议的选择，嵌入式 TCP/IP 主要包括实现 IP 地址到物理地址动态映射的 ARP 协议，其中包括 ARP 请求和 ARP 应答；实现网际传输的 IP 协议；检测主机是否可达的 ping 应答和端口不可达的 ICMP 协议；实现数据快速简单传输的 UDP 协议；实现数据可靠传输的 TCP 协议。其中 UDP 和 TCP 两种不同传输层协议应用于不同的场合，在不同的应用中可以只加载一种协议模块，也可以同时支持这两种协议。

3 系统总体硬件设计

3.1 系统网络接入方案的选择

目前 DSP 系统网络接入通常有两种方法[30]。

(1)DSP 芯片直接与网络接口芯片进行接口，TCP/IP 协议也直接在 DSP 中实现。这种方案可直接使系统与网络相连，有很大的灵活性。缺点是占用的系统资源较多，对 DSP 系统资源要求高，并且在 DSP 上实现 TCP/IP 协议栈的编程相对复杂，工作量较大，开发周期长。

(2)使用嵌入式网关来实现。DSP 系统首先和网关进行通信，通信方式采用传统的 RS－232、RS－485 等，由嵌入式网关来负责实现 TCP/IP 协议，实现 DSP 系统与网络的信息交互。缺点是嵌入式网关与 DSP 系统之间的通信受到距离和速度的限制。

本课题选择的 DSP 系统结合嵌入式网络接入模块的系统实现方案就类似于上述的第二种方法，其中嵌入式网络接口模块（嵌入式网关）由高性能单片机和通用网络接口芯片组成。DSP 系统模块则通过串口与单片机进行通信，从而实现系统的网络接入。所以本系统硬件电路的设计主要分为 DSP 系统模块、异步串口通信模块和嵌入式以太网接口系统模块三部分。

3.2 系统的总体结构框图

系统的 DSP 系统模块中的 DSP 最小系统以 TI 公司的 TMS320VC54X DSP 为核心处理器，DSP 系统模块与以太网接口模块的通信功能由 DSP 芯片的多通道同步缓冲串口 MCBSP 与通用异步串口收发器接口实现。而嵌入式以太网接口系统模块则使用高性能的 51 系列单片机作为核心处理器与网络控制器芯片进行接口来实现对以太网通信的控制。系统的硬件结构框图如图 3.1 所示：

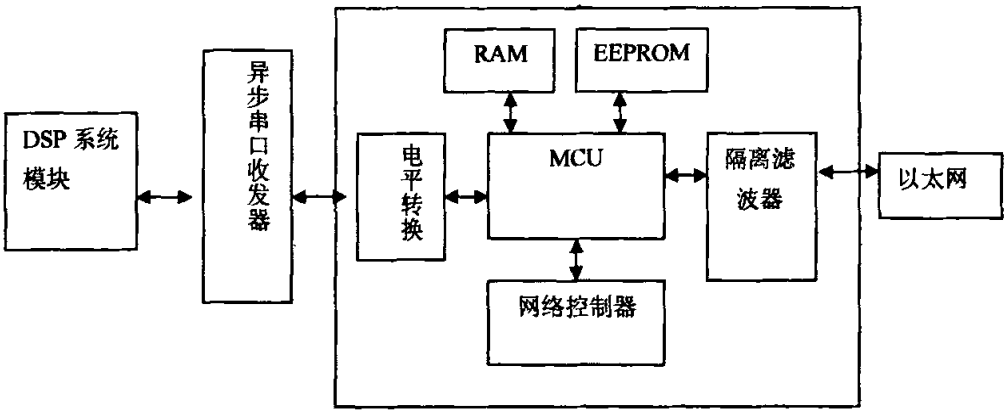


图 3.1 系统总体结构框图

Fig. 3.1 System total structure

3.3 系统总体结构及工作原理分析

从图 3.1 的系统总体结构图中可以看出，DSP 系统模块与以太网接口系统模块是两个相对独立的系统模块，以太网接口系统模块的工作状态受到 DSP 系统模块所发送的指令参数的控制和管理，DSP 系统模块可以动态配置以太网接口系统模块的 IP 地址、MAC 地址、Server/Client 工作模式等网络接入状态参数，从而 DSP 系统模块在整个系统中起着主导作用。

以太网接口系统模块则是整个系统实现网络接入的核心功能模块，由于嵌入式 TCP/IP 协议栈在该系统模块的核心处理器单片机上实现，因此网络通信的组织和管理工作也都由该模块承担。异步串口通信模块则在前面所述的两个模块之间起着沟通的桥梁作用，实现两个模块系统的数据交换。

4 DSP 系统模块设计

DSP 系统模块主要由以 TMS320VC5416 DSP 芯片为核心的 DSP 最小系统组成。

系统上电以后本模块需要完成以下工作：

（1） 将系统程序从快速闪存 FLASH 中导入系统快速 RAM 中

这主要利用 TI 公司的 DSP 芯片本身固化的 Bootloader 程序实现的，根据该程序引导方式的具体要求实现将调试通过的所有系统软件烧写到快速闪存中。

（2） 对 DSP 本身进行必要的初始化

这主要包括上电后 DSP 的运行速度、访问外部扩展器件时的等待周期、系统正常运行时的状态设定，特别是系统存储空间的合理分配等。

（3） 对异步串口收发器进行初始化

异步串口收发器MAX311E在本系统模块与以太网接口系统模块之间建立起可靠的通信，结构简单实用，利用 DSP 本身的资源即可实现对控制器的初始化以及数据的收发工作。

（4） 向以太网接入系统模块发送配置命令。

4.1 DSP 芯片概述

数字信号处理由于其精度高、灵活性大、可靠性好、易于大规模集成，可采用多种性能优良的数字信号处理方法和算法等优点，正得到迅速发展和广泛应用。数字信号处理器（Digital Siginal Processor,缩写为 DSP）正是适应这种需要出现的，并处于蓬勃发展之中，这反过来又为数字信号处理技术的迅猛发展提供了动力[31]。DSP 不仅具有高速运算和控制能力，而且根据实时数字信号处理的特点，在处理器结构、指令系统、指令流程上都做了很大的改动，具体有以下几个方面：

(1) 普遍采用数据总线和程序总线相分离的哈佛结构或改进的哈佛结构，比冯.诺依曼结构有更高的指令执行速度。

(2) 大多采用流水线技术，减少每条指令的执行时间。

(3) 片内有多条总线，并且提供方便的寻址方式，大大提高了指令的执行效率。

(4) 提供高速的寻找方式，如循环寻找、位反寻找等。

(5) 针对数字信号处理中大量用到的乘累加操作的特点，配有独立的硬件乘法器和加法器，可在一个指令周期内完成乘累加运算。

(6) 片内集成 DMA 控制器和串行通信口等，提高了数据的搬移能力。

(7) 具有软、硬件等待功能，可方便的与各种存储器接口。

(8) 单片系统，功耗低，易于小型化和便携式设计。

TMS320VC5416(简称 VC5416)是 TI 公司的 C54X DSP 家族的成员之一，它是基于先进的改进哈佛结构的 16 位定点 DSP，拥有一条程序总线和三条数据总线。片内集成有一个具有高速并行性的算术逻辑单元（ALU）、专用硬件逻辑、片内存储器和片内外设等几部分[32]。以下主要介绍 VC5416 的一些主要特点：

- 先进的多总线结构：三条 16 位数据总线和一条程序总线。
- 40 位桶形移位器和 40 位累加器。
- 可寻址 8M*16bit 的程序空间。
- 16K*16bit 片内 ROM 和 128K*16bit 的片内 RAM。
- 片内外设包括：
    1）软件可编程等待状态发生器；
    2）软件可编程锁相环（PLL）；
    3）三个多通道缓冲串行口（McBSP）；
    4）增强型 8/16 位主机接口（HPI8/16）；
    5）一个 16 位定时器
    6）六通道 DMA 控制器
- 软件设置进入省电模式。
- 指令周期 6.25/8.33ns(160/120MIPS)。
- 内核电压 1.6V，I/O 电压 3.3V。

本系统采用的 TMS320VC5416 单指令周期达 6.25ns(160MIPS)，软件可编程锁相环在 CPU 不工作时，降低时钟频率，从而可降低功耗，而正常工作时，又可很快提升时钟频率；软件可以利用空闲指令（IDLE1、IDLE2 和 IDLE3）将 VC5416 置于省电模式；片内的软等待状态发生器和 DMA 通道等外设，通过软件设置等待周期的个数，不仅降

低了系统硬件设计的复杂性，而且为系统带来了很大的灵活性，为硬件调换和软件编程带来了极大的便利。因此本部分中考虑到系统功能的可扩展性，所使用的核心处理器是由 TI 公司生产的 TMS320VC5416。

4.2 模块硬件电路设计

模块硬件电路的设计主要包括：DSP 最小系统电路的设计和 DSP 存储空间的设计。电路框图如图 4.1 所示：

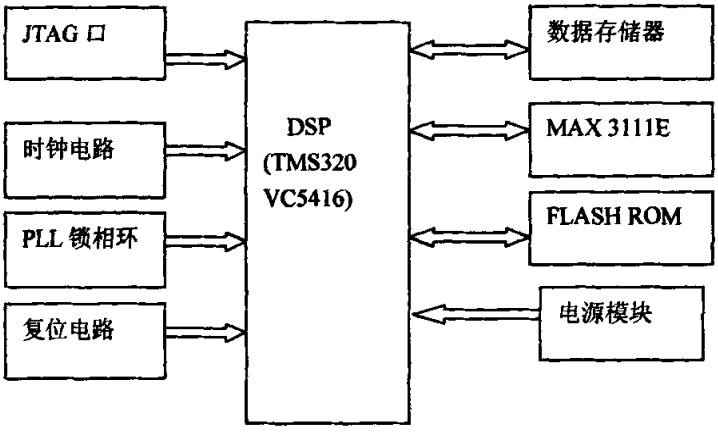


图 4.1 DSP 系统模块电路设计框图

Fig. 4.1 DSP Module Circuit

4.2.1 DSP 最小系统电路的设计

DSP 最小系统电路的设计主要包括电源管理模块、扫描仿真口（JTAG）、复位电路、时钟电路和 PLL 锁相环的设计。

（1）电源转换电路

为了降低芯片功耗，TI 公司的芯片采用的是双供电模式，即内核电压和 I/O 的电压分开的方式。本系统中 VC5416 的 I/O 口的工作电压为 3.3V，内核工作电压为 1.6V。由于目前没有输出固定电压为 3.3V 和 1.6V 的电源芯片，因此电源管理模块采用的是 TI 输出电压可调的 TPS767D301，通过计算输出电阻的大小使其输出 3.3V 和 1.6V 两种电压。电源模块需要注意的是滤波电容的选择，因为电压质量的好坏直接影响后面系统的

稳定性。滤波电容尽可能的选择滤波质量好，并且稳定性好的电容。综合上述，本系统中选用的是稳定性较好的胆电容。本部分电源电路如图 4.2 所示。

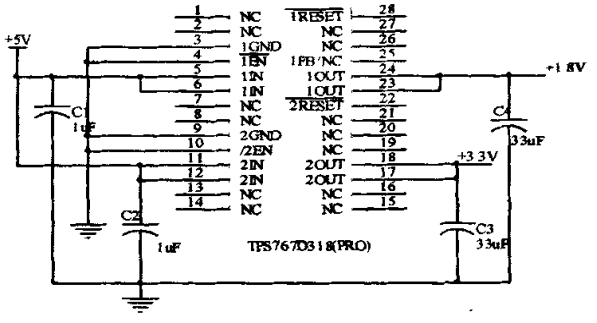


图 4.2 电源转换电路

Fig. 4.2 Power Supply Circuit

（2）扫描仿真口（JTAG）

仿真器即扩展开发系统，可用来进行系统级的集成调试，使进行 DSP 芯片软硬件开发的最佳工具。目前主要有两种仿真器：一种是传统的电路仿真器，另一种是先进的扫描仿真器。现在 DSP 常用的是第二种仿真器。DSP 芯片内部是通过移位寄存器扫描链实现扫描仿真，这个扫描链被外部的串行口访问。采用扫描仿真，即使芯片已经焊在电路板上，也可以进行仿真调试，这对于再生产过程中调试 DSP 系统也带来极大的方便。

（3）复位电路

对于实际的 DSP 应用系统，特别是产品化的 DSP 系统，其可靠性是一个不容忽视的问题，由于系统的时钟频率较高，在运行时极有可能发生干扰和被干扰的现象，严重时可能会出现死机现象。为了克服这种情况，除了在软件作一些保护措施外，硬件上也必须作相应的处理。硬件上最有效的保护措施就是采用具有监视功能的自动复位电路。本系统选用的 MAX 公司的复位芯片 MAX706RCPA 来实现对系统的复位功能。

（4）PLL 锁相环

早期的 DSP 芯片一般工作频率较低，因此其工作频率与外部提供的频率相等或是外部频率的 2 分频或 4 分频。随着 DSP 芯片速度的提高，外部频率很高，如果仍采用此种方式，必然产生高频干扰，影响系统的稳定性。因此，现在的 DSP 芯片提供了多种工作方式。不仅具有传统的分频方式，而且采用更加灵活的可编程 PLL 方式。TMS320VC5416 是属于软件可编程 PLL 方式，软件可编程 PLL 受一个存储器映射（地址为 58h）的时钟模式寄存器 CLKMD 控制，在 DSP 的程序中设置 CLKMD 的值，可以改变 PLL 时钟模块的配置。复位后的 CLKMD 的值是根据 DSP 芯片三根输入引脚 CLKMD1～CLKMD3 确定，从而确定 DSP 复位后的工作时钟，然后通过软件设置 PLL 改变工作频率。本系统中的 CLKMD1～CLKMD3 利用跳线被设置为 110,使系统上电的工作频率为 16MHZ。

4.2.2 DSP 存储空间设计

VC5416 处理器具有 23 根地址线，有三个独立编址的存储空间，其中程序存储空间为 8M，数据存储空间为 64K 和 I/O 存储空间为 64K。程序存储空间用于装载程序指令和常数表；数据存储空间存放程序指令使用的临时变量；I/O 空间则为外部设备提供了一个存储器映射接口，并且可以作为额外的数据存储器。在任何一个存储空间内，根据不同的映射不同空间的器件都可以驻留在片内或片外[33]。

VC5416 具有丰富的片上存储器资源，包括 64Kx16bit 的片上双访问 RAM（DARAM）和 64Kx16bit 的片上单访问 RAM（SARAM）。DARAM 被划分为 8 块不同的地址期间，每块 8K 字。在同一个可以对同一块 DARAM 进行两次读操作，或是一次读一次写操作，这样大大提高了程序执行的速度。SARAM 被划分为 8 块不同的地址期间，每块 8K。每一块都是单访问存储器，也就是在同一个指令周期一个指令字能将一个数据字从一个 SARAM 块写到另一个 SARAM 块。

可用软件编程方法，对影响片上存储器分配的三个控制位进行设置，控制片上存储器是否配置到存储空间，并指定片上存储器是配置到程序空间还是数据存储空间。

（1）MP/＃MC：当此位为 0 时，允许片上 ROM 配置到程序空间中；当此位为 1 时，禁止片上 ROM 配置到程序存储空间中。

（2）OVLY：当 OVLY=1 时，片上 RAM 配置到程序空间和数据空间；当 OVLY＝0 时，片上 RAM 仅配置到数据存储空间。

（3）DROM：DROM=1 时，片上 ROM 部分配置到数据存储空间；DROM=0 时，片上 ROM 部分不被配置到数据存储空间。DROM 与 MP/MC#的状态无关。

VC5416 使用一个页扩展存储器结构可寻址的程序空间为 8M，为了完成这种结构，VC5416 有许多不同于其它 54x 系列 DSP 的特征：

●地址总线为 23 根，而不是 16 根。

●具有一个扩展存储器寄存器 XPC。

●六条可访问扩展程序空间的指令。

VC5416 的程序空间有 128 页组成，每页 64K 字。当片上 RAM 映射到程序空间时，每页程序存储器由两部分组成：一个每页都公用的 32K 字，一个是每页独立的 32K 字。公用的 32K 存储空间为每页共同使用的同一个存储器，每页独立的 32K 存储空间通过特定的页进行访问[34]。

DSP 的 DROM、MP/MC#、OVLY 三个控制位决定了 DSP 的存储空间分配，使得 DSP 可以访问外部的数据空间、程序空间。本系统是在 DSP 内部执行程序，需要的是将内部 ROM 映射到程序空间，使得系统从在片 ROM 的 0xFF80 处开始执行程序，因此设置 MP/MC#=0。此位还必须与 DSP 的外部管脚 MP/MC#一致，即必须设置 DSP 的外围硬件 MP/MC#与地相接。

同时系统的要求是将外部程序存储器中的程序代码加载到内部的高速 RAM 中，需要将 DSP 内部的 RAM 映射到程序存储空间去，因此设置 OVLY=1，使得 DSP 芯片的内部 RAM 可以映射到程序空间和数据空间。但其中需注意的是内部 RAM 映射到程序空间和数据空间的地址空间不可以重叠。例如：DRAM 的 0x0080～0x2000 的地址空间映射到程序空间，则这部分的 RAM 就不能再映射到数据空间，并且也就丧失这段数据空间。

根据系统脱机运行的要求，本系统在 DSP 外扩了一片 FLASH 芯片 SST39VF400A，根据 DSP 系统程序加载的特点，将 FLASH 地址为 0x8000—0xffff 的存储区在程序下载的过程中映射到 DSP 的数据空间。

4. 3 TMS320VC5416 Bootloader 程序设计

DSP 应用系统设计的最后一步是 Bootloader 程序的实现。Bootloader 是对单片机的改进。通常单片机程序是通过把单片机放入专用烧写器中将程序烧入其中的 EEPROM 中，然后将单片机装入功能板上工作[35]。DSP 为了增加系统软件下载调试的灵活性，将这个 EEPROM 等存储器放置到片外，由一片或几片 FLASH ROM 来代替；DSP 的内部 ROM 固化了一个称为 Boot 的程序，在 DSP 上电硬复位（MP/MC=0）后，DSP 自动执行这个 Boot 程序，将外部 FLASH ROM 的系统程序导入 DSP 内部的高速 RAM 程序存储空间中。

可编程的数字化芯片如 TMS320VC5416 从上电复位到进入正常工作状态前一瞬间的这个阶段称为 Boot 阶段。有些简单的可编程数字化芯片，当上电复位后，它的程序指针自动指到一个固定的入口地址，程序设计者必须将程序可执行代码的首地址放在这个入口地址处。对于 C5000 系列 DSP 来说，当上电复位后，程序指针自动指向 ROM 中的一个称为 Bootloader 的程序，这个程序会根据环境选用相应的 bootloader 方法，将外部 FLASH ROM 中的程序搬移到 DSP 内部的 RAM 程序存储空间中，并将程序指针指向执行系统程序的首地址。

4.3.1TMS320VC5416 的 Bootloader 模式

按照 Boot 时系统程序由外部 FLASH ROM 等外部存储器导入到 DSP 片上快速 RAM 的方式不同分为多种 Bootloader 模式。一般地，C5000 系列均支持 HPI、并口、串行口等模式，有的还支持通用 I/O 口等模式；按照数据进入 DSP 时的字长又分为 8 位和 16 位模式[36]。DSP 上电后，会根据系统的设置自动选取相应的 Bootloader 模式。

VC5416 有四种 Bootloader 模式，且具有优先级，按照从高到低的顺序依次为：HPI 模式（增强主机接口）、并口模式、标准串行口模式和通用 I/O 口模式。上电复位后，VC5416 首先判断 INT2 是否有效。当 INT2 被激活时，则采取 HPI 的 Boot 模式；否则不采用 HPI 模式，接着去读取 I/O 空间的 0xFFFFh 地址，当该地址内容有效时，采用并口 Boot 模式；当该地址内容无效时，接着去读数据空间的 FFFFh 地址处，当该地址内容有效时，仍采用并口 Boot 模式。若无效，则初始化串口，然后将 XF 设置为低。当设置完 XF 时，串口主机通过 VC5416 的缓冲串口 McBSP0 或 McBSP2 开始发送 boot 表，

当缓冲串口接收到一个字节数据，则产生一个串口中断，采用串口 Boot 模式。若 Boot 表并不从串口进行传输，则可采用 I/O 口 Boot 模式，主机通过判断 BIO 输入是否为低来决定是否采用 I/O 口 Boot 模式。若 BIO 为低且关键字有效，则从 I/O 口地址为 0x0h 的空间开始读取数据[37]。若仍没有满足要求的，则 Bootloader 程序又会从 HPI 模式开始进行第二次的遍历，直到找到一种能够满足要求的 Boot 模式为止。在本系统中采用了并口模式进行 Boot。

4.3.2 TMS320VC5416 并口 Bootloader 模式

TMS320VC5416 上电复位后，当 INT2 没有中断触发时，MP/MC=0，Bootloader 程序去读取 I/O 空间的 FFFFh 地址。当这个地址包含了有效的地址内容时，Boot 程序将该地址中内容作为 FLASH ROM 中程序的首地址，Boot 程序会从这个首地址开始读取数据并复制到内部程序空间中。FLSAH 中的程序数据按一定规律构成 Boot 表，其中包含所有要下载到 DSP 内部程序区的程序代码。当 I/O 空间的 FFFFh 不含有效数据时，Boot 程序会读取数据空间的 FFFFh 地址处的内容。如果数据有效，Boot 程序将该数据作为 FLASH 中 Boot 表的首地址，去进行 Bootloader 操作。当数据区 FFFFh 处的内容无效时，VC5416 将不采用并口 Boot 模式，进行串行口 Boot 方式的判断。

读取 Boot 表的过程是：首先读取 Boot 表的第一个字，当该字为 0x10AA 时，采用 16 位的读取模式；当第一个字的后面 8 位为 0x08，下一个字的后面 8 位是 0xAA 时，采用 8 位的读取模式；接着读取下面的 R-1 个字初始化寄存器（根据不同的模式 R 具有不同的值，对于并口，R=3）；下面为程序完成 Boot 之后的入口偏移指针 XPC 和指针 PC；再下面的结构是相似的，依次为区段的长度、搬移到目的程序区的 XPC 和 PC 以及所需搬移的区段内容；Boot 程序中依次将各个区段搬移到内部 RAM 程序区中；最后，全部搬移完成后，程序指针会跳到程序入口处，DSP 进入工作状态。

除了 HPI 模式不需要 Boot 表之外，其它模式均需要建立 Boot 表。将系统编译通过的程序生成的可执行文件.out 作为输入文件，利用 Ccstudio 自带的 hex500 程序包在命令行模式下生成这个 Boot 表文件，文件名用户可以自己设置，后缀名为.hex。

4.3.3 FLASH 现场编程

FLASH 现场编程是指通过仿真器和 JTAG 口借助 DSP 芯片将 Boot 表写入 FLASH 的过程[38]。这个过程不需要将 FLASH 从 DSP 功能板上取下来，更不需要借助于编程器等其它的烧写程序的设备，而只限于仿真环境下即可。

现场 FLASH 编程又称为在线 FLASH 编程，这对整个系统有一定的要求，即必须将 FLASH 与 DSP 芯片的 HPI 口、并口或串口等数据通道管脚相连接。在线 FLASH 编程使得系统的灵活性大大加强，下载软件到 DSP 功能板上很方便，还可以通过加载不同的软件完成不同的功能，也可以根据需要不断地升级软件，从而增强了硬件平台的功能，提高了应用范围。

在本系统中选用了一款常用的高档 FLASH 芯片 SST39VF400A 存取程序。SST39VF400A 是一个低功耗 FLASH ROM，工作在 2.7V 至 3.6V 电压下，存储容量为 256KW，其中的数据可以保存 100 年以上，可重复编程次数达 10 万次[39]。

对 FLASH 的编程主要是在软件编程的条件下，对 FLASH 进行擦除、写入数据和读出数据的操作，各种 FLASH 芯片编程的原理都大同小异，都是通过对不同的 FLASH 寄存器写入不同的控制字来实现的。对 SST39VF400A 控制如表 4.1、表 4.2、表 4.3 所示：

表 4.1 FLASH 整片擦除步骤

Tab. 4.1 FLASH Whole erasure process

步骤	1	2	3	4	5	6
地址（0x）	5555	2AAA	5555	5555	2AAA	5555
本示例地址（0x）	D555	AAAA	D555	D555	AAAA	D555
数据（0x）	AA	55	80	AA	55	10

表 4.2 FLASH 编程写入数据步骤

Tab. 4.2 FLASH Write data process

步骤	1	2	3	4
地址（0x）	5555	2AAA	5555	编程地址
本示例地址（0x）	D555	AAAA	D555	编程地址
数据（0x）	AA	55	A0	编程数据

表 4.3 FLASH 进入正常工作状态的步骤

Tab. 4.3 FLASH put in normal work status process

步骤	1	2	3
地址（0x）	5555	2AAA	5555
本示例地址（0x）	D555	AAAA	D555
数据（0x）	AA	55	F0

由此可见，Bootloader 程序设计主要是如何在线写 FLASH。具体的 Bootloader 程序流程如图 4.3 所示：

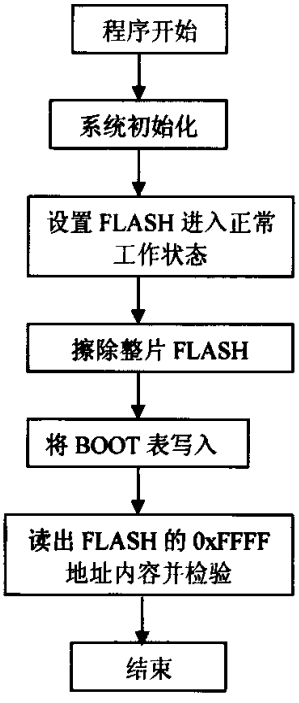


图 4.3 Bootloader 程序流程图

Fig. 4.3 Bootloader programme flow chart

综上所述，FLASH ROM 芯片 SST39VF400A 的编程操作步骤如下[40]：

第一步：通过编程使 SST39VF400A 进入正常工作状态(如果 SST39VF400A 已处于正常状态，这一步可以省略)。

第二步：将要编程存储数据的 FLASH 空间擦除。一般地，写数据到 FLASH 内部时，通常应该首先擦除这部分空间的内容；否则，可能会出现误码。因此，这一步是不可缺少的。

第三步：将 BOOT 表编程写入到 SST39VF400A 中。写入过程是由 CCStudio 的软件控制的，即写入的详细地址由软件控制。

第四步：将写入 SST39VF400A 中的内容读出，进行读出校验。

5 异步串口通信模块

本部分的设计目标是实现 DSP 系统模块与以太网接口系统模块的数据交换，即实现 TMS320VC5416 DSP 微处理器与单片机之间的数据通信。目前 DSP 与单片机之间的通信主要有串行和并行两种连接方式。

（1）DSP 与单片机通过 HPI（主机接口）以并行方式实现通信

DSP 芯片中的 HPI（主机接口）是为了满足 DSP 与其它的微处理器接口而专门设计的。它分为 HPI-8 和 HPI-16，分别针对具有 8 位和 16 位数据线的单片机。每一种又分为标准型和增强型。两者的区别在于标准型只可以访问固定的地址空间，而增强型可以访问整个 DSP 的片内存储器。TMS320V5416 使用的是增强型的 HPI8。

HPI 接口通过 DSP 片内的 DMA 控制器来访问片内存储器，不需要 DSP 的干预，从而大大节约了 DSP 对于通信过程的处理时间。可以说，HPI 接口是 DSP 的一个“后门”，单片机通过这个“后门”可以访问到 DSP 的片内存储器。只有当 HPI 接口和 DSP 同时对同一地址进行访问时，由于 HPI 具有访问优先权，这时 DSP 的执行会被延迟一个周期，而这种情况对系统实时性的影响是非常小的。所以目前大多数 DSP 与单片机结合的嵌入式系统都选择此种方式解决微处理器之间的接口问题。

（2）DSP 与单片机的串行通信方式

串行通讯分为同步串行通讯和异步串行通讯,同步串行通讯能成批地传输数据,其控制复杂,速度快,而异步串行通讯则比较灵活,适应于数据的随机发送与接收,控制比较简单,但速度较同步通讯慢。DSP 与单片机的串行通信通常选择异步方式。TMS320V5416 不像单片机那样已经普及了异步串行通信模块，片上只带有 2 个多通道同步缓冲串口（MCBSP），所以对于异步数据传输的应用必须在其片上扩展出异步串行接口。

综上所述，与并行方式相比，串行方式优点是减少了引脚数目、降低了设计的复杂性。缺点则是通信速度较慢，且占用系统资源。

本系统中的以太网接入系统模块为 5V 供电方式，接口输入输出信号为 TTL 电平。而 TMS320VC5416 DSP 芯片则为 1.8/3.3V 供电，接口输入输出信号为 CMOS 电平。所以 DSP 与单片机的并行方式连接需要进行电平转换，这无疑在原有并行方式复杂的硬

件连接上更增加了设计的难度与复杂性。所以，在满足系统通信速度要求的前提下，本系统选择使用串行方式实现本系统中 DSP 与单片机的数据通信。并且针对前面内容所述的 DSP 缺乏异步串口的情况，研究并实现了一种简单、可靠的串口连接方法。

5.1 扩展方案提出

目前，在 DSP 应用系统中扩展异步串口的方法有很多，其基本方法和优缺点分析如下：

（1）用软件模拟实现 UART。早期的定点 DSP(如 TMS320C2X 和 TMS320C5X 等)片上没有集成异步串行通信电路,一般是利用芯片的 I/O 信号线和一些中断信号相配合实现与 PC 机串行通信的。此种方法无需额外硬件开销,仅仅需要两个通用 I/O 脚(BIO 和 XF)、外部中断 INT0 以及一个定时器就可实现,在硬件连接上是非常简单的，但软件程序复杂，增加了 CPU 的负担，所以不适合通信数据量大的场合。

（2）在 DSP 的并行总线上扩展 UART 芯片，用硬件来实现异步数据传输。其优点是软件实现简单，缺点是在总线上还要扩展其它设备，使目标系统复杂化。

（3）利用 DSP 的多通道缓冲串行接口（McBSP），在扩展适当硬件的情况下，将同步数据变换为异步数据格式进行传输，这样就充分利用了 DSP 的片上资源，使硬件系统尽量简单化。

综合考虑三种串行传输方式硬件连接和软件编程的应用特点和本系统的性能要求，本系统采用第三种方案，应用 MAXIM 公司的异步串行收发器 MAX3111E 与 VC5416 的 McBSP 构成串行通讯接口。MAX3111E 是专门为小型微处理系统所设计的 UART，包括一个振荡器和一个可编程波特率发生器、一个可屏蔽的中断源、一个 8 字节的接收 FIFO 等。其应用 SPI 接口技术直接与主控制器之间进行通信，通信速率可达 230kbps，它还包括四个 RS-232 电平转换器，无需再接入额外的 MAX232 进行电平转换，这样只需一个芯片就实现了具有 SPI 接口的微控器与 PC 进行异步数据传输的功能。由于异步数据的发送和接收由 MAX3111E 以硬件方式实现，所以就软件编程而言，需要考虑的也只是 VC5416 与 MAX3111E 之间的同步数据通信。这样，就实现了以最简单的硬件连接和软件编程实现同步到异步的串行数据格式转换。

5.2 SPI 协议介绍

串行外设接口 SPI 是 MOTOROLA 公司推出的一种同步串行接口，是目前比较常用的串行总线接口标准。SPI 协议是一种主从方式工作的同步串行接口，该协议通常要求一个主设备和一个或多个从设备，其包括以下四种信号：

（1）串行数据输入信号（主入从出或 MISO）；

（2）串行数据输出信号（主出从入或 MOSI）；

（3）移位时钟信号（SCK）；

（4）从器件使能信号（SS#）。

主器件通过提供移位时钟信号 SCK 和从器件使能信号 SS#控制通信的进程。从器件使能信号 SS#是一个可选的低有效信号，由它控制从器件是否进行串行数据输入和输出。一个典型的 SPI 接口原理图如图 5.1 所示。

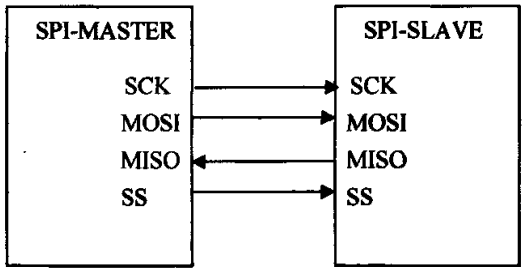


图 5.1 典型 SPI 接口图

Fig. 5.1 Tapical SPI Interface

5.3 McBSP 的时钟停止模式

McBSP 的时钟停止模式与 SPI 协议相兼容，即这时的 McBSP 可以作为 SPI 器件来使用。系统中将 VC5416 配置为 SPI 的主器件，MAX3111E 配置为 SPI 的从器件。为此必须对 McBSP 的寄存器进行相应的设置，才能将 VC5416 和 MAX3111E 进行连接，构成串口通讯电路。

串行口控制寄存器 SPCR1 中 CLKSTP 位和管脚配置寄存器 PCR 中的 CLKXP 位用于设置是否使 McBSP 工作在时钟停止模式下，它们两者共同提供了时钟停止模式的四种结构。CLKSTP 和 CLKXP 的设置参数如表 5.1 所示。

表 5.1 时钟停止模式的配置

Tab. 5.1 clock stop mode configuration

CLKSTP	CLKXP	时钟模式
0X	X	禁止时钟停止模式。
10	0	上升沿非延时模式。McBSP在CLKX上升沿发送数据，在CLKR下降沿接受数据。
11	0	上升沿延时模式。McBSP在CLKX上升沿的前半个时钟周期发送数据，在CLKR上升沿接受数据。
10	1	下降沿非延时模式。McBSP在CLKX下降沿发送数据，在CLKR上升沿接受数据。
11	1	下降沿延时模式。McBSP在CLKX下降沿的前半个时钟周期发送数据，在CLKR下降沿接受数据。

系统中，VC5416 通过 PCR 寄存器中的 CLKXM 位设置为 SPI 的主器件，所以它必须为 MAX3111E 提供串行时钟信号来控制数据的传输。因此发送时钟信号（BCLKX）必须设置为输出状态且输出的时钟信号只在数据包传输期间有效。当没有数据传输时，BCLKX 管脚受选择极性的控制而保持在高电平或低电平。McBSP 的采样率发生器时钟源（CLKSM）和时钟降频因子（CLKGDV）用于选择驱动 BCLKX 信号的时钟源和时钟频率，在时钟停止模式下发送时钟和接受时钟在内部实现同步，因此数据的发送和接收都是由 BCLKX 来提供时钟信号。

McBSP 还可通过发送帧同步信号（BFSX）来提供 MAX3111E 使能信号（SS#）。BFSX 管脚应配置成输出状态，并对帧发生器进行适当的配置，使之在每个数据包传输期间产生帧同步脉冲，即在数据包传输的第一位变为有效状态，然后保持此状态直到数据包传输结束。尽管可以进行数据的连续传输，但在每帧数据传输结束后 BCLKX 信号会停止，BFSX 信号会呈现无效状态。因此在连续的数据传输进程中，在每帧数据传输结束后都会产生两个时钟周期的空闲态。

数据延迟参数（XDATDLY 和 RDATDLY）在 SPI 主器件模式下要设置为 1b，因为在此模式下 0b 或 2b 没有意义。需注意的是在 McBSP 工作在时钟停止模式时，由于发送器和接收器使用同一个时钟信号，所以 RWDLEN1 和 XWDLEN1 必须设置为相同的值。详细的参数设置如表 5.2 所示。

表5.2 VC5416作为SPI主器件时需设置的参数

Tab.5.2 VC5416 in the master SPI mode configuration

位域	值	功能	寄存器
CLKSTP	1xb	使能时钟停止模式并选择延时模式	SPCR1
CLKXP	0bor1b	配置BCLKX信号的极性	PCR
CLKXM	1b	配置BCLKX为输出状态，即设置VC5416为主器件	PCR
CLKSM	1b	采样率时钟信号从CPU时钟中抽取	SRGR2
CLKGDV	1—255	设置采样率时钟分频系数	SRGR1
FSXM	1b	配置BFSX为输出状态	PCR
FSGM	0b	配置BFSX在每次数据传输时有效	SRGR2
FSXP	1b	设置BFSX为低电平有效	PCR
XDATDLY	01b	为BFSX信号提供正确的建立时间	XCR2
RDATDLY	01b	为BFSX信号提供正确的建立时间	RCR2
RWDLEN1	000b--101b	设置接收数据包的长度（和XWDLEN1一致）	RCR1
XWDLEN1	000b--101b	设置发送数据包的长度（和RWDLEN1一致）	XCR1

McBSP的帧同步和时钟源等控制信号，都是通过软件设置的。因此，McBSP中的各个模块的启动/激活次序对串口的正常操作是极为重要的。当McBSP工作在时钟停止模式时，其初始化必须按如下的顺序来进行：

（1）设置XRST#=RRST#=0，复位发送器和接收器。

（2）按表5.1、5.2设置McBSP各个寄存器的相应位的值，其余各位使用默认值。

（3）保持上述各位设置不变，设置GRST#=1，使采样率发生器退出复位状态。

（4）为保证McBSP逻辑正确，等待两个采样率时钟周期。

（5）设置XRST#=RRST#=1，使发送器和接收器退出复位状态。

（6）为保证McBSP逻辑正确，再等待两个采样率时钟周期。

5.4 硬件电路的设计

MAX3111E通过SPI接口与VC5416进行16位数据的全双工同步通信，即DSP向MAX3111E传送16位数据的同时也接收来自MAX3111E发送的16位数据。DSP所发送的16位数据中除了包含要发送的数据外，还包括传输格式控制字，而MAX3111E向主设备发送的16位数据中除了接收到的数据外，还包括中断标志等状态位。这样，两个设备控制、状态、数据信息的实时通信保证了数据传输的可靠性和稳定性。

VC5416作为主设备，MAX3111E作为从设备时接口电路如图5.2所示。

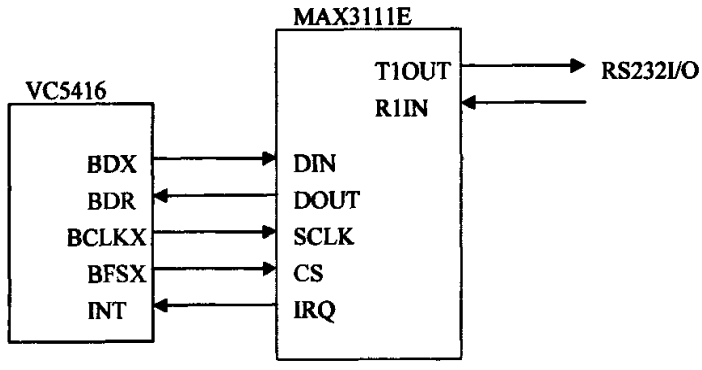


图 5.2 VC5416 和 MAX3111E 接口电路图

Fig. 5.2 VC5416 and MAX3111E Interface Circuit

DSP 的发送时钟信号（BCLKX）作为 MAX3111E 的串行时钟输入，发送帧同步脉冲信号（BFSX）作为 MAX3111E 的片选信号。 发送器输出信号（BDX）与 DIN 连接作为发送数据线，接收器输入信号（BDR）与 DOUT 连接为接收数据线。MAX3111E 的 TX 与 T1IN 连接，RX 与 R1OUT 连接，从而利用其片内的转换器实现 TTL 到 RS-232 电平的转换。MAX3111E 的中断信号(IRQ#)与 DSP 的外部中断相连。

SPI 串行协议中，主设备提供时钟信号并控制数据传输过程。因此，必须根据 MAX3111E 接口电路时序将 VC5416 的 McBSP 设置于适当的方式下才能保证其与 MAX3111E 的时序相匹配。

McBSP 的时钟停止模式 2（CLKSTP=11b, CLKXP=0b）的工作时序和 MAX3111E 数据收发格式相吻合，所以在系统中采用时钟停止模式 2 作为 VC5416 与 MAX3111E 协同工作的时序标准。

5.5 软件程序设计

5.5.1 McBSP 初始化程序设计

串口通讯程序包括将 VC5416 设置为 SPI 主器件的初始化程序和用于收发数据的传输程序。其中初始化程序是保障串行接口快速、准确的进行数据传输的基础。

McBSP 通讯初始化有严格的顺序，具体初始化程序如下所示：

（1）设置 XRST#=RRST#=0，复位发送器和接收器。

//系统复位后，RRST#=XRST#=0，所以在程序中不用再设置这两位为0。

（2）按表 5.1、5.2 设置 McBSP 各个寄存器的相应位的值，其余各位使用默认值。

```
*(volatile u16 *) SPSA_ADDR(1) = SPCR1;
*(volatile u16 *) SPSD_ADDR(1) = 0x1800;
//设置CLKSTP=11，使能McBSP的SPI功能。
*(volatile u16 *) SPSA_ADDR(1) = PCR;
*(volatile u16 *) SPSD_ADDR(1) = 0x0A08;
//设置CLKXP=0；CLKXM=1；FSXM=1；FSXP=1。
*(volatile u16 *) SPSA_ADDR(1) = SRGR1;
*(volatile u16 *) SPSD_ADDR(1) = 0x001D;
//设置CLKGDV=29，即设置时钟分频系数为30。
*(volatile u16 *) SPSA_ADDR(1) = SRGR2;
*(volatile u16 *) SPSD_ADDR(1) = 0x2000;
//设置CLKSM=1；FSGM=0。
*(volatile u16 *) SPSA_ADDR(1) = RCR1;
*(volatile u16 *) SPSD_ADDR(1) = 0x0040;
//设置RWDLEN1=010，即设置数据包长度为16位
*(volatile u16 *) SPSA_ADDR(1) = RCR2;
*(volatile u16 *) SPSD_ADDR(1) = 0x0001;
//设置RDATDLY=01，给BFSX信号提供正确的建立时间。
*(volatile u16 *) SPSA_ADDR(1) = XCR1;
*(volatile u16 *) SPSD_ADDR(1) = 0x0040;
//设置XWDLEN1=010，即设置数据包长度为16位。
*(volatile u16 *) SPSA_ADDR(1) = XCR2;
*(volatile u16 *) SPSD_ADDR(1) = 0x0001;
//设置XDATDLY=01，给BFSX信号提供正确的建立时间。
```

（3）保持上述各位设置不变，设置 GRST#=1，使采样率发生器退出复位状态。

```
*(volatile u16 *) SPSA_ADDR(1) = SPCR2;
```

```
*(volatile u16 *) SPSD_ADDR(1) = 0x0040;
//在不改变先前设置的情况下，设置GRST#=1，使采样率发生器退出复位状态。
```

（4）为保证 McBSP 逻辑正确，等待两个采样率时钟周期。

```
delay(100); //设置一定的延时，来满足配置时序的要求。
```

（5）设置 XRST#=RRST#=1，使发送器和接收器退出复位状态。

```
*(volatile u16 *) SPSA_ADDR(1) = SPCR1;
*(volatile u16 *) SPSD_ADDR(1) = 0x1801;
//在不改变先前设置的情况下，设置RRST#=1，使接收器退出复位状态。
*(volatile u16 *) SPSA_ADDR(1) = SPCR2;
*(volatile u16 *) SPSD_ADDR(1) = 0x0041;
//在不改变先前设置的情况下，设置XRST#=1，使发送器退出复位状态。
```

5.5.2 数据收发程序的设计

MAX3111E 的四种操作，即数据发送、数据接收、芯片配置和状态读取，通过高两个数据位（D15 和 D14）来区分，通过写操作来完成。在进行任何写操作时，MAX3111E 将相应的状态值和接收到的数据送到 McBSP 的接收转换寄存器（RSR），在 McBSP 的接收转换寄存器（RSR）中的数据自动通过接收缓冲寄存器（RBR）到达数据接收寄存器（DRR）中。对于 DSP 的 McBSP，DRR 寄存器具有数据保持功能，只有 DRR 中的数据被读取后，RBR 中的数据才会复制到 DRR 中。然而 RBR 不具有数据保持功能，即当 RSR 接收到 16 位数据后，就会立即更新 RBR，从而造成接收数据的丢失。当进行读操作时，通过 DRR 读出来的数据是以往没有读过的数据，而不一定是想要的数据。因此在编写程序时，每执行一次写操作后，不管 DRR 中的数据是否有用，都要对其进行一次读操作。然后根据读取的数据是否有效来选择相应的处理方式。

5.5.3 中断服务程序的设计

MAX3111E 有四个中断源，即 Pr（接收帧校验位中断）、R（接收数据有效中断）、RA/PE（帧格式错误中断）、T（发送缓冲器空中断），但只有一个中断输出管脚。因此当中断产生时，VC5416 必须要对 3111E 进行一次读操作，通过读取数据中的 R 和 T 等标志位来判断中断类型，然后进行相应的操作。

在进行中断方式数据传输时，虽然 VC5416 的 McBSP 有自身的发送和接收中断，但由于 McBSP 与 MAX3111E 同步串行数据传输速率高于 MAX3111E 将数据以一定波特率异步发送速率，如果应用 McBSP 的发送中断，将造成发送数据的丢失。同时在 SPI 协议中，数据的传输是由 SPI 主设备发起的，所以在 SPI 方式下的 McBSP 并不能产生接收中断。因此，系统设计中的关键之一是将 MAX3111E 的中断信号连接至 DSP 的外部中断，以实现在中断方式下可靠、正确的数据传输。

5.5.4 串口通信设计

在扩展了异步串口以后，DSP 就可以按照预先设置的波特率和通信协议与单片机进行串口通信了。通信协议数据帧格式内容如下表 5.3：

表 5.3 通信协议数据帧格式

Tab. 5.3 protocol data frame format

STX	LENGTH	CMD	DATA	BCC（校验位）	ETX

STX 为串行通信开始标志字段，值为 0x7E。即当系统收到数据为 0x7E 时，后面所接收到的数据为有效通信数据。

LENGTH 为标志通信帧中有效数据长度的字段。DATA 为数据帧中所包含的有效数据内容的字段，其最大长度为 80 字节。

CMD 为命令字段，以太网接口系统模块根据该字段的值对数据帧中的有效数据信息进行相应的处理。命令值与相应操作对应关系如下：

0x00： 以太网接口系统模块将 DATA 字段中的数据发送到以太网。

0x01：以太网接口系统模块将 DATA 字段中的数据通过串口发回到 DSP 系统模块。

0x02： DATA 字段中的数据作为系统接入网络的 IP 地址、MAC 地址以及通信端口号，且前四个字节的数据为 IP 地址，中间六个字节作为 MAC 地址，最后两个字节为端口号。

0x03： DATA 字段中的数据作为系统的网络通信模式选择标志。DATA 字段数据长度为 1 个字节，如果值为 0x33，则系统作为客户端接入网络。否则，系统将作为服务器监听相应的端口，等待客户端系统的连接请求。

0x04：请求发送数据。

BCC 是奇偶校验和字段。计算方法为：BCC＝DATA[0]⊙DATA[1]⊙……⊙DATA[LENGTH-1]。

ETX 为串行通信结束标志字段，值为 0x55。当系统接收到 0x55 时，一帧通信数据帧结束。

在发送方有数据发送时，先检测线路是否空闲，如线路空闲则向接收方发送“请求发送数据”命令帧。接收机收到后，如准备就绪，则回送“可以发送”的确认命令。发送方得到确认后开始发送数据。接收方对收到的每一帧数据进行和校验，校验正确发送“和校验正确”命令，发送方发送下一帧；否则发送“重发”命令，发送方重发此帧。协议中还具有等待超时处理、帧同步处理和线路冲突检测功能。VC 5416 DSP 与单片机通信数据的发送和接收程序流程图如图 5.3、5.4 所示：

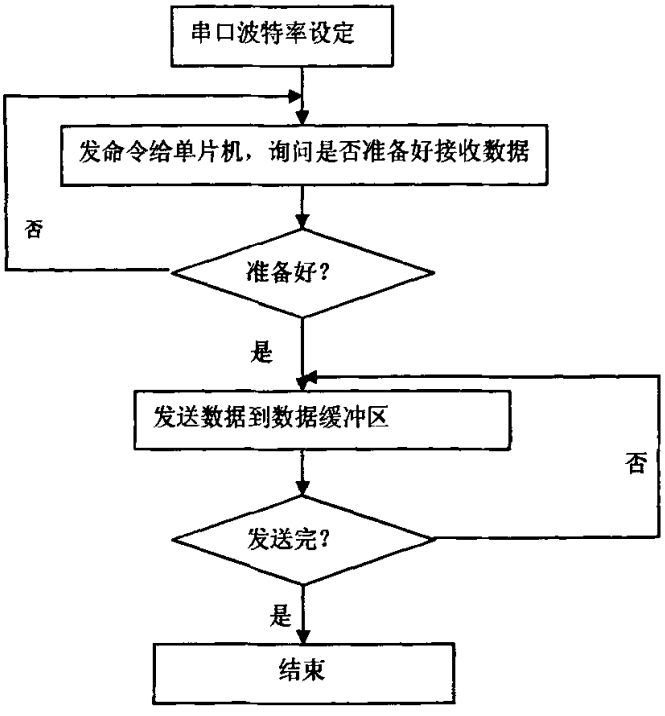


图 5.3 DSP 数据发送程序流程

Fig. 5.3 Send data programme flow chart

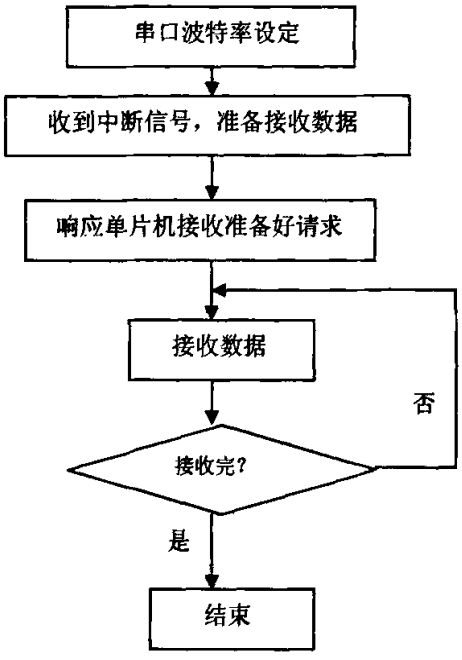


图 5.4 DSP 数据接收程序流程

Fig.5.4 Receive data programme flow chart

6 以太网接入系统模块的设计

6.1 网络接口控制器芯片简介

网络接口控制器(Network Interface Controller，简称 NIC)是网卡上的一个核心控制芯片，它负责完成开放系统互联参考模型(OSI/RM)的数据链路层的功能。通常所使用的网卡就是 NIC 与其它外围芯片与电路构成的。而网卡性能的好坏主要取决于 NIC 的结构功能和控制 NIC 的驱动程序。另外，它也是基于网卡进行网络编程的主要控制对象，程序通过对 NIC 的直接控制，就可以实现特定目的的网络软件。

6.1.1 网络接口控制器芯片的分类

网络接口控制器的种类很多，按照它所实现的数据链路层协议，把网络接口控制器分成以下几类：

(1)基于以太网的 NIC：这种类型的 NIC 是根据工 EEE802.3 的 MAC 层标准设计的，主要用于以太网环境。常见的 NIC 有 National Semiconductor 公司的 DP8390C/NS32490C 系列和 Intel 公司的 82501 系列。

(2)基于令牌总线网的 NIC：该类型的 NIC 是根据 I EEE802.4 的 MAC 层协议标准设计的，主要用于令牌总线网络环境。

(3)基于令牌环网的 NIC：该类型的 NIC 是根据 IEEE802.5 的 MAC 层协议标准设计的，主要用于令牌环网络环境。

本系统选用的是基于以太网的 NIC(又称以太网控制器，下文皆以此称)，又由于主机用的是 8 位单片机，因此要求所选的以太网控制器必须支持 8 位工作模式。实际上，只有部分基于 ISA 总线的以太网控制器才能满足此条件，所以基于 PCI 总线的以太网控制器不在考虑之列。其次，考虑到以太网控制器的片上缓存，具有足够片上缓存的以太网控制器，可以简化系统设计。所以系统采用了 REALTEAK 公司的 RTL8019AS 网络接口控制器。

6.1.2 网络接口控制器内部功能模块

NIC 主要用来控制网络和主机间的数据传输，主要由以下几个模块来实现数据传输。

接收功能模块：该模块主要用来把串行输入的比特流进行 8 位分割，然后把分割后的比特流(8 位)并行送入 16 字节的 FIFO。激活接收字节计数器对帧长度进行计数，同时，必须把串行输入的数据送往 CRC 生成逻辑，进行帧校验。

CRC 产生校验功能模块：该功能模块有两个作用：

（1）当发送数据时，它用来产生较验和，并把该校验和附在数据之后一起送入网络，供接收方用来校验接收到的数据。

（2）当接收数据时，它用来校验接收到的数据是否有错，并把错误信息存入接收状态寄存器，供接收程序检查。

发送串行功能模块：该功能模块负责把 FIFO 中的并行数据转变成串行数据并发送，同时把串行化数据送入 CRC 产生校验功能模块，以计算校验和，该校验和附加在数据帧之后发送出去。

地址识别逻辑：该功能模块用来识别接收到的帧的目的地址域中的 6 字节长的地址，并与地址寄存器组中的地址进行地址匹配。若不匹配，则拒绝接收该帧。判断依据是本地网卡的物理地址和多址寄存器组中的地址是否与接收帧的目的地址相符。

FIFO 和 FIFO 控制逻辑：NIC 只有 16 字节的 FIFO。当发送数据时，本地 DMA 把数据写入 FIFO，发送串行模块把 FIFO 中的数据并行读走，并串行发送该数据。当接收数据时，接收功能模块把并行(8 位)写入 FIFO，再由本地 DMA 把 FIFO 中的数据存入缓冲区。FIFO 控制逻辑用来对进入 FIFO 的字节数进行计数，以便在发生溢出前能够把数据取走。

协议逻辑阵列：该功能模块主要负责实现 IEEE802.3 协议，包括冲突的随机退避，装帧(加帧头)和拆帧(去帧头)，以及实现接收同步。

DMA 和缓冲控制逻辑：DMA 和缓冲控制逻辑用来控制本地 DMA 和远端 DMA，两个 16 位的 DMA 通道。当接收数据时，本地 DMA 用来把接收到的数据存入接收数据缓冲环；当发送数据时，本地 DMA 把发送缓冲区中的待发送数据传送到 FIFO。远端 DMA 用来实现本地缓冲区和主机间的数据传输。本地 DMA 的优先级高于远端 DMA。DMA 方式大为简化了该芯片的使用，提高了主机和网络接口、网络接口和传输线路之间数据自动传输的能力。

6.2 RTL8019AS 结构原理

6.2.1 特点

RTL8019AS 网络接口控制器是一个高度集成的芯片，它提供了遵守 IEEE802.3 标准所有的 MAC 和编解码功能。网络接口包括基于 AUI 的 10BASE－5 或 10BASE－2 和基于双绞线的 10BASE-T。RTL8019AS 以太网控制器能直接连到 PC-AT ISA 总线接口而无需任何外部设备。PC-AT ISA 总线接口完全兼容于 NE2000 以太网适配卡，因此所有面向 NE2000 设计的软件都能运行在 RTL8019AS 网卡上而无需调整。同时它也支持微软的即插即用 PnP 方式和跳线配置方式。PnP 和非 PnP 自动转换功能允许用户配置网卡。集成的 8Kx16 SRAM 和 10BASE-T 收发器使 RTL8019AS 更加节省成本。

RTL8019AS 具有如下主要特点:

- 单片集成解决方案，包括 IEEE802.3, 10BASE-T, 10BASE－5 和 10BASE－2;
- 集成 ISA 总线接口，8Kx16 SRAM，MAC, ENDEC 和 10BASE-T 转换器;
- 支持 ISA 即插即用配置;
- 与 NOVELL NE2000 兼容，软件移植性好;
- 支持 PnP 和非 PnP 自动转换模式;
- 可选择 PnP、非 PnP 或自动转换模式软件之一;
- 8 位中断线可选;
- 自动极性检测和校正;
- 可选择 8 或 16 位插槽模式;
- 提供对 10BASE-T 收发器和 AUI 的自动检测/自动转换功能;
- 支持 BOOT-ROM 页模式;
- 外部 EEPROM 可编程;

6.2.2 内部结构及工作原理

按数据链路的不同，可以将 RTL8019AS 内部划分为远端 DMA(remote DMA)通道和本地 DMA (local DMA)通道两个部分。本地 DMA 完成控制器与网线的数据交换，主处理器收发数据只需对远端 DMA 操作。当主处理器要向网上发送数据时，先将一帧数据通过远端 DMA 通道送到 RTL8019AS 中的发送缓存区，然后发出传送命令。RTL8019AS

在完成了上一帧的发送后，再完成此帧的发送。RTL8019AS 接收到的数据通过 MAC 比较、CRC 校验后，由 FIFO 存到接收缓冲区，收满一帧后，以中断或寄存器标志的方式通知主处理器。原理框图如图 6.1 所示。

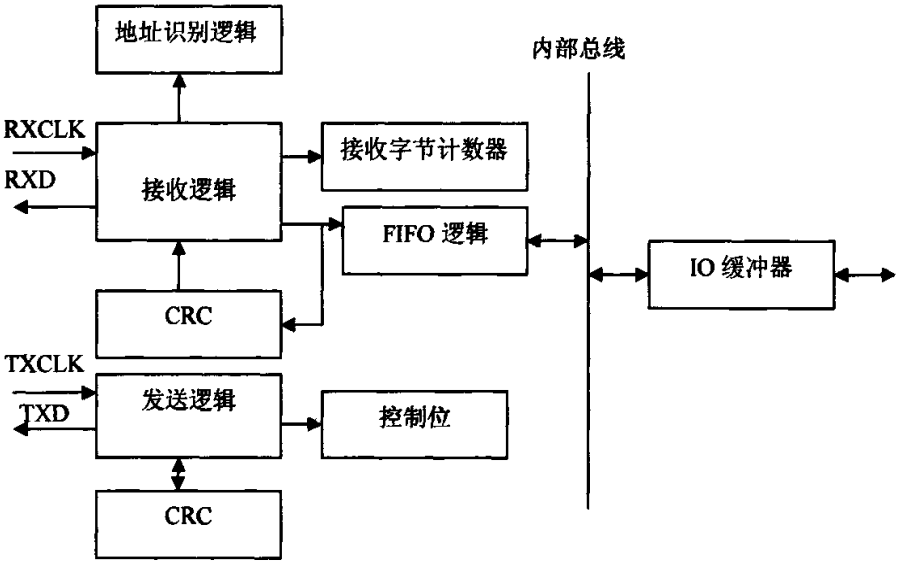


图 6.1 RTL8019 原理框图

Fig. 6.1 RTL8019 principle chart

在图 6.1 中，接收逻辑在接收时钟的控制下，将串行数据拼成字节送到 FIFO 和 CRC；发送逻辑将 FIFO 送来的字节在发送时钟的控制下逐步按位移出，并送到 CRC；CRC 逻辑在接收时对输入的数据进行 CRC 校验，将结果与帧尾的 CRC 比较，如不同，该帧数据将被拒收，在发送时 CRC 对帧数据产生 CRC，并附加在数据尾传送；地址识别逻辑对接收帧的目的地址与预先设置的本地物理地址进行比较，如不同且不满足广播地址的设置要求，该帧数据将被拒收；FIFO 逻辑对收发的数据作 16 个字节的缓冲，以减少对本地 DMA 请求的频率。

6.3 硬件接口电路设计

本设计采用 51 系列单片机（STC89C516RD+）与 RTL8019AS 的直接接口，实现的网络接口采用 UTP（无屏蔽双绞线）RJ-45 接口。用到的主要芯片有（STC89C516RD+）单片机、RTL8019AS、74HC573（8 位锁存器）、62256（32K 字节的 RAM）。这

里需要指出的是 RTL8019AS 的端口 I/O 基地址和以太网物理地址等设置信息被直接写到(STC89C516RD+)单片机的 EEPROM 中,所以系统上电后可从单片机内部 EEPROM 中直接读取配置参数对系统进行配置。

RTL8019AS 具有 32 位输入输出地址，地址偏移量为 00H～1FH。其中 00H～0FH 共 16 个地址，为寄存器地址。寄存器分为 4 页：PAGE0、PAGE1、PAGE2、PAGE3，由 RTL8019AS 的 CR（Command Register 命令寄存器）中的 PS1、PS0 位来决定要访问哪一页的 16 个寄存器。远程 DMA 地址包括 10H～17H，都可以用来做远程 DMA 端口，只要用其中的一个就可以了。复位端口包括 18H～1FH 共 8 个地址，功能一样，用于 RTL8019AS 复位。

配置寄存器 CONFIG1 的默认值为 00H，CONFIG1 的低 4 位 IOS3～0 用于选择 I/O 基地址。IOS3～0 值均为 0 时，RTL8019AS 选择的端口 I/O 基地址为 300H。RTL8019AS 的地址为 20 位，那么用到 RTL8019AS 的地址空间为 00300H～0031FH，用二进制表示 00300H～0031FH，可以发现第 19 位到第 5 位是固定的：000000000011000。根据上述分配原则，RTL8019 AS 的 20 根地址线 SA0～SA19 的物理连接方法如下：

SA19～SA10 接地 ；SA9～SA8 接单片机 P2 口的 P2.7，即地址总线 ADDR15 ；SA7～SA5 接地；SA4～SA0 对应为地址总线的 ADDR0～ADDR4 。

通过 ADDR15、I/OW、I/OR 来划分 RTL8019AS 和 62256 的地址空间。ADDR15 接 62256 的 CE 脚，低电平时选择 62256；高电平时选择 RTL8019AS 的地址空间。（STC89C516RD+）单片机的地址为 16 位，在程序里则通过使用 8000H～801FH 来选中的 RTL8019AS 地址空间。图 6.2 是硬件接口电路图。

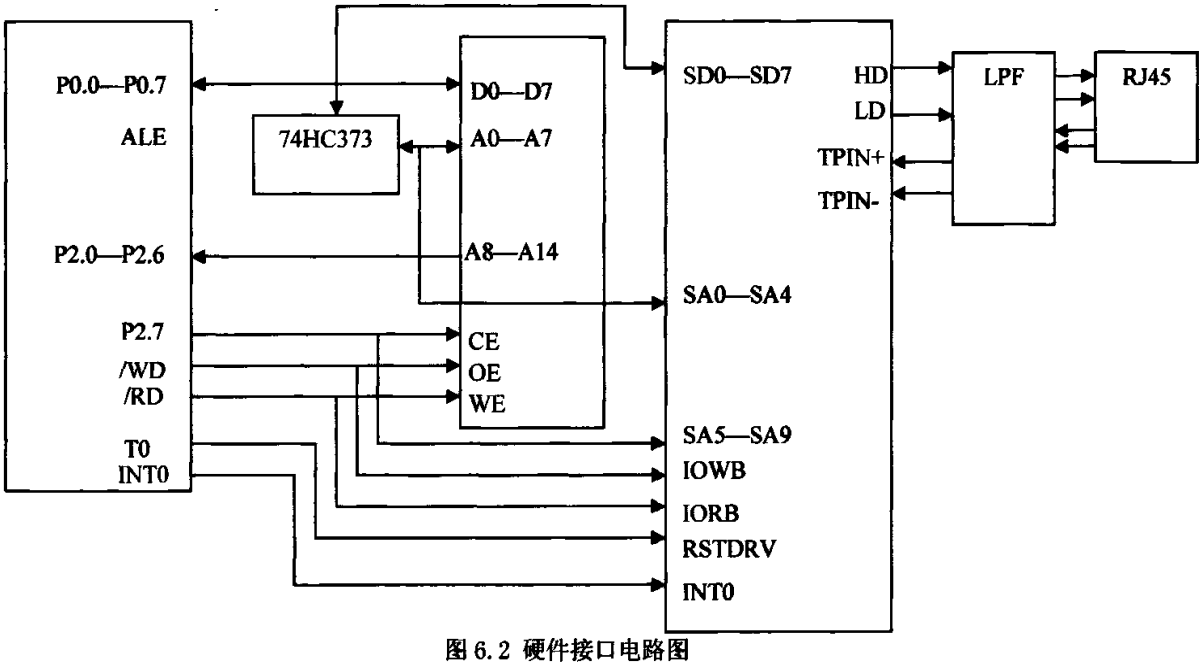


图 6.2 硬件接口电路图

Fig. 6.2 Hardware Interface Circuit

6.4 系统驱动程序设计

系统驱动程序包括 RTL8019AS 初始化程序和数据帧收发程序两部分。

6.4.1 RTL8019AS 初始化程序

为了启动网络接口控制器 RTL8019AS，并使之处于准备接收或准备发送的状态，必须对其进行初始化。

首先初始化页 0 和页 1 的寄存器：

（1） 初始化命令寄存器 CR:CR=0x21,选择页 0 的寄存器。

（2） 初始化数据配置寄存器 DCR:DCR=0xc8，使用 FIFO 缓存，普通模式，8 位数据 DMA。

（3） 初始化远程字节计数寄存器 RBCR ：指定远程 DMA 操作时传输数据的字节数，RBCR1=0，要读取的字节数的计数（高 8 位）；RBCR0=12，要读取的字节数的计数（低 8 位）共要读取 12 个字节。

（4） 初始化接收配置寄存器 RCR：RCR=0xcc，使用接收缓冲区，仅接收自己地址的数据包（以及广播地址数据包）和多点播送地址包，小于 64 字节的包丢弃，校验

错的数据包不接收。

（5） 初始化发送配置寄存器 TCR：TCR=0xe0，启用 CRC 自动生成和自动校验，工作在正常模式。

（6） 初始化接收缓冲环：PSTART=0x4c，PSTOP=0x80，构造缓冲环：0x4c～0x80。BNRY=0x4c。

（7） 初始化中断状态寄存器 ISR：ISR=0xff，初始化时，将各位值设置为 1，表示在初始化状态下，无中断发生。

（8） 初始化中断屏蔽寄存器 IMR： IMR=0x00，屏蔽所有中断。

（9） 重置命令寄存器 CR，使其选择页 1。

1） 初始化物理地址寄存器 PAR0—PAR5。

2） 初始化多址寄存器 MAR0—MAR7：均设置为 0x00。

3） 初始化当前页寄存器 CURR： CURR=0x4d，CURR 是 RTL8019AS 写内存的指针，指向当前正在写的页的下一页。

（10）重置命令寄存器的值，将 NIC 置于开始模式，即 CR=0x22。

6.4.2 RTL8019AS 数据帧接收程序

数据帧接收是指将网络上的数据帧接收并存于网络接口控制器的接收缓冲环中，而后由主控器程序将缓存于接收缓冲环的帧读走，并存入系统数据存储器供其它程序使用。

NIC 通过本地 DMA 自动将帧存入接收缓冲环，当条件 CURR-BNRY！=1 成立时，表示接收缓冲区中有数据，则进行下一步。

远程 DMA 在主控器的控制下将接收缓冲环中的帧读入系统数据存储器。具体程序实现过程如下：

（1） 设置命令寄存器 CR:CR=0x22，选择页 0 的寄存器。

（2）设置地址寄存器 RSAR:RSAR0=0,读取数据块的缓存起始地址(低 8 位);RSAR1=BNRY,读取数据块的缓存起始地址（高 8 位）。

（3） 设置远程字节计数寄存器 RBCR ：指定远程 DMA 操作时传输数据的字节数，RBCR1=0，要读取的字节数的计数（高 8 位）；RBCR0=4，要读取的字节数的计数（低 8 位）共要读取 4 个字节。

（4） 设置命令寄存器 CR：CR=0x0a，远程读 DMA。

（5） 读数据端口，读出 4 个字节：第 1 个字节表示接收状态，第 2 个字节为下一包开始地址指针，第 3～4 个字节为本数据包的长度（高位字节在前）。

（6） 根据接收状态，判断数据包是否接收正确。

（7） 如果接收正确，启动远程 DMA，收取该数据包并进行处理。

（8） 结束远程 DMA：CR=0x22。

（9） 重新设置读指针 BNRY，满足条件 BNRY=CURR-1，为下一次帧接收作准备。

6.4.3 RTL8019AS 的数据帧发送

数据帧的发送是指将待发送的数据以帧的形式发送到网络传输线上的过程，因此，帧的发送过程应包括以下步骤：

(1)装帧：数据包在发送前应该按规定的格式封装好。

(2)将帧送入 NIC 的发送缓冲区：

1） 设置地址寄存器 RSAR：RSAR0=0，发送数据帧的缓存起始地址（低 8 位）；RSAR1=40，发送数据块的缓存起始地址（高 8 位）。

2） 设置远程字节计数寄存器 RBCR ：设置为发送数据帧的长度。

3） 设置命令寄存器 CR：CR=12，远程 DMA 写。

4） 往数据端口写入发送数据。

(3)初始化发送的字节数寄存器 TBCR:设置为发送数据帧的长度。

(4)初始化发送页面地址寄存器 TPSR：TPSR=0x40。

(5)启动 NIC 将该帧发送到网络传输线上：CR=0x26，启动发送。

6.5 TCP/IP 协议程序的实现

根据嵌入式系统的特点，课题首先基于以太网传输介质对 TCP/IP 进行了选择，主要包括 ARP 协议、IP 协议、ICMP 内部的端口不可达协议和 ping 应答协议、UDP

协议和 TCP 协议。协议程序主要是完成各种协议类型报文的封装和解封装，以及对各种报文按照协议规定进行处理。

在数据报文封装和解封装的过程中，程序使用了开发环境中的 C 语言库函数。这样可以使开发的程序尽量简单和优化。

封装数据前首先为数据包分配一定的空间，程序中使用了 malloc( )函数。malloc 函数为长度为 size 的对象分配存储空间，并返回指向该空间的指针。如果 malloc()函数无法实现空间分配，则函数的返回值为 0。malloc 函数不修改被分配单元中的内容。

用例：

```
... ... ... ...
struct arp *ap;
uchar *outbuf;
outbuf=(uchar*)malloc(21);/*分配 21 字（42 字节）的空间*/
ap=(struct arp)(outbuf+7);
/*将以太网首部后的字节转换为 ARP 分组格式*/
ap->hwtype=ETHERNET;
/*填充 ARP 内容*/
... ... ... ...
```

在向分配的空间填充数组型数据时还使用了 memcpy( ) 函数和 memset()函数。

memcpy()函数将指针 s2 所指的 n 个字符复制到指针 s1 所指的单元中。如在组装以太网首部时，发送端硬件地址为本地地址，采用拷贝方式进行填充。

用例：

```
... ... ... ...
memcpy(eth->shwaddr, cg.myhwaddr, 3);
/*将本地硬件地址拷贝到以太网发送端硬件地址*/
... ... ... ...
```

memset()函数将字符 ch 复制到指针 mem 所指长度为 length 字符串中。如在填

充数据包字段时有时需要将该字段设置为全 0，此时采用 memset()函数简单方便。

```
… … … …
if(opcode==ARP_REQUEST)
memset(ap->thwaddr,0,3);
… … … …
```

在数据发送完毕之后程序使用 free()函数释放空间。free()函数释放由 malloc 函数等指令预先分配的存储空间（由 packet 制定起始地址）。这使存储空间可以反复使用。如果不释放 malloc 函数分配的空间将导致内存丢失，直至没有内存可用系统瘫痪。

在接收判断地址是否符合要求时使用了开发环境中的是内存比较库函数：memcmp( )函数。memcmp()函数比较指针 ct 所指单元和指针 cs 所指单元之间 n 个字符的内容，返回一下 3 个值中的一个：

<0 如果*cs 小于*ct;

0 如果*cs 等于*ct;

<0 如果*cs 大于*ct。

用例：

```
… … … …
if(memcmp(eth->thwaddr,cg.myhwaddr,3)==0
/*硬件地址是否为本地地址*/
 ||memcmp(eth->thwaddr,hbdcst,3)==0)
/*硬件地址是否为广播地址*/
{
处理数据
}
… … … …
```

在处理 IP 首部的选项字段时使用了 memmove()函数。memmove()函数将指针 s2 所指的 n 个字符转移到指针 s1 所指的 n 个单元中。memmove()函数可以实现两个重叠目标之间的正确复制。

用例：

```
... ... ... ...
memmove(inbuf + 17, inbuf + 7 + header_len, payload_len);
... ... ... ...
```

6.5.1 以太网数据包

在以太网作为链路层的 TCP/IP 协议中，数据包是按照以太网帧格式进行传输的。在 TCP/IP 中，以太网数据报的封装是在 RFC894 中规定的。以太网数据包的封装格式如表 6.1 所示。图中每个方块下面的数字是它们的字节长度。

表 6.1 以太网数据包的封装格式

Tab.6.1 Ethernet data frame format

目的地址(6字节)	源地址(6字节)	类型(2字节)	数据（46—1500 字节）	CRC(4 字节)

48 bit（6 字节）的目的地址和源地址就是我们所称的硬件地址或物理地址。目的地址为全 1 的特殊地址是广播地址。电缆上的所有以太网接口都要接收广播的数据帧。ARP 协议就是对 32bits 的 IP 地址和 48bits 的硬件地址进行映射。后面的类型字段指明后续数据的协议类型，对于 ARP 来说，该字段的值为 0x0806，对于 IP 协议则是 0x0800。CRC 字段用于帧内后续字节差错的循环冗余码检验（检验和）（它也被称为 FCS 或帧检验序列）。以太网帧标准要求数据长度最少要有 46 字节，也就是说加上地址、类型和 CRC 字段数据包的总长度不小于 64 字节。因此，为了保证这一点，程序实现中发送数据报时在不足的空间插入填充（pad）字节。

课题设计的以太网接入系统（以后简称系统）在接收到一个以太网数据包后，首先检查数据包的总长度，如果小于 64 字节就丢弃；接着检查目的地址是否为本地地址或广播地址，如果是本地地址或广播地址则将数据根据类型字段的内容交给 ARP 协议或 IP 协议处理程序，否则就丢弃数据包，直接返回。

6.5.2 ARP 协议

当一台主机把以太网数据帧发送到位于同一局域网上的另一台主机时，是根据 48 bit 的以太网地址来确定目的接口的。设备驱动程序从不检查 IP 数据报中的目的 IP 地址。ARP 的功能就是在 32 bit 的 IP 地址和硬件地址之间提供动态映射。

为了存放 IP 地址到硬件地址之间的映射记录，协议程序在数据存储区分配了一定的空间作为高速缓存区。根据协议规定，为了提高 ARP 运行的效率，高速缓存中每一项的生存时间一般为 20 分钟，起始时间从被创建时开始算起。程序实现 ARP 协议时首先要按照协议完成数据帧的封装。ARP 协议的数据帧的分组格式，如表 6.2 所示：

表 6.2 ARP 分组格式

Tab. 6.2 ARP grouping format

以太网目的地址	以太网源地址	帧类型	硬件类型	协议类型	硬件地址长度	协议地址长度	操作码	发送端以太网地址	发送端 IP 地址	目的以太网地址	目的 IP 地址

硬件类型字段表示硬件地址的类型。在这里是以太网，它的值设置为 1。协议类型字段表示要映射的协议地址类型。在课题设计中要完成以太网地址到 IP 地址的映射，它的值为 0x0800 即表示 IP 地址。接下来的两个 1 字节的字段，硬件地址长度和协议地址长度分别指出硬件地址和协议地址的长度，以字节为单位。以太网地址是 48bit—6 个字节，而协议地址也就是 IP 地址是 32bit——4 个字节，因此这两个字段的值分别位 6 和 4。操作码字段（OP）指出四种操作类型，它们是 ARP 请求（值为 1）、ARP 应答（值为 2）、RARP 请求（值为 3）和 RARP 应答（值为 4）。这个字段必需的，因为 ARP 请求和 ARP 应答的帧类型字段值是相同的。在课题设计中因为不考虑实现 RARP 协议，因而在程序实现 ARP 协议时也不考虑 RARP 请求和应答操作类型。最后四个字段是发送端的硬件地址（在本课题中是以太网地址）、发送端的协议地址（IP 地址）、目的端的硬件地址和目的端的协议地址。在进行 ARP 数据帧封装时，ARP 请求数据包中的目的以太网未知，因而设置位全 0。

当高层协议程序需要发送数据包时，首先查找高速缓存中是否存在该目的 IP 地址的硬件地址映射，如果没有则按照 ARP 协议封装 ARP 请求数据包并发送出去，查找该 IP 地址对应的硬件地址。

系统在发送一份 ARP 请求后即在高速缓存中建立一个地址映射表项（不完整），设置超时值为 3 分钟。当收到 ARP 应答后再更新表项并将超时值设置为 20 分钟。

在发送 ARP 请求时可能存在目的 IP 地址的主机不存在或已关机，这时不能收到 ARP 响应。同时考虑到不能收到响应也又可能是数据包在传输过程中丢失引起的，所以程序采用递增延时算法多次发送 ARP 请求，如果长时间不能收到响应然后再放弃。程序中是在第 1 次请求发生 6 秒后进行第 2 次请求，在 24 秒之后又进行第 3 次请求，在 75 秒后放弃。

为了能够检测是否有另一个主机设置了与本地系统相同 IP 地址，系统支持一种称为免费 ARP 的功能，即在系统启动引导进行接口配置的时候发送 ARP 请求查找自己的 IP 地址。对这个请求，系统不希望收到回答，如果这时收到一个回答，就说明有另外一个主机具有相同的 IP 地址。在启动时发送查找本地 IP 地址的 ARP 请求还能够通知其它主机本地的硬件地址，如果其它主机缓存中存在本系统 IP 地址的 ARP 映射，则可以根据这个 ARP 请求进行更新。为了防止因为数据包丢失而不能检测出 IP 地址冲突，课题设计中在系统启动时将查找本地 IP 地址的 ARP 请求发送 3 遍。

当系统收到一份 ARP 协议数据包后，首先对数据包中的各个字段进行检查，判断数据包是否符合本课题设计的要求。

如果发送端 IP 地址与本地 IP 地址相同则通过串行通信接口发送告警信息。

如果系统收到某个 IP 地址的 ARP 请求，而且它已经在系统的高速缓存中，那么就要用 ARP 请求中的发送端硬件地址（以太网地址）对高速缓存中相应的内容进行更新。如果该请求数据包的目的端 IP 地址是本地 IP 地址，它就把硬件地址填进去，然后用两个目的端地址分别替换两个发送端地址，并把操作字段置为 2（ARP 应答），最后把它发送回去。

如果系统收到的是 ARP 应答数据包且目的 IP 地址是本地 IP 地址则更新完善地址映射表项。如果这时有 IP 数据包等待发送，则发送 IP 数据包。

在 ARP 协议程序完成后，对程序进行了优化调整。ARP 接收数据处理程序流程如图 6.3 所示。

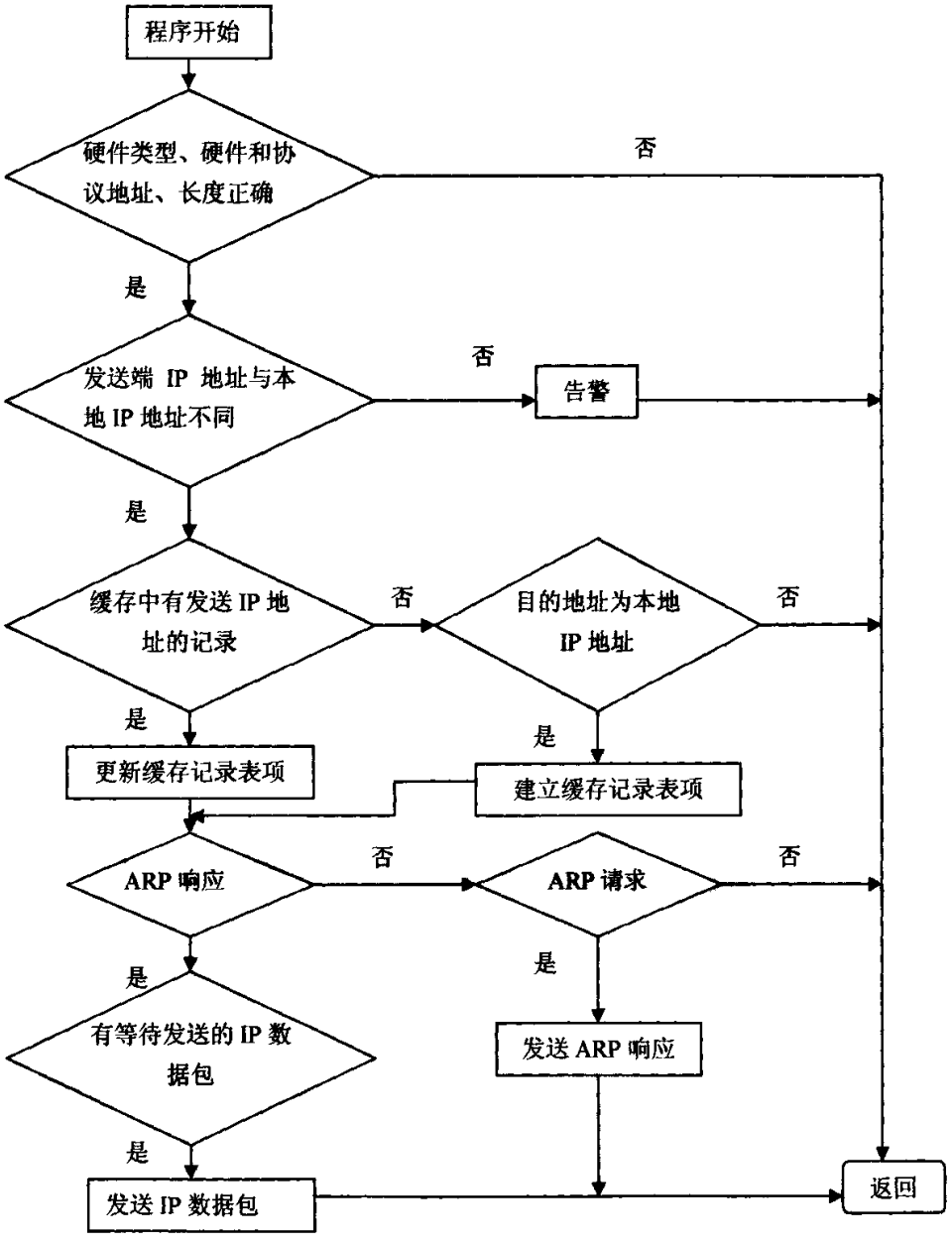


图 6.3 ARP 接收数据处理流程图

Fig. 6.3 ARP Flow Chart in Receiving Data

6.5.3 IP 协议

IP 协议是 TCP/IP 协议族中核心的协议，数据包跨越不同的网络必须通过 IP 协议的处理。也就是说所有高层协议（如 UDP、TCP、ICMP、IGMP）的数据都要按照 IP 分组的格式进行封装。IP 协议数据包有一定的分组格式，如表 6.3。

表 6.3 IP 数据包格式

Tab. 6.3 IP data frame format

4 位版本	4 位首部长度	8 位服务类型	16 位总长度（字节数）
16 位标识			16 位总长度（字节数）
8 位生存时间		8 位协议	16 位首部校验和
32 位源 IP 地址			
32 位目的 IP 地址			
选项			
数据			

普通的 IP 首部不包含选项字段时，长度为 20 个字节。在课题设计中用的 IP 协议版本是 4，即 IPv4，所以第一个字段的值设置为 0100（二进制）。

首部长度指的是首部占 32 bit 字的数目，包括选项。由于它是一个 4 比特字段，因此首部最长为 60 个字节。考虑到协议的系统实现受条件制约，在本课题设计中发送报文时只用普通 IP 数据报（没有任何选择项），因此该字段的值设置为 5。而在接收报文时如果该值不是 5，则说明报文有选项字段，程序将其删除，不进行分析处理。

服务类型（TOS）字段包括一个 3 bit 的优先级子字段，4 bit 的 TOS 子字段和 1 bit 未用位但必须置 0。4 bit 的 TOS 分别代表：最小时延、最大吞吐量、最高可靠性和最小费用。4 bit 中只能置其中 1 bit。课题设计时，在运输层使用 UDP 协议时采用最小时延，将 4 bit 设置为 1000（二进制）；在使用 TCP 协议时将 4bit 均设置为 0，采用一般服务。这个字段的内容主要是在路由器进行转发时使用。

总长度字段是 IP 首部中必要的内容，因为一些数据链路（如以太网）需要填充一些数据以达到最小长度。尽管以太网的最小帧长为 46 字节，但是 IP 数据可能会更短。如果没有总长度字段，那么 IP 层就不知道 46 字节中有多少是 IP 数据报的内容。总长度

字段是指整个 IP 数据报的长度，以字节为单位。利用首部长度字段和总长度字段，就可以知道 IP 数据报中数据内容的起始位置和长度。但是由于总长度字段与首部长度字段不同，首部长度是指 IP 数据包首部占 32bit 的数目，所以数据长度不是总长度与首部长度数值直接相减。由于总长度字段长 16 比特，所以 IP 数据报最长可达 65535 字节。尽管可以传送一个长达 65535 字节的 IP 数据报，但是大多数的链路层都会对它进行分片。在实际应用当中，一般限制用户数据报的长度小于 576 字节。所以在本课题设计中，用户数据长度限制在 512 字节。

标识字段唯一地标识主机发送的每一份数据报。通常每发送一份报文它的值就会加 1。IP 提供无连接的数据报传送。无连接（ connectionless）的意思是 IP 并不维护任何关于后续数据报的状态信息。每个数据报的处理是相互独立的。这也说明，IP 数据报可以不按发送顺序接收。如果某信源向相同的信宿发送两个连续的数据报（先是 A，然后是 B），每个数据报都是独立地进行路由选择，可能选择不同的路线，因此 B 可能在 A 到达之前先到达。接收方根据 IP 数据报的标识字段识别每一帧数据包，并将数据包按发送顺序组合起来。在课题设计中根据这个字段调整数据包的顺序进行组合，并删除重复的数据包。因为系统的缓存较小，所以在程序实现时只缓存一个数据包。如果收到的数据包标识差很大时，重新记录标识号，并报告有报文丢失。

TTL（time-to-live）生存时间字段设置了数据报可以经过的最多路由器数。它指定了数据报的生存时间。经过处理该数据报的一个路由器，它的值就减去 1。当该字段的值为 0 时，数据报就被丢弃，并发送 ICMP 报文通知源主机。在课题中将该字段值设置为 32，本系统发送的数据包最多可以经过 31 个路由器中转处理。

协议字段，表明是哪个协议向 IP 传送数据。在处理收到的 IP 数据包时，也根据这个字段判断 IP 数据应该交给哪个协议处理。在本系统中有 ICMP、TCP、UDP 向 IP 传送数据，因此这个字段的值可以是 1、6 或 17。

首部检验和字段是根据 IP 首部计算的检验和码。它不对首部后面的数据进行计算。为了计算一份数据报的 IP 检验和，首先把检验和字段置为 0。然后，对首部中每个 16 bit 进行二进制反码求和（整个首部看成是由一串 16 bit 的字组成），结果存在检验和字段中。当收到一份 IP 数据报后，同样对首部中每个 16 bit 进行二进制反码的求和。由于接

收方在计算过程中包含了发送方存在首部中的检验和，因此，如果首部在传输过程中没有发生任何差错，那么接收方计算的结果应该为全 1。

在 IP 数据包首部还有源 IP 地址和目的 IP 地址字段，用于标识 IP 数据包的发送主机和目的主机。它们都是 32 bit 的值。

最后一个字段是任选项，是数据报中的一个可变长的可选信息。选项字段一直都是以 32 bit 作为界限，在必要的时候插入值为 0 的填充字节。这样才能保证 IP 首部始终是 32 bit 的整数倍（这是首部长度字段所要求的）。

发送数据前，程序将高层协议传递过来的数据包封装成 IP 报文。在发送一份 IP 数据报之前，首先要在 ARP 缓存表中查找目的 IP 地址对应的硬件地址（以太网地址），然后按照硬件地址进行以太网数据包封装。如果不能查到目的 IP 地址对应的硬件地址，则发送查找该目的 IP 地址的 ARP 请求（如果该 IP 地址不是本地局域网内 IP 地址，则发送查找网关的 IP 地址的 ARP 请求），将待发送的 IP 数据包设置为等待发送状态，在收到对应的 ARP 响应并建立完整的地址映射之后再发送出去。

IP 提供不可靠、无连接的数据报传送。不可靠的就是它不保证 IP 数据报能成功地到达目的地。无连接就是 IP 并不维护任何关于后续数据报的状态信息，每个数据报的处理是相互独立的，也就是 IP 数据报可以不按发送顺序接收。

在接收到 IP 数据包后，程序根据首部标识字段判断 IP 数据包是否与前一个数据包连续或重复，如果数据包首部标识与前一个数据包标识相等（数据包重复）就将数据包丢弃。如果差值大于 1（数据包首部标识不连续），则重新记录数据包标识，并报告有数据包丢失。关于初始值的确定，在具有操作系统的主机根据系统引导时的时间来设置。在本系统中没有时间设置，所以在发送数据时，IP 数据包首部标识字段的值由系统启动时随机设定；而在接收数据包时，设置了对端 IP 数据包首部标识变量，并将该变量默认值设置为 0，在第一次收到数据包后则修改记录的标识字段值，以后根据该值判断 IP 数据包是否重复或连续。为了保证数据传输的可靠性，高层协议提供了超时重发的功能，因而这里也设定如果在超时重发的时间内没有收到连续的数据包，则对端 IP 数据包首部标识变量设置默认值，重新开始记录对端 IP 数据包首部标识，不报告有数据包丢失。

接收到的每一个数据包，系统程序都根据首部检验和、版本号和分片标志等字段的内容检测数据是否正确，如果数据内容正确再根据协议字段的内容交给相应的协议处理。IP 数据包处理程序流程如图 6.4 所示。

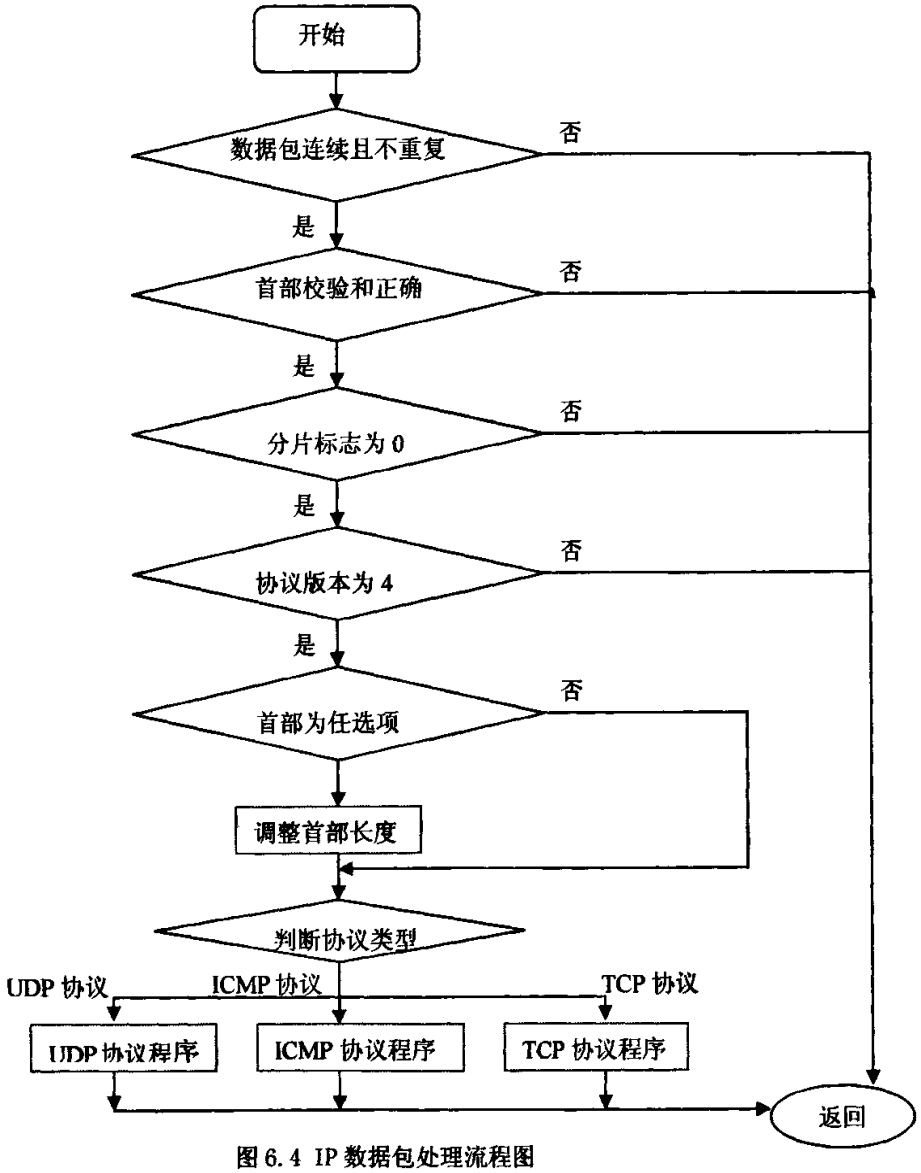


图 6.4 IP 数据包处理流程图

Fig. 6.4 IP Data Process Flow Chart

6.5.4 ICMP（Internet 控制报文协议）

ICMP 传递差错报文以及其他需要注意的信息，因此被看作是网络层的一个组成部分，但又因为它是在 IP 数据报内部中传输，所以有时又把它看作是高于 IP 层的协议。

ICMP 报文通常被 IP 层或更高层协议（TCP 或 UDP）使用。有些 ICMP 报文直接把差错报文返回给用户进程（应用层）。ICMP 也有自己的报文格式，如表 6.4 所示。

表 6.4 ICMP 报文格式

Tab.6.4 ICMP Message format

8 位类型	8 位代码	16 位校验和
（不同类型和代码有不同内容）		

类型字段描述特定类型的 ICMP 报文，可以有 15 个不同的值。某些 ICMP 报文还使用代码字段的值来进一步描述不同的条件。ICMP 的检验和字段与 IP 数据报不同，在 IP 数据报中检验和只是覆盖首部，ICMP 的检验和覆盖整个报文。

ICMP 报文不同类型由报文中的类型字段和代码字段来共同决定。在协议选择时根据实际需要，系统只支持 ICMP 中 Ping 应答协议和端口不可达协议。

Ping 程序是对两个 TCP/IP 系统连通性进行测试的基本工具，它只利用 ICMP 回显请求和回显应答报文，而不用经过传输层（TCP/UDP）。端口不可达协议则进一步检测高层协议（UDP）是否可以完成通信。

Ping 程序发送一份 ICMP 回显请求报文给主机，并等待返回 ICMP 回显应答。Ping 回显应答的数据内容就是 ping 回显请求的数据内容。一般来说，如果不能 ping 到某台主机那么就不能连接到那台主机。随着 Internet 安全意识的增强，出现了提供访问控制清单的路由器和防火墙，那么像这样没有限定的断言就不再成立了。一台主机的可达性可能不只取决于 IP 层是否可达，还取决于使用何种协议以及端口号。Ping 程序的运行结果可能显示某台主机不可达，但同样可以通过某种协议进行连接。但在本课题设计中只要系统能响应其它主机发出的 ping 请求，返回 ping 应答就可以了。也就是说系统在接收到类型为 8、代码为 0 的 ping 请求 ICMP 报文后，能够产生类型为 0、代码为 0 的 ping 应答，ping 应答的数据内容为 ping 请求的内容。

端口不可达报文是 ICMP 目的不可到达报文中的一种。在 UDP 的规则中，如果收到一份 UDP 数据报而目的端口与某个正在使用的进程不相符，那么 UDP 返回一个 ICMP 不可达报文。在课题设计中有两种情况：一是当系统向其它主机发送数据时，如果主机回端口不可达数据报，系统在接收到该报文后通过串行通信端口发出连接被拒绝的告警信息；二是当其它主机向系统发送的数据报的目的端口不存在时返回端口不可达的 ICMP 报文。

产生一份 ICMP 目的端口不可达差错报文时，报文包含 IP 的首部（包括选项）和产生 ICMP 差错报文的 IP 数据报的前 8 个字节。这样，接收 ICMP 差错报文的模块就可以把它与某个特定的协议（根据 IP 数据报首部中的协议字段来判断）和用户进程（根据包含在 IP 数据报首部后面的前 8 个字节中的 TCP 或 UDP 报文首部中的 TCP 或 UDP 端口号来判断）联系起来。系统接收到 ICMP 报文后也是根据报文的这些内容进行处理。

6.5.5 UDP(用户数据报协议)

UDP 是一个简单的面向数据报的运输层协议。应用程序将要发送的数据封装产生一个 UDP 数据报，并组装成一份待发送的 IP 数据报。UDP 不提供可靠性：它把要传给 IP 层的数据发送出去，但是并不保证它们能到达目的地。UDP 数据分组格式如表 6.5 所示。

表 6.5 UDP 数据分组格式

Tab. 6.5 UDP Data Grouping format

16 位端口号	16 位目的端口号
16 位 UDP 长度	16 位 UDP 校验和
数据（如果有）	

源端口号和目的端口号表示发送进程和接收进程。不同的端口号可以同时工作用于完成不同的任务。

UDP 长度字段指的是 UDP 首部和 UDP 数据的字节总长度。该字段的最小值为 8 字节（没有纯数据的 UDP 数据报）。IP 数据报长度指的是数据报全长，因此程序接收到装载 UDP 数据的 IP 数据报后计算 UDP 数据报长度时只是用 IP 数据报的全长减去 IP

首部的长度（该值在 IP 首部的首部长度字段中指定）。如果计算的 UDP 数据包长度与 UDP 首部长度字段的值不同，说明数据包存在错误，进行丢弃。

UDP 检验和是一个端到端的检验和。它由发送端计算，然后由接收端验证。其目的是为了发现 UDP 首部和数据在发送端到接收端之间发生的改动。UDP 检验和覆盖 UDP 首部和 UDP 数据。IP 首部的检验和只覆盖 IP 的首部——并不覆盖 IP 数据报中的任何数据。UDP 检验和的基本计算方法与 IP 首部检验和计算方法相类似（16 bit 字的二进制反码和），但是 UDP 数据报的长度可以为奇数字节，而检验和算法是把若干个 16 bit 字相加，所以在课题的程序实现中，如果 UDP 数据报的长度可以为奇数字节，计算检验和时则在最后增加填充字节 0，填充的字节只是为了检验和的计算，而不被传送。根据协议规定，为了让 UDP 两次检查数据是否已经正确到达目的地，程序中 UDP 数据报在计算检验和时还包含一个 12 字节长的伪首部。伪首部包含 IP 首部一些字段。

这里，UDP 数据报的长度在检验和计算过程中出现两次。如果检验和的计算结果为 0，则存入的值为全 1（65535）。如果传送的检验和为 0，则说明发送端没有计算检验和。

为了能够检验数据传输是否正确，在本课题实现中要求必须进行检验和计算。如果发送端没有计算检验和或者接收端检测到检验和有差错，那么 UDP 数据报就要被悄悄地丢弃，不产生任何差错报文（当 IP 层检测到 IP 首部检验和有差错时也是这样做的）。

当根据应用层协议满足发送条件时，UDP 协议将数据进行封装，并进一步组装成 IP 数据报，然后通过以太网封装并发送出去。当系统接收到一个 UDP 数据包后，首先根据首部内容进行检测后根据不同情况进行处理。UDP 数据包处理流程如图 6.5 所示。

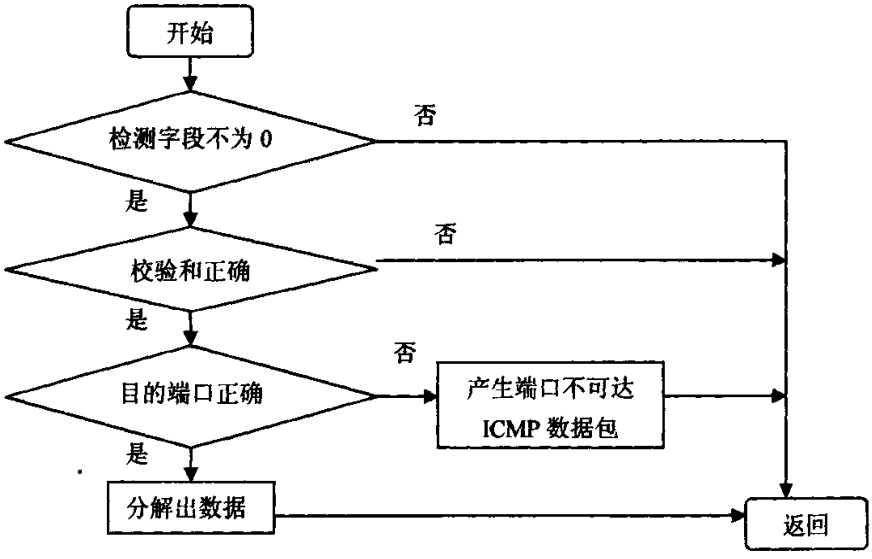


图 6.5 UDP 数据包处理流程

Fig. 6.5 UDP Datagram Process Flow Chart

6.5.6 TCP(传输控制协议)

TCP 是 TCP/IP 协议族中最复杂的协议。TCP 和 UDP 都使用相同的网络层（IP），TCP 却向应用层提供与 UDP 完全不同的服务。TCP 提供一种面向连接的、可靠的字节流服务。

面向连接意味着两个使用 TCP 的应用程序（通常是一个客户和一个服务器）在彼此交换数据之前必须先建立一个 TCP 连接。在一个 TCP 连接中，仅有两方进行彼此通信。

TCP 的可靠性主要依靠以下几个方式来保证：

- 当 TCP 发出一个段后，它启动一个定时器，等待目的端确认收到这个报文段。如果不能及时收到一个确认，将重发这个报文段。
- 当 TCP 收到发自 TCP 连接另一端的数据，它将发送一个确认。这个确认不是立即发送，通常将推迟几分之一秒。
- TCP 将保持它首部和数据的检验和。这也是一个端到端的检验和，目的是检测数据在传输过程中的任何变化。如果接收端的检验和有差错，TCP 将丢弃这个报文

段和不确认收到此报文段（希望发端超时并重发）。

- TCP 还能提供流量控制。TCP 连接的每一方都有固定大小的缓冲空间。TCP 的接收端只允许另一端发送接收端缓冲区所能接纳的数据。这将防止较快主机致使较慢主机的缓冲区溢出。

字节流服务是指两个应用程序通过 TCP 连接交换 8 bit 字节构成的字节流，TCP 不在字节流中插入记录标识符。TCP 对字节流的内容不作任何解释。TCP 不知道传输的数据字节流是二进制数据，还是 ASCII 字符或者其他类型数据。对字节流的解释由 TCP 连接双方的应用层解释。

（1） TCP 数据包的封装

TCP 数据也是被封装在一个 IP 数据报中，TCP 首部没有选项字段时，长度是 20 个字节，TCP 数据包的格式如表 6.6。

表 6. 6 TCP 数据包的格式

Tab. 6.6 TCP Datagram Format

16 位端口号								16 位目的端口号
32 位序号								
32 位确认序号								
4 位首部长度	保留 6 位	U R G	A C K	P S H	R S T	S Y N	F I N	16 位窗口大小
16 位检验和								16 位紧急指针
选项								
数据								

和 UDP 首部一样，每个 TCP 段都包含源端和目的端的端口号，用于寻找发端和收端应用进程。这两个值加上 IP 首部中的源端 IP 地址和目的端 IP 地址唯一确定一个 TCP 连接。有时，一个 IP 地址和一个端口号也称为一个插口（socket）。插口对（socketpair）(包含发送端 IP 地址、发送端端口号、接收端 IP 地址和接收端端口号的四元组)可唯一确定互联网络中每个 TCP 连接的双方。

序号用来标识从 TCP 发端向 TCP 收端发送的数据字节流，它表示在这个报文段中的第一个数据字节。如果将字节流看作在两个应用程序间的单向流动，则 TCP 用序号对每个字节进行计数。序号是 32 bit 的无符号数，序号到达 232－1 后又从 0 开始。序号字段包含由这个主机选择的该连接的初始序号 ISN（Initial Sequence Number）。在课题程序设计中，在系统初始化时初始的发送序号被初始化为 1。ISN 随时间而变化，每 0.5 秒增加 64000，约每隔 9.5 小时又回到 0，另外每次建立一个连接后，这个变量也增加 64000，这样每次连接都具有不同的 ISN。TCP 为应用层提供全双工服务，数据能在两个方向上独立地进行传输。因此，连接的每一端保持每个方向上的传输数据序号。因为在建立连接发送 SYN 标志（发送没有纯数据只有同步标志的数据包）时要占用一个序号，所以系统要发送数据的第一个字节序号为 ISN 加 1。

确认序号是上次已成功收到数据字节序号加 1。只有 ACK 标志为 1 时确认序号字段才有效。

首部长度给出首部中 32 bit 字的数目。它和 IP 首部相似，因为任选字段的长度是可变的，所以必须设置这个字段值，而 UDP 首部是固定长度，首部中也就没有这个字段。这个字段占 4 bit，因此 TCP 最多有 60 字节的首部。在本课题设计中没有任选字段，首部长度为 20 字节，这个字段的值也就是 5。

在 TCP 首部中有 6 个标志比特。它们中的多个可同时被设置为 1。每个标志比特的意义为：

URG 紧急指针（ urgent pointer）有效。

ACK 确认序号有效。

PSH 接收方应该尽快将这个报文段交给应用层。

RST 重建连接。

SYN 同步序号用来发起一个连接。

FIN 发端完成发送任务。

在系统程序中，根据连接状态在封装数据时打上不同的标记。

TCP 的流量控制由连接的每一端通过声明的窗口大小来提供。窗口大小为字节数，起始于确认序号字段指明的值，这个值是接收端期望接收的字节。窗口大小是一个 16 bit

字段，因而窗口大小最大为65535字节。在课题设计中考虑到系统数据处理速度一般低于普通的PC机，所以窗口的大小采用固定的256字节，使用小窗口。

检验和覆盖了整个的TCP报文段： TCP首部和TCP数据。TCP检验和的计算和UDP检验和的计算相似，使用一个伪首部。

紧急指针是一个正的偏移量，和序号字段中的值相加表示紧急数据最后一个字节的序号。TCP的紧急方式是发送端向另一端发送紧急数据的一种方式。只有当URG标志置1时紧急指针才有效。

本课题中用到的选项字段是最长报文大小，又称为MSS (Maximum SegmentSize)。每个连接方通常都在通信的第一个报文段（为建立连接而设置 SYN 标志的那个段）中指明这个选项。它指明本端所能接收的最大长度的报文段，用于流量控制，防止数据缓冲区溢出。

TCP报文段中的数据部分是可选的。在一个连接建立和一个连接终止时，双方交换的报文段仅有 TCP 首部。如果一方没有数据要发送，也使用没有任何数据的首部来确认收到的数据。在处理超时的许多情况中，也会发送不带任何数据的报文段。

（2）TCP连接的建立和关闭实现

在课题设计的协议程序中实现了建立 TCP 连接的三次握手过程和关闭连接的四次握手过程。

首先在使用TCP协议发送数据之前，请求端（通常称为客户）发送一个SYN段（没有数据，SYN标志变1的TCP数据包）指明客户打算连接的服务器的端口，以及本地初始序号；响应端（服务器）发回包含服务器的初始序号的 SYN 报文段作为应答，同时对客户端的数据包进行确认。确认序号设置为客户的ISN加1；客户再将确认序号设置为服务器的ISN加1对服务器的SYN报文段进行确认。此时，客户端已经收到确认建立了连接，所以在对服务器的SYN报文确认的同时可能传输数据。

建立一个连接是三次握手，终止一个连接是4次握手过程。一个TCP连接是全双工（即数据在两个方向上能同时传递），因此每个方向必须单独地进行关闭。当系统完成它的数据发送任务后就发送一个FIN报文来终止这个方向连接，当服务器端收到一个FIN报文，它通知应用层：另一端已经终止了那个方向的数据传送，然后发回一个ACK，

确认序号为收到的序号加 1。服务器端要求关闭连接时，也要发送一个 FIN 报文段，系统收到这个报文发回一个确认，并将确认序号设置为收到序号加 1，从而关闭另一方向连接。收到一个 FIN 只意味着在这一方向上没有数据流动，TCP 连接在收到一个 FIN 后仍能发送数据。

当对端主机没有处于正常状态时，就无法建立 TCP 连接。在这种情况下，就会出现建立连接超时。系统在发送第一个 SYN 报文之后 6 秒如果没有收到确认发送第二个 SYN 报文，如果仍然没有收到确认间隔 24 秒再发送第三个 SYN，在第三个分组发出后 75 秒没收到确认放弃连接。这里的时间间隔与 ARP 请求超时的时间间隔相同。

在连接建立和连接拆除的过程中，TCP 连接（完整，也可能不完整）可以处于多种状态，包括关闭连接状态、等待连接状态、同步发送状态、同步收到状态、连接建立状态、正在关闭状态、等待关闭状态、最后确认状态、发送 FIN 状态、FIN 发送接收状态、2MSL 等待状态。系统作为客户端在发送 TCP 连接请求数据包后建立到目的 IP 地址和目的端口的不完整连接，连接状态为同步发送状态（如果是服务器端主动要求建立连接则系统处于等待连接状态），然后根据收到的数据包不同进一步在不同的状态之间转换。

报文段最大生存时间 MSL（Maximum Segment Lifetime）是任何报文段被丢弃前在网络内的最长时间。当系统在收到 FIN 报文并发送回 ACK 之后，该连接在 TIME_WAIT 状态停留 2 倍的 MSL 的时间。这样可以防止最后的 ACK 丢失而对方重发 FIN 时再次发送确认。在本课题实现中采用的是常用值 30 秒。

（3） TCP 发送和接收数据

使用 TCP 协议发送数据，当数据满足发送条件时，如果存在需要的连接直接按照 TCP 协议封装并发送到 IP 协议层；如果不存在需要的连接则系统要求建立 TCP 连接，然后再按照 TCP 协议封装数据并发送到 IP 协议层。

TCP 提供可靠的运输层。它使用的方法之一就是确认从另一端收到的数据。但数据和确认都有可能会丢失。TCP 协议程序通过在发送时设置一个定时器来解决这种问题。如果当定时器溢出时还没有收到确认，它就重传该数据。

系统接收到 TCP 协议数据包，在进行行首部检查正确之后检查本地是否存在与发送数据包的 IP 地址和端口的连接，如果存在连接则继续分解数据包，得到纯数据；如

果没有连接则再根据端口是否正确或存在决定是发送端口不可达的 ICMP 数据包还是发送 TCP 确认建立连接表（不完整）。在进行完上述分析检测之后，系统程序接下来检测 TCP 首部的标志比特，根据标志位的不同和连接（可能不完整）的状态分别进行不同响应。TCP 协议数据处理流程如图 6.6 所示。

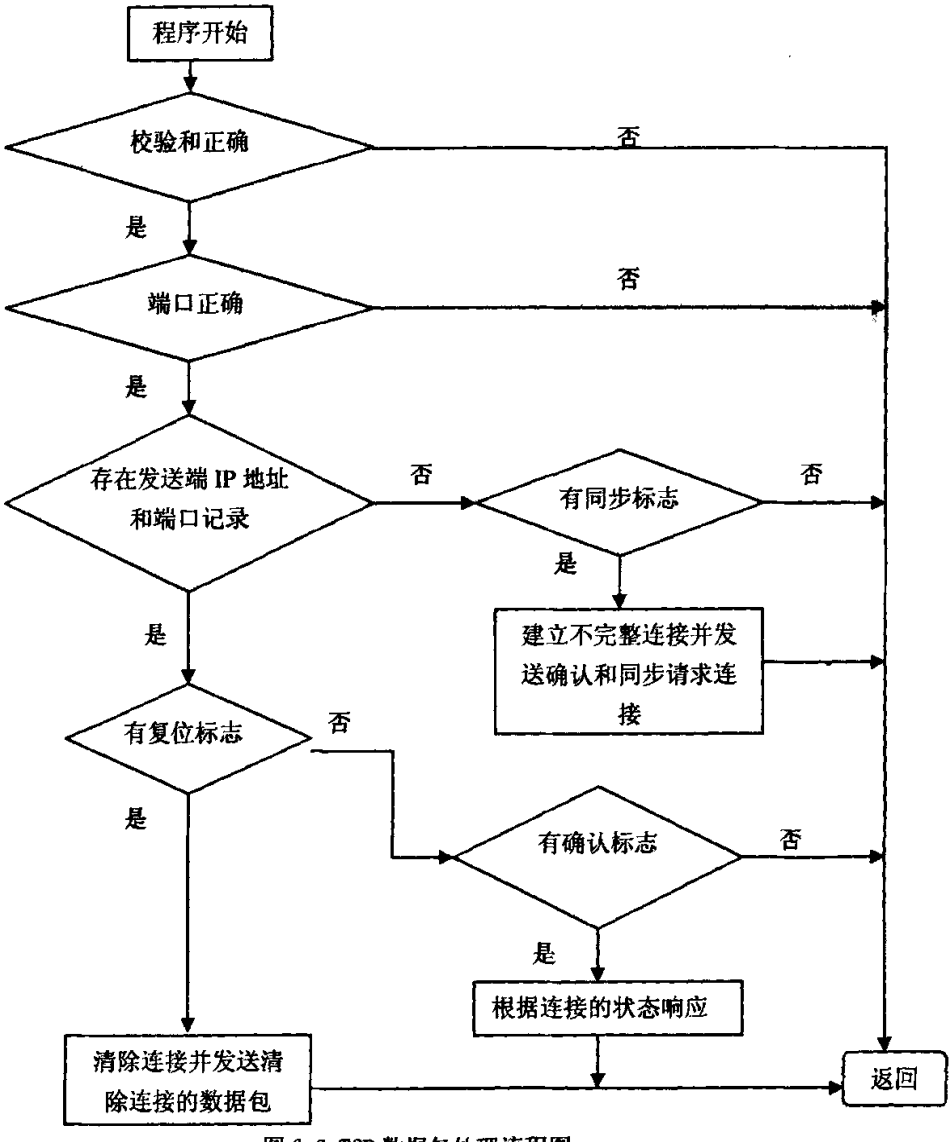


图 6.6 TCP 数据包处理流程图

Fig. 6.6 TCP Datagram Process Flow chart

如果没有其他控制，就可能出现没有任何数据流通过一个空闲的 TCP 连接的情况。也就是说，如果 TCP 连接的双方都没有向对方发送数据，则在两个 TCP 模块之间不交换任何信息。这意味着我们可以启动一个客户与服务器建立一个连接，然后离去数小时、数天、数个星期或者数月，而连接依然保持。中间路由器可以崩溃和重启，电话线可以被挂断再连通，但是只要两端的主机没有被重启，则连接依然保持建立；或者服务器端关机或重新启动后，TCP 连接已经丢失而本系统仍然存在该连接。在实际应用中这种情况是可能存在而又不合适的。在课题设计中，协议程序采用保活定时器来解决这个问题。也就是在 TCP 连接建立之后，启动一个定时器，当时间溢出时删除连接，如果业务需要再重新建立连接。课题中设计的保活定时时间为 20 分钟。

6.5.7 协议中的定时函数

在 TCP/IP 协议规定中有许多定时，如：ARP 协议中 ARP 地址映射记录有一定的存活时间，需要定时更新；在发送 ARP 请求而没有响应时也需要根据定时再次或多次发送请求。在 TCP 协议中也有超时重传定时和保活定时。所以在协议程序中还包括用于完成不同功能的定时程序。

ARP 定时函数有两个，一种是表项时间检测函数，每分钟执行一次，检测 ARP 记录表项的时间，完整表项 20 分钟（不完整表项 3 分钟）被删除；另一种是定时重发函数，每秒执行一次，在定时时间到达时没有收到响应则重发 ARP 请求。

TCP 协议定时函数有三种，一种是连接建立时请求重发定时，一种是连接保持时间定时，还有一种是发送数据包时没有收到确认进行重发的定时。这三种定时都是以秒为单位，在定时时间到达时重发连接请求或删除连接。

6.6 系统测试

在软硬件分模块调试完成以后，对整个系统网络接入功能进行了测试。测试网络连接方式如图 6.7 所示。

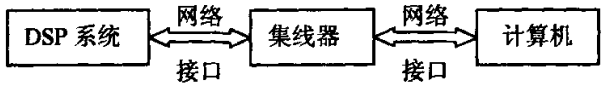


图6.7 测试网络连接方式

Fig. 6.7 Testing Network Connection

在嵌入式系统端，使用 DSP 系统仿真调试软件 CCS 观察嵌入式系统对网络数据的接收和发送情况，在另一端的计算机上，则通过上位机网络通信程序观察网络数据包的收发情况。系统功能测试主要分为如下四部分内容：

（1）计算机向系统发送网络数据包

计算机通过网络向第一次接入系统发送数据包时，因为计算机不知道接入系统的硬件地址，首先发送 ARP 请求，接入系统收到 ARP 请求后应该产生 ARP 响应并发送回计算机。计算机收到 ARP 响应并建立地址映射后开始向接入系统发送网络数据包，接入系统应该能接收到数据包并将数据分解出来。如果计算机发送的网络数据包端口不是本地端口，接入系统产生端口不可达数据包并发送回计算机。

（2）系统向计算机发送网络数据包

接入系统第一次向计算机发送网络数据包之前，首先应该发送目的地址为计算机IP地址的ARP请求，在收到计算机发回的 ARP 响应之后建立地址映射，然后再向计算机发送网络数据包。如果接入系统收到计算机发回的端口不可达数据包时，接入系统应该产生告警信息。

（3）计算机 ping接入系统的 IP 地址

当接入系统收到计算机主机发送的 ping请求后，应该返回ping应答数据包。

（4）IP 地址冲突

当计算机 IP 地址设置成与接入系统相同，然后再启动系统时，系统会收到计算机发送的发送端 IP 地址与接入系统地址相同的 ARP 响应，此时应该产生告警信息。如果先启动接入系统，后启动计算机，接入系统收到计算机发送的发送端 IP 地址与接入系统地址相同的 ARP 请求，也应该产生告警信息。

7 结论

本课题通过采用 DSP 系统与嵌入式网络接入模块结合的方案实现了 DSP 系统与以太网的互连。并通过测试工作得出以下结论：

（1）本课题在研究 TCP/IP 协议的基础上，结合嵌入式系统的特点，提出了一种对 TCP/IP 协议的选择方案。通过这种选择，在满足实际要求的前提下降低了 TCP/IP 协议实现的复杂程度。

（2）使用单片机与RTL8019网络控制器芯片接口实现了网络接口，从而DSP系统可以通过与网络接口系统模块的串口通信实现网络的接入。

（3）通过测试得出系统最大网络通信速度为：15.5K byte/s。

但是由于研究时间有限及课题涉及知识面较广等原因，系统仍存在以下问题：

（1）软件编程中的内存丢失问题。

系统软件在收发数据时根据网络数据包长度信息以动态的方式在内存中开辟数据缓冲空间，所以当该数据包处理完后需要及时释放该数据空间。否则，系统内存将会出现泄漏丢失，进而导致系统内存耗尽而停止工作。本课题中使用malloc和free函数完成上述操作，但效果不佳，所以系统在执行大数据量传输任务过程中容易发生死机现象。

改进方法：使用动态链表的编程方法编写内存管理程序函数取代malloc和free函数执行内存管理任务，提高内存利用率。

（2） DSP与单片机串口通信的速度瓶颈问题。

本课题中DSP通过串口与单片机通信进而实现网络接入功能。所以整个系统的网络通信速度受到串口通信速度串口通信速度的限制，且串口可实现的最大通信速度仅为115200Kbit/s，远低于以太网的通信速度能力。

改进方法：

(1)DSP使用HPI主机接口与单片机通过并行方式连接。但使用这种方法时需要注意的是，应选采用3.3V供电方式的单片机与网络接口芯片组成以太网接口模块，以省去了电平转换的设计环节。

(2) DSP与网络接口芯片直接接口实现嵌入式网络接入系统。这种方法硬件设计上较前一种方法可以简化很多，但TCP/IP协议栈也需要在DSP上直接实现，从而增加了编程开发的难度。

参考文献

[1] 戴明帧 周建江. TMS320C54X DSP 结构、原理及应用. 北京航空航天大学出版社. 2001.

[2] 张雄伟 曹铁勇. DSP 芯片的原理与开发应用（第 2 版）. 电子工业出版社. 2001.

[3] VC++ 6.0 网络编程实作教程 . 北京希望电子出版社. 2001

[4] 张勇 陈天霖. C /C ++语言硬件程序设计一基于 TMS320C5000 系列 DSP. 西安电子科技大学出版社

[5] W. Richard Stevens. TCP / IP 详解(卷 1：协议). 范建华等译. 北京：机械工业出版社，2000 4

[6] W. Richard Stevens. TCP / IP 详解(卷 2：实现 TCP / IP）. 陆雪莹等译。北京：机械工业出版社，2000. 7

[7] Jean J. Labrosse. uC / OS-11-源码公开的实时嵌入式操作系统. 邵贝贝译. 北京：中国电力出版社，2001

[8] Androws Tanenbaum，AlbertS Woodbull. 操作系统设计与实现[M]. 第二版。北京：清华大学出版社，2001

[9] 胡道元. 计算机局域网[M]. 北京：清华大学出版社，1996

[10] 马忠梅等. 单片机的 C 语言应用程序设计[M]. 北京：北京航空航天大学出版社，2001. 2

[11] 李农. 因特网技术在嵌入式系统中的应用. 测控技术，19（4），2002. 15—16

[12] 王勇，姚亦峰，蒋兴浩等. 嵌入式系统接入 Internet 的技术研究. 计算机工程与应用，37（4），2001：29—30

[13] 仇玉章等. 32 位微型计算机原理与接口技术[M]. 清华大学出版社 2000

[14] 谭浩强. C 程序设计. 北京，清华大学出版社. 1995

[15] 师明珠。嵌入式应用系统软件设计技术研究。计算机工程与应用，38(7), 2002, 127—129

[16] 老古开发网单片机与 RTL8019 的以太网解决方案. http://www.laogu.com/

[17] Tim Parker. Teach Yourself TCP/IP in 14 Days(Second Edition). Dean Miller Comments Department Sams Publishing 201 W. 103rd Street Indianapolis, IN 46290

[18] EasyWeb:Tiny TCP/IP Stack and Web Server. http://www.keil.com/

[19] Rodger Richey . Using Embedded Web Servers within Applications. Applications Manager, Microchip Technology Inc, USA

[20] uIP-A Free Small TCP/IP Implementation for 8 and 16 bit Microcontrollers. http://www.dunkels.com/

[21] Shear, David, Putting an Embeded system Internet, EDN. 1997(9):45-46

[22] Fieldbus technology, http://www.fieldbus.org, 1996.03.

[23] Dave Harrold, Ethernet Evewhere, http://www.isa.org/BookStore/1998.04

[24] Jonas Berge, Ethernet In Process Control, Industry Ethernet Book, Issue 3, http//:www.indusrial-neworking.com

[25] Doulas E.Comer. Internet Working with TCP/IP Volume1:Principles

[26] David Hudson. Software Vital in Net-Linked MCU Apps. http://www.eetimes.com,2001,8

[27] 李朝青编著. PC机及单片机数据通信技术. 北京：北京航空航天大学出版社，2000.45－132

[28] Richard Heathfield. 标准C语言实用全书. 北京：电子工业出版社，2001.

[29] 林锐等. 高质量程序设计指南－C＋＋/C语言. 北京电子工业出版社，2002

[30] 李刚. 数字信号处理器的原理及其开发应用[M]. 天津：天津大学出版社，2000.24-143

[31] 刘益成. TMS320C54X DSP应用程序设计与开发. 2002.120－189

[32] 郑红，吴冠. TMS320C54X DSP应用系统设计[M]. 北京：北京航空航天大学出版社，2002

[33] 戴逸民，梁晓霞. 基于DSP的现代电子系统设计[M]. 北京：电子工业出版社，2002.

[34] 朱铭皓，赵勇，甘泉. DSP应用系统设计[M]. 北京：电子工业出版社，2002.

[35] 杨亚军，胡仁杰. 实时嵌入式内核在DSP上的移植实现. 工业控制计算机，2002.06.10-32

[36] 孙先奎，秦岚. 远程测控技术的发展现状和趋势. 仪器仪表学报. 2004.8

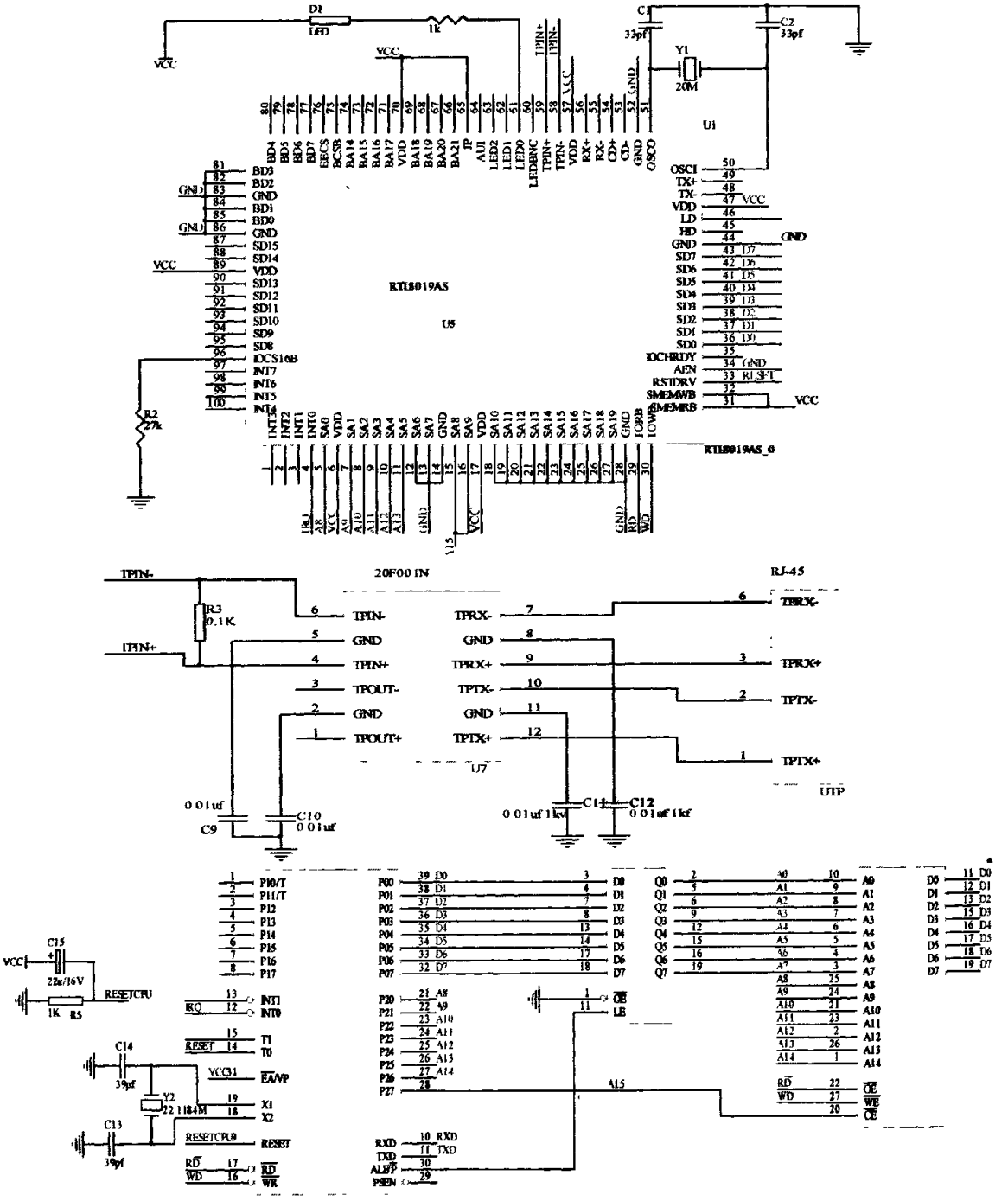
[37] 郑欣，郑宇恒. 远程计算机测控网络有关问题的研究. 机械与电子，2005（1）

[38] 郭瑞军，候成刚. DSP与MAX3111E UART数据通信的设计与实现. 仪器仪表用户，2005.2

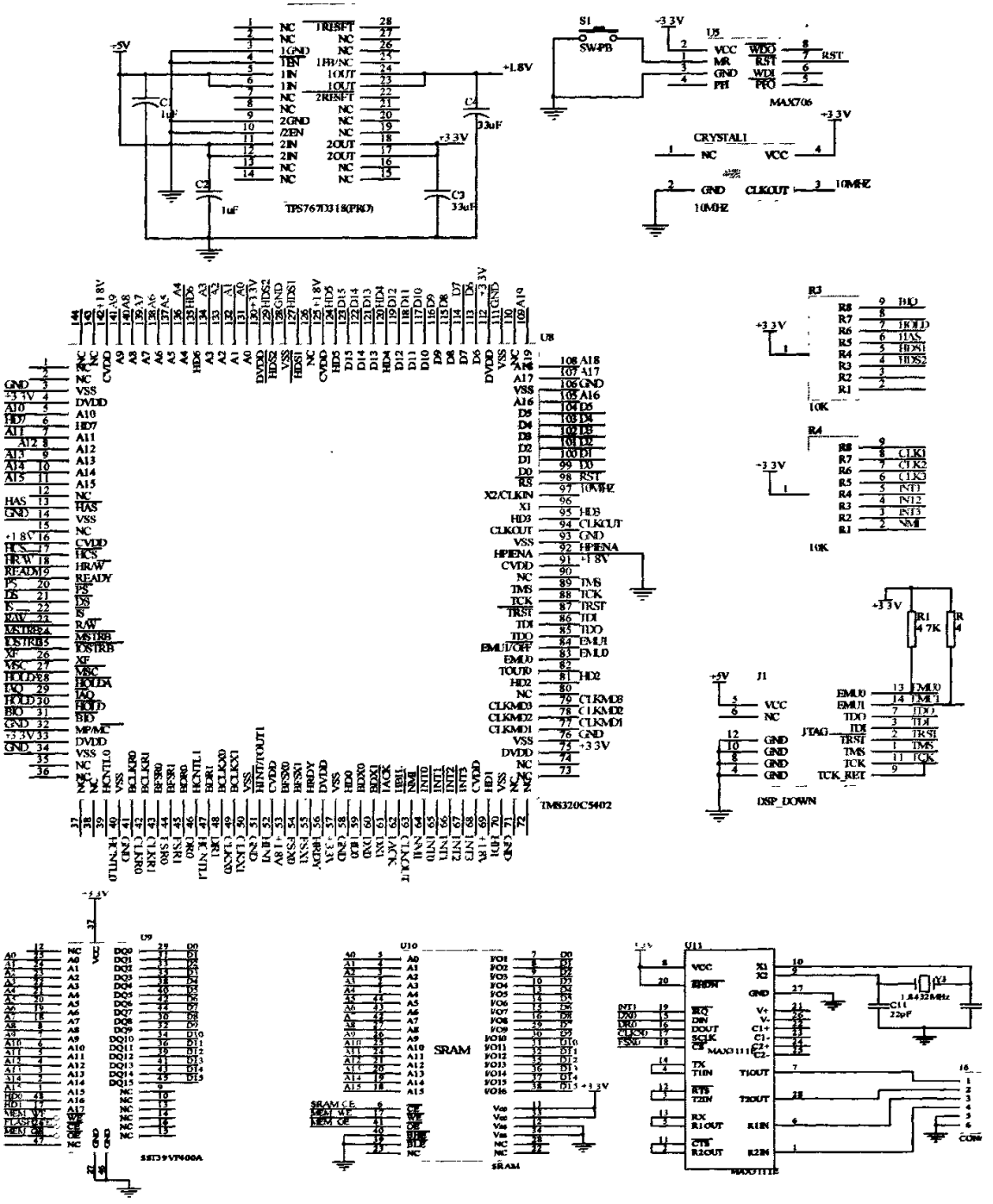
[39] 曹宇，魏丰，胡士毅. 用51单片机控制RTL8019AS实现以太网通信[J]. 电子应用技术 2003，29(1):21-23

[40] 习博方，彦军. 工业以太网中网络通信技术的研究[J].微计算机信息 2005，21（2）：148-149

附录 A 以太网接口系统模块电路原理图



附录 B DSP 系统模块电路原理图



在学研究成果

1. 苑玮琦，莫云鹏. 串口－以太网转换器的设计与实现，微计算机信息已录用 （中文核心期刊）

致 谢

在研究生的三年学习生活期间，我的导师苑玮琦教授给予了我极大的关心、帮助和鼓励，在学术方面，苑老师孜孜不倦的给予我指导，特别是她严谨的治学态度，使我受益非浅;在生活上，苑老师无微不至的关怀，平易随和的态度，乐观向上的精神都会使我终身难忘。苑老师不仅在学术上指引了我，而且他那豁达的生活态度也影响了我。今后在工作和生活当中，我都会谨记苑老师的教诲，踏实的工作，积极钻研技术，实现自己的理想。

同时要感谢我的家人，父母从小就教我做人的道理，在这二十几年来呕心沥血的教育、抚养我成长，给予我鼓励和支持，使我在求学生涯中，一直走下去，这是我毕生都不能回报的。

最后，要感谢我的师兄汤永华、刘军以及同实验室同学曹放、张永辉、刘宁、于青城、金晶晶，你们在生活和学习上对我的帮助和鼓励我也会铭记于心。