南京航空航天大学

硕士学位论文

嵌入式防火墙的研究与实现

姓名：葛广超

申请学位级别：硕士

专业：计算机应用技术

指导教师：顾其威;陈兵

20060301

摘　要

随着微电子技术和计算机软硬件技术的发展，嵌入式系统的性能得到了极大的提高，功能日益丰富，应用也越来越广泛。

传统的边界防火墙是保障网络安全的重要手段，但是它存在防外不防内、容易被绕过、易出现网络瓶颈等缺陷。嵌入式防火墙是一种基于嵌入技术的新型防火墙，它将安全策略延伸到了网络末端，有效地克服了传统边界防火墙的局限，为企业提供更为完善的网络安全解决方案。

嵌入式防火墙(Embedded Firewall，后面简称EFW)是将防火墙固化在网卡上，所有进出网卡的数据包都受到防火墙安全策略的控制，安全策略的制定和分发是由策略服务器完成。这种基于硬件来实现的防火墙具有不依赖主机操作系统、不能被绕过和管理更灵活的特点，是一种更加完善的安全基础架构。

本文详细介绍了一种基于ARM7TDMI处理器的EFW网卡的软硬件设计及其具体实现。EFW硬件设计及实现主要包括微处理器选型，NIC(Network Interface Card)芯片选型，FLASH芯片以及SDRAM芯片选型；整体电路设计；印制板制作；元器件焊接以及硬件调试等工作。EFW核心软件设计及实现主要包括BOOTLOADER程序的编写与调试；uC/OS-II嵌入式操作系统的移植；EFW网卡驱动程序的编写与调试；EFW包过滤程序的编写与调试等工作。

该嵌入式防火墙在局域网环境中进行了测试，它能够根据程序中定义的安全策略对进出EFW网卡的数据进行过滤。

关键词：EFW，分布式防火墙，ARM，RTL8029AS，RTL8019AS，uC/OS-II，BootLoader，包过滤

ABSTRACT

Along with the development of micro-electronic technology, the performance of embedded system has been greatly improved, the functions have been enhanced, and the application has been widely used.

The traditional boundary firewall is the important means to grant the network security, but it can just provide protection against outside, not inside, and it is easily been cheated, and also there is network bottle neck. The embedded firewall is a new type of firewall based on the embedded system. It expanded the security strategy to the network end, and overcome the limit of the traditional boundary firewall, and provide to the enterprises with the perfect methods to settle the network security.

EFW is to fix the firewall to the PCI network card, and all of the data packages in and out of the network card have to pass the security strategy of the firewall, and the make and delivery of the security strategy is accomplished by the strategy server. This kind of firewall is independent from host, isn't easy to be cheated and has flexible management. It is a more perfect security basic framework.

This thesis introduces the design of software & hardware as well as the detailed implement of the EFW network card based on the ARM7TDMI processor, including choosing the style of micro-processor, the style of NIC chip, FLASH chip as well as SDRAM chip; the design of integer electro-circuit; the make of print board; the jointing of apparatus and component; and hardware debugging. EFW software design mainly includes programming of Bootloader, EFW card drive, and EFW package filtering; the transplant of uC/OS-II.The implement of EFW has been tested in LAN environment, and it is able to filter the data in and out of EFW network card according to the security strategy defined by the program.

Key Words: Embeded Firewall,Distribute Firewall,ARM,RTL8029AS,RTL8019AS uC/OS-II,BootLoader,Package Filtering

承诺书

本人郑重声明：所呈交的学位论文，是本人在导师指导下，独立进行研究工作所取得的成果。尽我所知，除文中已经注明引用的内容外，本学位论文的研究成果不包含任何他人享有著作权的内容。对本论文所涉及的研究工作做出贡献的其他个人和集体，均已在文中以明确方式标明。

本人授权南京航空航天大学可以有权保留送交论文的复印件，允许论文被查阅和借阅，可以将学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或其他复制手段保存论文。

作者签名：_____

日　　期：_____

图表清单

第一章 绪论

1.1 引言

进入20世纪90年代以后，以因特网(Internet)为代表的计算机网络得到了飞速的发展，已从最初的教育科研网络逐步发展成为商业网络，并已成为仅次于全球电话网的世界第二大网络[1]。不少人认为现在已经是因特网的时代，这是因为资源共享、信息检索、电子邮件、网络会议、视频点播、远程教育等网络技术的应用给人们的生产和生活带来极大的便利。可以毫不夸大地说，因特网是自印刷术以来人类通信方面最大的变革[1]。然而，信息泄露、假冒用户、篡改信息以及恶意的攻击等网络安全问题也随之而来。人们在享受信息资源共享带来便利的同时，也越来越意识到必须有一种措施来保障自身数据的安全，防止非法的及恶意的访问。

根据我国计算机紧急反映小组/协调中心(CNCERT/CC)的统计数据[2]，2004年1 月~6 月，CNCERT/CC 及其各省分中心共接受来自国内外的网络安全事件报告总计13630 件，与2003 年上半年4675 件相比上升了2 倍多，也比2003 年下半年8639 件上升了近58%。安全事件报告数量持续大幅增长，说明人们对待网络安全事件的态度正在发生改变，人们开始意识到向应急组织报告网络安全事件的重要性。针对出现的安全问题人们都希望找到更好的防范措施，设置防火墙是防范外网非法入侵的有效方法之一。

防火墙是一种设置在网络可信任区域与不可信任区域之间的一道安全屏障。这些防火墙通常置于网络的入口处或者内部网络的边缘，因此也称为“ 边界防火墙 ”。

目前市场上的防火墙主要是个人防火墙和企业级防火墙。个人防火墙是一种能够保护个人计算机系统安全的软件，它可以直接在用户的计算机上运行，使用状态/动态检测的方式，保护一台计算机免受攻击；企业级防火墙，往往是专门的硬件产品，置于企业内部网络与外部网络(通常是因特网)之间，监视进出企业内部网络的数据，为企业内部网络提供安全保护。

传统边界防火墙(Perimeter Firewall)把内部网络一端的用户看成是可信任的，而外部网络一端的用户则都被作为潜在的攻击者来对待。然而，来自企业网络内部的威胁却远远大于来自外网的攻击。FBI和CSI曾对484家公司进行了网

络安全专项调查，调查结果显示：超过85%的安全威胁来自公司内部、有16%来自内部未授权的存取，有14%来自专利信息被窃取，有12%来自内部人员的财务欺骗，而只有5%是来自黑客的攻击；在损失金额上，由于内部人员泄密导致了60，565，000美元的损失，是黑客所造成损失的16倍，病毒所造成损失的12倍[3]，这组数据充分说明了内部人员泄密的严重危害。由于传统边界防火墙自身的缺陷，它对于拨号接入和无线接入等内部用户的攻击没有防范能力。同时，传统的防火墙通常是基于访问控制列表(ACL：Access Control List)进行包过滤的，它对于当前流行的加密通信也是无用武之地，随着网络带宽的不断扩大，传统的边界防火墙越来越成为系统的瓶颈。

正是因为传统防火墙存在诸多的弊端，很多拥有重要保密数据的组织和机构在接入Internet时都很慎重，比如军队的网络是和Internet完全隔离的，只有拥有保密机的单位才可以接入到因特网。随着防火墙技术的不断发展，一种新型的分布式防火墙应运而生。它是一种主机驻留式的安全系统，负责对网络边界、各子网和网络内部各节点之间的安全防护，它的防火墙体系结构由网络防火墙、主机防火墙、中心管理这三部分所组成。它是以主机为保护对象，它的设计理念是主机以外的任何用户访问都是不可信任的，都需要进行过滤。它不再依赖网络拓扑结构，有效地克服了传统防火墙“ 防外不防内 ”的缺陷，同时防火墙的策略实施延伸到了各个网络终端，有效地缓解了网络瓶颈的问题。

根据策略执行机构实现方法的不同，分布式防火墙可以分为软件实现和硬件实现两种。基于软件实现的分布式防火墙构建在主机操作系统之上，由于操作系统本身的不安全，就存在“ 谁保护谁的问题 ”。而基于硬件的实现具有更高的安全性，在系统启动后，防火墙的工作将与主机操作系统基本无关，攻击者将无法通过攻破操作系统的手段来影响防火墙的工作。

而另外一方面，信息时代、数字时代使得嵌入式技术获得了巨大的发展机遇，嵌入式技术的应用也越来越广泛，从电话、手机、PDA设备到电冰箱、洗衣机、电视机顶盒都可以看到嵌入式系统的身影。各种设备的互连是一种必然的趋势，嵌入式设备也需要必须适应网络发展的要求，从硬件上提供各种网络通信接口。基于硬件的分布式防火墙采用嵌入技术，实现对主机的安全防护。

1.2 国内外研究现状

1999年11月，Steven M. Bellovin在他的论文《Distributed Firewall》[4]分析

了传统防火墙的重要缺陷，即传统防火墙是基于拓扑结构的。针对这个特点，他提出了一种分布式的解决方案，将策略的实施分布到各端结点，同时保持集中式的策略管理。在Bellovin的文中，他还特别强调了策略语言和身份标识的作用，策略语言用于描述策略，可能为一个简单的访问列表(Access List),也可能是更复杂的形式，Bellovin给了一个选择，即KeyNote[43]。KeyNote是一种信任管理系统，由Matt Blaze, Joan Feigenbaum 和John Ioannidis设计，适用于很多种类的应用，如IPSec策略控制，电子支付系统等。身份标识用于“认证”内部用户，既然分布式防火墙不基于网络拓扑结构划分内外网，就不能使用IP地址这种不可靠的标识，Bellovin指出，更为可靠的方式是采用加密的IPsec证书[4]。

2000年，Sotiris Loannids, Angelos D. Keromytis, Steve M. Bellovin, Jonathan M.Smith在《Implementing a Distributed Firewall》[5]一文中，按照Bellovin 1999年提出的观点实现了一个分布式防火墙的原型系统。其实现建立在OpenBSD操作系统上，通过修改系统内核调用的方法来实现策略执行机制，整个实现包含内核扩展模块以及用户空间程序。系统采用KeyNote语言描述策略，使用IPsec协议分发策略。

2001年，Charles Payne, Tom Markham在《Architecture and Applications for a Distributed Embedded Firewall》[6]实现了一种基于硬件的分布式防火墙。该文对比2000年Bellovin等人的实现，指出基于硬件的分布式防火墙具有更高的可靠性。该实现的核心为3Com公司生产的3CR990系列网卡，该系列网卡嵌入有3XP处理芯片，称为嵌入式防火墙网卡EFW NIC(Embedded Firewall Network Interface Card)。EFW NIC负责执行策略，一个中心服务器(亦受EFW NIC保护)提供管理界面、策略制定和分发工具，审计和日志组件。该系统的运行与主机操作系统的关联最小，具有很高的安全性，可以应用于实际的网络中。

国内对于分布式防火墙的研究尚处于初级阶段。饶泓，陈炼，闻淑芳在[15]中对分布式防火墙的概念和原理作了一些介绍；陈春玲，雷世荣，陈丹伟在[16]中介绍了分布式防火墙三种不同的实现，其中包括一个嵌入式的实现。彭倩岚，李之棠以及纪羽中，胡静等人分别在[17]和[18]中研究了分布式防火墙的安全机制和信任管理系统。但国内目前还处于技术研究阶段，尚未有成型的嵌入式防火墙产品，然而可以预见，随着人们对于网络安全问题关注程度的提高，嵌入式防火墙将有很好的市场前景和发展空间。

1.3 论文研究内容

虽然分布式防火墙的概念已经出现有多年了，但对于分布式防火墙具体的实现方式仍未达成一致的见解。基于嵌入式技术的分布式防火墙是其中的一个变体。Tom Markham和Charlie Payne是这一变体的支持者，由硬件的强制性可以保障防火墙的安全性。然而，网卡上嵌入芯片的处理能力毕竟有限，不适宜做复杂的运算，在生产成本与所需功能之间需要做一个权衡。 Bellovin等人基于软件的实现具有更高的灵活性和更高级的策略定义，然而与操作系统联系过于紧密，易被黑客工具攻破，面临与个人防火墙同样的尴尬。

本课题设计并实现了一种嵌入式防火墙，采用32位嵌入式处理器(ARM7TDMI)。该防火墙的策略执行部件为嵌入ARM处理器的网络接口卡(NIC)，工作在Ehternet/IEEE802.3网络中，通过PCI接口与主机相连，嵌入式网络接口卡中运行防火墙包过滤程序作为策略执行部件。本文首先介绍了嵌入式防火墙的相关技术，并对目前国内外的软硬实现方法进行了对比分析，然后给出了一种嵌入式防火墙软硬件实现的方案，接着详细论述了该嵌入式防火墙硬件的选型，电路的设计及调试，最后给出了该嵌入式防火墙的核心软件设计与实现，其中包括系统引导程序BOOTLOADER的编写，uC/OS-II操作系统的移植以及嵌入式防火墙核心程序－包过滤程序的编写。希望本文能推动国内嵌入式防火墙的研究和发展，并希望能将嵌入式防火墙产品推广到实际应用中去。

1.4 论文结构

本文共六章，各章内容分别如下：

第一章：绪论，介绍了课题的背景，国内外的研究现状，本课题的研究意义，以及论文的主要内容和结构。

第二章：详细介绍了防火墙的概念和原理，重点介绍了分布式防火墙的特点、基本原理以及分布式防火墙基于软件和基于硬件的两种实现方法，并分析了两种方法各自的优缺点。

第三章：阐述了嵌入式防火墙的功能及技术要求，给出了嵌入式防火墙硬件、软件设计方案，以及相应的软硬件开发环境。

第四章：详细介绍了嵌入式防火墙的硬件设计与实现方案；各主要模块的功能，电路设计图、各模块之间的接口以及硬件电路的调试等。

第五章：详细介绍了嵌入式防火墙的核心软件设计与实现方案，包括系统

的初始化程序BOOTLOADER、嵌入式操作系统uC/OS-II的移植、网卡驱动程序的编写与调试以及防火墙包过滤程序等。

第六章：总结与展望，对下一步工作提出建议。

最后为致谢和参考文献。

第二章 防火墙概述

2.1 传统防火墙

2.1.1 防火墙技术概述

人们在木制结构房屋的周围，构筑起防止火势蔓延的屏障，这种屏障被称为防火墙[13]。在计算机网络中，也有这样一种屏障，用于保护内部的敏感数据，抵御外来的威胁，这种屏障亦被称为防火墙。

今天，防火墙仍为保障网络安全的一个重要手段，它作为一个门户，位于被保护的内部网络和外部非信任网络之间，选择性的允许和限制内网与外网之间的信息交互，既保障正常的因特网访问，也保护内部的重要数据不受破坏和越权访问。防火墙往往是运行在一台专用计算机之上的软件实体，用于识别和屏蔽非法的访问。

防火墙根据其防范的方式，大致可以归纳为两大类： 分组过滤(Packet Filtering)和应用网关(Application Gateway)[39]。分组过滤的实现比较简单，它工作在网络层和传输层，只需根据分组包头中的源地址，端口号、目的地址，端口号以及协议类型，对照规定的访问控制列表(ACL：Access Control List)，即可判断是否允许网络包的通过，比如大多数的路由器都具有ACL功能。由于其工作在网络层和传输层，对应用层透明，所以制定的规则相对简单，缺乏对应用程序一级的控制，比如无法检测FTP协议的PORT命令等。应用网关工作在应用层，起到用户程序代理的作用，代替用户与被访问的服务器对话。应用程序通过代理获取服务，所有对外网的访问均要通过应用代理。应用代理可以获取有关应用程序的更多的信息，可以制定更为精细的安全策略，例如允许一个Web浏览器接受一个ActiveX插件或是Java Applet，这样即可不必过滤整个Http会话中的网络包。在实际应用中往往都是两种形式结合使用的。

分组过滤和应用网关都是位于内部网与外部网的结合处，起着监视和隔离的作用，然而这也正是传统防火墙自身的一大局限性，随着网络技术的发展，人们开始认识到传统防火墙在现代网络环境中存在越来越多的局限性。

2.1.2 传统防火墙的局限性

传统防火墙因其位于网络的入口处，亦称为边界防火墙(Perimeter Firewall)。

这一特点导致了传统防火墙的很多局限，可以归纳为以下几点：

(1) 防外不防内。 传统防火墙一般位于网络的入口处，对于外来的攻击可以有效地抵制，但是对于网络内部的攻击却是无能为力。传统防火墙是基于这样一个假设的，即每一个外部用户都是一个潜在的敌人而内部用户均是可信任的。然而实际环境中，大多数的攻击是来自于内部，即使用户是诚实可靠的，一些恶意的病毒，蠕虫代码亦会将诚实的用户变成一个不知情的攻击者[40]。

(2) 瓶颈问题。防火墙位于网络的接入口，其吞吐量直接影响网络的性能。虽然计算机硬件的处理能力在不断提高， 但是更快的网络速度和更复杂的协议相结合产生的效果对防火墙的计算能力提出了严峻的挑战，使得防火墙容易成为网络的瓶颈和单点失效点。

(3) 易被绕过。现在计算机接入网络的方式多种多样，人们可以很轻易地建立一个非授权的接入点。各种隧道技术，无线接入技术和拨号访问都可以绕过防火墙的安全机制。纵然防火墙的策略定义的很完善，对它无法控制的接入也是无可奈何。对于这种网络外部的远程访问，亟需一种行之有效的保护和防范措施。

(4) 端到端的加密对传统防火墙是一个威胁。传统防火墙的分组过滤方法需要察看分组包头的信息来进行过滤，防火墙无法从加密的报文中获取其所需的信息。

(5) 策略的制定和维护复杂。传统防火墙是根据网络的拓扑结构制定规则的。在大型的网络中，往往有多个接入点和内部防火墙，这使得策略管理非常复杂，因为没有一种通用的管理机制，一般要依靠网络管理员的能力和经验。

为了解决传统防火墙的缺陷，同时保持其优点，有学者[5]提出了分布式防火墙的概念，将防火墙的功能下推到每个端节点。

2.2 分布式防火墙

2.2.1 分布式防火墙的提出

传统防火墙的很多缺陷都是来自于这样一个原因，即依赖于网络拓扑结构和单一接入控制。Tom Markham形象地比喻：“ …网络工程师和安全管理员被绑住脚踝与攻击者进行一场比赛，而网络拓扑结构就是这个绑绳 ”[40]。想要克服传统防火墙的缺点，就必须打破这一束缚。

Steven M. Bellovin于1999年在他的论文[5]中首次提出了分布式防火墙的概

念。在这种模式下，策略仍是由一个中心统一定义，而策略的执行确是由各个端结点完成。如此便消除了单一接入点，内网外网的划分并不依赖于网络的拓扑结构，因此内网的定义具有更多的逻辑意义，可以包含公司内无线接入的用户、拨号用户、通过VPN连接的用户，而不仅限于传统意义上某个房间或某栋建筑中的网络。相应地，防火墙的策略也不需按照网络拓扑结构来制定访问控制列表，管理员可以更专注于对被保护的对象来制定规则。 随后，Bellovin等人于2000年在OpenBSD系统上实现了一个原型系统，并举例说明了实际应用场景[5]。

2.2.2 分布式防火墙的主要特点

分布式防火墙的基本特征是：集中定义策略，分布实施策略。相对于传统防火墙，分布式防火墙具有以下特点：

(1)主机驻留。分布式防火墙的最主要特点就是采用主机驻留方式，它驻留在被保护的主机上，该主机以外的网络都被认为是不可信任的，因此可以针对该主机上运行的具体应用和对外提供的服务设定针对性很强的安全策略。这样就可以使安全策略不仅仅停留在网络与网络之间，而且可以延伸到每个网络末端，克服边界防火墙容易被绕过的缺陷。

(2)策略的制定和维护比较灵活机动。分布式防火墙的安全策略可以在企业内部网集中定义的基础上，各网络终端针对自己的需求定义安全策略。通过全方位的安全策略，构建了多层次立体的防范体系。

(3)分布式防火墙的安全策略延伸到了每个网络末端，从而消除了结构性瓶颈问题，提高了系统性能，同时也可以进行扩充。

2.2.3 分布式防火墙的基本原理

一个分布式防火墙系统包含三个基本组件：

策略语言，用于描述安全策略。分布式防火墙系统提供一个策略服务器，系统管理员利用策略服务器上提供的工具和策略数据库来制定和保存策略。策略最简单的形式就是传统防火墙中的包过滤规则，也可以使用更为高级的策略定义语言，如KeyNote。

安全的分发策略的机制。策略保存在策略服务器上，在向特定的主机发送策略的时候需要有一种机制保证策略不被篡改和伪造。端主机和策略服务器亦需要证实自身的身份。

策略的执行部分。分布式防火墙系统中策略的执行下推到主机，由各主机对进出其自身的分组或连接进行过滤。这也正是分布式防火墙体现其分布式的地方，每个结点都参与防火墙的工作。把策略的执行下放到各主机端，最直接的好处就是可以分散传统防火墙的工作，避免瓶颈并保证防火墙不会被绕过，因为任何进出网络的数据包都会被防火墙“看到”。

分布式防火墙的工作过程可以描述如下：由系统管理员在策略服务器上针对受保护的对象制定策略；策略服务器上的策略管理组件将策略编译成适用于各个主机的规则集；各主机在启动时从策略服务器上下载最新的规则集；主机上的策略执行模块根据规则集对进出的网络包或连接做出仲裁，决定是否接受；策略更新后，策略服务器应当通知主机下载最新版本；主机上的策略执行模块在正常工作时向策略服务器发送审计事务。

2.3 分布式防火墙的实现方法

分布式防火墙打破了网络拓扑结构对传统防火墙的束缚将安全策略分布到各个网络终端执行，从而使每一台主机都起到了传统防火墙的功能。目前分布式防火墙的实现分为两种方法：软件实现和硬件实现。

2.3.1 分布式防火墙的软件实现方法

首先我们要区别分布式防火墙的软件实现与个人防火墙，个人防火墙大都是安装在主机上的软件防火墙，如天网、瑞星、Norton等，它们的特点是在各个主机定义策略、实施策略，从而不能做到策略的集中管理。分布式防火墙的软件实现是通过修改操作系统的connect( )，accept( )函数调用对进出的连接应用安全策略，由于操作系统具有丰富的系统调用和库函数，所以安全策略可以针对某个用户或者某个应用程序来具体定义。

但是这种基于应用程序的软件防火墙依赖于主机操作系统，操作系统本身存在许多安全漏洞，所以很难说防火墙保护主机操作系统，还是主机操作系统保护防火墙，一旦不怀好意的用户利用操作系统的漏洞控制了主机，安装在主机操作系统之上的软件防火墙就形同虚设。

2.3.2 分布式防火墙的硬件实现方法

显然，基于主机操作系统的软件形式的防火墙无论是应用程序方式，还是嵌入操作系统内核方式都很难做到不被攻破，所以，我们需要把防火墙与主机

操作系统分离开。

基于硬件的分布式防火墙是在网络接口卡(NIC)上集成了处理器、存储器以及其他一些功能器件，这样即使是主机用户也无法干涉安全策略的执行机制，满足了独立于主机操作系统的要求。防火墙和主机操作系统的相互独立，使得恶意的攻击即使攻破了主机操作系统，也无法取得对防火墙的控制权，避免了攻击者以被攻破的主机为跳板进一步攻击网络其他部分的可能。

2.3.3 分布式防火墙软硬件实现方法的对比

针对以上分布式防火墙的软硬件的两种实现方法，我们对比如下：

1、运行的环境。分布式软件防火墙运行在主机的操作系统之上，由于主机操作系统自身的安全问题，需要不断增加操作系统的补丁来提高安全性；分布式硬件防火墙运行在独立的嵌入式操作系统上，是一个独立封闭的系统。

2、系统的运行速度。分布式软件防火墙受系统资源的影响比较大，而分布式硬件防火墙受硬件的配置影响比较大。

3、稳定性和兼容性。分布式软件防火墙因为对操作系统的依赖性，使得它的兼容性存在问题，而分布式硬件防火墙采用专门的软硬件系统，稳定性和兼容性更好。

4、功能的灵活性。分布式软件防火墙架构不依赖硬件，因此功能可以根据用户的需求进行定制，而分布式硬件防火墙则考虑硬件的成本，提供的功能和灵活性相对要差一些。

5、升级情况。分布式软件防火墙基于软件的，升级比较方便，而分布式硬件防火墙的升级涉及到硬件的升级，所以升级的代价比较大。

2.4 小结

本章简要分析了传统防火墙的缺陷，在此基础上提出了分布式防火墙的概念和基本原理，重点对分布式防火墙的软硬件实现方法做了对比分析。最后简要介绍了分布式防火墙实现所采用的一些技术。

第三章 嵌入式防火墙总体设计

前一章简单介绍了分布式防火墙的基本概念，工作原理以及软硬件两种实现方法，本章主要介绍基于硬件的分布式防火墙设计，即嵌入式防火墙的设计。

3.1 嵌入式防火墙结构、组成及技术要求

3.1.1 嵌入式系统简介

嵌入式系统的应用日益广泛，可以说无所不在、无处不在，在现实生活中，小到各种家电、手机、PDA、数码相机，大到汽车、飞机、火箭、宇宙飞船都有嵌入式系统的应用。嵌入式系统是指以应用为中心，以计算机技术为基础，软件硬件可裁减，适应应用系统对功能、可靠性、成本、体积和功耗严格要求的专用计算机系统[8]。嵌入式计算机是嵌入式系统的核心，它与实际应用的广泛结合，是在一切可能的设备上都使用计算机，将这些设备变得更智能化、可计算化。嵌入式计算机是构成未来数字化世界的基础细胞、元素[7]。

3.1.2 嵌入式系统的组成结构

嵌入式系统是专用计算机应用系统，它具有一般计算机组成的共性，即是由硬件和软件组成的。嵌入式系统的硬件是嵌入式系统软件环境运行的基础，它提供了嵌入式系统软件运行的物理平台和通信接口；嵌入式操作系统和嵌入式应用软件则是整个系统的控制核心，控制整个系统的运行，提供人机交互的信息等。图3.1描述了嵌入式系统软硬件各部分的组成结构。

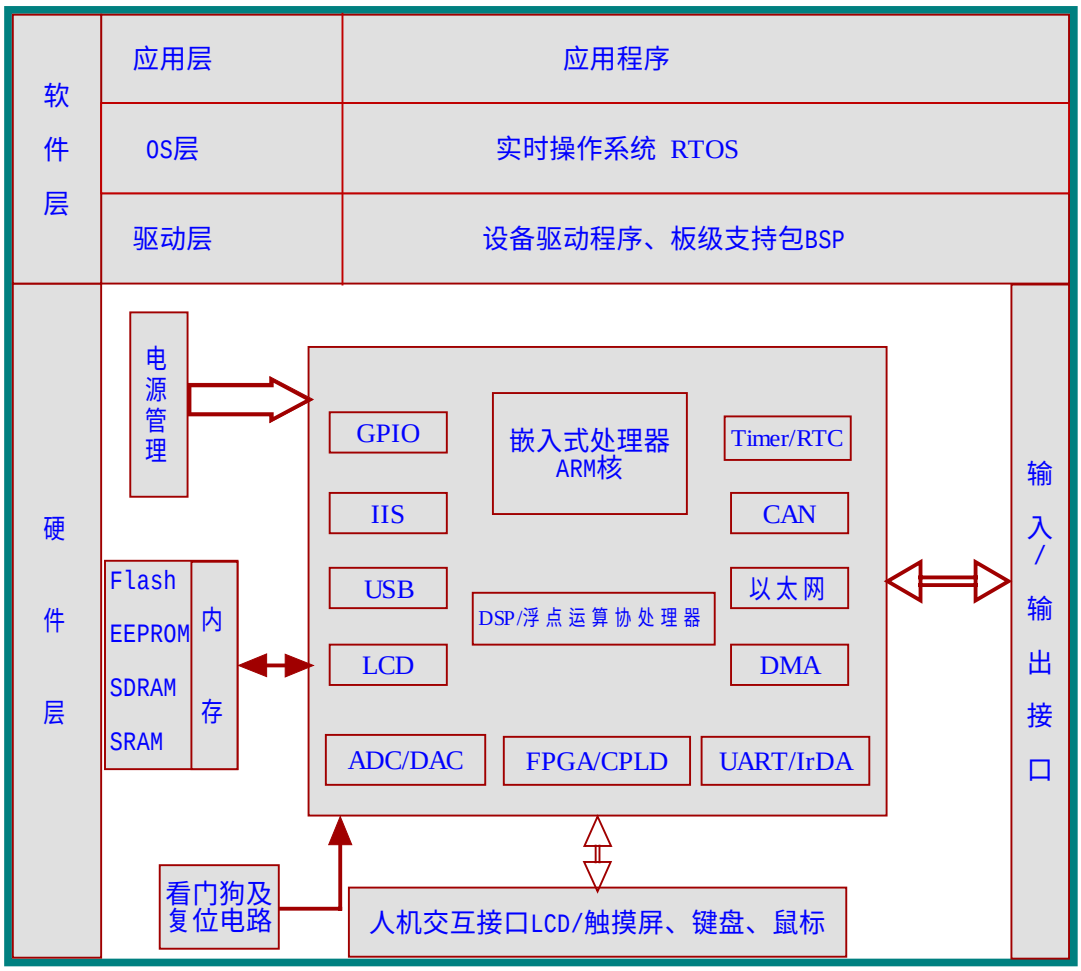


图 3.1 典型的嵌入式系统组成图

在图3.1中描述了一个典型的嵌入式系统的组成结构，它分为硬件层和软件层两个部分。硬件架构是以嵌入式处理器为中心，由存储器、I/O设备、通信模块以及电源等必要的辅助接口组成。嵌入式系统是根据用户的需求，量身定做的专用的计算机应用系统，在实际的应用中，除了处理器和基本的外围电路以外，其余的电路都可以根据用户需要进行裁剪、定制。

嵌入式系统的软件层由底往上分为驱动程序层、操作系统层和应用程序层。在设计简单的应用程序时，可以不使用操作系统，但是在设计复杂的程序时，使用操作系统可以方便管理和控制内存、多任务、中断以及资源和合理分配。利用操作系统提供的API接口可以减轻程序员的负担，方便应用程序的编写。

3.1.3 嵌入式防火墙的结构

嵌入式防火墙是一种在分布式防火墙基础上提出的解决方案，该解决方案

基于硬件实现，将分布式防火墙的实现嵌入到网卡中，以防止被保护的终端绕过基于软件的分布式防火墙对网络进行攻击。嵌入式防火墙是嵌入式系统的一种典型应用，嵌入式防火墙的结构如图3.2所示：

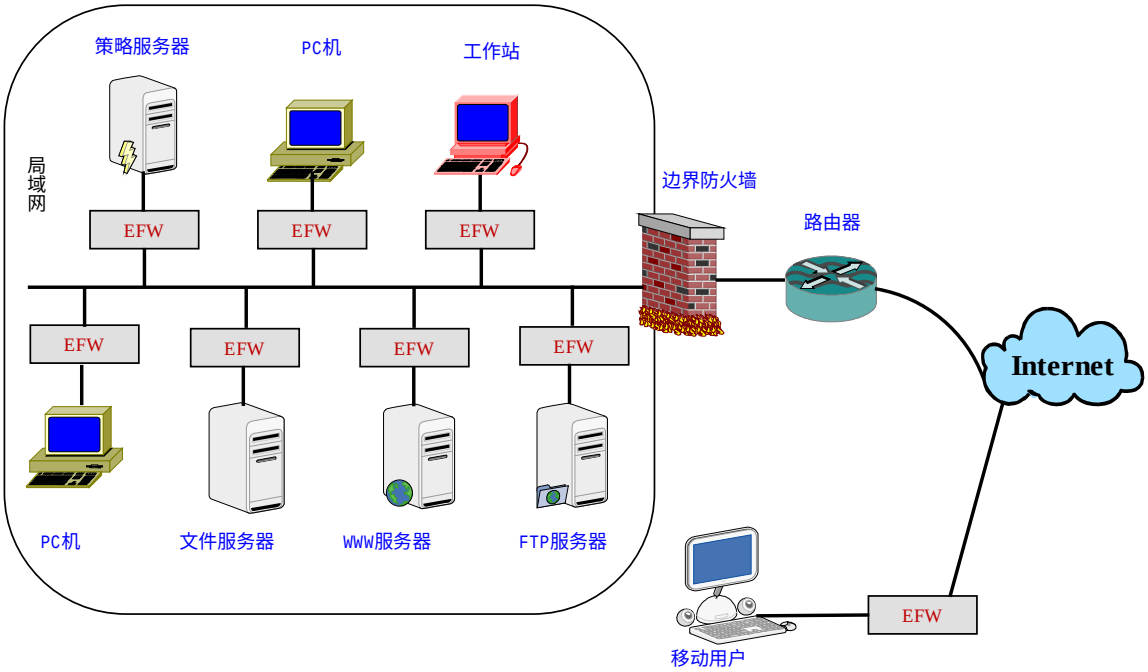


图 3.2 嵌入式防火墙的结构

在图3.2中，LAN表示一个受保护的内部网络，比如一个企业网络，Internet是因特网，代表一个不安全的区域。每一个EFW就是一块带有防火墙功能的网卡，该网卡在硬件一级实现对网络包的实时过滤，网卡上有自己的处理器和存储区，独立于主机操作系统工作。所有的EFW网卡构成分布式防火墙系统的策略执行组件，策略服务器负责管理这些EFW NIC。内部网络上的每一台PC，工作站或是服务器，包括策略服务器自身，均受到分布式防火墙的保护。企业内部的移动用户也可以安装上EFW NIC，获取分布式防火墙的保护以及获准访问企业内部资源。然而，使用分布式防火墙并不意味着完全放弃传统边界防护墙。边界防火墙可以作为内部网络对外的第一道屏障，可以有效地将大量的外部攻击抵御于内部网络之外，不必将其放入内部后再对其抵御，可以减少内部网络的数据流量和EFW NIC的工作，因此在实际应用中，采用两者结合的方法，可以获得更高的安全性能。

3.2 嵌入式防火墙硬件方案

嵌入式应用系统的设计包含硬件系统的设计和软件系统设计两个部分，并且这两部分的设计是互相关联、密不可分的，嵌入式应用系统的设计经常需要在硬件和软件的设计之间进行权衡与折中。这也是嵌入式应用系统设计与其他的纯粹的软件设计或硬件设计最大的区别。

3.2.1 嵌入式防火墙硬件设计方案

嵌入式防火墙硬件实际上就是这个嵌入式系统的硬件架构，根据本文3.1.2小节对嵌入式系统硬件架构的描述，嵌入式系统的硬件结构根据用户的需要可以自行裁剪，本文根据嵌入式防火墙的要求，给出了如下的嵌入式系统硬件布局，如图3.3所示：

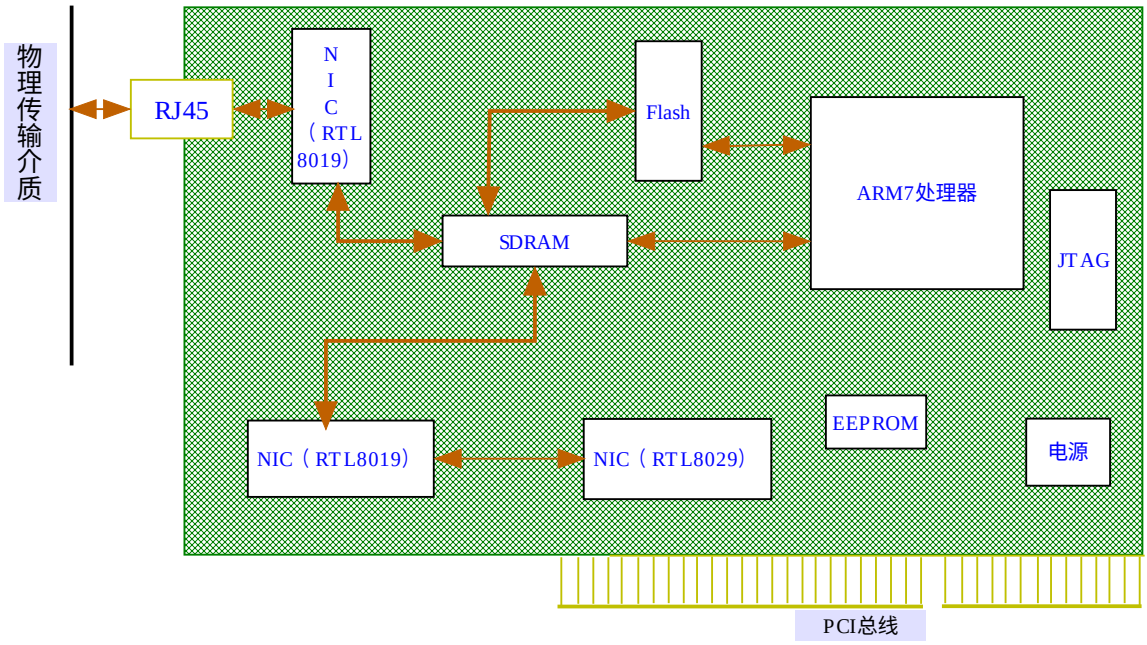


图 3.3 EFW 网卡的结构示意图

该硬件架构可以分为两大部分：处理器及外围电路和以太网接口模块。

1、选择处理器是嵌入式系统硬件设计中最为重要的一环。某一种处理器在设计时，就被确定为针对某个领域的特殊应用，而不是像PC机的处理器那样针对桌面系统这样的单个领域[9]。从图3.3可以看出EFW网卡ARM处理器仅仅处理进出网卡的数据，并没有复杂的算法运算，所以对处理器的性能要求并不高。一个处理器通常具有一个基本的外围电路，由处理器、时钟复位电路、SDRAM、

Flash ROM及JTAG口等部件构成。在本文下面章节中会根据EFW网卡的具体需要给出选择处理器的详细理由。

2、以太网接口模块由以太网控制器、隔离变压器及其外围电路组成，在处理器的控制下将主机发出网络的数据包和网络发回主机的数据包转移到网卡的SDRAM中进行分析处理。在EFW网卡设计中采用了两个支持10Mb以太网的RTL8019AS和一个支持10Mb以太网的RTL8029AS以太网控制器。这里的RTL8029AS以太网控制器与PC机中使用的网卡功能和电路结构相同，是采用PCI总线与处理器连接，可以实现即插即用。而RTL8019AS以太网控制器则是采用总线方式与ARM处理器相连接。

3.2.2 嵌入式防火墙的层次结构

根据上一小节对嵌入式防火墙硬件的设计，笔者进一步对嵌入式防火墙进行了分层细化。通过层次化的设计，便于分析该嵌入式防火墙的工作原理。嵌入式防火墙分层结构如图3.4所示。

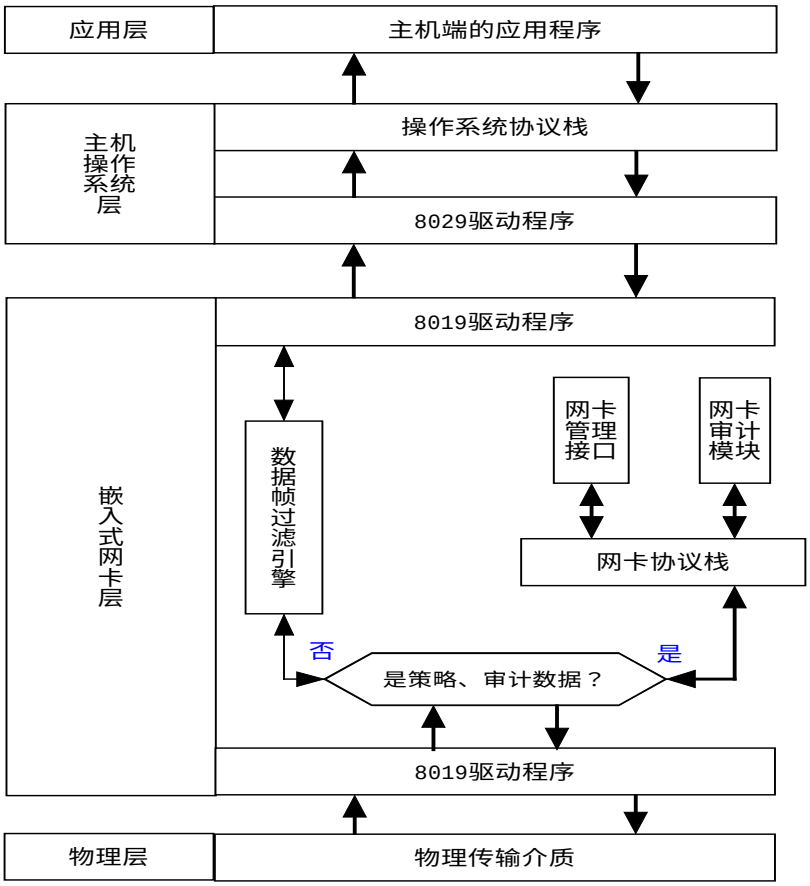


图 3.4 EFW 网卡的层次结构图

从图3.4中可以看出，经过EFW网卡的数据流分为两种：一种是普通数据流，另一种是服务器和客户端之间的策略数据和审计数据。

主机通过EFW网卡与其他的主机进行正常的通信，本文称这种数据流为普通数据流。普通数据流的传输有两种情况：

1)数据流是从主机端发起，数据的传输过程是：主机端的应用程序通过主机端的协议栈传给8029芯片，主机的处理器控制8029芯片将数据传输给8019芯片，ARM处理器控制8019芯片将数据接收到EFW网卡的SDRAM中，ARM控制包过滤程序对接收的数据包进行过滤，而后ARM处理器再控制另外一个8019芯片将过滤通过的数据包传输到网线上。

2)数据从网络中到来，则ARM处理器控制8019芯片将该数据包接收到EFW网卡的SDRAM中，接着ARM处理器控制包过滤程序对该数据包进行分析，如果该数据包不是策略和审计数据，就将它传给另外一个8019芯片，8019芯片又传给8029芯片，最后主机处理器将该数据包传给协议栈，进行相应的协议处理，这样就完成了普通数据的通信过程。

ARM处理器对来自网络中的数据分析时，如果是策略和审计数据则按照下面的流程来处理：ARM处理器将该数据包交由EFW网卡中的协议栈，协议栈根据数据包的协议类型与策略服务器建立连接，进行策略和审计数据的加密传输。

3.2.3 嵌入式ARM微处理器结构以及选型

3.2.3.1 RISC体系结构

传统的CISC(Complex Instruction Set Computer，复杂指令集计算机)结构有其固有的缺点，即随着计算机技术的发展而不断引入新的复杂的指令集，为支持这些新增的指令，计算机的体系结构会越来越复杂，然而，在CISC指令集的各种指令中，其使用频率却相差悬殊，大约有20%的指令会被反复使用，占整个程序代码的80%。而余下的80%的指令却不经常使用，在程序设计中只占20%，显然，这种结构是不太合理的。

基于以上的不合理性，1979年美国加州大学伯克利分校提出了RISC(Reduced Instruction Set Computer，精简指令集计算机)的概念，RISC并非只是简单地去减少指令，而是把着眼点放在了如何使计算机的结构更加简单合理地提高运算速度上。RISC结构优先选取使用频率最高的简单指令，避免复杂指令；将指令长度固定，指令格式和寻址方式种类减少；以控制逻辑为主，不用

或少用微码控制等措施来达到上述目的。

到目前为止，RISC体系结构也还没有严格的定义，一般认为，RISC体系结构应具有如下特点：

- 采用固定长度的指令格式，指令归整、简单、基本寻址方式有2～3种。
- 使用单周期指令，便于流水线操作执行。
- 大量使用寄存器，数据处理指令只对寄存器进行操作，只有加载/存储指令可以访问存储器，以提高指令的执行效率。
- 所有的指令都可根据前面的执行结果决定是否被执行，从而提高指令的执行效率。
- 可用加载/存储指令批量传输数据，以提高数据的传输效率。
- 可在一条数据处理指令中同时完成逻辑处理和移位处理。
- 在循环处理中使用地址的自动增减来提高运行效率。

当然，和CISC架构相比较，尽管RISC架构有上述的优点，但决不能认为RISC架构就可以取代CISC架构，事实上，RISC和CISC各有优势，而且界限并不那么明显。现代的CPU往往采用CISC的外围，内部加入了RISC的特性，如超长指令集CPU就是融合了RISC和CISC的优势，成为未来的CPU发展方向之一。

采用RISC架构的ARM微处理器一般具有如下特点：

- 体积小、低功耗、低成本、高性能。
- 支持Thumb(16位)/ARM(32位)双指令集，能很好的兼容8位/16位器件。
- 大量使用寄存器，指令执行速度更快。
- 大多数数据操作都在寄存器中完成。
- 寻址方式灵活简单，执行效率高。
- 指令长度固定。

ARM7系列微处理器为低功耗的32位RISC处理器，最适合用于对价位和功耗要求较高的消费类应用。ARM7微处理器系列具有如下特点：

- 具有嵌入式ICE－RT逻辑，调试开发方便。
- 极低的功耗，适合对功耗要求较高的应用，如便携式产品。
- 能够提供0.9MIPS/MHz的三级流水线结构。
- 代码密度高并兼容16位的Thumb指令集。
- 对操作系统的支持广泛，包括uC/OS、uCLinux、Palm OS等。
- 指令系统与ARM9系列、ARM9E系列和ARM10E系列兼容，便于用户的

产品升级换代。

- 主频最高可达130MIPS，高速的运算处理能力能胜任绝大多数的复杂应用。

ARM7系列微处理器的主要应用领域为：工业控制、Internet设备、网络和调制解调器设备、移动电话等多种多媒体和嵌入式应用。

ARM7系列微处理器包括如下几种类型的核：ARM7TDMI、ARM7TDMI-S、ARM720T、ARM7EJ。其中，ARM7TMDI是目前使用最广泛的32位嵌入式RISC处理器，属低端ARM处理器核。TDMI的基本含义为：

- T： 支持16为压缩指令集Thumb。
- D： 支持片上Debug。
- M： 内嵌硬件乘法器(Multiplier)。
- I： 嵌入式ICE，支持片上断点和调试点。

3.2.3.2 ARM微处理器的寄存器结构

ARM处理器共有37个寄存器，被分为若干个组(BANK)，这些寄存器包括：

- 31个通用寄存器，包括程序计数器(PC指针)，均为32位的寄存器。
- 6个状态寄存器，用以标识CPU的工作状态及程序的运行状态，均为32位，目前只使用了其中的一部分。

同时，ARM处理器又有7种不同的处理器模式，在每一种处理器模式下均有一组相应的寄存器与之对应。即在任意一种处理器模式下，可访问的寄存器包括15个通用寄存器(R0～R14)、1至2个状态寄存器和程序计数器。在所有的寄存器中，有些是在7种处理器模式下共用的同一个物理寄存器，而有些寄存器则是在不同的处理器模式下有不同的物理寄存器。

3.2.3.3 ARM微处理器的指令结构

ARM微处理器在较新的体系结构中支持两种指令集：ARM指令集和Thumb指令集。其中，ARM指令为32位的长度，Thumb指令为16位长度。Thumb指令集为ARM指令集的功能子集，但与等价的ARM代码相比较，可节省30%～40%以上的存储空间，同时具备32位代码的所有优点。

3.2.3.4 ARM微处理器的应用选型

鉴于ARM微处理器的众多优点，随着国内外嵌入式应用领域的逐步发展，ARM微处理器必然会获得广泛的重视和应用。但是，由于ARM微处理器有多达

十几种的内核结构，几十个芯片生产厂家，以及千变万化的内部功能配置组合，给开发人员在选用时带来一定的困难，所以，对ARM芯片做一些对比研究是十分必要的。

以下从本课题具体应用的角度出发，对选择ARM微处理器时所应考虑的主要问题做一些简要的探讨。

1、ARM微处理器内核的选择

从前面所介绍的内容可知，ARM微处理器包含一系列的内核结构，以适应不同的应用领域，用户如果希望使用WinCE或标准Linux等操作系统以减少软件开发时间，就需要选择ARM720T以上带有MMU(Memory Management Unit)功能的ARM芯片，ARM720T、ARM920T、ARM922T、ARM946T、Strong-ARM都带有MMU功能。而ARM7TDMI则没有MMU，不支持Windows CE和标准Linux，但目前有uCLinux、uC/OS等不需要MMU支持的操作系统可运行于ARM7TDMI硬件平台之上。

2、系统的工作频率

系统的工作频率在很大程度上决定了ARM微处理器的处理能力。ARM7系列微处理器的典型处理速度为0.9MIPS/MHz，常见的ARM7芯片系统主时钟为20MHz-133MHz，ARM9系列微处理器的典型处理速度为1.1MIPS/MHz，常见的ARM9的系统主时钟频率为100MHz-233MHz，ARM10最高可以达到700MHz。不同芯片对时钟的处理不同，有的芯片只需要一个主时钟频率，有的芯片内部时钟控制器可以分别为ARM核和USB、UART、DSP、音频等功能部件提供不同频率的时钟。

3、芯片内存储器的容量

大多数的ARM微处理器片内存储器的容量都不太大，需要用户在设计系统时外扩存储器，但也有部分芯片具有相对较大的片内存储空间，如ATMEL的AT91F40162就具有高达2MB的片内程序存储空间，用户在设计时可考虑选用这种类型，以简化系统的设计。

4、芯片内外围电路的选择

除ARM微处理器核以外，几乎所有的ARM芯片均根据各自不同的应用领域，扩展了相关功能模块，并集成在芯片之中，我们称之为片内外围电路，如USB接口、IIS接口、LCD控制器、键盘接口、RTC、ADC和DAC、DSP协处理器等，设计者应分析系统的需求，尽可能采用片内外围电路完成所需的功能，

这样既可简化系统的设计，同时提高系统的可靠性。

我们选择三星公司的ARM7TDMI内核S3C44B0X处理器，主要考虑到它的通用性、经济性。

S3C44B0X虽然是一款不带MMU的ARM7TDMI微处理器，但已经有uCLinux、uC/OS-II移植到ARM7TDMI微处理器平台上的成功案例，为我们在EFW上移植操作系统提供了便利。

S3C44B0X处理器工作频率为66MHz，能够满足数据处理需求，它提供了丰富的内置部件，包括：8KB cache，内部SRAM，带自动握手的2通道UART，4通道DMA，外部存储器控制器(片选逻辑，FP/EDO/SDRAM控制器)，RTC等等，它具有多种低功耗工作模式。

S3C44B0X片内只有8K的SRAM，需要外接2M的Flash ROM芯片用来保存操作系统和防火墙包过滤程序 运行程序、作为数据缓存还需要外接8M的SDRAM。为了便于调试，JTAG口在EFW网卡设计中也是必须的。

3.2.4 以太网控制器选型

RTL8019AS和RTL8029AS均为高度集成的以太网控制器。它们都遵循IEEE802.3协议；内置16KB的SRAM，用于收发缓冲；全双工，收发同时达到10Mbps；支持10Base5、10Base2、10BaseT,并能自动检测所连接的介质。RTL8019AS支持8位或16位数据总线；而RTL8029AS则支持32位数据总线。RTL8019AS通过总线方式同处理器连接，而RTL8029AS则可以通过PCI总线同主机进行连接。按功能它们均可以划分为：

- 接收/发送模块(Transceiver)：将电缆上传输的0，1比特反串行化，形成字节；或是将待发字节串行化为比特，放到电缆上传输。
- FIFO及FIFO控制逻辑：接收时暂存由收发器收到的字节，到一定阀值后送入接收缓存区。发送时暂存来自发送缓冲区的字节，然后交由收发器串行化为bit。
- 地址识别逻辑：收到的数据帧的目的地址字段与网卡自身的物理地址和多播地址比较，匹配者接受，否则拒绝。
- CRC校验/生成逻辑：收到的数据帧若未通过CRC(循环冗余码)校验，则丢弃。发送数据帧时由CRC生成逻辑自动计算校验值，附在数据帧之后发送。
- 协议逻辑阵列单元：用于实现IEEE802.3协议，包括冲突检测和自动重

传。此外还负责生成帧的前导同步码。

- DMA及缓存控制逻辑：控制收发FIFO与网卡缓存的数据交互(本地DMA)，以及本地缓存与主机内存之间的数据交互(远程DMA)。

RTL8019AS和RTL8029AS性价比都比较高。RTL8019AS是一款即插即用的全双工NE2000兼容网络适配器，收发速率可达到10Mbps，它与我们选择的ARM7 S3C44B0X处理器兼容，但它不支持PCI接口；而RTL8029AS除了具有RTL8019AS的基本特征外，它的突出特点就是支持PCI接口，但它却与ARM7 S3C44B0X不兼容。考虑到EFW网卡要与主机PCI接口相连，我们必须选择2个RTL8019AS芯片网络接口芯片和1个RTL8029AS网络接口芯片作为EFW网卡的网络适配器，其中1个RTL8029AS负责与主机PCI接口通信，一个RTL8019AS用来接收从网络中传来的数据，另一个RTL8019AS用来接收从主机经过RTL8029AS传来的数据。

3.3 嵌入式防火墙软件方案

为了减少软件开发的复杂程度，也为了充分利用32位嵌入式处理器的强大功能，在嵌入式系统开发中通常使用嵌入式操作系统。目前常用的有uC/OS-II和 uCLinux两种嵌入式操作系统。uC/OS-II具有优秀的实时性能，而uCLinux则在网络和文件系统方面具有优势。

3.3.1 uC/OS-II 和 uCLinux

uC/OS-II和uCLinux操作系统，是当前得到广泛应用的两种免费且公开源码的嵌入式操作系统。uC/OS-II适合小型控制系统，具有执行效率高、占用空间小、实时性能优良和可扩展性强等特点，最小内核可编译至2k。uCLinux则继承了标准Linux的优良特性，并针对嵌入式处理器的特点进行了专门设计，具有内嵌网络协议、支持多种文件系统，开发者可利用标准Linux经验知识等优点。其编译后目标文件可控制在几百k量级。

uC/OS-II是一种免费公开源代码、结构小巧、具有可剥夺实时内核的实时操作系统。其内核提供了进程调度、时钟管理、内存管理和进程间的通信与同步等基本功能，而没有包括I/O管理、文件系统、网络等额外模块。

uCLinux是一种优秀的嵌入式Linux版本。uCLinux是micro-control-Linux的缩写。同标准Linux相比，它集成了标准Linux操作系统的稳定性、强大网络功能和出色的文件系统等主要优点。但是由于没有MMU(内存管理单元)，其多任务的

实现需要一定技巧。

嵌入式操作系统是嵌入式系统软硬件资源的控制中心，它以尽量合理的有效方法组织多个用户共享嵌入式系统的各种资源。所谓合理有效的方法，指的就是操作系统如何协调并充分利用硬件资源来实现多任务。嵌入式操作系统还有一个特点就是针对不同的平台，系统不是直接可用的，一般需要经过针对专门平台的移植操作系统才能正常工作。

EFW网卡在接收和发送数据的过程中，要对两个网卡芯片进行操作，因此对系统的实时性要求较高。考虑到uCLinux的实时性能的不足，uC/OS-II在实时性能方面有上佳的表现，所以选择在S3C44B0X上移植uC/OS-II操作系统，并在uC/OS-II操作系统环境下编写网卡驱动程序和包过滤程序。

3.3.2 uC/OS-II的进程调度

任务调度主要是协调任务对计算机系统内资源(如内存、I/O设备、CPU)的争夺使用。进程调度又称为CPU调度，其根本任务是按照某种原则为处于就绪状态的进程分配CPU。由于嵌入式系统中内存和I/O设备一般都和CPU同时归属于某进程，所以任务调度和进程调度概念相近，很多场合不加区分，下文中提到的任务其实就是进程的概念。

进程调度可分为“剥夺型调度”和“非剥夺型调度”两种基本方式。所谓“非剥夺型调度”是指一旦某个进程被调度执行，则该进程一直执行下去直至该进程结束，或由于某种原因自行放弃CPU进入等待状态，才将CPU重新分配给其他进程。所谓“剥夺型调度”是指一旦就绪状态中出现优先权更高的进程，或者运行的进程已用满了规定的时间片时，便立即剥夺当前进程的运行(将其放回就绪状态)，把CPU分配给其他进程。

作为实时操作系统，uC/OS-II是采用的可剥夺型实时多任务内核。可剥夺型的实时内核在任何时候都运行就绪了的最高优先级的任务。uC/OS-II中最多可以支持64个任务，分别对应优先级0~63，其中0为最高优先级。调度工作的内容可以分为两部分：最高优先级任务的寻找和任务切换。其最高优先级任务的寻找是通过建立就绪任务表来实现的。uC/OS-II中的每一个任务都有独立的堆栈空间，并有一个称为任务控制块TCB(task control block)数据结构，其中第一个成员变量就是保存的任务堆栈指针。任务调度模块首先用变量OSTCBhighrdy记录当前最高级就绪任务的TCB地址，然后调用os_task_sw()函数来进行任务切换。

综上所述，uC/OS-II内核是针对实时系统的要求设计实现的，相对简单，可

以满足较高的实时性要求，EFW网卡系统就移植使用了uC/OS-II实时操作系统进行进程调度。

3.4 开发环境

基于PC环境的系统开发，操作系统提供标准的API调用，不用考虑资源使用情况。而进行嵌入式系统开发时，需要针对不同的应用，设计不同的软硬件环境。功能强大、性能优良的开发工具对于加快开发进度必不可少的。下面介绍嵌入式防火墙开发中用到的软硬件开发环境。

3.4.1 ADS集成开发环境

ADS集成开发环境是ARM公司推出的ARM核微控制器集成开发工具，英文全称是ARM Developer Suite，成熟版本为1.2[38]。ADS1.2支持ARM10之前的所有ARM系列微控制器，支持软件调试及JTAG硬件仿真调试，支持汇编、C、C++源程序，具有编译效率高、系统库功能强等特点，可以在Windows98、Windows2000、WindowsXP以及RedHat Linux上运行。

ADS 1.2是由代码生成工具、CodeWarrior IDE集成开发环境、AXD仿真调试器、指令模拟器、ARM开发包和ARM应用库六个部分组成。一般用户直接使用CodeWarrior IDE集成开发环境和AXD调试器。

鉴于ADS以上的优良特性，EFW网卡中的核心软件都是在ADS开发环境下进行编译、仿真和调试的。

3.4.2 Protel99 SE电路设计制版系统

Protel99 SE是基于Windows的纯32位电路设计制版系统[34]。它提供一个集成的设计环境，包括原理图设计和PCB(Printed Circuit Board，印刷电路板)布线工具，集成的设计文档管理，支持通过网络进行工作组协同设计功能。课题中的所有的原理图以及PCB板图都是用Protel99 SE设计。

3.5 小结

本章首先介绍了通用嵌入式系统的定义及组成结构，而后给出了一种典型的嵌入式防火墙结构图，接着详细论述了嵌入式防火墙网卡的硬件方案和软件方案，并根据硬件方案进行了嵌入式防火墙网卡所需主要硬件的选型，最后介绍了具体的软件设计方案以及软件开发环境。

第四章 嵌入式防火墙硬件设计及实现

嵌入式防火墙硬件采用模块化设计，分成四个模块：存储模块、8019接口模块、8029接口模块和JTAG调试接口模块。下面就从功能和结构两个方面来介绍EFW网卡的硬件设计。

4.1 嵌入式防火墙的功能模块

嵌入式防火墙的各个功能模块组成如图4.1所示，各模块的功能如下：

存储模块主要包括一个闪速存储器(Flash Memory)和一个SDRAM，FLASH是用来存储程序，SDRAM提供程序运行的内存空间。

8019接口电路主要包括两个RTL8019AS网络芯片，这部分电路的功能是通过一个RTL8019AS从网络接收数据包，另一个RTL8019AS将ARM处理过的数据包传输给RTL8029AS芯片。

8029接口电路主要包括一个RTL8029AS网络芯片和一个EEPROM芯片，其电路的功能和普通网卡的功能相同，主要负责主机中到嵌入式防火墙之间数据包的转接。

JTAG调试接口电路包括一个调试接口，便于烧写FALSH芯片和调试EFW网卡。

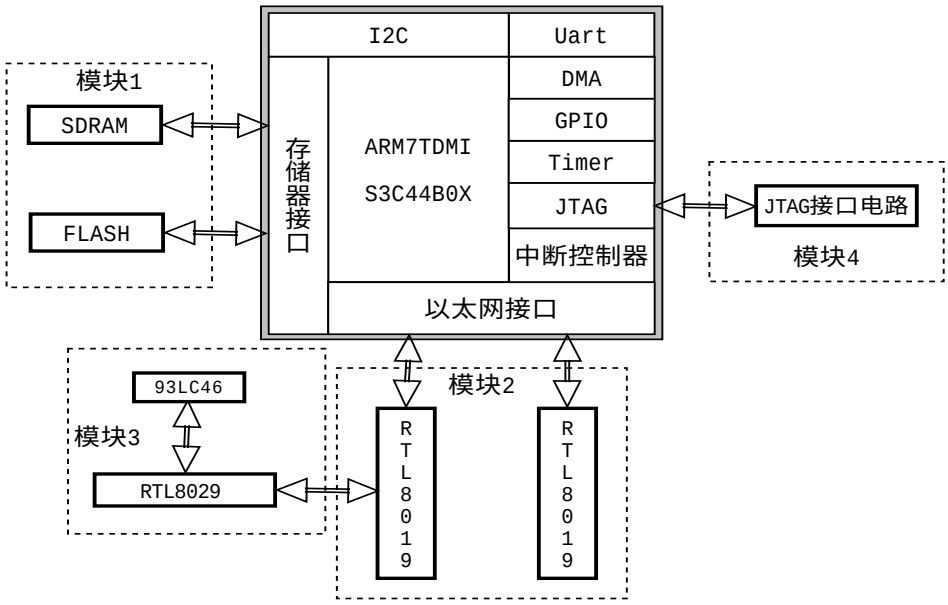


图 4.1 EFW 网卡各功能模块

4.2 S3C44B0X处理器的结构与特点

EFW网卡的硬件设计中,采用的是S3C44B0X处理器，它使用的是16/32位RISC结构的ARM7TDMI内核，工作频率为66MHz。为了降低总成本和减少外围器件，芯片中还集成了下列部件：8KB Cache、外部存储器控制器、LCD控制器、4个DMA通道、2通道UART、1个多主IC总线控制器、以及5通道PWM定时器和一个内部定时器、71个通用I/0口、8个外部中断源、实时时钟、8通道10位A/D。具体配置如下：

两路带DMA和中断的UART口，最高波特率为115200bps。支持IrDAl.0，可用于红外通信，当传送/接收时支持双向握手，每个通道有2个32位FIFO。

S3C44B0X有6个16位定时器，每个可以以中断模式或DMA模式来工作。定时器0, 1, 2, 3, 4有PWM(Pulse-Width Modulation 脉宽调制)功能，可以控制蜂鸣器输出不同声音作为系统提示信息。定时器5是一个内部定时器没有输出脚。定时器0有一个DEAD ZONE 产生器，定时器0、 1共享一个8位预定标器，定时器2、3共享另一个8位预定标器，定时器4、5也共享一个8位预定标器。定时器0、1、2、3有一个时钟除法器(1/2、1/4、1/8、 1/16、1/32)。定时器4、5有一个时钟除法器(1/2、1/4、1/8、1/16)和一个输入时钟TCLK/EXTCLK。每个定时器从时钟除法器接收时钟输入，而时钟除法器的输入时钟为相应的预定标器。每个定时器有一个计数缓冲寄存器保持定时器的计数初始值，当定时器允许和它的向下计数器计到0时，该初始值重新加载到定时器的向下计数器，并产生中断请求。每个定时器还有一个比较缓冲寄存器，初始值加载到定时器的比较寄存器，当比较缓冲寄存器的值与定时器的向下计数器值相同时，定时器控制逻辑改变TOUT的输出电平。

S3C44B0X有71个复合功能的I/0口引脚，分成了7个端口：端口A (10位I/0端口)；端口B (11位I/0口)；端口C (16位1/0口)；端口D和G (8位I/0口)；端口E和F (9位I/0口)。在主程序开始前，必须定义每个I/0管脚的功能。在特殊功能不用时，作为I/0脚使用。图4.2 为 S3C44B0X处理器的体系结构框图:

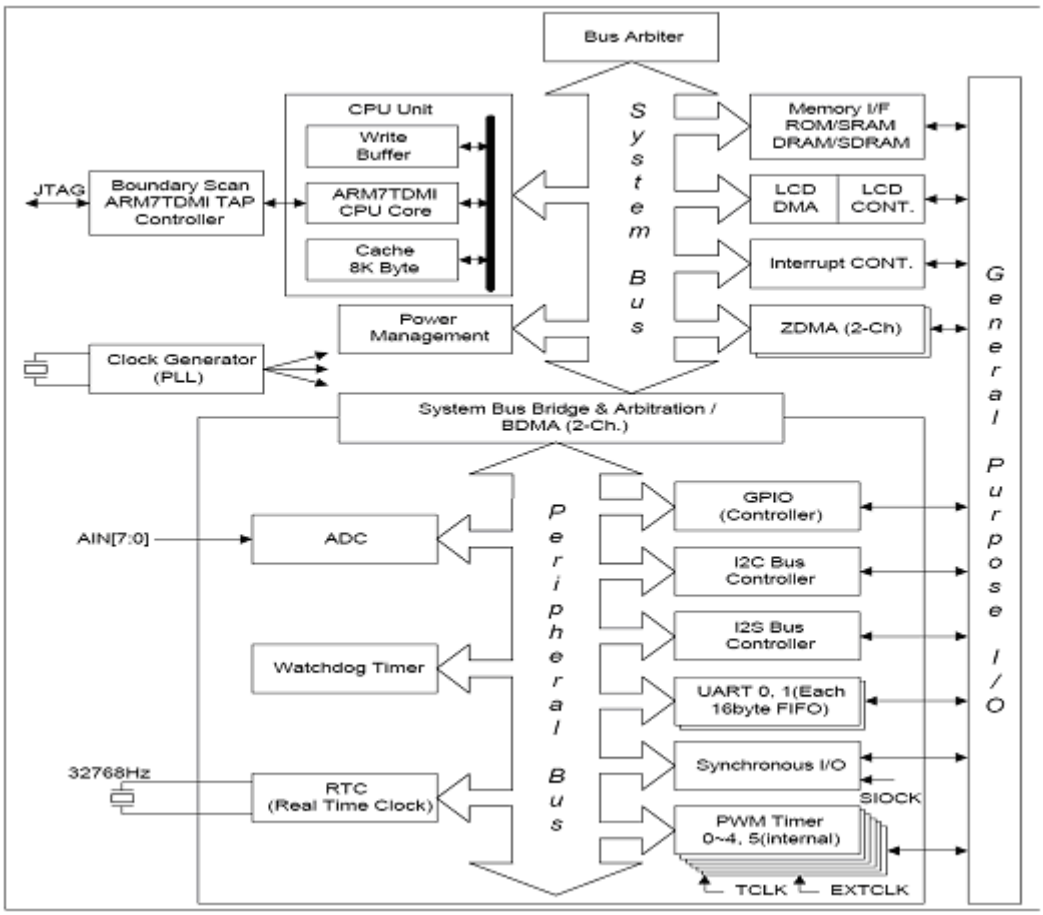


图 4.2 S3C44B0X 体系结构框图

4.3 存储模块设计

S3C44B0X的存储系统支持大小端选择(通过外部引脚)，有8个MEMORY BANKS，用于管理外部储存器，最大可以访问的存储容量达到256MB。每个BANK支持8/16/32位的数据格式，可寻址32MB(除BANK0，其后4MB保留给内部寄存器使用)。其中BANK0专门用于系统启动，它必须是线性寻址且有记忆功能；BANK6、BANK7专门用于DRAM/SDRAM，各个BANK都有其独立的片选信号，分别是nGCS0到nGCS7。

图4.3为ARM7 S3C44B0X处理器的地址空间图：

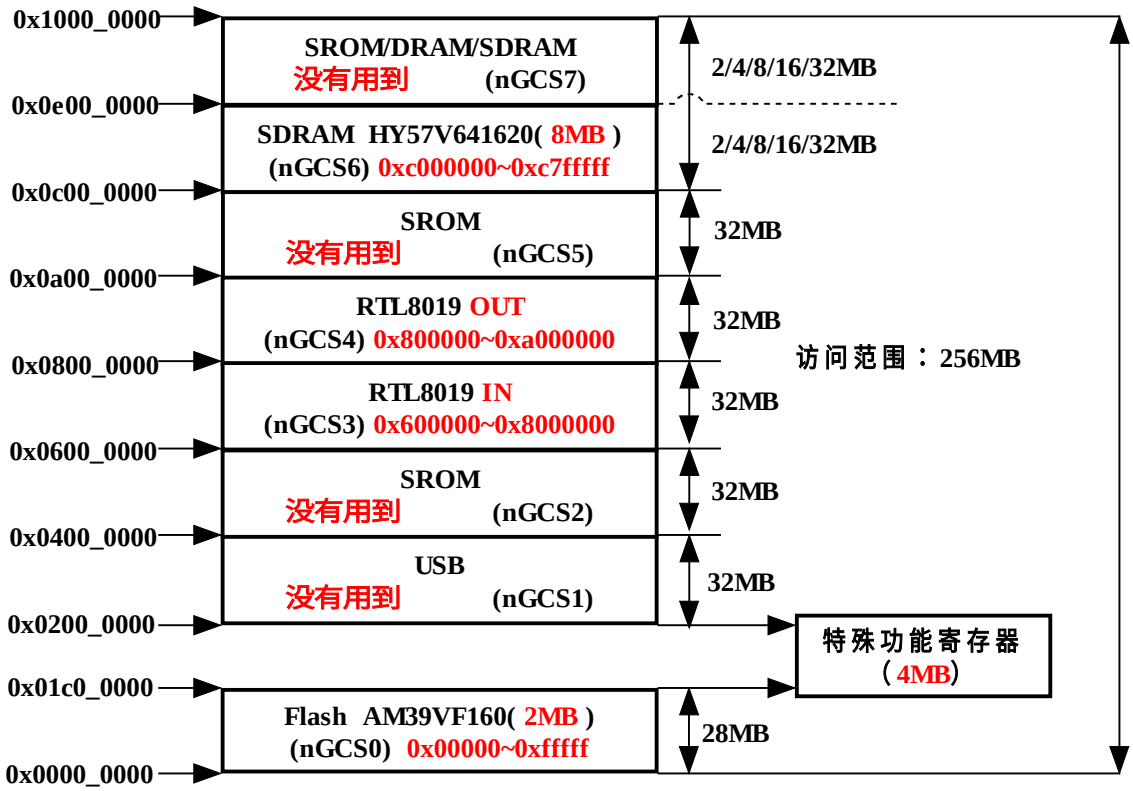


图 4.3 ARM 处理器的地址空间

从图4.3可以看出嵌入式防火墙系统中NOR Flash(SST39VF160)接在S3C44B0X芯片的nGCS0片选引脚，地址空间为0x00000——0xfffff，宽度为16M bits(2M字节)。SDRAM(HY57V641620)接在S3C44B0X芯片的nGCS6片选引脚，容量大小为64M bits(8M字节)，地址空间为：0xc000000——0xc7fffff。S3C44B0X芯片的nGCS3和nGCS4片选引脚分别接在RTL8019 IN芯片和RTL8019 OUT芯片的片选端。S3C44B0X芯片的nGCS1、nGCS2、nGCS5、nGCS7这几个片选引脚悬空未用。

4.3.1 Flash接口电路设计

闪速存储器(Flash Memory)是一类非易失性存储器(NVM，Non-VolatileMemory)即在供电电源关闭后仍能保持片内信息；而DRAM，SRAM是易失性存储器，当供电电源关闭时片内信息随即丢失。Flash Memory与其它类非易失性存储器的特点:与EPROM相比较，闪速存储器明显的优势是——当对它进行电擦除和可重复编程时，不需要特殊的高电压(某些第一代闪速存储器也要求高电压来完成擦除或编程操作)；与EEPROM相比较，闪速存储器具有成本低、

密度大的特点。闪速存储器独特的性能使其广泛地运用在包括嵌入式系统在内的各个领域。

FLASH分为NOR型和NAND型两种。NOR型FLASH的特点是：可靠性高、随机读取速度快，在擦除和编程操作较少而直接执行代码的场合，尤其是纯代码存储的应用中广泛使用，如PC的BIOS固件、移动电话、硬盘驱动器的控制存储器等。 而NAND型FLASH能提供极高的单元密度，可以达到高存储密度，并且写入和擦除的速度也很快，适合于纯数据存储和文件存储，如SM卡，CF卡等。

EFW网卡中使用的HY29LV160芯片是NOR型FLASH存储器，它的基本特性：单片存储容量为16M位(2M字节)，工作电压为2.7V～3.6V，采用48脚TSOP封装或48脚FBGA封装，16位数据宽度，可以以8位(字节模式)或16位(字模式)数据宽度的方式工作。

HY29LV160仅需单3V电压即可完成在系统的编程与擦除操作，通过对其内部的命令寄存器写入标准的命令序列，可对Flash进行编程(烧写)、整片擦除、按扇区擦除以及其他操作。

由于ARM微处理器的体系结构支持8位/16位/32位的存储器系统，所以对应的可以构建8位的Flash存储器系统、16位的Flash存储器系统或32位的Flash存储器系统。32位的存储器系统具有较高的性能，而16位的存储器系统则在成本及功耗方面占有优势，8位的存储器系统现在已经很少使用。EFW网卡中使用的是16位Flash存储器系统，见图4－4。

从EFW网卡的需求和成本考虑，笔者选用一片16位的Flash存储器芯片HY29LV160构建16位的Flash存储器系统，其存储容量为2MB，用于存放程序代码。系统上电或复位后要从Flash存储器中获取指令并开始执行，因此，应将存有程序代码的Flash存储器配置到ROM/SRAM/FLASH Bank0，即将S3C44B0X的nGCS0接至HY29LV160的CE＃端。

- HY29LV160的RESET＃端接系统复位信号；
- OE＃端接S3C44B0X的nOE；
- WE＃端S3C44B0X的nWE；
- 地址总线[A19～A0]与S3C44B0X的地址总线[ADDR20～ADDR1]相连。
- 16位数据总线[DQ15～DQ0]与S3C44B0X的低16位数据总线[DATA15～DATA0]相连。

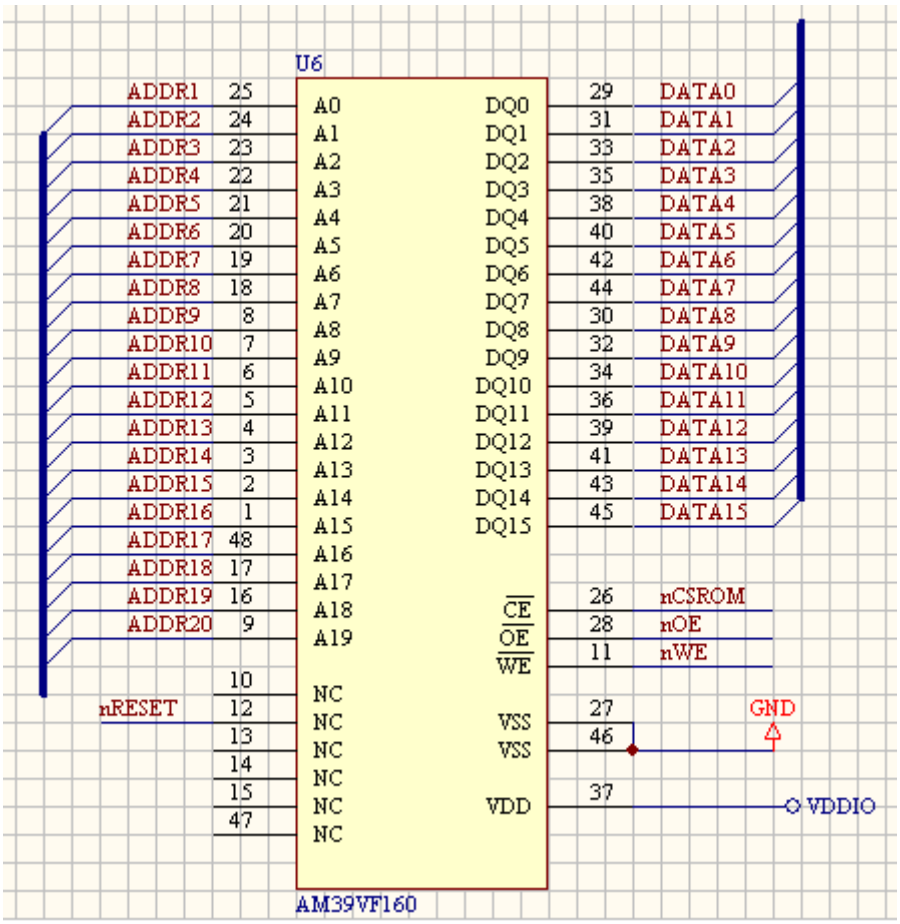


图 4.4 Flash 存储器的电路图

4.3.2 SDRAM接口电路设计

与Flash存储器相比较，SDRAM不具有掉电保持数据的特性，但其存取速度大大高于Flash存储器。SDRAM在系统中主要用作程序的运行空间，数据及堆栈区。当系统启动时，CPU首先从Flash中的复位地址0x0处读取启动代码，在完成系统的初始化后，程序代码调入SDRAM中运行，以提高系统的运行速度；同时，操作系统、用户堆栈及应用程序也都被加载到SDRAM中。

EFW网卡使用的SDRAM是HY公司的HY57V641620芯片，它的容量为8Mbytes，是由4个大小为1048576X16bit的Bank组成。这4个Bank由BAO和BA1决定。行地址为AO~A11，寻址空间为4K，列地址为A0~A7，寻址空间为256bytes。这样通过A0~A11，BAO/BAl信号就能够寻址到该SDRAM具体的地址空间。工作电压为3.3V，常见封装为54脚TSOP，兼容LVTTL接口，支持自动刷新(Auto-Refresh)和自刷新(Self-Refresh)，16位数据宽度。

图4－5为16位SDRAM存储器系统的实际应用电路图。与Flash存储器相比，

SDRAM的控制信号较多，其连接电路也要相对复杂。可将HY57V641620配置到DRAM/SDRAM Bank6～DRAM/SDRAM Bank7的任一位置，本文把它配置到DRAM/SDRAM Bank6，即将S3C44B0X的nGCS6接至HY57V641620的nSCS端。

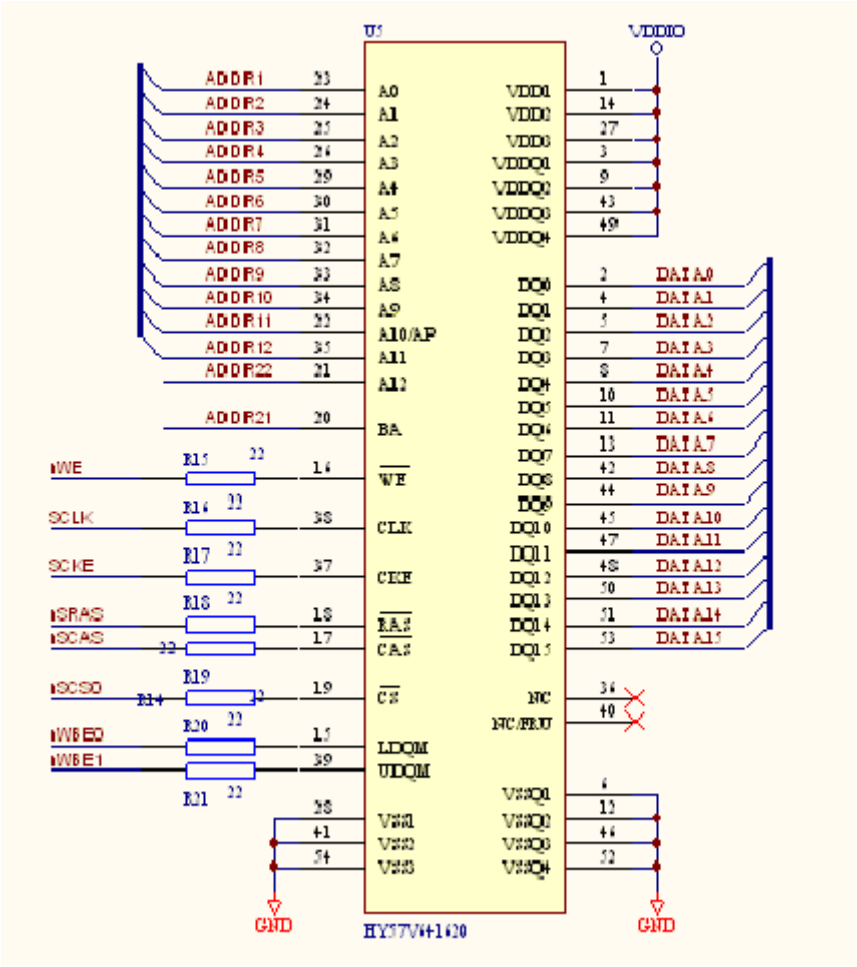


图 4.5 SDRAM 存储器的电路图

HY57V641620的SCLK端接S3C44B0X的SCLK端； HY57V641620的SCLE端接S3C44B0X的SCLE端； HY57V641620的nSRAS、nSCAS、nWE端分别接S3C44B0X的nSRAS端、nSCAS端、nDWE端；HY57V641620的A11～A0接S3C44B0X的地址总线ADDR<12>～ADDR<1>；HY57V641620的BA1、BA0接S3C44B0X的地址总线ADDR<21>、ADDR<22>。

4.4 以太网模块设计

以太网模块是本系统的设计重点，它包括EFW网卡的模块2和模块3两个部分(见图4－1所示)。模块2主要是RTL8019AS接口电路，它的功能是在S3C44B0X

处理器的控制下，接收网络上到来的数据包和主机通过RTL8029AS传出来的数据包。模块3主要是RTL8029AS接口电路，它的功能是在主机操作系统控制下接收从RTL8019AS传输来的数据包、发送从主机到网络的数据包。

在嵌入式防火墙系统中两个RTL8019AS分别接到S3C44B0X的nGCS3和nGCS4，映射的地址空间分别为Bank3(0x06000000 — — 0x08000000) 和Bank4(0x08000000——0x0a000000)。

4.4.1 RTL8019AS接口电路设计

1、RTL8019AS主要性能

RTL8019AS网络控制芯片是台湾RealTek公司生产的一款性能优良、价格低廉、使用广泛的以太网接口芯片，其主要性能如下[10]：

- 符合Ethernet II与IEEE802.3(10Base5、10Base2、10BaseT)标准；
- 全双工，收发可同时达到10Mbps的速率；
- 内置16KB的SRAM，用于收发缓冲，降低对处理器的速度要求；
- 支持8/16位数据总线，8个中断申请线以及16个I/O基地址选择；
- 可连接双绞线和同轴电缆，并可自动识别所连接的介质；
- 允许4个诊断LED引脚可编程输出；
- 采用CMOS工艺，功耗低。单一电源5V供电。

2、RTL8019AS内部逻辑功能

RTL8019AS内部包含远程DMA接口、本地DMA接口、介质访问控制逻辑、数据编码解码逻辑和其他端口。本地DMA完成RTL8019AS与网线之间的数据交换，远程DMA负责ARM与RTL8019AS之间的数据交换。

3、RTL8019AS寄存器

RTL8019AS共有32个输入输出地址，对应的地址偏移量为00h～1Fh：

- 00h～0Fh的16个地址为寄存器地址；
- 10h～17h的8个地址为数据读写端口地址，它们都是一样的，每个都可以用来做数据读写端口，只要用其中的一个就可以了；
- 18h～1Fh的8个地址为复位端口，这8个地址的功能也都一样，只用其中的一个就可以了。但需要注意，实际上只有18h、1Ah、1Ch、1Eh这几个复位端口是有效的，其他不要使用，因为有些兼容卡不支持19h、1Bh、1Dh等奇数地址的复位。

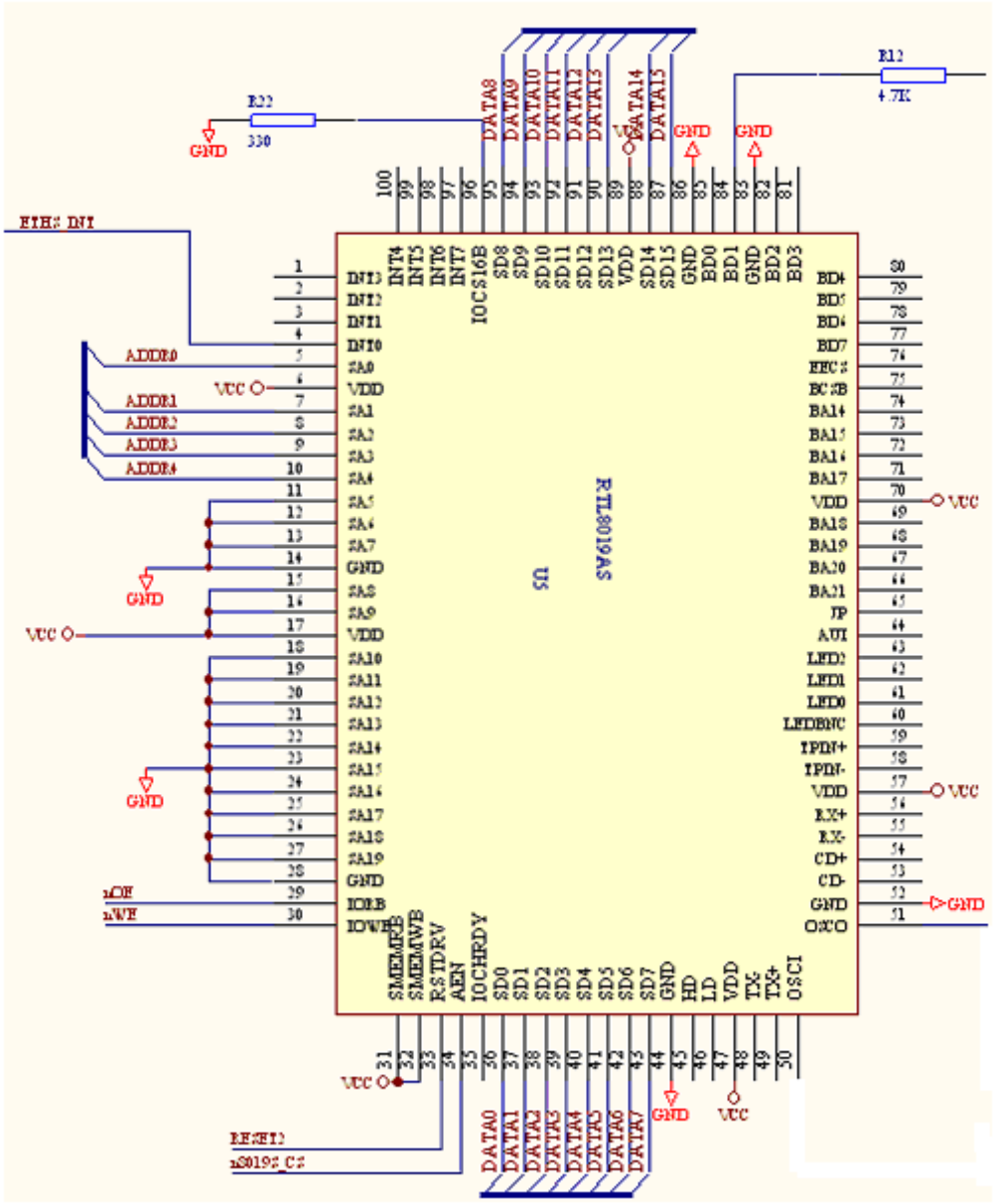


图 4.6 RTL8019AS 的电路图

4.4.2 RTL8029AS接口电路设计

RTL8029AS以太网控制器支持PCI接口，基本功能与RTL8019AS相似，遵循IEEE802.3协议；内置16KB的SRAM，用于收发缓冲；全双工，收发同时达到10Mbps；支持10Base5、10Base2、10BaseT，并能自动检测所连接的介质。它支持32位数据总线，而RTL8019AS只能支持8、16位数据总线。

这部分电路的功能实际上是普通网卡的全部功能。RTL8029As的功能与RTL8019AS基本相似。

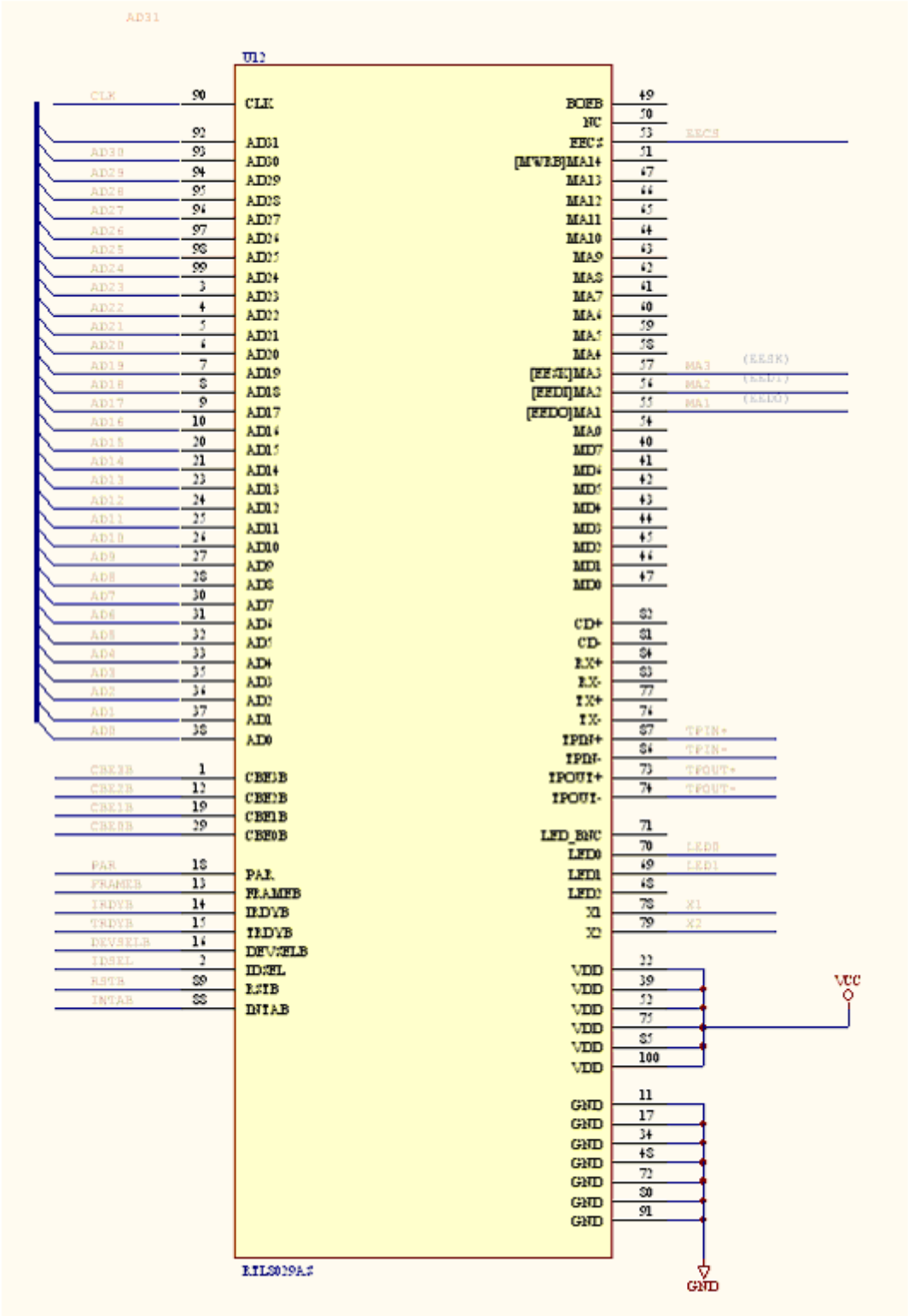


图 4.7 RTL8029AS 的电路图

4.5 EFW网卡调试

系统硬件调试是确保系统正确工作的重要步骤。下面就介绍在EFW网卡调试时所遇到的问题以及解决方法。进行硬件调试，首先要仔细检查电路板有无短路、断路等明显缺陷，确保电路无误后方可焊接元器件。因为EFW网卡采用的是模块化的设计，在焊接时以各个模块电路为单位，一个个焊接调试，以便遇到困难时缩小故障范围。

调试工具需要示波器、万用表等，还要配合调试软件ADS(AXD)以及ARM JTAG仿真器，ADS软件的使用方法参考《ARM与嵌入式系统基础教程》[44]。

4.5.1 电源、晶振的调试

EFW网卡需要的9V电源通过PCI插槽供电，在调试时也可以通过电源接头给EFW网卡供电。电源芯片LM1117_3.3和LM1117_2.5输入9V电源为系统提供3.3V(VCC)和2.5V(CVCC)两路电压。

S3C44B0X芯片使用了10MHz的有源晶振，用示波器观察可以看到10MHz的输出波型；RTL8019AS和RTL8029AS使用20MHz的晶体，用示波器观察可以看到20MHz的输出波型。

电源、晶振是系统正常工作的基础，首先应确保它们正常工作。 在笔者调试过程中出现过晶振不起振的现象，在检查电路没有问题的情况下，将不起振的晶振换掉后，重新用示波器测试，晶振工作正常。

4.5.2 S3C44B0X及JTAG接口电路调试

在保证电源、晶振和复位电路正常工作的前提下，可以通过JTAG接口调试S3C44B0X。确保JTAG接口的TMS、TCK、TDI、TDO信号引脚与S3C44B0X的对应引脚连接正确。其次应该保证S3C44B0X的输入引脚nWAIT上拉、nExtMREQ下拉，S3C44B0X才能正常工作。调试器可以选择专业的功能强大的仿真器，也可以选择简易的JTAG仿真器。下面的说明都以使用简易的JTAG仿真器为例。

系统上电后，可以通过示波器察看S3C44B0X的对应引脚输出波形，判断是否正常工作。在使能片内PLL电路的情况下，SDCLK/MCLKO引脚应有输出的波形。

4.5.3 SDRAM和Flash接口电路调试

1、SDRAM电路调试

在系统用到的SDRAM存储器接口电路相对于flash存储器接口，需要的控制信号较多，但是由于S3C44B0X片内集成了SDRAM控制器，几乎不需要用户干预就能正常工作，反而调试相对简单。在S3C44B0X正常工作的前提下，只要连线正确，SDRAM就应该能正常工作。反之，Flash存储器，接口较为简单，但是flash存储器的编程、擦除操作均需要用户程序控制，需要使用SDRAM，所以调试时应该先调试SDRAM。

SDRAM存储器连接在SDRAM Bank0，Flash存储器连接在ROM Bank0；在系统复位之后只有ROM Bank0被映射到地址空间0x0000,0000~0x0200,0000的位置，特殊功能寄存器的基地址被映射到0x03FF,0000，片内8K一体化SRAM的起始地址被映射到0x03FE,000,他们是可以访问的，其他存储器均未被映射，是不可访问的。

因此，在调试SDRAM存储器之前，应该首先配置相关的特殊功能寄存器，使系统中的SDRAM能被访问。

2、Flash电路调试

Flash存储器的调试主要包括Flash存储器的编程(烧写)和擦除。除了读操作，其他的编程/擦除等工作，用户需要对flash存储器发出相应的命令序列，Flash通过内部算法即可完成对芯片的操作，由于不同厂商的Flash操作命令可能有细微的差别，所以flash接口电路相对难调试。笔者采用flash编程工具软件flashpgm，配合Wiggler JTAG接口可以很方便的进行Flash的在线擦/写。

Flashpgm是通过调试接口(BDM or JTAG)进行flash在线编程的工具软件，通过GUI界面可以方便的对多种平台的flash芯片进行编程、擦除、填充、查空等操作。支持Motorola(.s19)、Intel(.hex)和ELF(.elf) 等文件格式。

4.5.4 网络接口调试

EFW网卡的网络接口芯片包括3个，其中两个RTL8019AS的调试方法相同。所以在调试时分成两步：

1、RTL8029AS电路调试

这部分电路完成普通网卡的功能，所以在调试的时候，只要在隔离变压器的电路后面引出1，2，3，6四条线，与RJ45连接，然后连接网线测试网络是否

连通。这部分的电路比较成熟，如果电路连接正确，一般都没有问题。

2、RTL8019AS电路调试

RTL8019AS和S3C44B0X均具有MII接口，对应的引脚以及功能定义明确，只要正确连接，一般都能正常工作。当EFW网卡接上网线后，RTL8019AS的发送时钟引脚RX＋、RX－，接收时钟引脚TX＋、TX－均应有波形输出，对应的LED也能指示工作状态。

4.6 小结

本节首先给出了EFW网卡硬件各功能模块的结构图，然后以S3C44B0X处理器为核心给出了存储器模块、以太网接口模块以及调试接口模块的具体设计过程，在兼顾效率与成本的同时给出了芯片的选型原则，并附上相应的电路图。此外，笔者还描述了EFW网卡的硬件调试过程。

第五章 嵌入式防火墙核心软件的实现

5.1 BootLoader程序

BootLoader是在操行系统内核或用户应用程序运行之前运行的一段小程序。大家都知道PC机中的引导加载程序是由主板BIOS和位于硬盘MBR中的引导程序一起组成。主板中的BIOS程序在完成硬件检测和资源分配后，将硬盘MBR中的引导程序读到系统的RAM中，然后将控制权交给引导程序，引导程序将内核映像从硬盘上读到RAM中，然后跳转到内核的入口点去启动操作系统。在嵌入式系统中，通常并没有像BIOS那样的固件程序，因此整个系统的加载启动任务就完全由BootLoader来完成。

BootLoader可以初始化硬件设备、建立内存空间的映射图，从而将系统的软硬件环境带到一个合适的状态，以便为最终调用操作系统内核或用户应用程序准备好正确的环境。通常，BootLoader是依赖于硬件而实现的，特别是在嵌入式领域，为嵌入式系统建立一个通用的BootLoader是很困难的。

5.1.1 BootLoader操作模式

BootLoader的操作模式通常有两种：启动加载模式和下载模式。

启动加载(Bootloading)模式：也称为“ 自主 ”(Autonomous)模式。BootLoader从目标机上的固态存储设备上将操作系统加载到RAM中运行，整个过程没有用户的介入。

下载(Downloading)模式：在这种模式下，目标机上的BootLoader将通过串口连接或网络连接等通信手段从主机下载文件，如下载应用程序、数据文件、内核映像等。从主机下载的文件通常首先被BootLoader保存到目标机的RAM中，然后再被BootLoader写到目标机上的固态存储设备上。这种模式通常在系统更新时使用。

本设计需要同时使用两种模式：通常使用启动加载模式；在策略更新时，使用下载模式。

5.1.2 BootLoader与主机之间进行文件传输所用的通信设备及协议

BootLoader与主机之间通信可以采用串口和网口，本课题所采用的是网口通信，利用TFTP协议进行。TFTP是一个传输文件的简单协议，它其于UDP协议而

实现。此协议设计的时候是进行小文件传输的，因此它不具备通常的FTP的许多功能，它只能从文件服务器上获得或写入文件，不能列出目录，不进行认证，它传输8位数据。而在EFW网卡与主机之间进行传输的通常都是很小的策略和审计文件，所以TFTP完成符合要求。而具体的操作是：当主机的策略或审计文件要分发到EFW网卡时，在主机端开启TFTP服务，EFW网卡通过TFTP客户端从主机端下载相应的文件。

5.1.3 BootLoader的主要任务及启动过程

系统加电或复位后，S3C44B0X在复位时从地址0x00000000取它的第一条指令，而EFW上面的FLASH被安排这个起始地址上，因此在系统加电后，ARM处理器S3C44B0X将首先执行BootLoader程序。

从操作系统的角度看，BootLoader的总目标就是正确地调用内核来执行，由于BootLoader的实现依赖于CPU的体系结构，比如设备初始化代码等都要用汇编语言来实现，以达到短小精悍的目的，而一些复杂的功能，通常用C语言来实现，使代码具有更好的可读性和可移植性。从而BootLoader的启动可以分为汇编语言和C语言两个阶段，两个阶段的任务分别如下：

1．BootLoader的汇编语言阶段通常完成如下任务：

(1)硬件设备初始化，RAM初始化，设置各个部件的时钟和片选；

(2)为加载BootLoader的C程序准备RAM空间；

(3)拷贝BootLoader的C程序到RAM空间中；

(4)设置好程序运行的堆栈；

(5)跳转到C程序的入口点。

图5.1是BootLoader汇编语言阶段的启动过程：

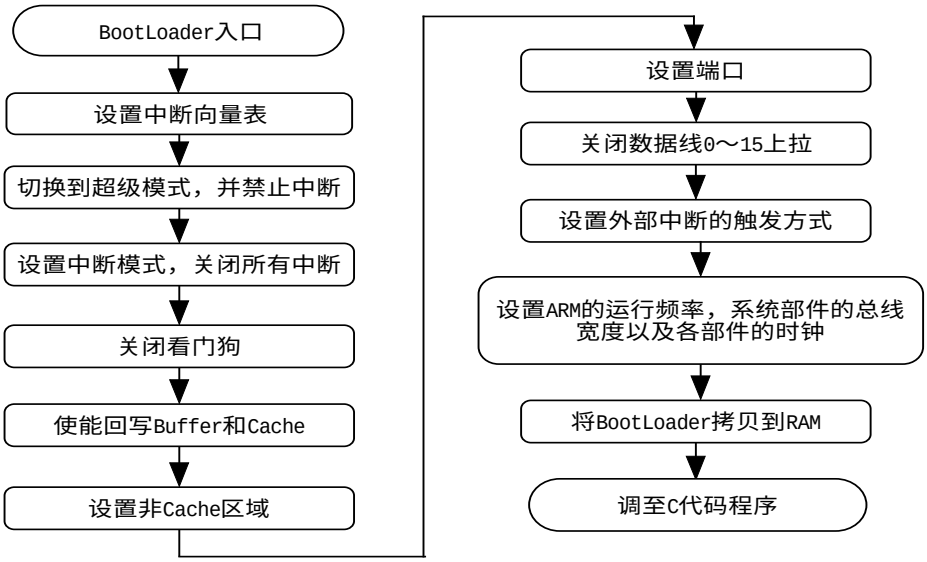


图 5.1 BootLoader 程序第一阶段运行图

2．BootLoader的C语言阶段完成的任务是：

(1)初始化本阶段要使用到的硬件设备；

(2)检测系统的内存映射；

(3)调用应用程序或者启动操作系统内核。

图5.2是BootLoader C语言阶段的启动过程：

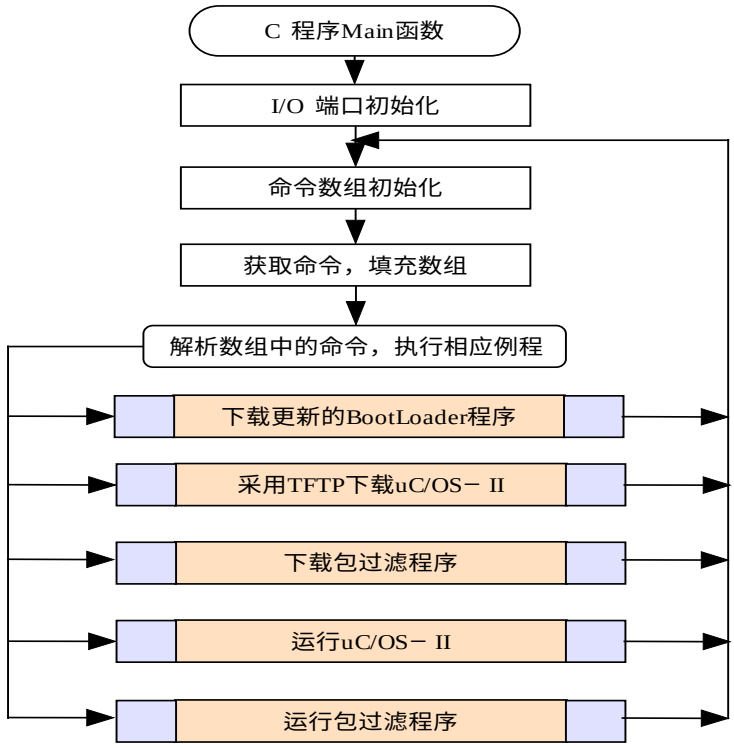


图 5.2 BootLoader 程序第二阶段运行图

5.2 uC/OS-II移植

5.2.1 uC/OS-II概述

uC/OS-II是一个完整的，可移植、固化、裁剪的占先式实时多任务内核。它是用ANSI的C语言编写的，包含一小部分汇编语言代码，使之可供不同架构的微处理器使用。至今，从8位到64位，uC/OS-II已在超过40种不同架构的微处理器上运行。

要使uC/OS-II可以正常工作，处理器必须满足如下要求：

1．处理器的C编译器能产生可重入代码。

可重入的代码指的是一段代码(如一个函数)可以被多个任务同时调用，而不必担心会破坏数据。也就是说，可重入函数在任何时候都可以被中断执行，过一段时间以后又可以继续运行，而不会因为函数中断的时候被其他的任务重新调用，影响函数中的数据。

代码的可重入性是保证完成多任务的基础，除了在C程序中使用局部变量以外，还需要C编译器的支持。本课题使用的是ARM ADS的集成开发环境，可以生成可重入的代码。

2．在程序中可以打开或者关闭中断。

在uC/OS-II中，可以通过OS_ENTER_CIRTICAL( )或者OS_EXIT_CIRTICAL( )宏来控制系统关闭或者打开中断。这需要处理器的支持，在ARM7TDMI的处理器上，可以设置相应的寄存器来关闭或者打开系统的所有中断。

3．处理器支持中断，并且能产生定时中断(通常在10Hz～1000Hz之间)。

uC/OS-II是通过处理器产生的定时器的中断来实现多任务之间的调度的。在ARM7TDMI的处理器上可以产生定时器中断。

4．处理器支持能够容纳一定量数据的硬件堆栈(可能是几千字节)。

5．处理器有将堆栈指针和其他CPU寄存器的内容读出、并存储到堆栈(或者内存)中去的指令。

uC/OS-II进行任务调度的时候，会把当前任务的CPU寄存器存放到此任务的堆栈中，然后，再从另一个任务的堆栈中恢复原来的工作寄存器，继续运行另一个任务。所以，寄存器的入栈和出栈是uC/OS-II多任务调度的基础。

5.2.2 uC/OS-II的移植

uC/OS-II的移植就是使它的实时内核能在其他的微处理器上运行，为了方便移植，大部分uC/OS-II的代码是用C语言来编写的。但是仍需要用C语言和汇编语言编写一些与处理器硬件相关的代码，这是因为uC/OS-II在读/写处理器寄存器只能通过汇编语言来实现。由于uC/OS-II在设计之初就充分考虑了可移植性，所以uC/OS-II的移植只需要修改与处理器相关的部分代码就可以了。下面给出uC/OS-II的硬件、软件的体系结构图。

图 5.3 uC/OS_II 硬件/软件体系结构图

在图中我们可以看到，uC/OS_II的移植其实就是修改与处理器相关的三个文件：OS_CPU.H,OS_CPU_A.S,OS_CPU_C.C。具体的移植不是本文的重点，在此不做详细的论述。

5.3 EFW网卡驱动程序

一个完整的以太网控制器驱动程序应包括硬件初始化、以太帧的发送和接收三个部分。EFW网卡的驱动程序包括对RTL8029AS和两个RTL8019AS网络接口芯片的驱动。

5.3.1 RTL8029AS的驱动程序

EFW网卡中8029电路和普通网卡的相同，是相对独立的一部分电路，它的驱动程序是安装在主机操作系统中，可以使用RealTek公司针对RTL8029AS、RTL8139AS等芯片提供的驱动程序进行工作。

5.3.2 RTL8019AS的驱动程序

EFW网卡上的两个RTL8019AS网卡芯片都担负着接收和发送数据包的任务，ARM处理器对于它们的驱动程序大致相同，不同的就是映射的内存空间不同，一个映射的地址空间为：0x600000~0x8000000，另一个为0x800000~0xa000000，下面笔者给出的是其中一个的驱动程序，另一个只要将基地址改成0x800000。

5.3.2.1 RTL8019AS内存缓冲区初始化设置说明

RTL8019AS内部有两块存储区，一块是用于存放收发数据包的RAM区，地址范围是：0x4000～0x7fff，共16KB；它是按页存储的，每256字节为1页，RAM采用16位地址方式，高8位称为页码，页码的范围从0x40到0x7f，共64页；通常将0x4000～0x4bff的12页3KB的空间作为发送缓冲区，0x4c00～0x7fff的52页13KB的空间作为接收缓冲区。而另一块是用于存放以太网物理地址的PROM，地址范围是0x0000～0x00ff，但只有前32字节可被访问，且通常不用。

RTL8019AS的内部寄存器分成4页，每页有16个寄存器，下面介绍常用的寄存器：

- CR(Command Register,命令寄存器)，用来设置RTL8019AS的寄存器页、控制远程DMA操作方式以及一些对RTL8019AS的操作命令；
- TCR(Transmit Configuration Register,传输配置寄存器),用来配置RTL8019AS的传输模式；
- DCR(Data Configuration Register,数据配置寄存器)，与TCR配合用来配置RTL8019AS的数据传输模式；

- RBCR0、RBCR1(Remote Byte Count Register,远程传输字节计数寄存器)，用来存储RTL8019AS远程DMA传输的字节数；
- RSAR0、RSAR1(Remote Start Address Register,远程DMA传输开始寄存器)，用来设置远程DMA的开始地址；
- RCR(Receive Configuration Register,接收配置寄存器)，用来设置RTL8019AS的接收模式；
- PSTART(Page Start Register,页起始寄存器)、PSTOP(Page Stop Register,页结束寄存器)，用来设置接收缓冲区的范围，通常设置PSTART＝0x4c，PSTOP＝0x80，这表示将使用0x4c00～0x7fff的RAM区来存储接收的数据包。这里PSTOP应该设成0x80而不是0x7f，因为PSTOP代表从该页开始的页不能作为接收缓冲区，PSTART的代表从这一页开始作为接收缓冲区；
- TPSR(Transmit Page Start Register，发送起始页寄存器)，用来设置发送缓冲区的起始页，通常将TBSR设置成发送缓冲区的起始地址0x40。一个以太帧的最大长度为1518字节，一页有256字节，那么一帧需要1518/256＝5.92页，即6页。RTL8019AS发送缓冲区的大小是12页，因此最多可以存放两个以太帧；
- BNRY(Boundary Register,边界寄存器)、CURR(Current Page Register,当前页寄存器)，用来读取RTL8019AS接收缓冲区的数据，BNRY是读指针，指向处理器下一次要读的页，CURR写指针，指向RTL8019AS接收的数据要写的页；
- ISR(Interrupt Status Register,中断状态寄存器)、IMR(Interrupt Mask Register,中断屏蔽寄存器)，用来设置中断方式，设置成0x00时，屏蔽所有的中断，设置成0xFF将允许中断。

5.3.2.2 RTL8019AS初始化过程描述

RTL8019AS的初始化主要完成两个RTL8019AS正常工作时有关参数的设置，如：收发缓冲区的设置、数据通信格式的设置、收发模式的设置等。具体的初始化过程如下：

1、复位RTL8019AS：RTL8019AS的RSTDRV引脚为高电平有效，且至少需要800ns的宽度。在系统上电时，ARM处理器向PTF4(PTF4与RSTDRV引脚硬件上相连)输出高电平，且让高电平持续10ms左右，然后施加一个低电平，等待2ms

左右，这样就完成了复位工作；

2、设置发送缓冲区的首地址：令TPSR＝0x40，当ARM处理器要向网络发送数据时，RTL8019AS先将该数据暂存储在页0x40开始的缓冲区中等待发送；

3、设置接收缓冲区环：令PSTART＝0x4c、PSTOP＝0x80，表示接收缓冲区环的地址范围是0x4c～0x80；

4、初始化接收缓冲区环读写指针：令BNRY＝0x4c表示ARM处理器下次要读的页为0x4c，令CURR＝0x4c，表示下一次当RTL8019AS从网络接收到的以太帧后，将它存放在0x4c开始的页；BNRY是由ARM来进行修改，而CURR是由RTL8019AS自身来修改；

5、设置RTL8019AS的收发模式：令RCR＝0xcc，表示仅接收与本地MAC地址匹配的数据包和广播包，不接收小于64字节或者错误的数据包；令TCR＝0xe0表示启用CRC自动生成和自动较验；令DCR＝0xc9表示使用FIFO缓存、普通模式、16数据DMA；令IMR＝0x00，表示允许所有中断；

6、设置网卡地址寄存器PAR0～PAR5；

7、设置工作方式：令CONFIG＝0x41，表示选择全双工工作方式；

8、清中断状态寄存器：令ISR＝0xff，表示清除中断状态寄存器所有位；

9、设置完成，启动RTL8019AS：令CR＝0x22，表示选择页0的寄存器，并启动RTL8019AS的网络功能。

下面就是RTL8019AS初始化的程序代码：

```
#define RPSTART    0x4c
#define RPSTOP     0x80
#define SPSTART    0x40
#define BaseAddr    0x6000000
static U8 SrcMacID[ETH_ALEN] = {0x00,0x80,0x48,0x12,0x34,0x56};
static U8 net_start;
static void InitRS8019( )
{
    net_start = 1;
    SetRegPage(0);
    outportb(BaseAddr, 0x21); /*使芯片处于停止模式,这时进行寄存器设置*/
    outportb(TPSR, SPSTART);        /*设置发送起始页寄存器为0x40 */
```

```
    outportb(Pstart, RPSTART);        /*设置页起始寄存器为0x4c */
    outportb(Pstop, RPSTOP);          /*设置页结束寄存器为0x80 */
    outportb(BNRY, RPSTART);      /*  设置边界寄存器为0x4c    */
    outportb(RCR, 0xcc);             /*将RCR设置成监视模式,不接收数据包*/
    outportb(TCR, 0xe0);              /*  设置TCR为loop back模式*/
#ifdef   RTL8019_OP_16
    outportb(DCR, 0xc9);          /*  设置数据配置寄存器0xc9, 16bit DMA */
#else
    outportb(DCR, 0xc8);          /*  设置数据配置寄存器0xc8, 8bit DMA */
#endif
    outportb(IMR, 0x00);          /*  设置中断标志为0x00 */
    outportb(ISR, 0xff);          /*  清除所有中断标志位  */
    SetRegPage(1);
    outportb(CURR, RPSTART+1);   /*设置当前页寄存器为0x4d*/
    outportb(MAR0, 0x00);       /*设置组播地址寄存器，MAR0～MAR7*/
    outportb(MAR1, 0x41);
    outportb(MAR2, 0x00);
    outportb(MAR3, 0x80);
    outportb(MAR4, 0x00);
    outportb(MAR5, 0x00);
    outportb(MAR6, 0x00);
    outportb(MAR7, 0x00);
    outportb(RCR, 0x00);        /*将RCR设置成正常模式,跟外部网络连接*/
    outportb(BaseAddr, 0x22);     /*启动芯片开始工作  */
    net_start = 0;
}
```

5.3.3 发送数据帧

在介绍RTL8019AS发送以太帧之前，首先让我们了解一下RTL8019AS是如何实现微控制器与网络之间的数据交换的。RTL8019AS自身有本地DMA通道和远程DMA通道，本地DMA通道负责以字节或者字的方式在本地缓冲区和FIFO之间的数据传输；同时在发送数据帧产生碰撞时，它可以在处理器不介入的情况

下自动重发；远程DMA通道负责本地缓冲区和内存之间以字或者字节的方式进行数据传输。

当主机有数据需要发送到以太网上时，主机操作系统中的TCP/IP协议栈将待发送的数据封装成以太帧的格式，通过RTL8029AS将封装好的数据帧发送到RTL8019AS(out)，ARM处理器将RTL8019AS(out)接收缓冲区中的数据读到RAM中；该数据包经过RAM中防火墙包过滤程序分析后，如果符合安全策略要求，ARM处理器将再通过I/O通道和RTL8019AS(in)的远程DMA通道把该数据帧写入RTL8019AS(in)的发送缓冲区，最后发送传输命令将它发送到网络上。

图5.4是RTL8019AS网络芯片发送数据的流程图：

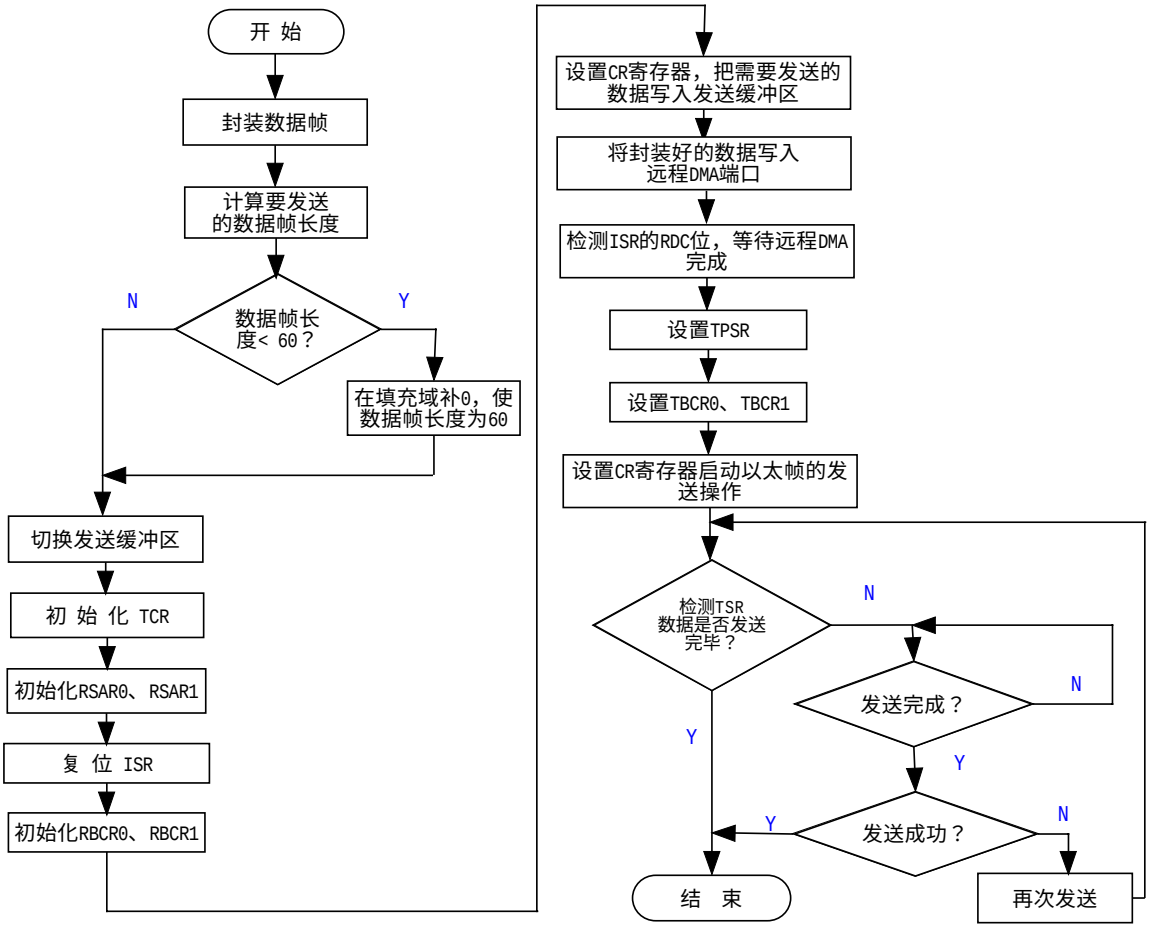


图 5.4 RTL8019 发送数据帧流程图

RTL8019AS发送数据帧的详细过程可描述如下：

1、封装数据帧，计算数据帧的长度，如果长度小于60字节，则在数据帧的

PAD字段填充0，使待发送的数据长度不小于60字节；

2、初始化有关寄存器，以启动远程DMA写操作，RSAR0和RSAR1寄存器用来指定远程DMA写操作时数据存放的缓冲区首地址，RBCR0和RBCR1用来指明本次远程DMA操作时传输数据的字节数，当远程DMA写操作完成后，RTL8019AS将ISR中的RDC位置0，通过此标志位的状态可判别远程DMA写操作是否结束；

2、设置有关寄存器，指定发送帧的起始地址和帧长度，设置TPSR用来指定待发送帧的起始地址，设置TBCR0和TBCR1用来指明待发送帧的长度；

3、通过设置CR寄存器中的RD2、RD1和RD0这三个位来启动远程写操作和发送数据帧的操作，在发送完毕后，测试传输状态寄存器TSR(Transmit Status Register)中的各标志位来检验数据帧是否已正确无误的发送完成，如果没有发送成功，则循环发送。

5.3.4 接收数据帧

在RTL8019AS初始化过程中，通过PSTART和PSTOP分配了52页的接收缓冲区，地址范围是0x4c00～0x4fff；并且把BNRY和CURR两个指针都赋成0x4c。当RTL8019AS接收到一个数据包时，按页存储，根据数据包的大小，计算所需的寄存器页数，并从CURR指针所指地址开始存储接收的数据包，不满一页的也要占用一页，并移动CURR到相应的页数。例如：当CURR＝0x4c时，如果收到一个小于256字节的数据包，则该数据包用一页来存储，CURR自动移动到0x4d(0x4c＋1)，下一个数据包从0x4d开始存储；如果收到一个1518字节的数据包，则需要分配接收缓冲区中0x4c以后的6页存储这个数据包，且CURR自动移动到0x52(0x4c＋6)，那么下一个数据包到来时，从0x52开始存储；如果CURR超过PSTOP，即CURR>0x7f时，CURR将被重新设置成0x4c。ARM处理器读数据的开始地址是BNRY指针所指的地址，每次读完一次数据，BNRY指针会移动到尚未读取数据的开始地址。

当网络上有数据包到达网卡时，RTL8019AS(in)接收逻辑将数据包进行地址匹配和CRC检验后，通过本地DMA通道将到来的数据包存储在CURR指针所指的接收缓冲区中；ARM处理器查询到CURR与BNRY不相等后，会通过远程DMA通道将接收缓冲区中的数据包读入到网卡的RAM中，该数据包经过RAM中防火墙包过滤程序分析后，如果符合安全策略要求，ARM处理器将再通过I/O通道和RTL8019AS(in)的远程DMA通道把该数据帧写入RTL8019AS(out)的发送缓冲

区，然后将该数据包发送到网卡上的RTL8029AS接收缓冲区，最后主机操作系统中RTL8029AS驱动程序接收该数据包，并交给主机操作系统协议栈进行相应的协议处理。

图5.5是RTL8019AS网络芯片接收数据帧的流程图：

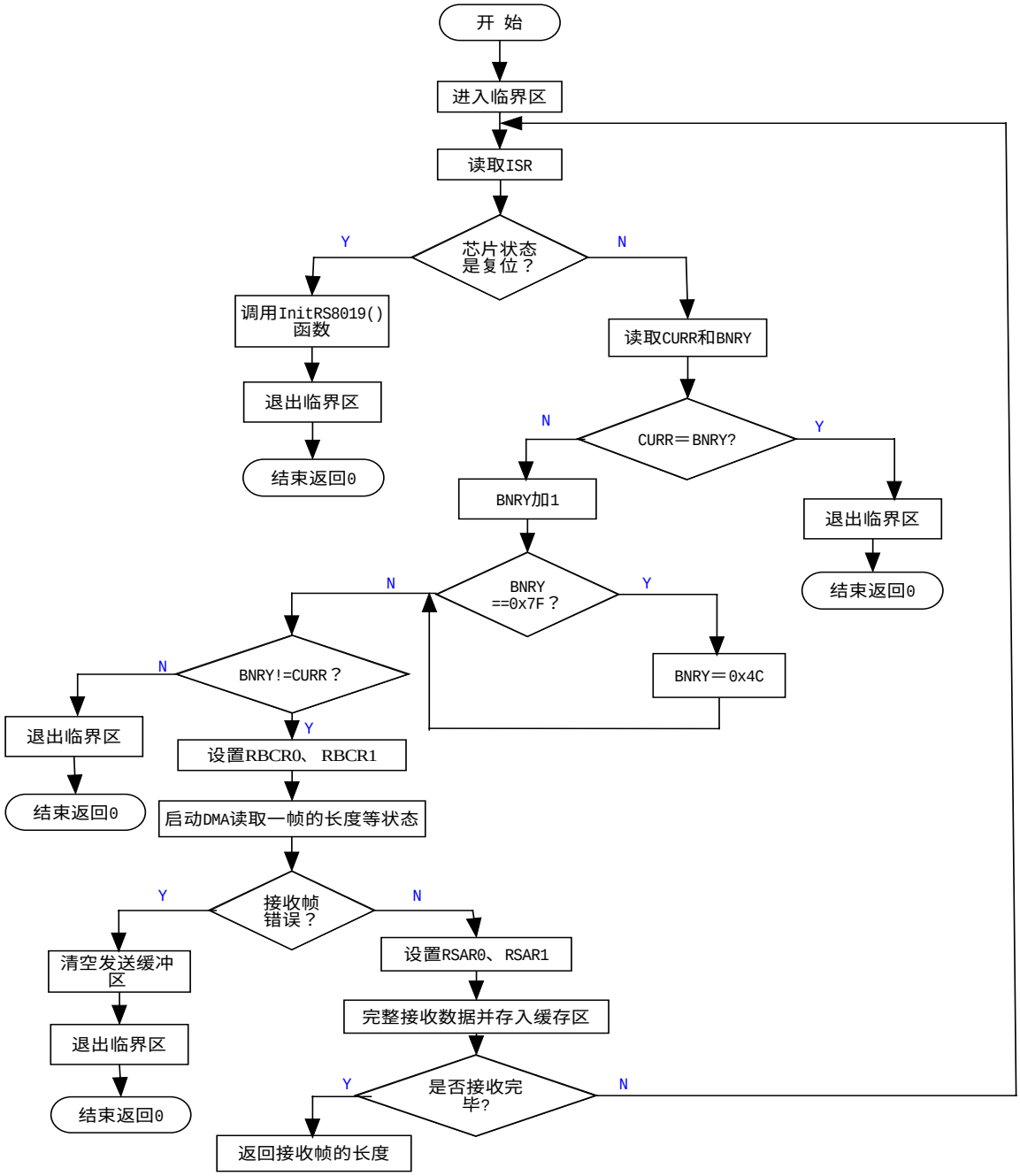


图 5.5 RTL8019 接收数据帧流程图

RTL8019AS接收数据帧的详细过程可描述如下：

1、RTL8019AS在接收一个网络数据包后，将中断状态寄存器ISR中的无错接收标志位PRX置1，ARM处理器可以通过两种方式读取该数据包，即中断方式和查询方式。中断方式是指在硬件初始化时，将中断屏蔽寄存器IMR的PRXE位置1，即允许中断，那么当接收到数据包时RTL8019AS就会产生一个中断请求信号，ARM处理器接收到这个中断请求信号后，就可以从RTL8019AS的接收缓冲区中读数据；而查询方式是指在硬件初始化时没有设置中断允许，ARM处理器只能通过查询ISR中的PRX位或者比较CURR与BNRY的值来确定RTL8019AS是否接收到数据。

2、EFW网卡采用中断的方式接收数据帧，当RTL8019AS有数据到来时，将产生一个中断信号，ARM处理器捕获到该中断信号后，就可以通过RTL8019AS的远程DMA读取数据。ARM通过读取RTL8019AS的CR中RD2、RD1和RD0组合来设定DMA的操作。“ 001 ”表示启动远程读操作；“ 010 ”表示启动远程写操作；“ 011 ”表示发送数据包；“ 1** ”表示中止或结束DMA的读写操作。当ARM处理器读取了一个完整的数据包后，就会修改BNRY的值，以便为下一次读数据包做好准备。当远程DMA读操作完成后，RTL8019AS将ISR中的RDC位置0，ARM处理器通过读取此标志位的状态来判断远程DMA读操作是否结束。

3、数据接收溢出处理。当网络上数据流量比较大或者ARM处理器没有及时将RTL8019AS接收缓冲区中的数据读走时，会导致数据接收的溢出，表现为CURR指针追上BNRY指针。出现这种情况后接收缓冲区的数据就会遭到破坏，RTL8019AS会中止接收数据，并且产生溢出，即将ISR寄存器的OVW位置1。ARM处理器对溢出进行处理的方法是：先中止RTL8019AS当前的工作，再清除(RBCR0、RBCR1)、重置CURR和BNRY指针、清除溢出标志位，重新启动RTL8019AS进入正常工作模式。

5.4 EFW包过滤程序

包过滤程序是EFW网卡中的主要软件模块，是分布式防火墙三要素中的策略执行组件。EFW网卡工作在数据链路层，因此可以获取数据帧的源和目的MAC地址、协议类型、源和目的IP地址以及TCP/UDP端口号等信息，根据这些信息包过滤程序可以进行基于MAC地址的过滤、基于IP地址的过滤以及基于端口号的过滤。这种过滤方式与传统防火墙的分组过滤(Packet Filtering)方式相类似。

5.4.1 工作流程

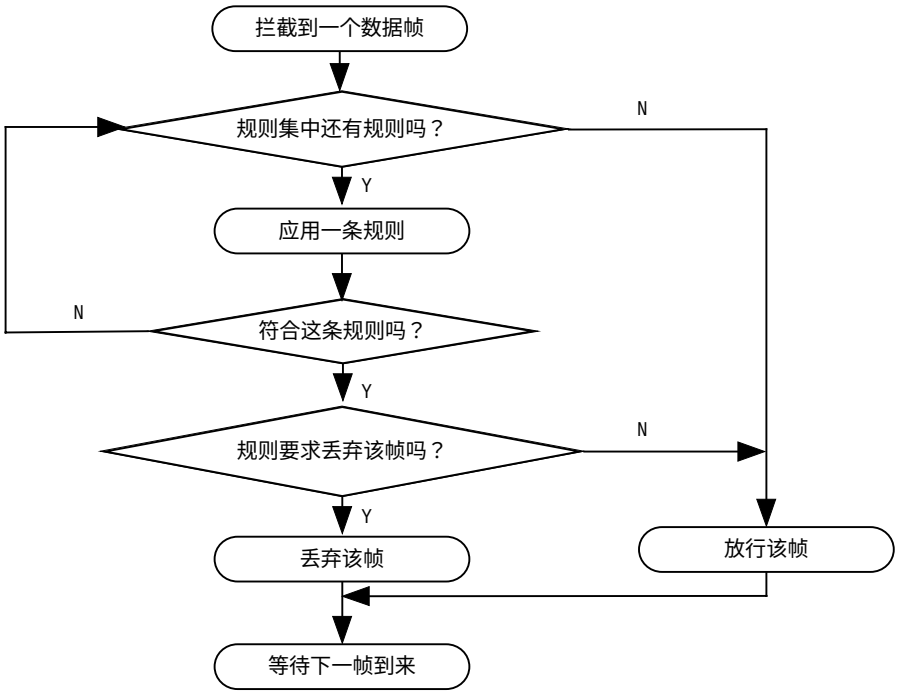


图 5.6 包过滤程序的工作流程图

EFW包过滤流程如图5.6所示，具体可描述如下：包过滤程序拦截到一个新的数据帧后，遍历过滤规则集，根据源、目的MAC地址，协议类型，源、目的IP地址，源、目的端口号，TCP选项(SYN，ACK，FIN，RST)，数据包方向(IN/OUT)等搜寻匹配规则，搜索到匹配规则之后则应用规则，根据规则对数据帧进行丢弃、放行和放行并做日志等操作。

5.4.2 报文头结构

EFW包过滤程序所涉及到的报文头格式包括：RTL8019AS接收帧格式，IP报文格式和TCP/UDP报文格式。

1、RTL8019AS接收帧格式

数据帧在网络上传输时，为了便于区分数据帧，由物理层硬件自动在数据帧的前面添加62位前导位(101010…10)和2位帧起始位(11)，在RTL8019AS芯片接收数据帧的时候会去掉这64位帧前缀，但是它会在以太帧的前面添加接收状态(8位)、下一页指针(8位)和以太网帧长度(16位)三个字段。

【接收状态】：8位，内容是接收状态寄存器(Receive Status Register)的值，用来标识数据包的接收状态，以及接收数据包的类型；

【下一页指针】：8位，用来指示下一个以太帧存放的页地址，如某时刻该字段的值为0x68，表示下一个数据帧将从0x68的页开始存放；

【帧长度】：2字节，存放数据帧的长度，按照先低字节后高字节的顺序存放，如某时刻该字段内容为：0x5000，则表示该以太帧的长度为80(0x5000)字节。

在EFW网卡的RTL8019AS驱动程序中，接收数据帧的时候，是判断数据帧的最小长度不小于64字节，就是因为有4个字节的数据帧头。

下面是网络上传输的以太帧格式与RTL8019AS接收的以太帧格式的对比图：

62位	2位	6字节	6字节	2字节	46～1500字节	X 字节	4字节
前导位	帧 起始 位	目的MAC 地 址	源MAC 地 址	类型	数据域	PAD数据域小于46字节时填充0	CRC较验

图 5.7 网络上传输的以太帧格式

1字节	1字节	2字节	6字节	6字节	2字节	46～1500字节	X 字节	4字节
接 收状 态	下一页指 针	帧长度	目的MAC 地 址	源MAC 地 址	类型	数据域	PAD数据域小于46字节时填充0	CRC较验

图 5.8 RTL8019AS 接收的以太帧格式

2、IPv4报文格式

IP是用于传输IP分组的协议，负责将传输层传下来的数据进行IP数据报的封装，再传给数据链路层处理；同时，也负责将数据链路层传上来的数据帧进行拆包分析后交给上层处理。IPv4的报文格式如下：

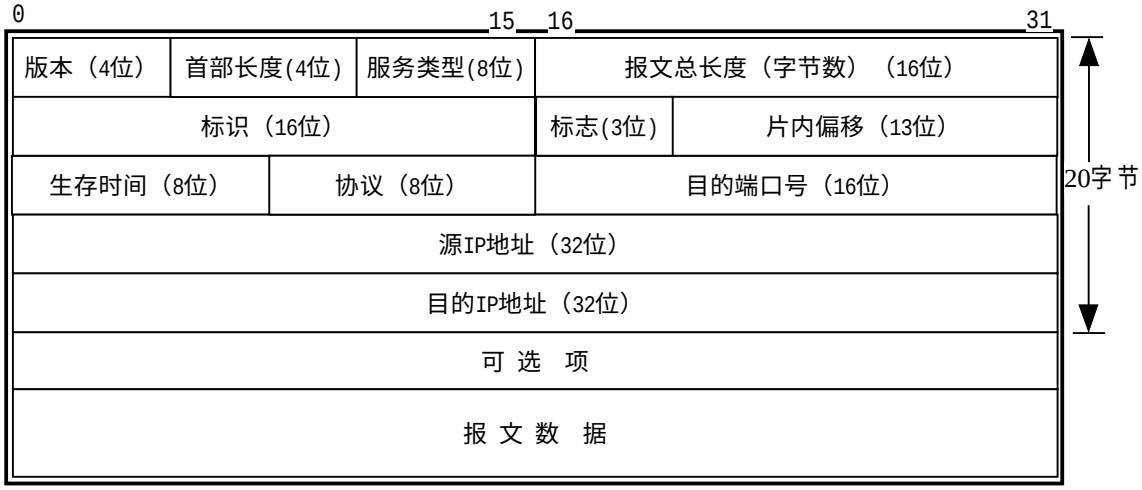


图 5.9 IPv4 报文格式

IPv4报文中的协议字段是指包含在数据部分的协议类型，如ICMP，IGMP，TCP，UDP，连同源、目的IP地址字段都是过滤引擎匹配规则所需的信息，首部中其它字段请参考[RFC 791]。

3、UDP和TCP报文格式

UDP和TCP是传输层的两个协议，传输层的功能是：当应用层将数据下传到传输层时，传输层将数据封装成UDP报文后再传给网络层；而当网络层将报文上传到传输层时，传输层将报文拆包处理后上传给应用层处理。其中TCP提供的是面向连接的、可靠的字节流服务，而UDP是一种简单的面向非连接、不可靠的数据报传输协议。下面是它们的报文格式：

图 5.10 UDP 报文格式

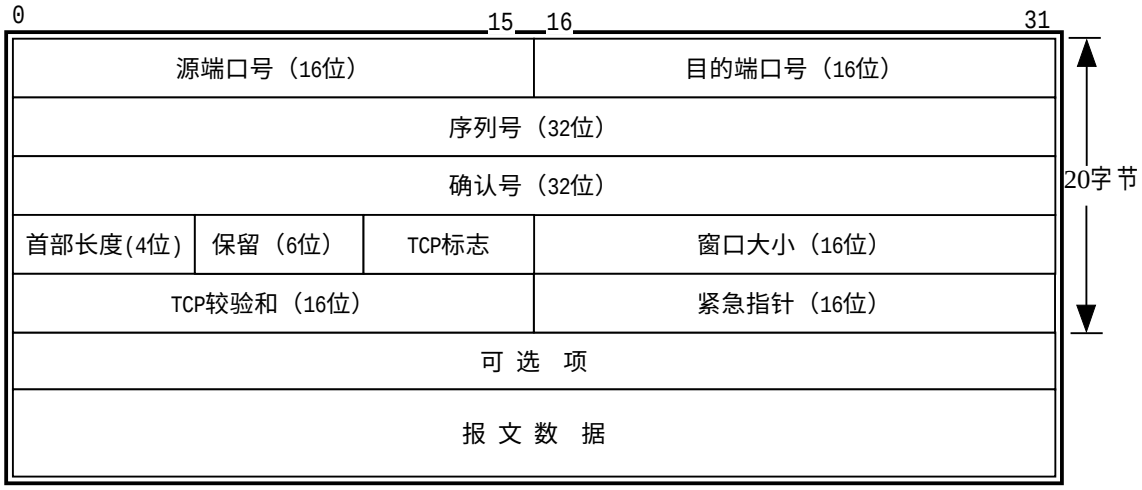


图 5.11 TCP 报文格式

UDP和TCP首部中的源、目的端口字段是过滤引擎查看的字段，其它字段请参看[RFC 768，RFC793]。

5.4.3 包过滤规则定义

EFW使用的是静态包过滤技术，在用户定义的规则中，通过检查数据帧头、IP包头的信息，来准许或拒绝包从什么地址来，到什么地址去，可能是一个地址

或一组地址；通过检查TCP/UDP包头的信息，来准许或拒绝包到达或来自相关具体服务的端口。

以下就是笔者定义的过滤规则结构体，如图5.12所示：

```
typedef enum _direction        /* 定义报文方向 */
{
 DIR_INVALID = -1,             /* 非法值 */
 DIR_IN = 0,                   /* 进入  */
 DIR_OUT,                      /* 外出 */
 DIR_BOTH                      /* 进入或外出 */
}Direction;

typedef enum _action           /* 定义对报文采取的操作 */
{
 PASS = 0,                     /* 通过 */
 SILENT_DROP,                  /* 丢弃 */
 LOG_DROP,                     /* 丢弃并记录日志 */
 LOG_ONLY                      /* 仅记录日志 */
}Action;

typedef struct _rule_item      /* 定义规则体 */
{
 int eth_proto;                /* 以太帧头的协议类型 */
 int proto;                    /* Ip头的协议类型 */
 uint32 remoteip_s;            /* 远程IP地址开始值 */
 uint32 remoteip_e;            /* 远程IP地址结束值 */
 uint16 srcport_s;             /* 源端口开始值 */
 uint16 srcport_e;             /* 源端口结束值 */
 uint16 dstport_s;             /* 目的端口开始值 */
 uint16 dstport_e;             /* 目的端口结束值 */
 Direction dir;                /* 报文方向 */
}Rule_Item, *Prule_Item;
```

图 5.12 包过滤程序的规则结构

该结构中eth_proto字段表示以太帧头的协议类型，如ARP，RARP，IP等，结构中的proto字段表示IP头的协议类型，如TCP，UDP等。在结构中定义了一个远程IP开始值和结束值，它是一个地址范围，如开始值为192.168.0.0，结束值为192.168.0.255，同样对于源和目的端口值也定义了这样一个范围。对于没有端口定义的协议比如ICMP也可以利用该字段来定义协议相关信息，本文设计各条规则的长度是一致的，这样便于数组的处理。方向字段表明该条规则应用于外出包(DIR_OUT)还是进入包(DIR_IN)，或者对于进入外出包都适用(DIR_BOTH)。所有这样的规则构成包过滤程序的规则集，包过滤程序将遍历该规则集，为一个进入或外出的报文查找匹配规则。

在匹配一条规则时，首先查看该规则的各项条件是否满足，即报文的协议

类型是否为规则指定的类型，对于外出的数据包，则匹配它的目的IP地址是否在remoteip_s到remoteip_e地址范围，而对于进入的数据包，则匹配它的源IP地址是否在这个地址范围,源和目的端口是否匹配，报文的方向是否匹配。如果各项条件都满足，则停止搜索规则集，并且应用当前规则中的动作，笔者定义了四种类型的动作，PASS：放行数据帧；SILENT_DROP：丢弃数据帧但不记录日志；LOG_DROP：丢弃数据帧并记录日志；LOG_ONLY：仅记录日志，用于调试。

5.4.4 包过滤算法及实现

EFW网卡在接收到一个数据包时，上面的包过滤程序要将数据帧头、IP包头和TCP包头的信息与定义的规则表的信息进行比较，决定是放行还是丢弃，规则表就是用户定义的安全规则。规则是按顺序进行检查的，直到有规则匹配为止，如果没有规则匹配，则按缺省的规则执行。

实际上，有两种思想决定包过滤的缺省规则：一是易于使用，二是安全第一。易于使用，一般都设置为准许放行；安全第一，一般都设置为禁止放行。在本课题的测试过程中，笔者通过修改规则的定义对这两种方法都进行了测试。下面是笔者按照图5.12中的_rule_item结构定义的安全规则，其中：

第一条规则定义是 :对外部IP地址为10.0.0.1~10.0.0.254的主机经过端口值为0~0xFFFF进入包过滤程序的数据进行丢弃操作。

第二条规则定义是：对外部IP地址为172.18.134.1~172.18.134.58的主机经过端口值为0~0xFFFF进入包过滤程序的数据进行放行操作。

第三条规则是缺省规则，其定义是：对外部IP地址为0~0xFFFFFFFF的主机经过端口值为0~0xFFFF进入包过滤程序的数据进行丢弃操作，并做日志。它被放在规则集的最后一条，这样就使得包过滤程序对于缺省规则的处理如同搜索普通规则一样，不需另作判断。

三条规则的具体形式如下：

```
Rule_Item rules[] = {
    {
        0,                  /*以太帧协议值，0表示任意协议*/
        0,                  /* IP包头协议值，0表示任意协议*/
        0x0A000001,         /*远程IP地址起始值10.0.0.1*/
        0x0A0000FE,         /*远程IP地址结束值10.0.0.254*/
        0,                  /*源端口起始值*/
```

```
        0xffff,             /*源端口结束值*/
        0,                  /*目的端口起始值*/
        0xffff,             /*目的端口结束值*/
        DIR_IN,             /*匹配所有进入到网卡的包*/
        SILENT_DROP         /*丢弃操作*/
    },
    {
        0,                  /*以太帧协议值，0表示任意协议*/
        0,                  /* IP包头协议值，0表示任意协议*/
        0xAC128601,         /*远程IP地址起始值172.18.134.1*/
        0xAC12863A,         /*远程IP地址结束值172.18.134.58*/
        0,                  /*源端口起始值*/
        0xffff,             /*源端口结束值  */
        0,                  /*目的端口起始值*/
        0xffff,             /*目的端口结束值*/
        DIR_IN,             /*匹配所有进入到网卡的包*/
        PASS                /*放行操作*/
    },
    {/*缺省规则*/
        0,                  /*以太帧协议值，0表示任意协议*/
        0,                  /* IP包头协议值，0表示任意协议*/
        0,                  /*远程IP地址起始值*/
        0xffffffff,         /*远程IP地址结束值*/
        0,                  /*源端口起始值*/
        0xffff,             /*源端口结束值  */
        0,                  /*目的端口起始值*/
        0xffffffff,         /*目的端口结束值*/
        DIR_BOTH,           /*双向，匹配所有进出包*/
        LOG_DROP            /*缺省为丢弃并做日志记录*/
     }
};
```

根据以上的安全规则，笔者给出了包过滤程序的主要函数实现：

Action　FilterPacket(Direction dir,struct sk_buff *skb1)

参数说明：dir是规则对数据操作方式结构体的指针，skb1是数据接收缓冲区的指针。

该函数负责对EFW网卡接收到的数据进行规则匹配。具体实现是，网卡驱动程序将网卡接收数据缓冲区的指针传递给该函数，通过移动该指针，可以得到相应的以太帧帧头，IP报头，以及TCP、UDP报头信息，从而根据上面定义的规则进行逐步匹配，最后返回一个对数据包操作的结构指针。以下是函数的具体实现：

```
int rules_count = 3;
Action FilterPacket(Direction dir,struct sk_buff *skb1)
{
    PRule_Item p_rule = (PRule_Item)NULL;
    Action ret ;
    struct ethhdr *eth_hdr;
    struct iphdr *ip_hdr;
    struct tcphdr *tcp_hdr;
    struct udphdr *udp_hdr;
    uint32 srcip;
    uint32 dstip;
    uint16 srcport = 0;
    uint16 dstport = 0;
    uint8 proto;
    int iphlen;
    int i = 0;
    eth_hdr = (struct ethhdr *)(skb1->data);/*得到以太网帧头指针*/
    if (ntohs(eth_hdr->h_proto) == ETH_P_ARP)/*ARP报文缺省通过*/
        return PASS;
    if (ntohs(eth_hdr->h_proto) == ETH_P_IP)/*IP报文*/
    {
skb_pull(skb1, ETH_HLEN);/*数据缓冲区指针skb1向后移动以太帧的长度*/
```

```
ip_hdr = (struct iphdr *)(skb1->data); /*得到IP头指针*/
iphlen = (ip_hdr->vhl&0x0f)<<2;/*得到IP头长度*/
srcip = ntohl(ip_hdr->saddr); /*得到源IP地址*/
dstip = ntohl(ip_hdr->daddr); /*得到目的IP地址*/
proto = ip_hdr->protocol;/*得到协议类型*/
p_rule = rules;/*过滤规则的结构指针指向定义的规则*/
while(i<rules_count)/*循环匹配安全规则*/
    {
    if(0= =p_rule->proto||p_rule->proto = = proto)/*IP头协议匹配*/
    {
if(srcip<p_rule->remoteip_s||srcip>p_rule->remoteip_e) /*远程IP地址范围匹配*/
                {
                p_rule++;
                continue;
                }
            if(TCP= =proto||UDP= =proto) /*TCP、UDP协议匹配*/
                {
                if(TCP= =proto)/*如果是TCP*/
                {
        skb_pull(skb1, iphlen); /*数据缓冲区指针skb1向后移动IP头的长度*/
                tcp_hdr = (struct tcphdr *)(skb1->data);/*得到TCP头指针*/
                srcport = ntohs(tcp_hdr->source);/*得到源端口号*/
                dstport = ntohs(tcp_hdr->dest); /*得到目的端口号*/
                }
                else/*如果是UDP*/
                {
        skb_pull(skb1, iphlen); /*数据缓冲区指针skb1向后移动IP头的长度*/
                udp_hdr = (struct udphdr *)(skb1->data); /*得到UDP头指针*/
                srcport = ntohs(udp_hdr->source); /*得到源端口号*/
                dstport = ntohs(udp_hdr->dest); /*得到目的端口号*/
                }
```

```
if(srcport<p_rule->srcport_s||srcport>p_rule->srcport_e) /*源端口范围匹配*/
                        {
                        p_rule++;
                        continue;
                    }
if(dstport<p_rule->dstport_s||dstport>p_rule->dstport_e) /*目的端口范围匹配*/
                        {
                        p_rule++;
                        continue;
                    }
                }
if(p_rule->dir= =DIR_BOTH||p_rule->dir = = dir) /*数据包的方向匹配*/
                    {
                    ret = p_rule->action;/*得到对数据包的操作方式*/
                    break;
                    }
            }
            p_rule++;
            }
        }
        return ret;/*返回对数据包的操作方式*/
}
```

5.5 小结

本节给出了EFW网卡软件的具体实现，详细介绍了EFW网卡的BootLoader启动程序的加载过程、uC/OS－II在EFW网卡系统中的移植、EFW网卡的驱动程序以及包过滤防火墙的具体实现。

第六章 总结与展望

6.1 论文的主要工作及贡献

首先，本文在分析了国内外嵌入式防火墙的实现方法后，确定了嵌入式防火墙硬件的开发方案，接下来查阅了大量的关于嵌入式系统开发、ARM处理器以及网络接口芯片的资料，给出了详细的嵌入式防火墙硬件及软件实现方案。

方案实施后，笔者进行了系统整体设计以及对主要芯片的选型。在此期间，通过在开发板上进行uC/OS-II操作系统移植，通过测试表明ARM的处理能力、网络传输速度和中断响应时间能够满足要求。同时进一步熟悉了开发环境，为下一步硬件设计工作打下基础。

硬件设计和调试是课题工作的主要部分，在进行各单元电路设计时尽可能参考成熟的电路，比如RTL8029AS网络接口电路就是参考了普通网卡的基本电路进行设计的。在本系统中也进行了借鉴，S3C44B0X的最小系统的设计也借鉴了相应的开发板电路。本课题设计的EFW网卡采用了4层板设计，这样既使得网卡硬件布局比较合理，又方便了焊接调试。硬件调试采取分阶段分步骤的进行，根据硬件的模块化顺序，一步一步的进行了元器件的焊接和调试，这样处理的好处是发现故障便于定位故障点。在笔者的焊接调试过程中就遇到过这样的情况，由于采用了分步骤的焊接调试方法，出现问题后很快就找到了故障的根源，从而在短短的两个月的时间内保障了课题的顺利完成。

核心软件设计是课题的另外一个主要部分，本课题的核心软件主要包括：系统加载程序BOOTLOADER、移植的uC/OS-II操作系统、网卡驱动程序以及EFW包过滤程序四个部分。系统加载程序以及uC/OS-II操作系统的移植在参考了优龙公司的S3C44B0X开发板程序后进行了软件的优化和修改，编写了两个RTL8019AS网络接口芯片的驱动程序，开发了基于MAC地址、IP地址和端口号进行数据包过滤的EFW包过滤引擎。

本课题的应用前景非常广阔。一种为了简化智能设备联网过程的UPnP(Universal Plug and Play Protocol，通用即插即用协议)协议为以后的家庭数字家电网络带来可能，在不久的将来，家庭的电视，冰箱，钟表都可连入网络[25]。这些智能设备，家电联入Internet后也必然会面临安全问题，倘若在UPnP设备上整合本课题设计的嵌入式防火墙，便可使用家庭中的PC机统一进行安全管理，

从而可以减少由网络攻击带来的烦恼，充分享受Internet带来的便利。

6.2 对下一步工作的思考

本文给出了嵌入式防火墙网卡硬件以及软件的设计与实现，根据预先定义好的安全策略能够完成对进出嵌入式防火墙网卡的数据进行简单过滤，但是整个嵌入式防火墙系统的功能还有待于进一步的完善。就下一步的工作笔者有如下建议：

1、策略服务器与客户机之间的策略安全分发。具体包括策略语言的选择，策略的定义，策略的加密解密以及策略数据库的设计等工作。

2、策略服务器对客户端和策略文件的维护。具体应该在策略服务器端能够实现应用层的管理界面，方便系统管理员方便对客户授权和分发策略。

3、本文给出的嵌入式防火墙网卡的工作速率为10Mbps，速度比较慢，可以考虑升级到100Mbps的网卡。

致　谢

本篇论文是在导师顾其威教授、陈兵副教授的悉心指导下完成的。两位导师不仅授业解惑，其严谨的治学态度，谦虚热心的为人都是值得学习的。陈老师更是热心的帮助寻找论文资料，指导研究方法，极大了激发了我对嵌入式防火墙研究的兴趣。特别要感谢杨志伟同学在百忙之中对我们给予硬件技术上的大力支持，还要感谢蒋俊锋同学在我撰写论文期间给予的无私的帮助。

两年多紧张充实的研究生生活对我而言是一段十分珍贵的人生履历。这期间，许多老师和同学给予我很大的关心和帮助。在我完成这篇论文的时刻，心中充满了对他们的感激之情。

首先，衷心的感谢导师顾其威和陈兵老师。我的毕业设计和论文撰写工作是在他们悉心指导和热心帮助下完成的。两位导师不仅授业解惑，其严谨的治学态度，谦虚热心的为人都是值得学习的。尤其是陈老师更是热心的帮助寻找论文资料，指导研究方法，他严谨的治学态度、勤勉的工作作风和谦逊宽厚的为人深深影响了我，是我今后工作和学习的榜样。

其次，我还要感谢杨志伟同学，是他在百忙之中给予我们硬件技术上的大力支持，并协助我们完成了对硬件的调试工作。感谢我们同届的蒋俊锋、陈海贵、胡莹和佘恒斌同学，他们在我的毕业课题设计过程中给予了很好的建议和帮助，使得我的毕业课题得以顺利完成。

再次，感谢宗亮、谢峰、文旭、仲婷、王彩霞、彭星和孟英博等教研室师弟师妹给予我的大量的支持和鼓励。

还要感谢在两年来一直和我生活在一起的同学们，是你们使我有了一个值得回忆的年代。

最后我要感谢我的父母对我长期的支持和帮助，特别感谢我的妻子李中慧，正是她在课题研究和论文撰写期间所给予的关心和支持，才使我今天有机会能够顺利完成学业。

参考文献

[1] 谢希仁，计算机网络－第四版，电子工业出版社，2003 年 6 月。

[2] 国家计算机网络应急技术处理协调中心。CNCERT/CC 网络安全工作报告。2004 年 1－6 月，第 8 页。

[3] 范英磊。防火墙技术发展趋势浅析。http://www.istis.sh.cn/list/list.asp?id=2032

[4] Bellovin S. Distributed Firewall. November 1999, ;login:, pages 37~39

[5] Bellovin S, Smith J, Keromytis A D, etal. Implementing a Distributed Firewall. In 7th ACM Conference on Computer and Communications Security, Athens, Greece, November 2000.

[6] Payne C, Markham T. Architecutre and Applications for a Distributed Embedded Firewall. Http://www.acsac.org/2001/papers/73.pdf.

[7] 田泽。嵌入式系统开发与应用教程。北京航空航天大学出版社。2005 年 3 月。

[8] 马忠梅，李善平，康慨等。ARM & Linux 嵌入式系统教程。北京航空航天大学出版社。2004 年 9 月。

[9] 胥静。嵌入式系统设计与开发实例详解－基于 ARM 的应用。北京航空航天大学出版社。2005 年 1 月。

[10] RTL8019AS Data Sheet[Z]。REALTEK SEMI-CONDUCTOR CO.LTD,2000。

[11] W.Richard Stevens。 TCP/IP 详解 卷 1：协议(范建华， 胥光辉，张涛等译)。机械工业出版社。2000 年 4 月。

[12] 周立功等。ARM 嵌入式系统软件开发实例。北京航空航天大学出版社。2004 年 12 月。

[13] 楚狂等。网络安全与防火墙技术。人民邮电出版社。2000 年 4 月。

[14] 陈建恩。温室数据采集系统远程通信接口设计研究。农业工程学报。

[15] 饶泓，陈炼，闻淑芳。一种新型的分布式防火墙系统。南昌大学学报(理科版)。2002 年 9 月，第 26 卷第 3 期，282～284 页。

[16] 陈春玲，雷世荣，陈丹伟。分布式防火墙的原理、实现及应用。南京邮电学院学报。2002 年 12 月，第 22 卷第 4 期，5～10 页。

[17] 彭倩岚，李之棠。分布式防火墙系统的安全机制设计。计算机工程与科学。2003 年，第 25 卷第 2 期，11～15 页。

[18] 纪羽中，胡静，陈春玲。分布式防火墙中的信任管理系统。2003 年 9 月，第 23 卷第 3 期，43～47 页。

[19] Tim Parker, Mark Sportack。TCP/IP 技术大全(前导工作室译)。机械工业出版社。2001

年 7 月。

[20] Markham T, Payne C. Security at the network edge: A distributed firewall. In DISCEX II, Anaheim, CA, June 2001. DARPA, IEEE.

[21] 杜春雷。ARM 体系结构与编程。清华大学出版社。2003 年 2 月。

[22] 符意德。嵌入式系统设计原理及应用。清华大学出版社。2004 年 11 月。

[23] Jean J.Labrosse 著，邵贝贝等译。嵌入式实时操作系统 uC/OS-II(第二版)。北京航空航天大学出版社。2003 年 5 月。

[24] 牟英峰，徐殿国，张东来。基于嵌入式 TCP/IP 协议栈的信息家电连接 Internet 单芯片解决方案。电子技术应用。2002 年第 6 期，16～18 页。

[25] 陈武，雷航。基于精简 TCP/IP 协议栈的信息家电网络服务器。单片机及嵌入式系统应用。2004 年第 6 期，64～67 页。

[26] 陈良银。嵌入式系统开发。四川大学计算机学院。2005 年 6 月。

[27] 杜云海。ARM 学习笔记。http://www.seajia.com

[28] 3com Corporation 著 .3com 嵌入式防火墙解决方案 . http://www.3com.com.cn/newsolutions/security/embedded.asp

[29] A. White 著，New Trojan disables firewall defense， Network News， May 2001

[30] D. Nessett and P. Humeen. The multilayer firewall. In Network and Distributed System Security Symposium, March 1998.

[31] 防火墙技术综述 http://network.ccidnet.com/pub/disp/Article?columnID=241&articleID=49048&pageNO=1

[32] Stephen Kent and Randall Atkinson. Security architecture for the Internet Protocol,RFC-2401, November 1998.

[33] 张淼。嵌入式分布式防火墙的研究，硕士论文，2005 年 3 月。

[34] 京辉热点工作室。Protel 99 电路设计实用指南。人民邮电出版社。2000 年 7 月。

[35] 网络安全又见硝烟——2002 年重点行业网络安全应用调查报告 http://network.ccidnet.com/pub/disp/Article?columnID=232&articleID=21850&pageNO=1

[36] ARM44B0 中文说明。

[37] 4510B BOOTLOADER 的实现与分析。

[38] ARM 开发环境介绍。

[39] 郭宁，张有志，孙英明。对称式密码体制数据加密算法的分析。山东工业大学学报。2001 年 8 月，第 33 卷第 4 期，365～369 页。

[40] Rivest R L, Shamir A, Adleman L. A Method for Obtaining Digital Signatures and Public Key Cryptosystems. Comm ACM, 1978;21(2)

[41] 陆玉娟。嵌入式分布式防火墙关键技术，硕士论文，2004 年 2 月。

[42] M. Blaze, J. Ioannidis, A. D. Keromytis. Trust Management for IPsec. http://www.cis.upenn.edu/~keynote/Papers/tmipsec.pdf

[43] M. Blaze, J. Feigenbaum, J. Ioannidis, etal. The keynote trust management system version 2. Internet RFC 2704.

[44] 周立功等。ARM 与嵌入式系统基础教程。广州周立功单片机发展有限公司。2004 年 9 月。

[45] 周立功等。ARM 与嵌入式系统基础实验教程。广州周立功单片机发展有限公司。2004 年 9 月。

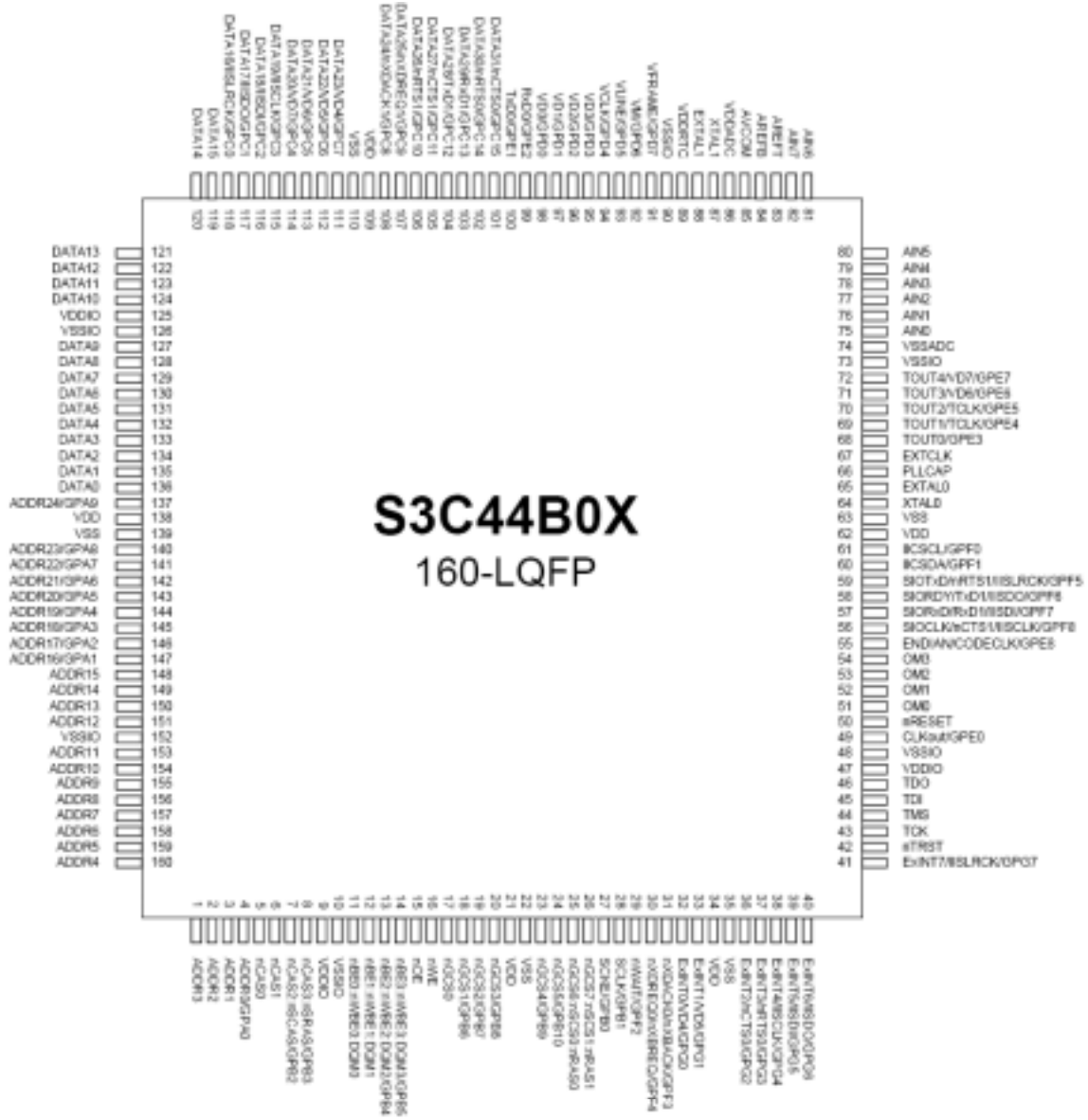
[46] 张亚旭。嵌入式电力系统动态记录装置智能采集前端，硕士论文，2005 年 3 月。

[47] 牟英峰，徐殿国，张东来。基于嵌入式 TCP/IP 协议栈的信息家电连接 Internet 单芯片解决方案。电子技术应用。2002 年第 6 期，16～18 页。

[48] Naganand Doraswamy, Dan Harkins, IPSec: the new security standard for the Internet, Intranet, and Virtual Private Networks, Prentice Hall PTR, 1999.

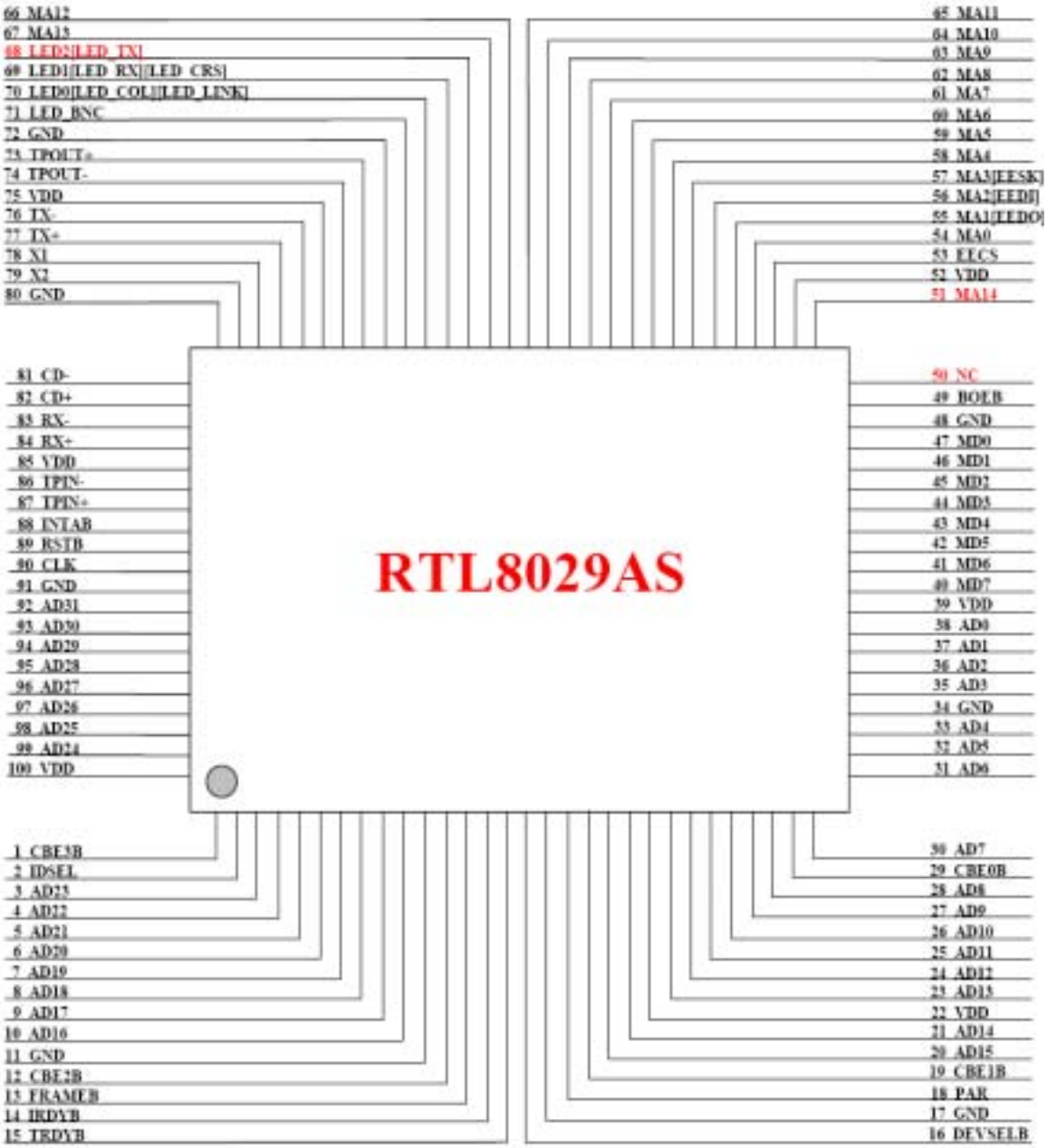
附录 I　S3C44B0X (Samsung)，ARM7TDMI 芯片资料

S3C44B0X引脚封装图



附录 II　RTL8029AS 相关资料

RTL8029AS的管脚图

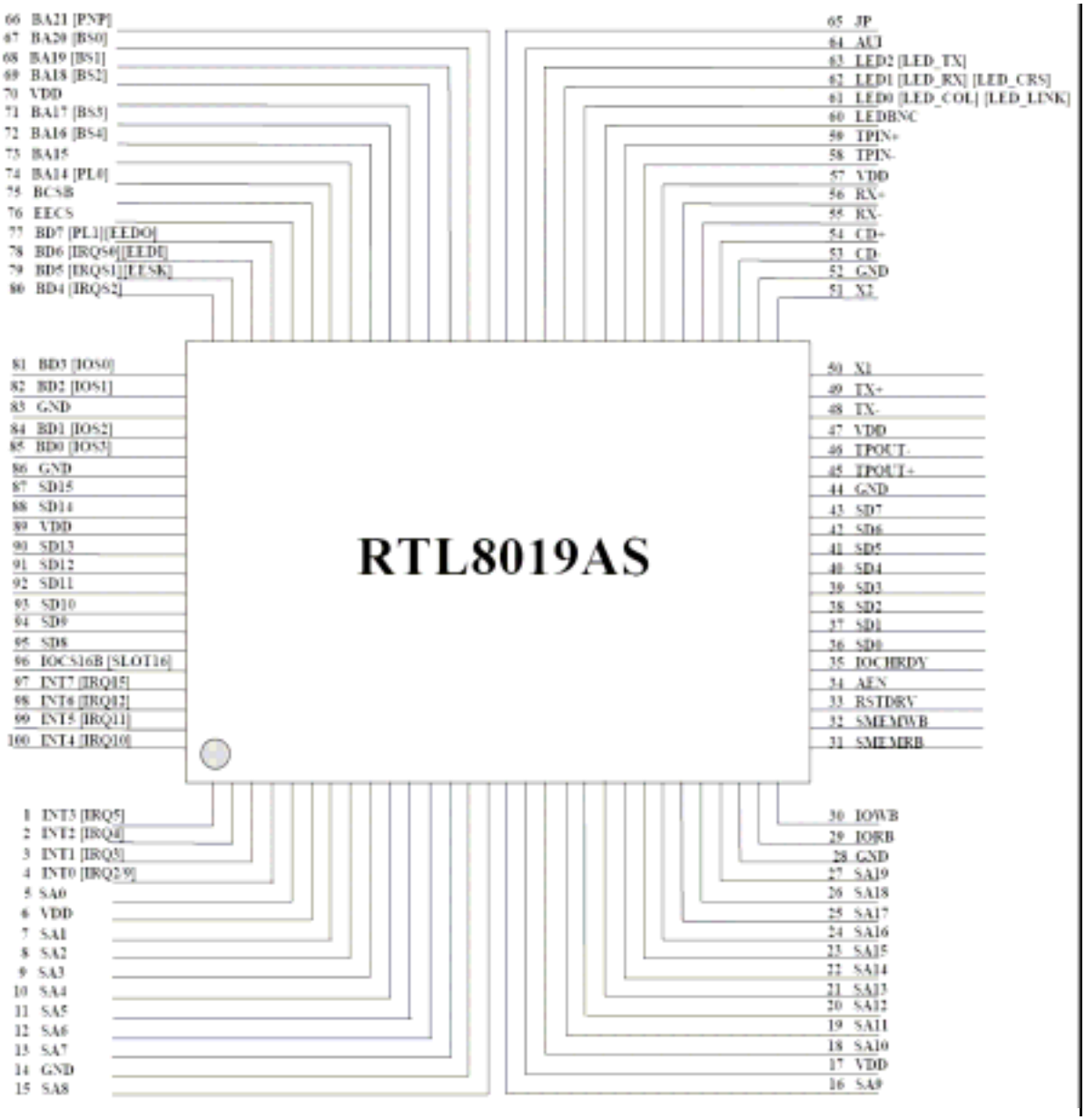


RTL8029AS的内部寄存器列表

No (Hex)	Page0		Page1	Page2	Page3	
	[R]	[W]	[R/W]	[R]	[R]	[W]
00	CR	CR	CR	CR	CR	CR
01	CLDA0	PSTART	PAR0	PSTART	*9346CR*	*9346CR*
02	CLDA1	PSTOP	PAR1	PSTOP	-	-
03	BNRY	BNRY	PAR2	-	*CONFIG0*	-
04	TSR	TPSR	PAR3	TPSR	-	-
05	NCR	TBCR0	PAR4	-	*CONFIG2*	*CONFIG2*
06	FIFO	TBCR1	PAR5	-	*CONFIG3*	*CONFIG3*
07	ISR	ISR	CURR	-	-	-
08	CRDA0	RSAR0	MAR0	-	-	-
09	CRDA1	RSAR1	MAR1	-	-	*HLTCLK*
0A	*8029ID0*	RBCR0	MAR2	-	-	-
0B	*8029ID1*	RBCR1	MAR3	-	-	-
0C	RSR	RCR	MAR4	RCR	-	-
0D	CNTR0	TCR	MAR5	TCR	-	-
0E	CNTR1	DCR	MAR6	DCR	*8029ASID0*	-
0F	CNTR2	IMR	MAR7	IMR	*8029ASID1*	-
10-17	Remote DMA Port					
18-1F	Reset Port					

附录 III　RTL8019AS 相关资料

RTL8019AS的管脚图



RTL8019AS的内部寄存器列表

No (Hex)	Page0		Page1	Page2	Page3	
	[R]	[W]	[R/W]	[R]	[R]	[W]
00	CR	CR	CR	CR	CR	CR
01	CLDA0	PSTART	PAR0	PSTART	*9346CR*	*9346CR*
02	CLDA1	PSTOP	PAR1	PSTOP	*BPAGE*	*BPAGE*
03	BNRY	BNRY	PAR2	-	*CONFIG0*	-
04	TSR	TPSR	PAR3	TPSR	*CONFIG1*	*CONFIG1*
05	NCR	TBCR0	PAR4	-	*CONFIG2*	*CONFIG2*
06	FIFO	TBCR1	PAR5	-	*CONFIG3*	*CONFIG3*
07	ISR	ISR	CURR	-	-	*TEST*
08	CRDA0	RSAR0	MAR0	-	*CSNSAV*	-
09	CRDA1	RSAR1	MAR1	-	-	*HLTCLK*
0A	*8019ID0*	RBCR0	MAR2	-	-	-
0B	*8019ID1*	RBCR1	MAR3	-	*INTR*	-
0C	RSR	RCR	MAR4	RCR	-	*FMWP*
0D	CNTR0	TCR	MAR5	TCR	*CONFIG4*	-
0E	CNTR1	DCR	MAR6	DCR	-	-
0F	CNTR2	IMR	MAR7	IMR	-	-
10-17	Remote DMA Port					
18-1F	Reset Port					

攻读学位期间公开发表的论文和参与的项目

[1] 葛广超，陈兵．基于ARM的嵌入式防火墙的设计与实现．计算机应用。(已录用)

[2] 蒋俊锋，陈兵，葛广超．嵌入式分布式防火墙策略分发机制的研究与实现．仪器仪表用户。(已录用)

嵌入式防火墙的研究与实现

作者： 葛广超
学位授予单位： 南京航空航天大学
被引用次数： 10次

本文读者也读过(10条)

1. 孟英博 嵌入式防火墙的研究与实现[学位论文]2007
2. 钟夏 基于IXP425网络处理器的嵌入式防火墙设计[学位论文]2008
3. 王江峰 基于嵌入式的主机防火墙研究[学位论文]2007
4. 蒋俊锋.陈兵.葛广超.JIANG Jun-feng.CHEN Bing.GE Guang-chao 嵌入式防火墙策略分发机制的研究[期刊论文]-仪器仪表用户 2006,13(1)
5. 王彩霞 嵌入式防火墙环境中基于IPsec的策略传输机制的研究[学位论文]2007
6. 张岳嵩.何先波.张上游.荆友波.岳淼.ZHANG Yue-song.HE Xian-bo.ZHANG Shang-you.JING You-bo.YUE Miao 嵌入式防火墙的研究[期刊论文]-西南民族大学学报（自然科学版）2009,35(3)
7. 陈金牛 嵌入式IPv6防火墙设计与实现[学位论文]2007
8. 林楠.向春枝.LIN Nan.XIANG Chun-zhi 基于Linux的嵌入式防火墙的设计与实现[期刊论文]-微计算机信息2008,24(32)
9. 史涛 防火墙嵌入式Web网管服务器的设计与实现[学位论文]2010
10. 李辉.Li Hui IPv6下嵌入式防火墙的设计与实现[期刊论文]-计算机光盘软件与应用2010(14)

引证文献(8条)

1. 张功萱,王平立,何广生,张冲 基于ARM的数据包提取转发机制的研究[期刊论文]-南京师范大学学报（工程技术版） 2008(04)
2. 何祥滨 基于ARM的嵌入式防火墙的研究与实现[学位论文]硕士 2010
3. 何广生 基于双核的安全Windows终端系统辅核通信模块研究与实现[学位论文]硕士 2008
4. 李献增 基于双核的安全信息交互机制的研究[学位论文]硕士 2008
5. 周琦 开放式控制器的安全模式研究及实现[学位论文]硕士 2008
6. 俞平 EPA安全网关状态防火墙技术的研究与实现[学位论文]硕士 2008
7. 干开峰 嵌入式EPA安全网关开发——安全功能模块的设计与实现[学位论文]硕士 2007
8. 张文杰 面向电力系统的嵌入式Web网关安全性研究与设计[学位论文]硕士 2008