中文摘要

Internet 正在把全世界的办公系统和通信系统连接起来，这为现场信息的远程访问提供了可能；现场总线为现场设备接入 Internet 提供了基础。论文探讨了 CAN 总线与以太网互连的方法，利用8 位微控制器 MC68HC908GZ60 设计并实现了 CAN 总线与以太网的互连，并将该互连系统成功地应用于远程数据采集系统。

本文首先介绍了课题的相关背景和概念，接着详细阐述了以MC68HC908GZ60 为主控 MCU 的 CAN-以太网互连系统的设计思想及软硬件实现方法。系统利用 MSCAN08 实现嵌入式 CAN 接口，利用RTL8019AS 实现嵌入式以太网接口。给出了协议网关的软硬件设计描述与关键技术的实现要点。最后将互连系统应用于苏州瑞萨半导体公司的稼动率(正常运行效率)数据采集系统。在该数据采集系统中，利用CAN 总线将现场设备连接起来，并通过协议网关实现 CAN 总线与公司的信息网相连，从而实现工业现场数据的远程采集。文中也述及了抗干扰问题、测试问题及实际研发过程中的一些体会，同时，还对工业以太网在远程数据采集中的应用进行了探讨。

关键词：CAN 总线，uIP，协议网关，MC68HC908GZ60，RTL8019AS，MSCAN08

作者：汤龙梅
指导老师：王宜怀

ABSTRACT

The Internet connects the office systems worldwidely with the communication systems and that makes it possible for the remote access to bottom information. At the same time, the field bus provides a foundation for the field devices to connect to the Internet directly. Based on the research of Ethernet connecting solution of CAN-bus, an interconnection system between CAN-bus and Ethernet is accomplished in the thesis, which is initialized with MC68HC908GZ60 MCU, an 8-bit micro-controller, and is successfully applied in a remote data collection system.

Firstly, some background information and conception related to the thesis are introduced. Secondly, the design idea and the implementing method of hardware and software of the interconnection system hosted by MC68HC908GZ60 are described in detail. The system takes MSCAN08 module to achieve an embedded CAN-bus interface and a RTL8019AS chip to complete an embedded Ethernet interface. Then the description about design of a protocol gateway 's hardware and software and some key points of the implementation are given. Finally, the interconnection system is applied in the collection system of Renesas Corporation in Suzhou, which gathers devices' running percentage data. In this system, devices located in field are connected each other by CAN-bus, while the CAN-bus is connected to the information network of the corporation via the protocol gateway, thus the remote data collection system is accomplished. Some issues about anti-jamming, testing and experience of project development are also presented. Further more, the application of industrial Ethernet in remote data collection is discussed.

Key words: CAN-bus, uIP, Protocol Gateway, MC68HC908GZ60, RTL8019AS, MSCAN08

Written by Tang Longmei
Supervised by Wang Yihuai

目 录

第一章 概述

在竞争日趋激烈的信息化和网络化时代，企业管理决策人员如何以最快速度获取生产第一线的情况，进而作出正确的生产经营决策，是保证企业生存的关键因素之一。企业信息网络化是企业与世界最新技术同步、保持其竞争力的必然要求和趋势。本文的工作是通过构建 CAN 总线与以太网互连系统，并将该系统应用于远程数据采集。本章首先介绍了相关的设计背景、作者的设计构思和课题意义，最后给出了本文的工作内容及论文结构。

1.1 背景

随着控制技术、通信技术和网络技术尤其是 Internet 的迅速发展，如何将现场测控网络接入 Internet，通过 Internet 对现场设备进行远程诊断、维护和服务，实现企业从现场控制层到上层管理层的无缝信息集成是现代化时代的基本要求[1]。

而在传统的现场设备控制方案中，受当时技术水平的限制，现场信息往往止步于“现场”[2]。现场数据通常是由现场操作员通过报表的方式汇报给上层管理者。因此，上层管理者不能快速、直接地了解现场生产情况，更不用说通过 Internet 远程访问现场设备了。这使得从数据产生到管理者作出反应之间存在较大的延迟时间。在竞争激烈、各厂商都在与时间赛跑的今天，这样的控制方案显然不能满足要求。

1.1.1 嵌入式以太网发展情况及应用前景

由于 Internet 的飞速发展，以太网通信技术得到了越来越广泛的应用。以太网是当前主流的信息传输网络，因为速度快、互操作性好、可扩展性强、价格便宜等特点已逐步在嵌入式领域得到了广泛的应用。当一台设备具有网络功能时，人们可以在任意时间、任何地点、使用任何平台随时浏览设备的实时状态，在远程实现对这台设备的监视、控制、诊断、测试和配置。由于市场对嵌入式以太网产品的需求急剧上升，投入以太网研究的公司越来越多，有些大公司已经推出带以太协议的微控制器，如 Freescale(原 Motorola)半导体公司已于 2004 年年底推出集成的 16 位微控制器 MC9S12NE64。可以预见，在不久的将来，人们身边的各种设备，都将可能成为互联网上的一员。这将是一个嵌入式系统和 Internet 相结合的时代，具有联网功能的嵌入式系统将代替 PC 在 Internet 网络中占据主导地位，这反过来又将大大地促进嵌入式以太网和嵌入式 Internet 技术的发展。

1.1.2 现场总线发展情况及应用前景

现场总线是用于现场仪表与控制系统和控制室之间的一种全分散、全数字化、智能、双向、多变量、多点、多站的通信网络。IEC(International Electrotechnical Commission，国际电工技术委员会)对现场总线一词的定义为：现场总线是一种应用于生产现场，在现场设备之间、现场设备与控制装置之间实行双向、串行、多节点数字通信的技术[3]。它是一项以智能传感、控制、计算机、数据通信为主要内容的综合技术，是当今自动化领域发展的热点，被誉为自动化领域的局域网[4]。

现场总线所涉及的应用领域十分广泛，几乎覆盖了所有连续和离散的工业领域，如过程控制自动化、制造加工自动化、楼宇自动化、家庭自动化等等。由于众多领域需求各异，加上商业利益的驱使，这使得统一的现场总线标准至今仍未完成。目前国际上存在几十种现场总线标准，比较流行的主要有 FF(基金会现场总线)、CAN(控制器局域网)、LonWorks、Profibus 等[5]。各种总线均有自己的特点，已在不同应用领域形成了各自的优势。因此，在未来几年内，将出现几大总线标准共存，甚至在一个现场总线系统内，几种总线标准的设备通过路由网关互连实现信息共享的局面。

CAN 总线是德国 BOSCH 公司于 20 世纪 80 年代为解决现代汽车中众多的控制与检测仪器之间的数据交换而开发的一种高性能串行通信协议。由于其独特的设计思想、良好的功能特性、极高的可靠性和现场抗干扰能力，已经被广泛应用于各个领域[6][7]。目前，支持 CAN 协议的有：Intel、Motorola、Philips、NEC、Siemens 等百余家国际著名大公司。CAN 总线已成为欧洲总线标准之一，是唯一被批准为国际标准的现场总线。在国际上，一些著名的汽车制造厂商如 BENZ(奔驰)、BMW(宝马)、PORSCHE(保时捷)等，都已经使用 CAN 总线实现汽车内部控制系统与检测和执行机构间的数据通信[8]。近几年来 CAN 总线已开始进入我国各个应用行业，正逐渐被大家重视。如清华大学、广州周立功单片机有限公司已着手开始研究和生产与 CAN 总线的相关产品[9]。

1.2 现场总线与以太网互连系统构思

Internet 正在把全世界的办公系统和通信系统连接起来，这为现场信息的远程访问提供了可能；现场总线为现场设备接入 Internet 提供了基础。设备在完成测量控制任务的同时，向上层提供各种现场信息。为实现企业纵向的信息集成，位于现场的测量控制设备通过现场总线互联构成底层的设备网络；设备网再通过网关等连接到 Internet 上，从而实现底层设备与上层管理网络之间的信息交互。企业管理决策人员将可以在世界的任何一个角落，通过 Internet 及时了解现场的情况，实现远程监

控[10]。因此，设备网络与 Internet 的结合将成为自动化领域信息技术发展的方向。

基于上述考虑，在完整的企业网构架中应具有 3 层结构，如图 1-1 所示，从底层向上依次为：现场过程控制层、制造执行层和企业资源规划层。

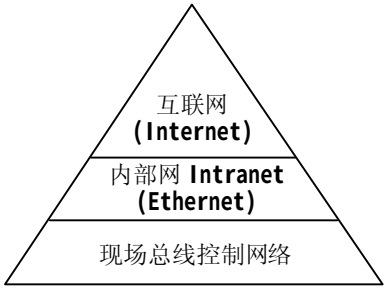


图 1-1 企业网络系统的层次结构图

本文选用比较具有代表性的 CAN 总线构建底层设备网，而以以太网(Ethernet)构建企业信息局域网，并通过 CAN-以太网网关有机地将设备网和信息网连接起来，共同完成信息网对设备网的控制和远程数据采集[11]。互连系统构思见图 1-2。

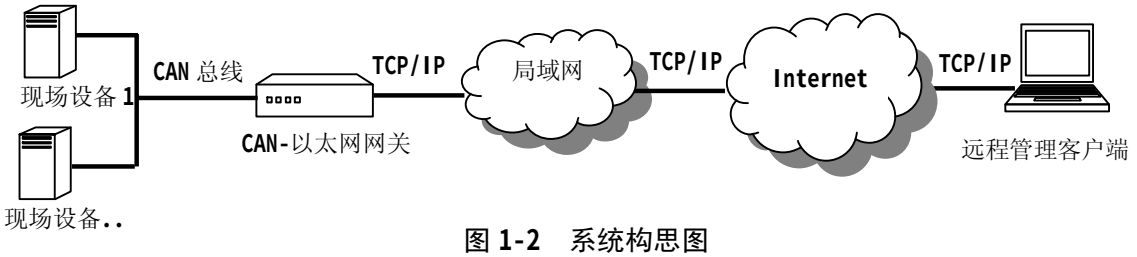


图 1-2 系统构思图

本互连系统设计核心是要解决三个问题：嵌入式以太网接口、CAN 总线接口和协议网关的实现。在本设计中，从可行性和成本等多方面考虑，均采用 8 位 MCU 作为网络节点的主控芯片。

1.3 课题意义

CAN 总线已逐步进入我国自动化领域，并越来越受到大家的重视。实现现场总线与以太网互连，具有以下现实意义：

(1) 为将工业现场与互联网互连提供简便、低成本的网络接口技术

(2) 开发具有自主知识产权的 CAN 总线节点和嵌入式 CAN-以太网网关

(3) 为 CAN 总线及工业以太网产品的国产化提供借鉴和经验

(4) 为其他现场总线在我国的应用推广提供借鉴和经验

1.4 本文工作和结构

1.4.1 本文工作

为实现现场总线与以太网互连并将之应用于远程稼动率数据采集系统，需完成如下工作：

1．需求分析

在设计硬件和软件之前，根据互连系统功能需求分析，完成系统软硬件总体设计。

2．制作实验电路板

(1) 芯片选型，分析以 **8** 位 **MCU MC68HC908GZ60**为主控芯片实现互连系统的可行性；

(2) 结合主控芯片的特点，分析 **MCU** 控制以太网接口芯片的最佳工作方式；

(3) 了解芯片的外围电路，分析芯片间的接线方式，设计硬件原理图；

(4) 绘制 **PCB** 电路图，联系厂家制作电路板；

(5) 其他元器件的选型与采购等；

(6) 焊接、测试，完成硬件系统。

3．软件设计及调试

(1) 根据 **MSCAN08** 的编程原理，完成 **CAN** 接口通信；

(2) 根据以太网接口芯片的时序要求，编写以太网接口芯片驱动程序；

(3) uIP 协议栈**(**简化的 **TCP/IP)**的选取与实现；

(4) 应用系统的系统总体设计、软硬件设计与调试；

(5) 实际应用经验总结。

1.4.2 本文结构

全文共分成七章，各章的内容安排如下：

第一章介绍了互连系统的设计背景，给出了毕业设计的核心内容及工作；

第二章简述了系统设计相关的概念；

第三章详细讲述了硬件选型、互连系统的硬件设计及测试情况，包括最小硬件系统、**SCI** 接口、**CAN** 接口和以太网接口等；

第四章阐述了在硬件基础上的软件设计，包括 **MSCAN08** 模块编程、网络接口芯片的驱动、**uIP** 协议栈、**CAN-TCP/IP** 协议转换等的实现；

第五章讲述了以互连系统为基础的稼动率采集系统的设计及实现；

第六章讲述了稼动率系统现场测试情况，分析了现场出现的干扰等问题，并给出了相应的解决措施；

第七章对本文的工作进行了总结，并提出了一些尚待研究的问题。

第二章 相关的基础知识概要

在实现互连系统设计之前，有必要了解一些相关的基础知识。本章主要对嵌入式以太网、TCP/IP 协议、CAN 总线等有关概念进行简要说明。

2.1 网络基础知识

2.1.1 网络的概念

计算机网络是计算机技术与通信技术相结合产生的技术，是用通信介质将地理上分散且具有独立功能的多台计算机相互连接，按照网络协议进行数据通信以实现资源共享的信息系统[12][13]。其主要功能实现数据传送和资源共享。

2.1.2 TCP/IP 协议

TCP/IP 协议是传输控制协议/网际协议，是网络中使用的基本的通信协议。通常说 TCP/IP 是 Internet 协议族，而不单是 TCP 和 IP，它包括上百个各种功能的协议，如：ARP、IP、UDP、TCP、HTTP 等[14]。TCP/IP 协议分层见图 2-1 [15]。数据在传输时每通过一层就要在数据上加个包头，而在接收端，每经过一层要把用过的包头去掉，以保证传输数据的格式完全一致。

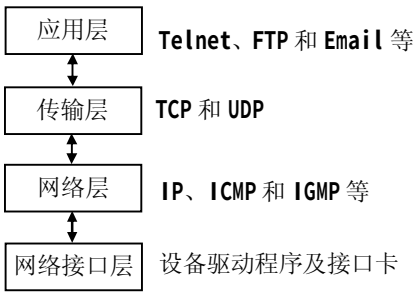


图 2-1 TCP/IP 协议层

TCP 协议和 IP 协议是 TCP/IP 中两个最基本的协议。TCP 协议即传输控制协议，提供端到端的连接，用于解决多用户或多任务操作系统中不同应用程序之间的通信。TCP 协议通过端口号标识不同的应用程序或进程。传输层协议主要有两个：传输控制协议 TCP 和用户数据报协议 UDP[16]。TCP 提供可靠的有连接传输，通过确认和超时重传机制保证数据的完整和正确，通过滑动窗口机制实现拥塞控制。一般 FTP、HTTP 等协议采用 TCP 进行传输。而 UDP 实现的是不可靠、无连接的数据报服务，它通过检验和提供数据的完整性。由于 UDP 比 TCP 简单，它不使用繁琐的流量控制或错误恢复机制，只充当数据报的发送者和接收者，因此开销小、效率高，适合于低延迟的局域网。

IP 协议提供在同一物理网络中无连接但简单有效的数据的点到点通信。如果传输过程中出现数据丢失或出错等情况，TCP 协议会自动要求数据重新传输，并重新组

包。为标识出网络中不同的点，需给该点分配一个地址即 IP 编址。每个 IP 地址标识了网络中的一个连接，连接到多个物理网络的路由器和多宿主机必须有多个 IP 地址。

2.1.3 协议网关

协议网关(Gateway)又叫协议转换器，是能够连接不同网络的软件和硬件的结合产品，它们可以使用不同的格式、通信协议或结构连接起两个系统，实现不同协议网络之间的互连[17]。网关实际上通过重新封装信息以使它们能被另一个系统读取。为了完成这项任务，网关必须提供几个网络接口并支持相应网络接口所采用的协议。

2.2 嵌入式以太网（Embedded Ethernet）和uIP

以太网(Ethernet)是 20 世纪 70 年代研制开发的一种基带局域网技术，采用载波多路访问和冲突检测(CSMA/CD)机制。由于以太网成本较低，性价比很好，已得到了广泛的应用并成为当今现有局域网采用的最通用的通信协议标准。

将以太网技术应用于嵌入式系统联网的技术称为嵌入式以太网技术[18]。以太网通信时遵循 TCP/IP 协议，因此，嵌入式以太网也需要支持 TCP/IP 协议。由于 TCP/IP 协议非常复杂，考虑到 8 位或 16 位嵌入式系统资源有限以及应用的特定性，一般并不实现全部 TCP/IP 协议，而采用经过裁减的 TCP/IP 协议栈--uIP[19]。在保留传统 TCP/IP 协议优点的同时，必须对它进行精简，尽可能做到代码精简、存储开销小，从而满足嵌入式系统的要求。

2.3 CAN（Controller Area Network）总线简介

2.3.1 CAN 总线简介

CAN 总线最初是为解决汽车中各部件之间相互通信而设计的。它采用非破坏总线仲裁、短帧结构等技术，具有通信速率高、传输距离远、抗干扰能力强等特点，且在严重出错时具有自动关闭总线功能，主要被应用在对抗干扰能力和实时通信能力要求较高，但单次通信量较小、通信距离在 3~5km 以内的一些场合。由于性能卓越，其应用范围已不再局限于汽车工业，而向过程控制、远程数据采集、机器人、智能化家居等领域发展。

2.3.2 CAN 的分层结构

CAN 遵从 OSI 模型，按照 OSI 标准模型，分为两层：数据链路层和物理层[20]。数

据链路层又包括逻辑链路控制子层 LLC 和媒体访问控制子层 MAC，如图 2-2 所示。

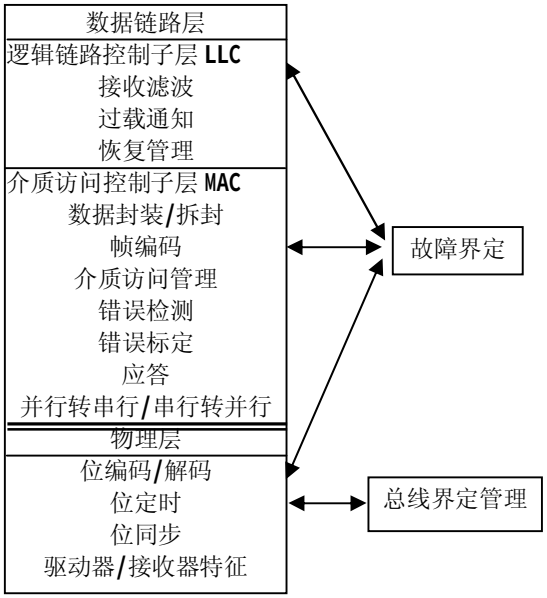


图 2-2 CAN 的分层结构

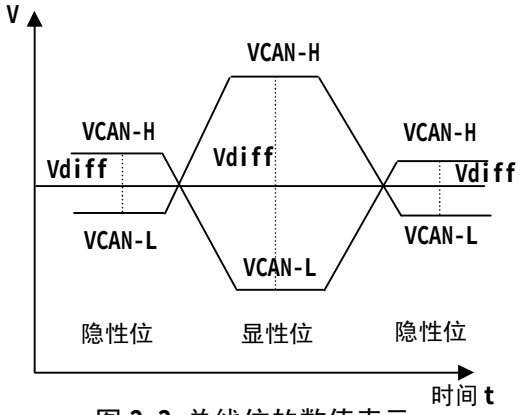


图 2-3 总线位的数值表示

2.3.3 CAN 总线的位数值表示与通信距离

CAN 总线上用显性(Dominant)和隐性(Recessive)两个互补的逻辑值表示 0 和 1[21]。当在总线上出现同时发送显性位和隐性位时，总线上数值将出现显性。如图 2-3 所示，VCAN-H 和 VCAN-L 为 CAN 总线收发器与总线之间的两接口引脚，信号是以两线之间的“差分”电压形式出现。在隐性状态，VCAN-H 和 VCAN-L 被固定在平均电压附近，Vdiff 近似于 0。在总线空闲或隐性位期间，发送隐性位。显性位以大于最小阀值的差分电压表示。

CAN 总线上任意两个单元之间的最大传输距离与位速率有关，表 2-1 列出了距离与位速率的相关数据[22]。这里的最大距离是指不接中继器的两个单元之间的距离。

表 2-1 CAN 总线上任意两单元最大距离及位速率对应表

位速率/kbps	1000	500 250 125 100		50	20	10	5
最大距离/m	40	130 270 530 620 1300			3300	6700	10000

2.3.4 相关概念

【帧】：即报文，CAN 协议规定信息以帧为单位进行传输。每一种帧都有一固定的类型格式。

【位速率】：总线的传输速率。在一个给定的 CAN 系统中，位速率固定的。

【仲裁】：CAN 协议是多主系统，若同时有两个或两个以上节点发送数据则会发生冲突。此时，总线访问冲突运用逐位仲裁规则，借助标识符 ID 解决。仲裁期间，

每一个发送器都对发送位电平与总线上检测到的电平进行比较，若相同则该单元继续发送。当发送的是一隐性电平而监视到的是一显性电平，则该节点退出发送状态，直到监听到总线空闲时再开始发送。

【应答】：所有接收器对接收到的报文进行一致性检查，即检查该报文是否为本节点能够接收的信息。只有一致的报文才能被接收。

【报文滤波】：每个报文都有一特定的标识符 ID，定义为可接收该标识符的节点可以接收该报文。通过接收报文滤波，特定的节点可以选择接收特定的信息帧。

【错误检测】：当任何节点检测到一个报文出现 CRC 检验错、位填充错或报文格式错等会将该报文标志为出错。标志为出错的报文会失效并自动重传。

2.3.5 报文传输和帧结构

在 CAN 协议中，数据是以帧为单位进行传输的，只有当一帧的所有数据都正确传送，数据才算正确传送，否则必须重新传。在 CAN 技术规范中，规定了总线上传输的 4 种帧类型[23]：

数据帧：数据帧将数据从发送器传输到接收器。

远程帧：总线单元发出远程帧，请求发送具有同一标识符的数据帧。

错误帧：任何单元检测到总线错误就发出错误帧。

过载帧：过载帧用于在先行和后续数据帧（或远程帧）间提供附加的延时。

数据帧与远程帧之间以帧间空间隔开。

1．数据帧

数据帧由 7 个不同的位场组成：帧起始、仲裁场、控制场、数据场、CRC 场、应答场、帧结束，如图 2-4 所示。

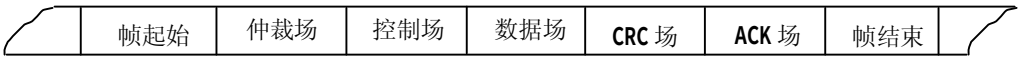


图 2-4 数据帧格式

帧起始：数据帧和远程帧的起始，仅由一个单独的“显性”位组成。只有在总线空闲时，才允许站开始发送报文。

仲裁场：由标识符和远程传送请求组成，发生冲突的两个节点将在这个场中进行仲裁。标识符有两种格式：11 位的标准格式和 29 位的扩展格式。

控制场：由 6 位控制信息组成，包括 4 位长度代码和 2 位保留位。

数据场：最多 8 字节数据。

CRC 场：利用 CRC 检验保证数据的正确性。

ACK 场：在 ACK 场中，发送节点释放总线，所有接收到有效数据的节点改写总线

状态，以通知发送节点它们已经接收到数据。

帧结束：标志整个帧的结束。

2．远程帧

作为数据接收的站，可以借助于发送远程帧启动其资源节点传送数据。远程帧由 6 个不同的位场组成：帧起始、仲裁场、控制场、CRC 场、应答场、帧结束。与数据帧相比，没有数据场。

3．错误帧

错误帧由两个不同的场组成。第一个场是由不同节点提供的错误标志的叠加；第二个场是错误界定符。当一个节点检测到错误时，该节点强行控制总线电平，这违背了正常的通信协议，因此每个节点都能识别到这个错误并做出相应的出错处理。当节点检测到错误信息并等待该错误信息接收后，各个节点都会维持一定的总线空闲状态位时间，以便界定总线错误信息，并恢复正常通信。

4．过载帧

过载帧由过载标志和过载界定符组成，其定义和错误帧类似。

2.4 差分信号原理

传统的并行总线大多使用“正信号”技术或“负信号”技术来表示二进制数，这两者都是利用一条线路来传输一个比特位，称为“单端”(Signal_ended)。若在数据传输过程中受到只是低强度的干扰，高低电平信号都不会突破临界值，信号能正常传输。但如果受到强烈的干扰，高低电平完全可能因此而产生突破临界值的大幅度扰动，例如，当低电平被抬高至临界值时，接收端就会认为它收到的是高电平信号 1，由此造成数据受损，传输失败。同样，传输高电平时也会由于干扰降低电压，而出现低于临界值的情况。因此，单端技术无法适应高速总线的需要。

而差分信号使用两条线路才能表达一个比特位，这两条线路传输信号的电压差值作为判断 1 还是 0 的依据。差分信号技术有以下三个优点[24]：

(1) 能够很容易地识别小信号。在一个单端正信号方案的系统里，由于以“地”做基准的测量信号，其精确值依赖系统内“地”的一致性。信号源和信号接收器距离越远，它们局部地的电压值之间有差异的可能性就越大。而从差分信号恢复的信号值在很大程度上与“地”的精确值无关。

(2) 对外部电磁干扰(EMI)高度免疫。在遭受干扰时，差分信号的两端受干扰的方向和幅度几乎相同，其差值就可始终保持相对稳定，因此数据的准确性不会因干扰、

噪声而有所降低。并且，除了对干扰不大灵敏外，差分信号比单端信号生成的 EMI 还要少。

(3) 在单电源系统，能够精确地处理“双极”信号。为了处理单端，单电源系统的“双极”信号，必须在地和电源干线之间某任意电压处(通常是中点)建立一个虚地。用高于虚地的电压来表示正极信号，低于虚地的电压来表示负极信号。接下来，必须把虚地正确地分布到整个系统里。而对于差分信号，不需要这样一个虚地，这就使在处理和传播双极信号有一个高逼真度，而无须依赖虚地的稳定性。

当然，由于需要两条数据线路，在总线位宽相等的条件下，差分技术需要的数据线条数就是单端技术的两倍。

第三章 互连系统硬件设计及测试

互连系统的目标是为实现位于两个不同网络/总线中的设备能够相互通信，因此首先要解决的的问题是如何将这些设备连接到相应的网络/总线中，使之成为通信网络上的节点。在本文的设计中，均采用嵌入式方案实现CAN接口与以太网接口。对于一个嵌入式应用系统，硬件设计是实现整个系统的基础。本章先介绍芯片选型的相关知识，再从硬件的最小系统开始，依次介绍主控芯片的支撑电路、SCI接口、以太网接口、CAN接口的硬件设计及相应测试，最后给出了CAN总线-以太网网关的硬件设计。有了稳定可靠的硬件系统，才有可能保证在此之上的软件系统的正常运行，同时，也使作者后期的工作集中在软件的编写和完善上。

3.1 硬件选型

针对一定的用途，选择合适的主控芯片是十分重要的。选择功能过少的单片机，无法完成系统功能；选择功能过强的单片机，则会造成资源浪费。选择的原则一般主要有以下几点[25]。

(1) 单片机对应用系统的适用性。例如：是否有足够的I/O引脚、是否含有所需的外围部件、极限性能是否满足要求等；

(2) 单片机的可购买性；

(3) 单片机的可开发性。例如：程序下载工具、调试工具、开发工具等；

(4) 技术支持；

(5) 语言体系与熟悉程度。

在以往的学习开发过程中，作者分别用 MC68HC908GP32、MC68HC908JL8 和 MC9S08GB60 单片机开发过不同项目，对 Freescale 半导体公司的 08 系列 MCU 有比较深入的了解。由于 08 系列 MCU 的兼容性和相通性，作者采用该系列中资源比较丰富的 MC68HC908GZ60 MCU(见下面微控制器特性介绍)作为互连系统的主控芯片。

确定主控芯片后，接下来应该考虑系统其他部分的需求情况，此处主要是 CAN 通信接口和以太网通信接口。由于 MC68HC908GZ60 MCU内部已经集成了 CAN 协议控制器，故在实现 CAN 接口时不需再外接 CAN 协议控制器，可直接与 CAN 总线驱动器相连；由于目前并没有集成了以太网控制器的 8 位 MCU，为实现以太网通信，需另外接以太网控制芯片实现以太网接口。以太网控制芯片种类很多，考虑 8 位嵌入式系统的实际应用要求，在此选用台湾 Realtek 公司生产的 RTL8019AS 以太网控制器作为以太网接

口芯片(NIC)，稍后将介绍该芯片的特性。另外，为了完成程序的下载和调试工作，需要通过串行接口和 PC 机相连。

3.2 芯片介绍

3.2.1 MC68HC908GZ60 微控制器

1．MC68HC908GZ60 微控制器主要性能

MC68HC908GZ60 MCU 是 Freescale 半导体公司于 2004.5 月推出的一款低功耗、高性能的 8 位 MCU。该芯片内部集成了 CAN 协议控制器 MSCAN08 模块；具有 2KB 片内 RAM、60KB 片内 FLASH 和 352B 监控 ROM；并带有 24 路 10 位 A/D 转换器和多达 53 根通用 I/O 引脚[26]。其内部功能模块框图和存储器组织及地址分配分别参见附录 A.1 和附录 A.2。

MC68HC908GZ60 MCU 采用 64 脚 QFP 封装，体积小，因而寄生参数减小，可靠性高，适合高频应用，其封装形式参见附录 A.3，各引脚分布如下：

A 口(8)：PTA0/KBIP0/AD8~PTA7/KBIP7/AD15，还可用作键盘中断输入口和 AD 采集输入口；

B 口(8)：PTB0/AD0~PTB7/AD7，还可用作 A/D 采集输入口；

C 口(7)：PTC0/CANTx~PTC1/CANRx，PTC2~PTC6，其中，PTC0、PTC1 与 CAN 接口引脚 CANTx、CANRx 复用；

D 口(8)：PTD0~PTD7，与时钟输入捕捉复用；

E 口(6)：PTE0/TXD~ PTE1/RXD，PTE2~5，其中 PTE0、PTE2 与 SCI 接口引脚 TXD、RXD 复用；

F 口(8)：PTF0~PTF7；

G 口(8)：PTG0/AD16~PTG7/AD23，还可用作 A/D 采集输入口。

2．MSCAN08 简介

MSCAN08 是 Freescale 为 8 位 MCU 设计的通用 CAN 通信模块，该模块的设计符合 CAN2.0A/B 协议标准，具有如下特性：

(1) 模块化的设计；

(2) 符合 CAN2.0A/B 协议标准，支持标准和扩展帧格式，支持远程请求帧；

(3) 一帧最多 8 字节数据，高达 1Mbps 的可编程通信速率；

(4) 4 个 FIFO 收发缓冲区，其中 3 个为具有局部优先级的发送缓冲区；

(5) 灵活的标识符验收模式，可配置成 1 个 32 位过滤码、2 个 16 位过滤码和 4

个 8 位过滤码；

(6) 内置低通滤波的远程唤醒功能；

(7) 可编程为方便调试的自环工作模式。

MSCAN08 模块的内部结构见图 3-1。

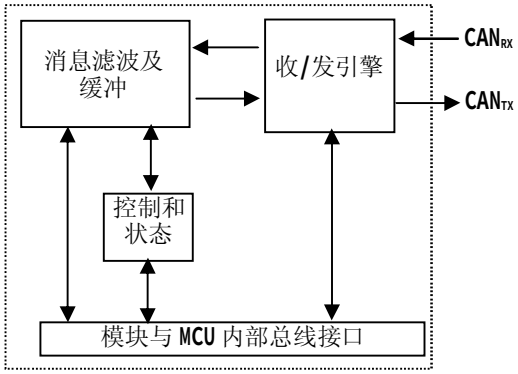


图 3-1 MSCAN08 模块内部结构图

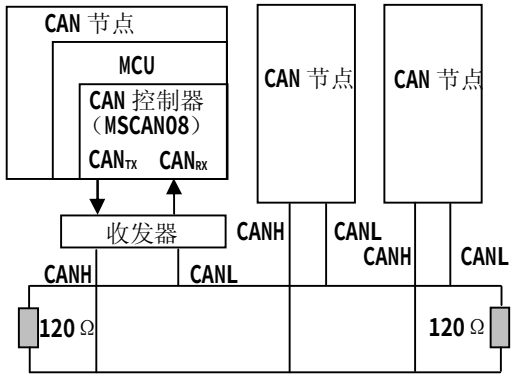


图 3-2 由 MSCAN08 模块构建的 CAN 系统

MSCAN08 使用 2 个外部引脚，一个输入（CANRX），一个输出（CANTX）。CANTX 输出引脚代表了 CAN 上的逻辑电平：0 为显性，1 为隐性。由 MSCAN08 模块构建的典型的 CAN 系统如图 3-2 所示。

3.2.2 RTL8019AS 网络接口芯片

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[27]：

(1) 符合 Ethernet II 与 IEEE802.3（10Base5、10Base2、10BaseT）标准；

(2) 全双工，收发可同时达到 10Mbps 的速率；

(3) 内置 16KB 的 SRAM，用于收发缓冲，降低对主处理器的速度要求；

(4) 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；

(5) 可连接双绞线和同轴电缆，并可自动识别所连接的介质；

(6) 允许 4 个诊断 LED 引脚可编程输出；

(7) 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、MAC(介质访问控制)逻辑、数据编码解码逻辑和其他端口。本地 DMA 完成控制器与网线的数据交换。单片机与 RTL8019AS 之间的数据交换则通过远程 DMA 完成。当单片机要向以太网发送数据时，先将数据通过远程 DMA 送到 RTL8019AS 的发送缓冲区，再启动 RTL8019AS 的发送命令，

由 RTL8019AS 将数据发到网线上；RTL8019AS 接收到的数据通过 MAC 比较、CRC 检验后，由 FIFO 存入接收缓冲区，收满一帧后，置有关寄存器标志并产生接收中断(若中断允许)，单片机可以查询相应标志位来判断是否有数据到达。原理框图见图 3-3。

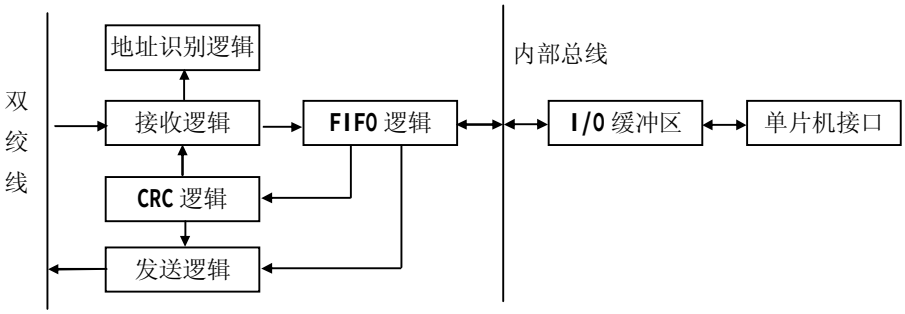


图 3-3 RTL8019AS 原理框图

图 3-3 中，接收逻辑在接收时钟的控制下，将串行数据拼成并行数据送到 FIFO 和 CRC，CRC 逻辑对输入的数据进行 CRC 检验，若检验出错，则拒收该数据；发送逻辑将 FIFO 送来的数据在发送时钟的控制下按位逐步移出，CRC 逻辑对欲发送数据进行 CRC 计算，并将计算结果附在数据尾传送；地址识别逻辑对接收帧的目的地址与预先设置的本地 MAC 地址进行比较，如不相同且又不是广播地址，则拒收该数据帧；FIFO 逻辑对收发的数据作 16 字节的缓冲，以减少对本地 DMA 请求的次数[28]。

3.3 MC68HC908GZ60 支撑电路

虽然单片机将 CPU、ROM、RAM 以及 I/O 都集成在一个集成电路芯片中，但仍需要一些外部电路的支持才能工作起来，如电源、时钟等，硬件设计的第一步是让 GZ60 “跑”起来。图 3-4 为由 MC68HC908GZ60、晶振电路及电源供电电路组成的支撑电路。

在图 3-4 中，引脚 OSC1 和 OSC2 分别为晶振输入引脚和输出引脚，晶振电路所用的元件有：4M 晶振，22p 晶体固定电容，15p 晶体微调电容及 10M 反馈电阻；在 PCB 板布线时，晶振电路应靠近 MCU，用最短的连线连接，并远离其他布线[29]。VSS 和 VDD 为 MCU 的电源引脚，VSS 接电源地，VDD 接+5V，为提高电源稳定性，在 VDD 和 VSS 间连接两个滤波电容。

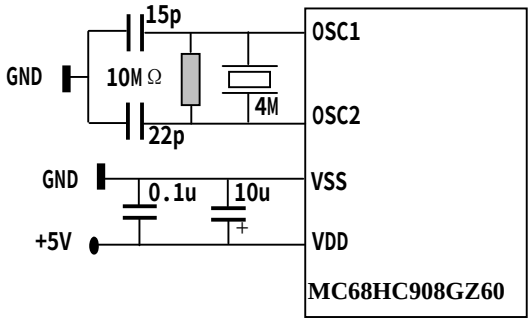


图 3-4 MC68HC908GZ60 支撑电路

支撑电路设计完成后，应检测一下电源是否正常供电、晶振能否起振、芯片能否正常工作。通过电源指示灯和万用表，可以了解电源是否正常供电以及电压值是否正确；通过示波器检测晶振频率，可以知道晶振的运行情况。芯片正常工作与否则应通

过软件方式进行测试。通过 MC68HC908GZ60 写入器，将包括芯片初始化和简单 I/O 控制的程序下载到芯片中，通过观察受 I/O 引脚控制的指示灯的亮暗变化，可以确定芯片是否正常运行。

3.4 MC68HC908GZ60最小系统硬件连接及测试

主控芯片的支撑电路设计好后，接下来需要解决用户程序的写入问题。芯片在出厂前已经被固化了 352 字节的监控 ROM，用户可以通过监控 ROM 提供的读、写、运行等操作命令实现程序的写入，即监控方式下的在线编程。然而在这种方式下进行程序的写入存在两个大的缺点：一是需给特定引脚加 9V 高压，且还需满足其他一系列引脚的特定电平值才能进入监控方式，操作非常不方便；二是监控方式下的程序下载是半双工，且只有一根通信线，下载速度慢。

基于监控方式下在线写入的不足，为了方便快捷地下载和调试程序，现在多数解决方案都是利用 MCU 的 FLASH 空间较大且一般都带有串行通信(SCI)模块的特点，在 FLASH 中划出部分存储空间(如 2KB)，驻留用户自定义的监控程序[30]。监控程序的主要功能便是通过 SCI 与 PC 主机通信，与PC 方的在线编程系统共同完成程序的下载和调试。带有 SCI 通信和支撑电路的系统就是一个能完成程序下载、调试和运行的最小系统。作者便是采用编写自己的监控程序来完成实验程序的下载和调试的。

1．最小系统硬件设计

MC68HC908GZ60 最小系统的硬件连接图见图 3-5。

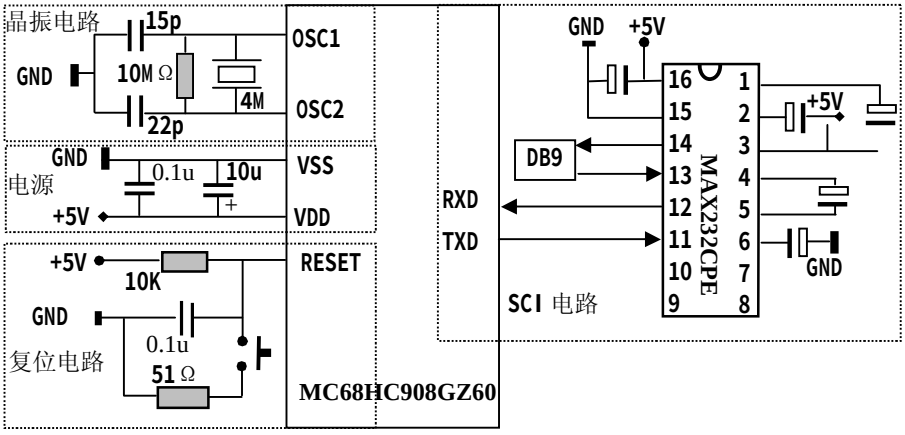


图 3-5 带有串行通信的 MC68HC908GZ60 最小系统

从图3-5中可以看出，最小系统由以下几部分组成：

(1) 晶振电路：本系统直接接4M外部晶振，可作为芯片内SCI、MSCAN08等模块的时钟源；

(2) 电源部分：电源部分电路给MC68HC908GZ60提供+5V直流电压；

(3) 复位电路：MC68HC908GZ60的复位引脚RESET平时被10K电阻外部上拉到5V，为高电平；当按下复位按扭时，通过51Ω电阻被拉到地，变为低电平，芯片复位；

(4) SCI电平转换电路：MCU的串行通信引脚TXD、RXD分别接TTL电平转换芯片MAX232的11（T1IN）、12（R1OUT），MAX232的13（R1IN）、14（T1OUT）为分别232电平的接收与发送引脚。当MCU需要往外发送数据时，MCU的TXD(TTL电平)信号经过MAX232的11（T1IN）送到MAX232内部，在内部TTL电平被“提升”为232电平，通过14（T1OUT）发送出去；当外部有数据发往MCU时，外部232电平经过MAX232的13（R1IN）进入到MAX232的内部，在内部232电平被“降低”为TTL电平，经过12（R1OUT）送到MCU的13（RXD），进入MCU内部。

2．最小系统硬件测试

在保证原理正确的硬件电路设计完成后，需要对硬件电路进行测试，包括硬件的稳定性和可靠性，这一点在嵌入式应用系统中是非常重要的。首先，若硬件电路设计不正确，软件写得再好也是无法运行的；其次，硬件系统的稳定性和可靠是保证软件正常运行的前提条件。硬件系统往往也是按功能模块分块设计的，硬件设计人员必须保证每一块硬件功能模块各自的正确性、可靠性和稳定性，才有可能保证整个硬件系统稳定可靠地运行。这一点与模块化程序设计的思想是相通的。故完成最小系统的硬件设计后，应进行最小系统的硬件测试。

在最小系统的硬件测试方案中，作者通过监控方式下的在线编程将包含串行初始化、串行发送和接收子程序的用户监控程序写入GZ60的FLASH区，并运行该程序，然后利用PC机方的在线编程系统将用户应用程序通过PC机串口发往GZ60。GZ60均能成功接收并完成用户程序的写入，说明最小系统能正常工作。

3.5 MSCAN08接口硬件设计及测试

1．MSCAN08 接口硬件设计

MSCAN08模块主要是完成GZ60与CAN总线之间的数据通信，MSCAN08接口硬件连接如图3-6所示。

在图3-6中，PCA82C250是CAN总线收发器[31]，它是协议控制器和物理传输线路之间的接口，能以高达1Mbit/s 的位速率在两条有差动电压的总线电缆上传输数据。在干扰较弱的场合，可以采用MSCAN08与收发器直接相连的方式，以简化电路。在本设计中，为了提高CAN节点的抗干扰能力，MSCAN08的CANTx和CANRx并不直接与PCA82C250的TXD和RXD相连，而是通过高速光耦6N137后再与之相连，以实现总线上各CAN节点间的电气隔离[32][33]。需特别说明的一点是，光耦部分电路所采用的两个电源VCC和VDD

必须完全隔离，否则采用光耦就失去了意义。在实际使用中，为实现电源隔离，可采用多电源输出模块供电。图3-6中虚线左边采用与MCU相同的电源，而虚线右边采用另外一个5V电源供电。

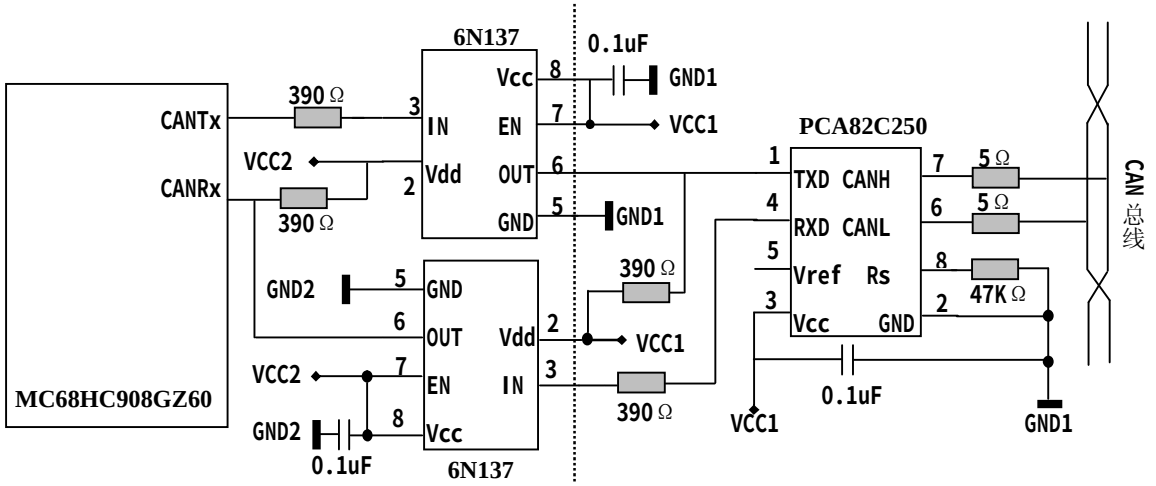


图 3-6 CAN 接口部分电路原理图

CANTx和CANRx经光电隔离后分别变为6N137T的OUT引脚和6N137R的IN引脚，此两脚再分别与82C250的TXD和RXD相连，以便连接到82C250的内部收发器；收发器通过有差动发送和接收功能的两个总线终端CANH和CANL连接到总线电缆，两个5Ω电阻用于限流，以免82C250受到过流的冲击；输入Rs用于模式控制；参考电压输出VREF的输出电压是额定VCC的0.5倍，其中收发器的额定电源电压是5V。详细资料请参阅PCA82C250技术文档。

2．MSCAN08 接口硬件测试

MSCAN08接口硬件测试按以下两步进行：

(1) 由于MSCAN08模块提供了方便的自测模式，在不加任何外围电路的情况下，可使CAN的发送和接收引脚在芯片内部相连，故先对MSCAN08模块进行在自测模式下进行测试，以确定MSCAN08协议控制器能正常工作。

(2) 构建CAN总线，并将两个CAN结点接入总线中，其中一个作发送结点，另一个作接收结点，测试两个结点是否能正常收发数据。

3.6 以太网接口硬件设计及测试

1．MCU 对 RTL8019AS 控制原理分析

以太网接口通过RTL8019AS实现。RTL8019AS有100根外部引脚(引脚图参见附录B.1)，在微处理器实现对RTL8019AS的控制以实现数据收发功能时，需考虑下列引脚的连接：

(1) 16位数据总线，SD0～15；

(2) 20位地址总线，SA0～19；

(3) I/O读控制，IORB，低电平有效；

(4) I/O写控制，IOWB，低电平有效；

(4) 使能线AEN，低电平有效；

(5) 复位引脚RSTDRV，高电平有效；

(6) 16位/8位数据选择线IOCS16B，高电平为16位模式，低电平为8位模式；

(7) 网络接口采用10BaseT，需要TPIN+、TPIN-、TPOUT+、TPOUT-；

(8) 电源脚和接地脚。

尽管主控芯片GZ60通用I/O引脚较多(53根)，但不能也没必要全部用于对上述众多引脚进行控制，只要能实现对RTL8019AS的控制以及与后者之间的数据交换即可。GZ60通过控制总线、地址总线及数据总线完成对RTL8019AS内部寄存器的控制以及与RTL8019AS进行数据通信，因此，可以从三种总线上进行分析，进而得到简化的网络接口控制电路。

RTL8019AS的寄存器地址为0x00~0x0f，有page0~page3共4页(参见附录B.2)，每页有16个寄存器，均以0x00~0x0f访问。寄存器的访问由三个因素来决定的：一是命令寄存器的PS1和PS0位，决定访问寄存器的哪一页；二是四条地址线RA0～RA3，决定访问某页中的哪一个寄存器；三是读写信号，决定对寄存器是读还是写操作。此外，还有两个寄存器不属于任何页，它们是DMA读写端口(地址为0x10)和Reset地址(地址为0x1f)。由此可见，要完成对RTL8019AS的寄存器访问，至少需要5条地址线(其它不用的地址线接地即可)、2条控制线(读写)和8根数据线。

在实现读写操作时，为了使控制简单，可以把地址使能信号AEN直接接地，使之始终有效，但这会降低系统的稳定性，IORB或IOWB信号线抖动时会产生错误信息，所以通常还是通过一条I/O引脚来控制AEN。另外，为使RTL8019AS复位，还需1条复位控制线。

由于GZ60通用I/O引脚较多，为提高通信速率，RTL8019AS与GZ60之间的数据通信可采用16位模式。

综上所述，GZ60 MCU只要提供16条数据线、5条地址线、4条控制线就可以完成对RTL8019AS的控制和数据通信。

2．GZ60 与 RTL8019AS 硬件连接

图3-7为在上述分析下设计的MC68HC908GZ60对RTL8019AS的控制连接图。

下面以RTL8019AS各引脚出发，分析网络控制芯片的硬件连接图。

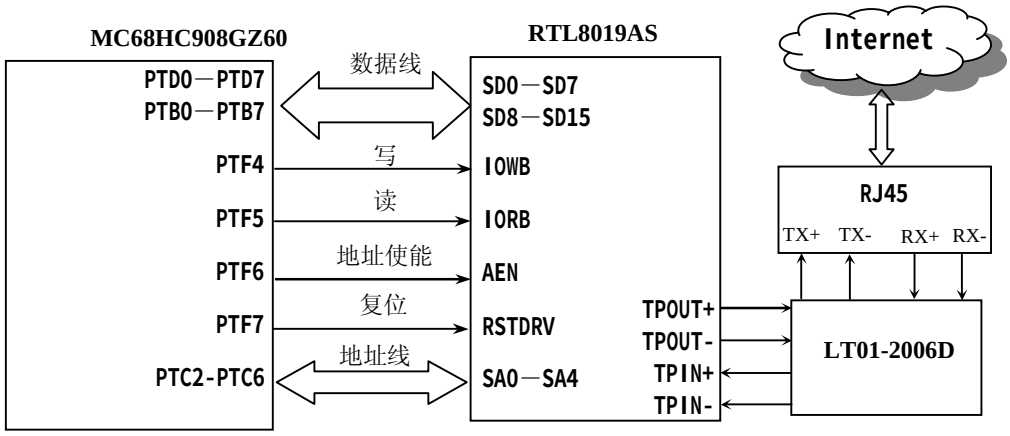


图 3-7 MC68HC908GZ60 与 RTL8019AS 连接图

【RSTDRV 引脚】：RTL8019AS 复位信号引脚。当该引脚置高电平超过 800ns，RTL8019AS 复位。该引脚与 GZ60 的 PTF7 相连；

【IOS3-0 引脚】:基地址选择引脚。这 4 个引脚接地时，IO 基地址为 300H；

【SA4~0 引脚】：地址线。根据 IOS3-0 基地址引脚的选择，IO 基地址为 300H，即 0011 0000 0000 ，所以地址线 SA9，SA8 接+5V。而寄存器地址偏移量为 00H~1FH 共 32 个(对应于 300H~31FH)，所以只需将地址线 SA0~4 接 GZ60 的 PTC2~6，其余地址线均接地；

【IOCSl6引脚】：数据通信模式选择引脚。在RSTDRV信号的下降沿，RTL8019AS 判断该引脚的电平高低，以决定采用8位还是16位数据总线模式。在本设计中采用16位数据模式，该引脚通过300Ω电阻上拉到+5V，为高电平；

【SD15~SD0 引脚】：16 根数据线；

【AEN 引脚】：地址使能引脚，为低电平时 I/O 操作有效，为高电平时 I/O 操作无效，与 GZ60 的 PTF6 相连；

【IORB 引脚】：I/O 读控制线，与 GZ60 的 PTF4 相连；

【IOWB 引脚】：I/O 写控制线，与 GZ60 的 PTF5 相连；

【AUI 引脚】：AUI 或 BNC 接口选择。如果是低电平，使用 BNC 接口，支持双绞线或同轴电缆。最常见的是采用双绞线为通信介质。在本设计中 AUI 接地；

【TPIN+、TPIN-、TPOUT+、TPOUT-引脚】：与藕合隔离变压器相连，利用 RJ45 插头实现与网络的连接；

【PL0，PLl引脚】：决定网络接口类型10BaseT，10Base2或10Base5。接地，保持低电平，设置为自动选择方式；

【OSC1，OSC2引脚】：接20MHZ晶振，过30pF的电容后接地，以减少干扰；

【LED0、 LED1、LED2引脚】：过1KΩ电阻，串接1个发光二极光接地。当向以太

网发送数据有冲突时，LED0闪烁；当RTL8019AS发送数据时，LED1闪烁；当数据到达RTL8019AS时，LED2闪烁；

【TPIN+、TPIN-引脚】:RTL8019AS差分信号接收引脚；

【TPOUT+、TPOUT-引脚】: RTL8019AS差分信号发送引脚。

为实现以太网上各节点的电气隔离，RTL8019AS通过网络隔离芯片LTL-2006再与外面的Internet相连接。LTL-2006是双绞线驱动/接收器，内部有2个耦合变压器，用来传输信号，同时抑制来自介质的共模噪声/干扰。

3．RTL8019AS 硬件连接测试

由于 RTL8019AS 的工作状态比较复杂，并且需要主控芯片的控制信号，所以测试时要分步进行，确保上一步正常后再进行下一步的测试工作。其测试流程如下：

(1) 用交叉网线将实验板与 PC 机连起来，并给实验板上电，看 PC 机是否能检测到有 10M 以太网卡与它相连。若没有，表示网络接口硬件上有故障。参考电路图，检查各引脚的电平状态，确定相关引脚是否满足电路图的要求；

(2) 根据 RTL8019AS ISA总线的读写时序，完成主控芯片对 RTL8019AS 的寄存器正确读写。读写时序如图 3-8 所示；

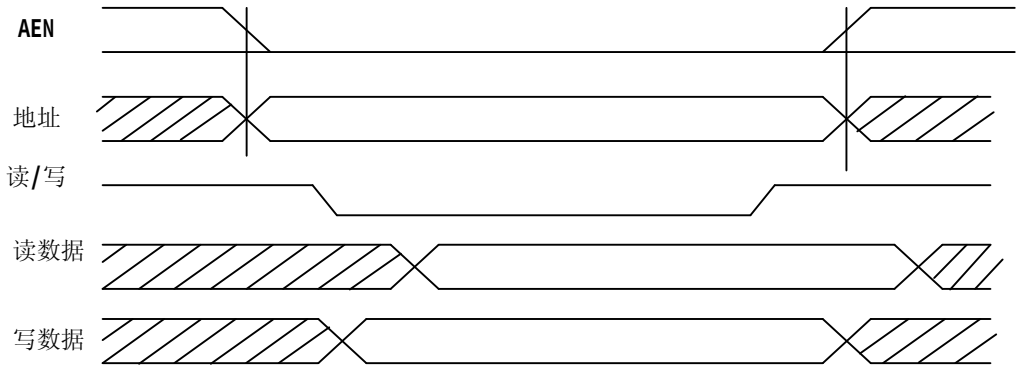


图 3-8 以太网芯片 I/O 读写时序

(3) 初始化 RTL8019AS，配置 MAC 地址和 IP 地址；

(4) 从 PC 机 Ping 这个 IP 地址，观察 RTL8019AS 的接收指示亮是否闪烁，闪烁表示正在收到数据；

(5) 向 RTL8019AS 的发送缓冲区中写入数据并启动发送命令，观察发送信包指示亮是否闪烁，闪烁表示正在发送数据；

(6) 实验板发送 ARP 请求，接收 PC 机发来的 ARP 响应，并将响应报文通过串口发往 PC 以检测以太网接口一次完整的收发操作。

3.7 CAN-以太网网关硬件设计及测试

1．网关硬件设计

单独的 CAN 接口和以太网接口实现后，要实现 CAN-以太网网关，硬件上只需将这两个接口结合起来即可。图 3-9 为包含支撑电路、SCI 电路、CAN 接口电路和以太网接口电路的 CAN-以太网网关硬件连接框图。

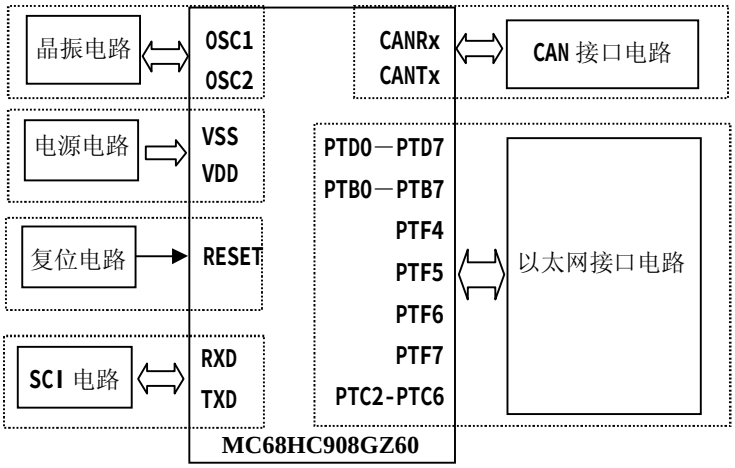


图 3-9 CAN 与以太网互连系统硬件结构框图

2．网关测试

网关硬件设计完成后，先按上述方式单独测试 CAN 接口和以太网接口，在各自硬件工作正常的前提下，再测试网关的协议转换功能。测试网关硬件上连接是否正确，可将网关通过 DB9 口连到 CAN 总线中，再用交叉网线将网关与 PC 机相连。完成网关有关模块初始化之后，PC 机通过网关向 CAN 总线中的另一个节点发送数据，该节点收到数据后通过串口再将它发往 PC 机比较。

第四章 互连系统软件设计

在前一章中重点阐述了互连系统的硬件设计，接下来重点讲述系统的软件设计。实际上，在实验过程中，软硬件设计是交互进行的，任何硬件模块的设计正确与否，都需要相应的软件来验证；当不能达到软件的预期目的时，需要改进硬件设计，以排除一些不合理的硬件设计。本章从用户监控程序的编写开始，逐步深入到各通信接口的软件设计，最后阐述协议网关的软件实现。相应程序的测试将在第六章中进行说明。

软件设计中首先应确定开发语言。由于C语言可以在计算机上仿真，用C语言编制的程序具有很好的可移植性、程序易读易修改、便于程序调试、便于使用实时操作系统、有丰富的库函数等诸多优点，C语言已成为开发嵌入式应用软件的有力工具。但是，在嵌入式应用系统中程序需要与硬件直接打交道，而C语言是与硬件无关的通用程序设计语言，因而，与硬件有关的部分如初始化堆栈指针还是必须用汇编编写；另外，时序要求严格的场合也必须用汇编编写。根据嵌入式应用系统的特点，作者采用Motorola汇编和ImageCraft C共同编写系统的软件部分，用汇编编写与硬件有关的部分，而用C语言编写与硬件无关的部分。

4.1 MC68HC908GZ60用户监控程序

监控程序的主要功能是对应用系统硬件及底层软件进行调试[34]，是最基本的调试工具。监控程序功能不足会给应用程序的开发调试带来麻烦，但是，若用大量精力研究监控程序，则主次不分。监控程序最重要的功能是通过串行口将程序下载到单片机的存储器中，因此，程序下载是监控程序必须实现的；而程序的调试功能可以通过在程序运行时，在欲调试处设断点，并将断点处的有关寄存器值、变量值通

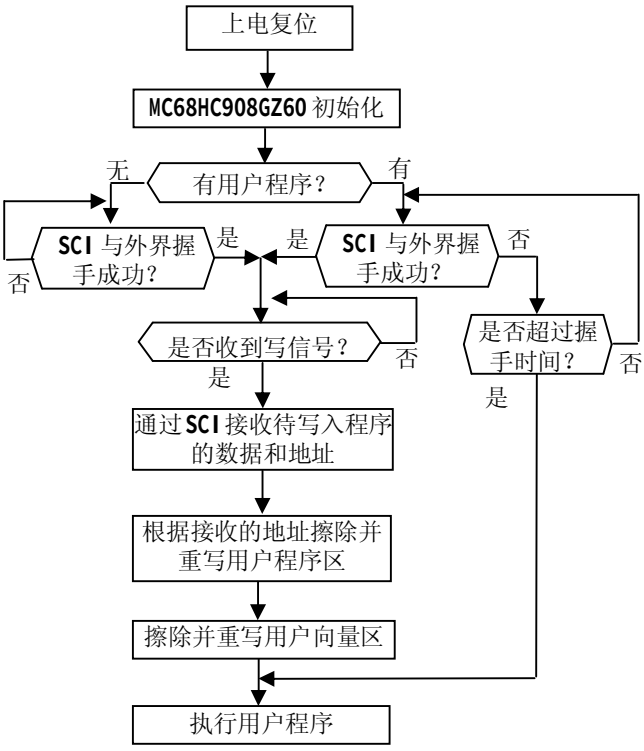


图 4-1 用户监控程序的实现流程图

过串行口发出。根据这一思想，此处MC68HC908GZ60的用户监控程序只实现了最基本的程序下载功能，从而取代了条件苛刻的监控模式程序下载方式。待应用程序定版后，可将监控程序擦除。监控程序的执行流程见图4-1。

4.2 MSCAN08模块软件设计

MSCAN08模块编程主要通过读写该模块提供的13个寄存器和4个CAN报文存储缓冲区，实现MSCAN08的初始化、报文接收和报文发送。

4.2.1 初始化MSCAN08

1．初始化过程

MSCAN08初始化主要对MSCAN08工作方式、接收滤波方式、接收屏蔽器和接收代码寄存器、波特率等有关参数进行设置。MSCAN08初始化子程序部分代码如下：

```
//-------------------------------------------------------//
//程序名：void CANInit(void)
//功能：CAN初始化
//入口：无
//出口：无
//说明：在2M CAN总线频率下将总线速率设为200kbps，采用标准帧格式，
//      节点标识符设为10(即0b00000001,010)
//-------------------------------------------------------//
void CANInit(void)
{
   CMCR0|=(1<<SFTRES);  //模式控制寄存器的软复位标志为1
   CMCR1&=~(1<<CLKSRC); //MSCAN08时钟源选择外部晶振(2分频)
   CIDAR0=0b00000001;   //本地标识符设为10
   CIDAR1=0b10100000;   //低5位无效
   CIDMR0=0b00000000;   //11位屏蔽位均有效
   CIDMR1=0b00011111;
   CIDAC=0b00000000;    //单验收滤波器模式
   CBTR0=0b00000000;    //将位速率设置成200kbps，单采样方式
   CBTR1=0b00100101;
   CRIER=0b00000001;    //接收到数据允许产生中断
```

```
    CTCR=0b00000000;       //发送缓冲区空不产生中断
    CMCR0&=~(1<<SFTRES); //正常模式
}
```

2．初始化说明

(1) 在进行寄存器赋值操作时，**MSCAN08**规定必须在软复位状态下才能进行写操作，故必须先将控制寄存器**0**中的软复位标志置成**1**，即**CMCR0|=(1<<SFTRES)**；

(2) MSCAN08模块的时钟源可以选用外部晶振，也可以采用**PLL**锁相环输出，频率相同时外部晶振比**PLL**稳定。此处选用外部晶振；

(3) 在**CAN**总线中传输的报文不指明目的地址，而是采用生产者**-**消费者模式，即只要节点的接收代码寄存器中的值与总线上报文标识符一致，该节点就将报文接收下来。**CAN2.0B**协议中定义了两种标识符格式，标准格式和扩展格式。其中，标准格式为**11**位标识符，而扩展格式为**29**位标识符。用户可以根据实际节点数目，选用标准格式或扩展格式。若采用标准格式，则只使用前**2**个代码寄存器；若采用扩展格式，则需使用**4**个代码寄存器。在本初始化中，采用标准格式，设该节点标识符为**10**，故给**CIDAR0-1**分别赋**0b00000001**和**0b01000000**；

(4) 报文标识符与代码寄存器中的内容逐位进行比较后，会得到一个“匹配”或“不匹配”的结果，然而，该报文最终接收与否，还受屏蔽寄存器设置的影响。当屏蔽寄存器某位为**1(**即“无关”**)**时，标识符比较的结果并不影响报文的接收；当屏蔽寄存器某位为**0(**即“相关”**)**时，则只有“匹配”的报文才会被接收。由此可见，报文最终能否接收，受两个因素影响，一是代码接收寄存器的值，二是屏蔽寄存器的值。图**4-2**是报文验收过程示意图；

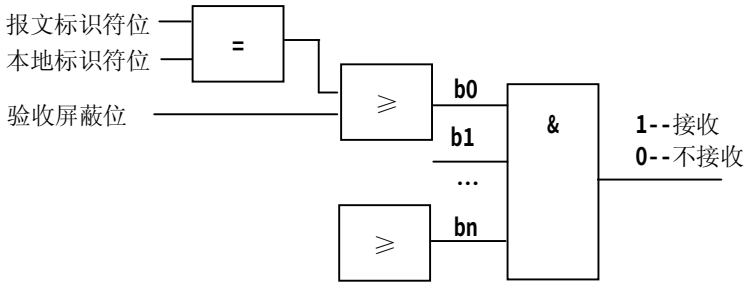


图 4-2 报文验收过程示意图

(5) CAN总线中的通信速率是可以通过软件设置的，但在一个固定的**CAN**总线中，所有**CAN**节点的通信速率必须一致。总线上传输一位**(bit)**的时间长短决定了**CAN**总线的通信速率。**CAN**总线的位时间各组成部分如图**4-3**所示。

从图**4-3**可以看出，位时间由同步段、时间段**1**和时间段**2**组成，而时间份额**(Time Quanta)**由**MSCAN08**时钟频率和预分频因子共同决定。

例如：在MSCAN08时钟频率fMSCAN08=2M时，

CBTR0=0b00000000，CBTR1=0b00100101，

则：预分频因子PreScale=1，时间份额Tq= $f_{MSCAN08}$/PreScale = 2M，

同步跳转宽度=1(Tq)，时间段1=0b0101=6(Tq)，TSEG2=0b010=3(Tq)，

位时间=1+6+3=10(Tq)，故通信波特率=2M/10=200(kbps)

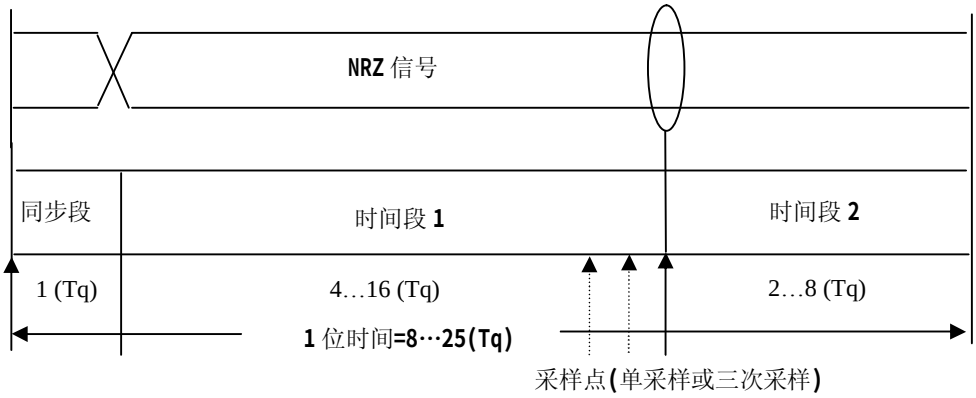


图 4-3　CAN 通信位时间总体结构图

4.2.2 发送CAN协议报文

1．MSCAN08的发送缓冲区结构

当某时刻有超过 1 帧数据需要发送时，为保证报文连续发送，即在节点获得了总线发送权后，在发送完前一帧时，不会被标识符优先级比自己高的节点剥夺总线发送权，MSCAN08 模块提供了三个发送缓冲区。所有缓冲区结构都相同，每个报文缓冲区均分配 16 字节存储空间，其中，报文数据结构占 13 字节。报文缓冲区的组织如图 4-4 所示。报文缓冲区首先存放 4 字节报文标识符，当采用标准帧格式时，标识符寄存器 IDR0 和 IDR1 有效，IDR2 和 IDR3 无效；当采用扩展帧格式，IDR0-4 均有效。标识符后面是最多 8 字节报文数据，具体有效数据长度由数据长度寄存器 DLR 的低 5 位指定。发送缓冲区优先级寄存器则规定了该报文的发送级别，当不只 1 个报文等待发送时，MSCAN08 先发送优先级别最高的报文。

地址(1)	寄存器名
$05x0	标识符寄存器 0(IDR0)
$05x1	标识符寄存器 1(IDR1)
$05x2	标识符寄存器 2(IDR2)
$05x3	标识符寄存器 3(IDR3)
$05x4	数据段寄存器 0(DSR0)
$05x5	数据段寄存器 1(DSR1
$05x6	数据段寄存器 2(DSR2)
$05x7	数据段寄存器 3(DSR3)
$05x8	数据段寄存器 4(DSR4)
$05x9	数据段寄存器 5(DSR5)
$05xA	数据段寄存器 6(DSR6)
$05xB	数据段寄存器 7(DSR7)
$05xC	数据长度寄存器(DLR)
$05xD	发送缓冲区优先级寄存器
$05xE	未用
$05xF	未用

说明：(1) 发送缓冲区 0，x 取值 5
发送缓冲区 1，x 取值 6
发送缓冲区 2，x 取值 7

图 4-4 报文缓冲区组织图

2．CAN 协议报文的发送过程

报文的发送由 MSCAN08 根据 CAN 协议规范自动完成。当 MCU 需要发送数据时，必

须先将数据按 CAN 协议格式进行封装，再将 CAN 报文存入空闲发送缓冲区，并清除相应缓冲区空标志。MSCAN08 模块根据 MCU 的清标志操作知道待发数据已经准备好，可以将发送缓冲区中的数据发往 CAN 总线。报文的发送既可以采用查询方式也可以采用中断方式。报文在被成功发送出去之前 MCU 可以通过发送“中止发送”请求以中止当前报文的发送，利用这一机制可以实现一个紧急报文的优先发送。发送一个报文的流程见图 4-5。

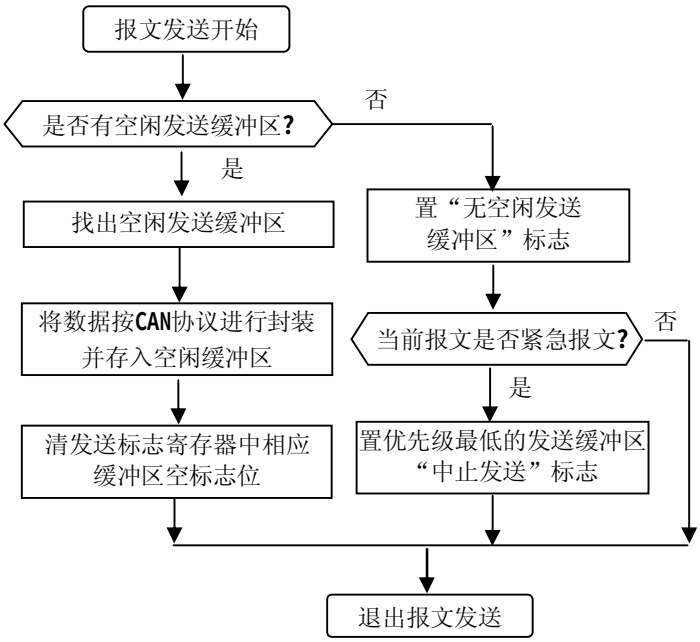


图 4-5 发送一个 CAN 协议报文的流程图

4.2.3 接收CAN协议报文

当 CAN 总线上有报文在传输时，总线上除发送结点以外的所有节点都将成为接收节点；接收节点将报文暂时存放在本地 MSCAN08 的后台(Background)接收缓冲区 RxBG 中并进行滤波比较，即将该报文的标识符与本地节点标识符进行比较；若该报文通过本地滤波，MSCAN08 则将此报文复制到前台(Foreground)接收缓冲区 RxFG 中，并置接收缓冲区满标志位(RXF)通知 MCU 去读取 RxFG 中的数据。MCU 只能访问 RxFG 而不能访问 RxBG。

MCU 在查询到 RXF 标志为 1(查询接收方式)或收到 MSCAN08 模块产生的接收中断后，便到接收缓冲区中读取数据进行分析，并作相应处理。报文接收中断处理流程如图 4-6 所示。

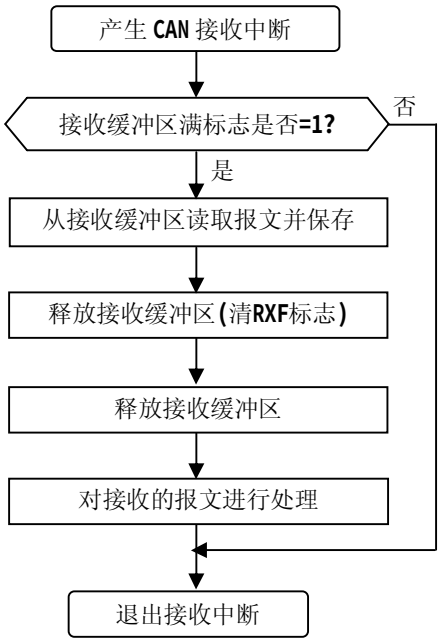


图4-6 CAN报文接收处理流程

4.3 以太网接口软件设计

4.3.1 RTL8019AS驱动程序设计

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。

1．硬件初始化

(1) RTL8019AS 的初始化过程描述

硬件初始化主要完成 **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、本地 **MAC** 地址的设置等。初始化过程描述如下：

① 复位 **RTL8019AS**：**RTL8019AS**的 **RSTDRV** 引脚为高电平有效，且至少需要 **800ns** 的宽度。可按如下步骤将 **RTL8019AS** 复位：主控芯片向 **PTF4** 输出高电平**(PTF4** 与 **RSTDRV** 引脚硬件上相连**)**该 **RSTDRV** 引脚施加一个高电平，为确保完全复位，可让高电平持续 **10ms** 左右；然后施加一个低电平，等待 **2ms** 左右，复位结束；

② 配置发送缓冲区首地址：令 **TPSR=0x40**，表示当 **MCU** 要向网络发送数据时，**RTL8019AS** 先将该数据暂存在页 **0x40** 开始的缓冲区中等待发送；

③ 配置接收缓冲区环：令 **PSTART=0x4C**、**PSTOP=0x80**，表示接收缓冲区环为 **0x4C~0x7F**；

④ 初始化缓冲区环维护指针：令 **BNRY=0x4C** 表示主控制芯片**(GZ60)**下次要读的页为 **0x4C(BNRY)**，该寄存器值由主控芯片修改；令 **CURR=0x4C**，表示下次当 **RTL8019AS** 从网络上接收到以太帧后，将它存放在 **0x4C(CURR)**开始的页，该寄存器值由 **RTL8019AS** 内部修改；

⑤ 设置 **RTL8019AS** 的收发模式：令 **RCR=0xCC**，表示使用接收缓冲区、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** 字节或检验错的数据包；令 **TCR=0xE0** 表示启用 **CRC** 自动生成和自动校验、工作在正常模式；令 **DCR=0xC9**，表示使用 **FIFO** 缓存、普通模式、**16** 位数据 **DMA**；令 **IMR=0x00**，表示屏蔽所有中断；

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**，清除中断状态寄存器所有位；

⑨ 配置完成，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

注：在读写寄存器时，需先写 **CR** 以选择欲操作的寄存器所在的页，再进行读写操作。例如：在读写 **PSTART** 时，由于 **PSTART** 在第 **0** 页，因此，需先令 **CR=0x21**，再

执行读写操作。寄存器分配参见附录B.2。

程序中的语句主要是读写有关寄存器。下面以选择页0(往命令寄存器CR写0x21)为例介绍写寄存器的操作。其C语言程序如下：

```
//=========================================================//
//程序名：void writeRTL(unsigned char rtl_addr, unsigned char data)
//功能：向RTL8019某个寄存器中写入数据
//入口: 1. unsigned char rtl_addr - 寄存器偏移地址
//      2. unsigned char data - 写到寄存器的值
//出口：无
//说明：需用到的控制信号为：读控制IOWB，写控制IORB，使能线AEN
//备注: 地址线只用了 5 根，在输出地址信号时，要注意程序中地址是否
//       与硬件实际连线相符。
//=========================================================//
void writeRTL(unsigned char rtl_addr, unsigned char data)
{
   asm("sei");                  //关中断，因此处操作与时序有关
   asm("sta $1800");            //看门狗喂食
   DataPortDD=0xff;             //数据口输出
   CtrlPortDD |=(1<<IOWB) ;     //CtrlPort.IOWB输出
   CtrlPortDD |=(1<<IORB) ;     //CtrlPort.IORB输出
   CtrlPortDD |=(1<<AEN);       //CtrlPort.AEN输出
   AddrPortDD |= 0xF8;          //地址口输出
   rtl_addr =( rtl_addr << 3);//地址左移3位是因为地址线为高5位
   CtrlPortD |=(1<<IOWB) ;      //置CtrlPort.IOWB(0)为高
   CtrlPortD&=~(1<<AEN);        //CtrlPort.AEN(2)  低(有效)
   AddrPortD= rtl_addr;         //地址上线
   CtrlPortD&=~(1<<IOWB);       //清CtrlPort.IOWB(0)为低(有效)
   DataPortD=data;              //数据上线
   asm("nop");
   CtrlPortD|=(1<<IOWB);        //CtrlPort.IOWB(0) 高
   CtrlPortD|=(1<<AEN);         //CtrlPort.AEN(2) 高
   asm("cli");                  //开中断
```

```
}
```

读寄存器的操作与写操作相似，只是读写控制信号不同而已。

对于数据通信(远程 DMA)操作，由于采用 16 位模式，故跟寄存器读写操作略有不同。下面是以写数据端口(地址 0x10)为例介绍远程 DMA 的写操作：

```
//=========================================================//
//程序名：void writeDMA(unsigned char addr, unsigned char dataL,
//        unsigned char dataH)
//功能：将一个字(2 字节)写入 RTL8019 远程 DMA
//入口：1 unsigned char addr  -  远程 DMA 端口(0x100)
//      2 unsigned char dataL -  写到远程 DMA 端口的低字节
//      3 unsigned char dataH -  写到远程 DMA 端口的高字节
//出口：无
//调用举例：writeDMA(RDMAPORT,localBuffer[i],localBuffer[i+1]);
//说明：写 RTL8019 远程 DMA 时，为提高通信速度,采用 16 位 DMA 方式
//=========================================================//
void writeDMA(unsigned char addr, unsigned char dataL,
        unsigned char dataH)
{
   asm("sei");                  //关中断，因此处操作与时序有关
   asm("sta $1800");            //看门狗喂食
   DataPortDD=0xff;             //数据口输出
   HDataPortDD=0xff;            //高字节数据口输出
   HDataPortD = dataH;          //高字节数据上线
   DataPortD = dataL;           //低字节数据上线
   CtrlPortDD |=(1<<IOWB);      //CtrlPort.IOWB 输出
   CtrlPortDD |=(1<<IORB);      //CtrlPort.IORB 输出
   CtrlPortDD |=(1<<AEN);       //CtrlPort.AEN  输出
   AddrPortDD |= 0xF8;          //地址口输出
   addr=(addr << 3);            //地址左移 3 位是因为地址线为高 5 位
   CtrlPortD |=(1<<IOWB) ;      //CtrlPort.IOWB(0) 高
   CtrlPortD&=~(1<<AEN);        //CtrlPort.AEN(2)  低(有效)
   AddrPortD=addr;              //地址上线
```

```
    CtrlPortD&=~(1<<IOWB) ;     //CtrlPort.IOWB(0) 低(有效)
    asm("nop");
    CtrlPortD|=(1<<IOWB);       //CtrlPort.IOWB(0) 高
    CtrlPortD|=(1<<AEN);        //CtrlPort.AEN(2) 高
    asm("cli");                 //开中断
  }
```

(2) RTL8019AS 的内存缓冲区的初始化设置说明

在上述初始化过程中,构造了接收缓冲环(初始化的第 4 步),这涉及到 RTL8019AS 的 RAM 结构。

RTL8019AS 内部有两块存储区，一块是用于存放收发信包的 RAM 区，地址为 0x4000~0x7FFF，共 16K；RAM 区按页存储，每 256 字节为 1 页，16 位 RAM 地址的高 8 位又称页码,从页 0x40 到页 0x7F，共 64 页。通常将从 0x4000~0x4BFF 的 12 页作为发送缓冲区，从 0x4C00~0x7FFF 的 52 页作为接收缓冲区。另一块是用于存放以太网物理地址的 PROM，地址为 0x0000~0x00FF，但只有前 32 字节可被访问，且通常不用。图 4-7 为 RTL8019AS 的内存空间结构[35]。

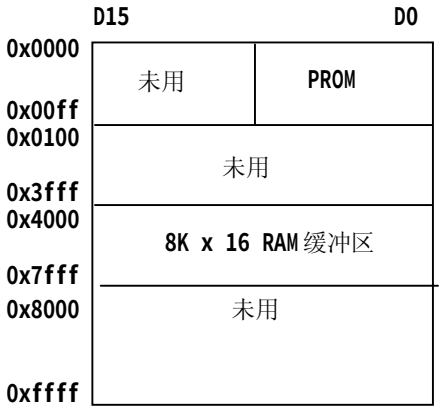


图 4-7 RTL8019AS 存储空间结构图

通过写 TBSR(Transmit Page Start Register，发送页起始寄存器)可以设置发送缓冲的起始页，此处，往 TBSR 写入 0x40；一个以太帧最长为 1518 字节(1500 字节数据+14 字节头+4 字节校验)，因此，一帧最多需要 1518/256=5.92(页)，即需要 6 页。此处分配 12 页作为发送缓冲区，因而可以存放两个最大的以太帧。前 6 页 0x40~0x45 作为发送缓冲 1，后 6 页 0x46～0x4B 作为发送缓冲 2。用户可以将以太帧放在发送缓冲 1，然后启动发送。在 RTL8019AS 发送数据的过程中，若用户又有数据需要发送，则可将要发的数据包放入发送缓冲 2(不影响发送缓冲区 1 的处理)，一旦发送缓冲 1 的数据包发送完就可以马上启动发送缓冲 2 中的数据包。这样网络控制芯片就可以不间断地进行数据发送，从而提高数据包的发送速度。

通过设置 PSTART(Page START Register，页起始寄存器)、PSTOP(Page STOP Register，页结束寄存器) 可设置接收缓冲区的范围。此处，设置 PSTART=0x4C，PSTOP=0x80，这表示将使用 0x4C00~0x7FFF 的 RAM 区来存储从以太网中接收的数据包。需指出的是，PSTOP 应设成 0x80 而不是 0x7F。PSTOP 的意思是，从该页开始的页不

能作为接收缓冲区，而 PSTART 的意思是从这一页开始作为接收缓冲区。RTL8019AS 接收缓冲环示意图如图 4-8 所示。

在设置了接收缓冲区之后,微控制器(MC68HC908GZ60)如何从 RTL8019AS 读取数据呢？这由两个寄存器决定：当前页寄存器 CURR 和边界寄存器 BNRY。CURR 是网卡写缓冲区的指针，指向当 RTL8019AS 从网络接收到数据时要写的页；BNRY 是读指针，指向用户下一次要读的页。在上面的初始化中，CURR=0x4C，BNRY=0x4C，意思是 RTL8019AS 将把第一个收到的数据包存放在 0x4C 开始的页缓冲区中，用户下一次要读取数据则是 4C 页的内容。

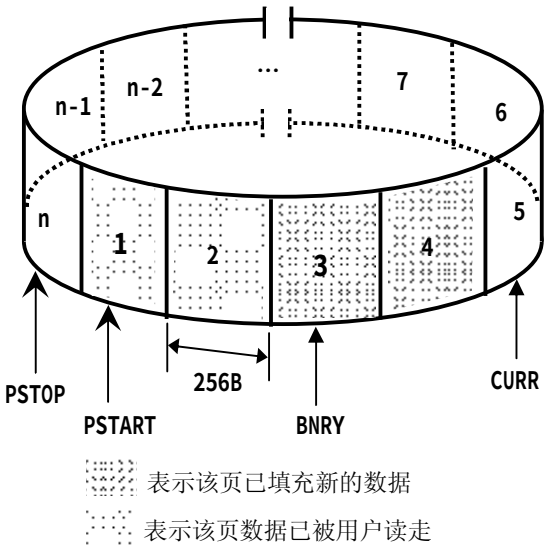


图 4-8 RTL8019AS 接收缓冲环示意图

2．发送以太帧

在实现以太帧的收发操作之前，必须先了解 RTL8019AS 是如何实现微控制器与网络之间的数据交换的。RTL8019AS 提供了本地 DMA 通道和远程 DMA 通道。本地 DMA 通道负责本地缓冲区和 FIFO 之间的数据传输；一方面实现两者之间在发送和接收帧时以字节或字方式的数据传输；另一方面在发生发送帧碰撞时，可在处理器不介入的情况下自动重发。而远程 DMA 通道负责本地缓冲区和处理器内存之间的以字或字节方式下的数据传输。

因此，当有数据需要发送到以太网中时，微处理器将待发送的数据依次按 TCP 报文格式、IP 报文格式、以太帧格式进行封装，再通过 I/O 通道和 RTL8019AS 的远程 DMA 通道将数据写入 RTL8019AS 的本地发送缓冲区，最后发送传输命令让 RTL8019AS 将以太帧通过 FIFO 发送到网线上，由接收方接收。由于以太帧的长度需在 64~1518 字节之间，故当长度小于 46 时，需要填充一些无用的数据；而当超过 1518 字节时，需要拆分成多个以太帧再发送。详细发送过程如下：

首先，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。远程字节计数寄存器 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。

其次，设置有关寄存器，指定发送帧的起始地址和帧长度。发送寄存器 TPSR 用来指定待发送帧的起始地址，传输字节计数寄存器 TBCR0 和 TBCR1 用来指明待发送帧的长度。帧长度是以字节数表示的整个帧长度，是目的地址、源地址、类型数据长度字段及数据域的长度之和。

最后，通过设置 CR 寄存器中的 RD2，RD1，RD0 这 3 个位来启动远程写操作和发送数据帧的操作。在发送完毕，测试发送状态寄存器 TSR 中的各标志位来检验数据帧是否已正确无误地发送完成。发送流程如图 4-9 所示。

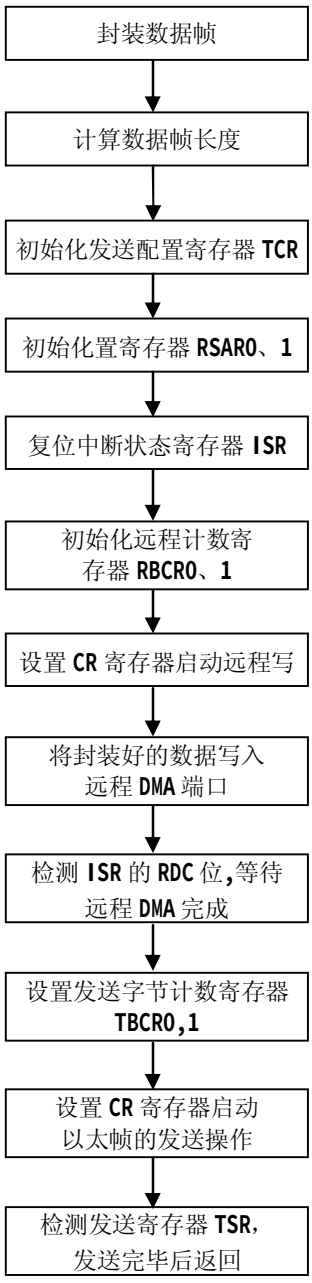


图 4-9 发送帧的流程

3．接收以太帧

(1) 接收缓冲环的维护

在上述初始化程序中，已经通过 PSTART 和 PSTOP 分配了 52 页的本地 DMA 接收缓冲区环，并且当前写指针 CURRT 和边界寄存器 BNRY 都赋成 0x4C。当 RTL8019AS 收到一个数据包时，根据数据包的大小，计算出所需的页数，将数据包存放在以 CURR 指向的页为首页的若干个连续的页中，并且 CURR 后移相应页数。例如：当 CURR=0x4C 时，若收到一个 300 字节的数据包时，则需分配 0x4C 和 0x4D 对应的页缓冲区给该包，且 CURR 自动变为 0x4E(=0x4C+2)，即下一个数据包到达时，从 0x4E 页开始存放。RTL8019AS 存储时是按页存储的，不满一页，也需占用一页，下一个数据包将用下一页开始存储。如果 CURR 超过结束页 PSTOP，也就是 CURR>0x7F 时，CURR 将被重新设置成 PSTART=0x4C。

初始化后，假如收到 1 个小于 256 字节的数据包（该数据包只需 1 页来保存），则 CURR=0x4C+1=0x4D，BNRY=CURR-1，而不是 BNRY=CURR，两个指针相差了 1 页，也就是说当 CURR ，BNRY两个指针差 1 页或 1 页以上时，表示 RTL8019AS 收到了新的数据包。编程时正是根据这个条件来判断是否有新的数据包到达。

(2) 以太帧接收过程描述

当网络上有数据包到达时，RTL8019AS 接收逻辑将数据包进行地址比较和 CRC 检验后，通过本地 DMA 通道将接收到的数据帧缓存在 CURR 所指向的页接收缓冲区中，

并修改 CURR 寄存器值。主控芯片(GZ60)查询到 CURR 与 BNRY 不相等时，通过远程 DMA 通道将接收缓冲环中的数据包读入主控芯片的存储区内以供用户程序读取。

在从 RTL8019AS 接收缓冲环中读取数据之前，需先了解数据即以太帧在缓冲区中的存放格式。表 4-1 为总线上传输的以太帧格式，其中，前导位、帧起始位和检验由硬件自动添加/删除，与上层协议无关，因此以太网驱动程序只需要处理其他字段。表 4-2 为 RTL8019AS 接收帧格式[36]。接收帧格式在以太帧的前面添加了接收状态、下页指针和以太网帧长度三个字段，但不包括以太帧格式中的前导位和帧起始位。表 4-2 中各字段含义如下：

表 4-1 总线上传输的以太帧格式

62bit	2bit	48 位	48bit	16bit	<=1500byte	X byte	32bit
前导位	帧起始位	目的 MAC 地址	源 MAC 地址	类型	数据域	数据域小于 46 字节时的填充	CRC 校验

表 4-2 RTL8019AS 接收帧格式

8bit	8bit	16bit	48 位	48bit	16bit	<=1500byte	X byte	32bit
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	数据域小于 46 字节时的填充	CRC 校验

【接收状态】：1 字节，内容为接收状态寄存器(RSR)的值。若某时刻该字段值为 0x21，表示数据包被正确接收，且为广播包；

【下一页指针】：1 字节，指示下一个以太帧存放页地址。若某时刻该字段值为 0x50，表示下一个数据包将存储在 0x50 页开始的地址，即 0x5000；

【帧长度】：2 字节，存放本帧数据长度，按照先低字节再高字节存放。若某时刻该字段内容为 0x4A00，则表示以太帧的长度为 74(0x004A)字节。接收程序在从 RTL8019AS 读数据时，可先预读 4 个字节，取得该数据帧长度后，再把完整的帧读入；

【目的 MAC 地址】：6 字节，为数据接收方的硬件地址；

【源 MAC 地址】：6 字节，为数据发送方的硬件地址；

【类型】：2 字节，为以太帧的协议类型。当类型字段=0x0806 时，表示报文数据部分是 ARP 报文，而当类型字段=0x0800，表示报文数据部分是 IP 报文；

【数据域】：最大不超过 1500 字节，为以太帧的数据部分；

【填充】：若干字节，当数据域小于 46 字节时，需填 0，以使以太帧长度最少为 64 字节；

【CRC 检验】：4 字节，从目的 MAC 地址开始的数据进行循环冗余检验的结果。

RTL8019AS 在接收完网络中的一个数据包时，将中断状态寄存器 ISR 中的无错接收标志 PRX 置 1。若设置接收中断允许，即中断屏蔽寄存器 IMR 的 PRXE 标志位置 1

时，将产生一个中断信号。微控制器捕获到该中断信号后，可以立即从 RTL8019AS 接收缓冲区中读取数据；若未设置接收中断允许，微控制器则只能通过查询 ISR 中的 PRX 位或比较 CURR 与 BNRY 来确定 RTL8019AS 是否接收到数据。

微控制器接需要从 RTL8019AS 读取数据时，可以通过 RTL8019AS 的远程 DMA 读取数据。CR 寄存器中的 RD2，RD1，RD0 这 3 个位组合起来设定 DMA 的操作。“001”启动远程读操作；“010”启动远程写操作；“011”发送网卡数据包；“1**”终止或结束 DMA 的读写操作。微控制器读取完当前数据包后，必须修改边界寄存器 BNRY 的值，以便为下一次读数据包做准备工作。当远程 DMA 读操作完成后，RTL8019AS 将中断状态寄存器 ISR 中的 RDC 位置 0，微控制器可通过此标志位的状态可判别远程 DMA 读操作是否结束。在此处以太网接口的实现方案中，采用查询方式接收数据帧，接收流程如图 4-10 所示。

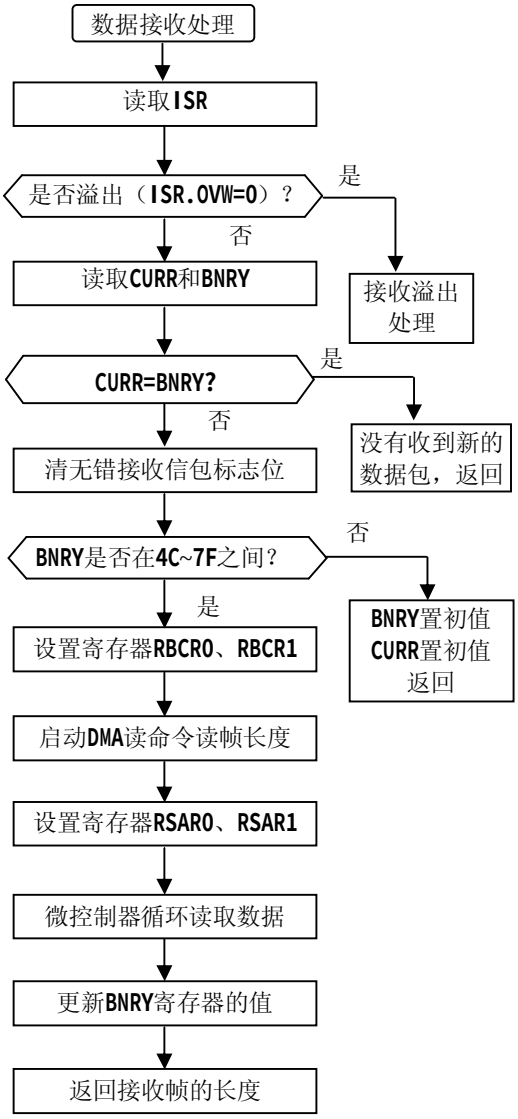


图 4-10 接收帧的流程图

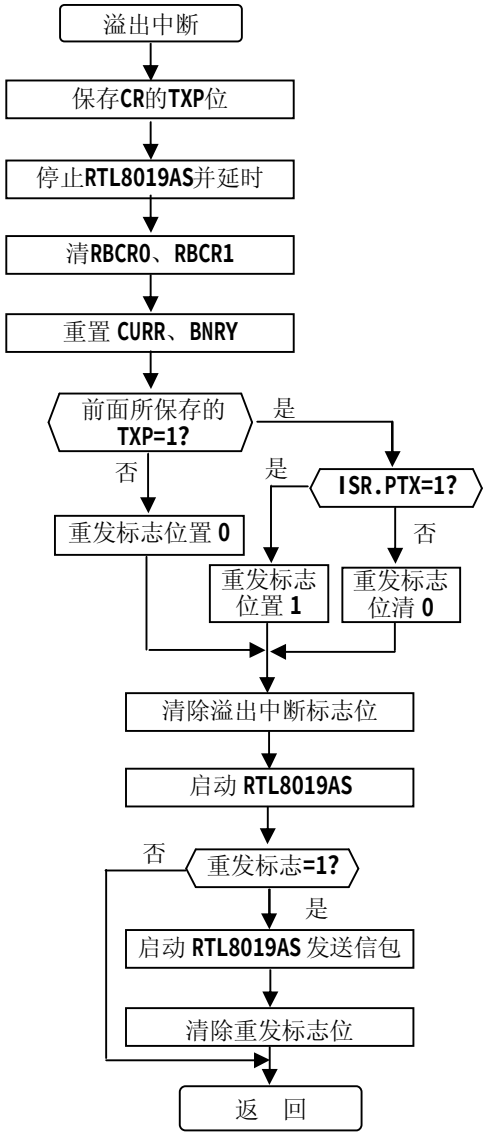


图 4-11 溢出处理流程图

(3) 数据接收溢出处理

当网络上数据流量太大或微处理器未及时将网络控制芯片接收缓冲区内的数据取走时，会导致数据接收溢出，表现为 CURR 指针追上了 BNRY 指针，因而接收缓冲环遭到破坏，此时 RTL8019AS 会中止接收数据，并且产生溢出(ISR 寄存器的 OVW 位置 1)。当出现这种情况下时，微控制器必须进行溢出处理，否则 RTL8019AS 将进入不可预知的工作方式。溢出处理的方法是先中止 RTL8019AS 当前的工作，再清除远程计数寄存器(RBCR0、RBCR1)、重置 CURR 和 BNRY 指针、清除溢出标志位，最后重新启动 RTL8019AS 进入正常工作模式。需要指出的是，中止 RTL8019AS 工作可能使得正在进行的数据包发送工作被中止，因此，在溢出处理的开始应该要检查当前是否有数据包正在发送，若有，则在溢出处理结束后，应该重新发送数据包。溢出处理的程序流程如图 4-11 所示。

4.3.2 uIP协议栈

uIP 协议通常都包括 TCP/IP 协议组中以下几个基本的协议：ARP、IP、ICMP 和 TCP。但在实际应用中，由于应用需求不同，相应地，对 TCP/IP 的裁减也有所不同，因此，得到的 uIP 并不都完全一样。下面较为详细地介绍 ARP、IP、ICMP 和 UDP 等 uIP 协议的软件实现过程。

1．ARP 协议

要实现网络通信，仅有网络地址即 IP 地址还不够，因为以太网硬件并不理解 IP 地址，它有自己独特的寻址模式，这种寻址模块是基于每个网络适配器惟一的 6 字节地址，即通常说的媒体访问控制(MAC)地址[38]。因此，TCP/IP 协议栈需要具有将 IP 地址转换为 MAC 地址的功能。ARP 协议便是为实现这一功能而制定的。

ARP 协议主要工作流程如下：当通信的一方(称之为 A)需要与另一方(称之为 B)通信，但不知道 B 的 MAC 地址时，A 先向 B 发送 ARP 请求信包；B 接收到 A 的 ARP 请求后，发送一个带有自己 MAC 地址的 ARP 响应信包；A 收到 B 的 ARP 响应信包后更新本地路由表。从该工作流程中可以看出，ARP 协议需要处理 2 两种信包：ARP 请求和 ARP 响应。ARP 请求用于根据 IP 地址索取 MAC 地址，而 ARP 响应则用于向 ARP 请求者提供自己的 MAC 地址。

(1) 发送 ARP 请求报文

当 A 需要向 B 发送 ARP 请求时，须先按 ARP 报文格式生成一个 ARP 请求报文，再按 Ethernet(以太帧)格式生成一个以太帧，最后通过网络发往 B。ARP 报文格式见表 4-3，各字段含义如下：

硬件类型：发送者网络接口类型，1 表示是以太网卡；

上层协议类型：网络层协议类型，0x0800 表示是 IP 协议；

硬件地址长度：查询物理地址的字节长度，以太网卡长度为 6；

上层协议地址长度：查询上层协议地址的字节长度，IPv4 时为 4；

操作号：表示报文要进行的操作，1 为 ARP 请求，2 为 ARP 响应，3 为 RARP 请求，4 为 RARP 响应；

源 MAC 地址：报文发送者硬件地址；

源 IP 地址：报文发送者 IP 地址；

目的硬件地址：对方硬件地址；

目的 IP 地址：对方 IP 地址。

表4-3 ARP报文格式

<table>
<tr><td colspan="2">硬件类型(2B)</td><td>上层协议类型(2B)</td></tr>
<tr><td>硬件地址长度(1B)</td><td>上层协议地址长度(1B)</td><td>操作(2B)</td></tr>
<tr><td colspan="2" rowspan="2">源 MAC 地址(字节 0-5)</td><td></td></tr>
<tr><td>源IP地址(字节0-1)</td></tr>
<tr><td colspan="2">源IP地址(字节2-3)</td><td rowspan="2">目的 MAC 地址(6B)</td></tr>
<tr><td colspan="2"></td></tr>
<tr><td colspan="3">目的IP地址(字节0-3)</td></tr>
</table>

按表 4-3 生成 ARP 请求报文后，再按以太帧格式在报文前添加目的 MAC 地址、源 MAC 地址和类型(0x0806)封装成以太帧，并由网卡驱动程序将该帧发送到网络中。以太帧报文格式见上面表 4-1。由于表 4-1 中的前导位、帧起始位和检验由硬件自动添加/删除，因此以太网驱动程序只需要添加目的 MAC 地址、源 MAC 地址和类型字段。

在发送 ARP 请求报文时，由于不知道对方的 MAC 地址，故以太帧中的目的 MAC 地址应全部填 0xff，以便网络中能接收广播帧的节点都可接收到并处理该以太帧。类型字段赋值 0x0806，表示报文数据部分是 ARP 报文。

ARP 请求报文的发送过程见图 4-12。

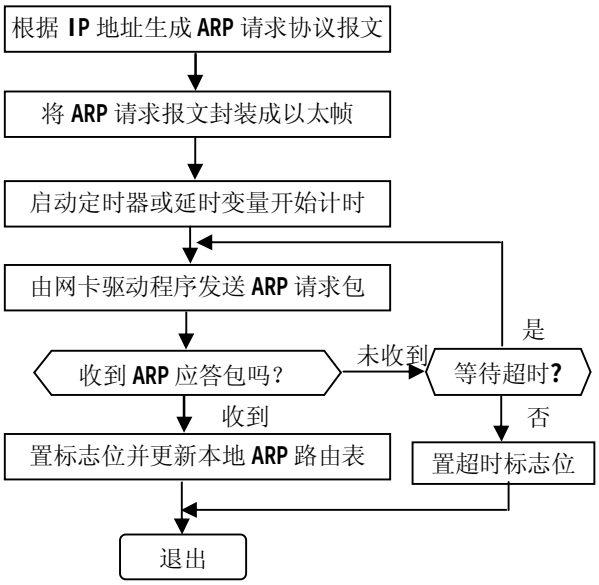


图 4-12 发送 ARP 请求协议流程图

(2) 发送 **ARP** 响应报文

当某以太网节点从网络上接收到一个以太帧时，若经解包后分析出是请求自己 **MAC** 地址的 **ARP** 请求报文时，应向对方发送 **ARP** 响应报文，告诉对方自己的 **MAC** 地址，以便后续网络通信能正常进行。此时，只需将以太帧头和 **ARP** 请求报文的源 **IP** 与目的 **IP**、源 **MAC** 地址与目的 **MAC** 地址互换，**ARP** 报文中的操作字段改成 **2**，并将本地**MAC** 地址替换该报文的源 **MAC** 地址字段，最后将以太帧发往网络中。

2．IP 协议

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** 封装再由网络接口层处理，同时，也负责将网络接口层传上来的数据进行拆包分析再交给传输层处理。

IP 报文格式见表 **4-4**。各字段含义如下：

表**4-4 IP**报文格式

<table>
<tr><td colspan="2">版本(4位)</td><td>头长度(4位)</td><td>服务类型(8位)</td><td colspan="2">报文总长度(字节数)(16位)</td></tr>
<tr><td colspan="3">标识(16位)</td><td>标志(3位)</td><td>片内偏移(13位)</td></tr>
<tr><td colspan="2">生存时间(8位)</td><td>协议(8位)</td><td colspan="2">头检验和(16位)</td></tr>
<tr><td colspan="5">源IP地址(32位)</td></tr>
<tr><td colspan="5">目的IP地址(32位)</td></tr>
<tr><td colspan="5">可选项</td></tr>
<tr><td colspan="5">数据</td></tr>
</table>

版本和报文头长度：使用 **IPv4**，通常的报文头大小为 **5**，即 **20** 字节，所以该字段的值通常取 **0x45**；

服务类型：提供诸如优先级等服务类型，由于该信息会被大部分的路由器忽略，通常将它设为 **0**，即“正常”的优先级；

标识与分段：**IP** 被设计用来支持广泛的物理介质，每种介质都有最大可传输字节数的限制**(MTU)**。若一个数据报大于 **MTU**，则进行 **IP** 封装时将被拆分成较小的分组再传输；而分组到达目的地后需重新组装起来；

生存时间：每一个在网络上传输的数据报都需要设置一个时间值作为该数据报的生存时间。该生存时间在数据报经过一个路由器时递减一次，如果减至 **0**，则这个数据报会被丢弃；

协议：这个值标识了该 **IP** 报的数据所使用的协议。若为 **1**，表示数据部分是 **ICMP** 报文，为 **6** 是 **TCP** 报文，而为 **17** 则是 **UDP** 报文；

头检验和：头检验和是对 **IP** 报文头进行校验，方法是按 **16 bit**字的二进制反码

进行求和，以保证 IP 报头的正确性；

可选项：报文头字段可以扩展为包含一些选项，允许更严格地控制路由过程和增加安全性。

当传输层有数据传到 IP 层时，IP 层按表 4-3 的格式在原数据的前面添加 IP 头。为简化应用，此处采用不带可选项的 IP 头(即 IP 头长 20 字节)，也不实现 IP 协议的分片功能。

当 IP 层接收到网络接口层传上来的 IP 报文后，判断是否发往本 IP 地址的报文，并检测版本、头检验是否正确，正确后再根据协议的取值上传给传输层协议或 ICMP 协议进行处理。

3．ICMP 协议

ICMP 为 Internet 控制报文协议，用于传递差错报文以及其他需要注意的信息，可以完成检查目的地址是否可以到达，用于拥塞和数据流控制，通知路由器的改变路由信息，检查路由是否形成回路等工作。ICMP 报文是在 IP 数据报内部被传输的，因此常被认为是 IP 层的一个部分。ICMP 报文类型很多，其中以测试目的主机是否可达的 Ping 命令最为常用。为方便网络调试，在 uIP 中实现了 Ping 命令。

一般称发送回显请求的 Ping 程序为客户，而称被 Ping 的主机为服务器。当服务器收到回显请求时，应向客户发送回显应答报文。ICMP 回显请求和回显应答报文格式如表 4-5 所示。

表4-5 ICMP回显请求和回显应答报文格式

类型(1字节)	代码(1字节)	检验和(2字节)
标识符(2字节)		序列号(2字节)
数据		

在表 4-5 中，Ping 报文的类型为 8 或 0，分别对应请求和应答，代码取 0；检验和是整个 ICMP 报文参与加运算后取反的结果；标识符能够识别出同一台机器上不同进程发出的 ICMP 报文；序列号从 0 开始，每发送一次新的回显示请求则加 1。

主动进行 Ping 操作的实现步骤如下：

(1) 解析 IP 地址

① 发送 ARP 请求

② 接收 ARP 应答

(2) 发送 ICMP 请求

(3) 等待 ICMP 应答

4．UDP 协议

UDP 和 TCP 是传输层的两个协议。TCP 提供一种面向连接的、可靠的字节流服务，非常适合诸如远程注册和文件传输之类的应用。而 UDP 是一个简单的面向数据报的传输层协议，不提供可靠性，它把应用程序传给 IP 层的数据发送出去，但是并不保证它们能到达目的地。

虽然 TCP 能保证数据传输的可靠性，但它的可靠性是靠复杂的协议完成的。若采用 TCP 通信，则需建立连接、维持连接和拆除连接，这将大大增加嵌入式设备的处理时间。而 UDP 头结构相对于 TCP 头结构更加紧凑，因而具有较高的传输效率。虽然它提供的是不可靠连接，但数据的完整性可以很容易由应用层超时和包重传机制解决，因此，UDP 仍然被应用层经常使用。在嵌入式应用中，通信往往是在一个局域网内的两个节点之间进行，基本上不需要出网关，报文被转发的次数很少，这大大降低了数据丢失的可能性。采用 UDP 传输时，由于不需要维持连接，从而可以使用一种更轻便更快速的接口，在低档单片机的嵌入式应用中更具有优越性[38]。因此，在嵌入式应用中，传输层采用 UDP 通信更合适。此处以 UDP 协议为例，介绍传输层协议的实现。

传输层处理过程大致如下：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，传输层负责将报文拆包处理，再将处理结果上交给应用层处理。UDP 报文格式见表 4-6。

表 4-6 UDP 报文格式

源端口号(2 字节)	目的端口号(2 字节)
UDP 报长度(2 字节)	UDP 报检验和(2 字节)
报文数据	

在表 4-6 中，源端口号和目的端口号分别对应发送进程和接收进程。UDP 长度字段指的是 UDP 首部和 UDP 数据的字节长度之和。该字段的最小值为 8 字节（没有数据部分），这个 UDP 长度是有冗余的。IP 数据报长度指的是数据报全长，因此 UDP 数据报长度是全长减去 IP 首部的长度。UDP 检验和覆盖了 UDP 首部和 UDP 数据，同时还包括 12 字节的伪头；伪头部分包含源 IP 地址、目的 IP 地址、8 位协议(17)和 16 位 UDP 长度。检验和算法与 IP 头检验相同。

5．应用层协议

TCP/IP 实现了 OSI 层次模型中传输层及以下层协议，而未实现应用层。嵌入式设备不像计算机那样具有丰富的资源，可以存储并处理大量数据，提供图文并茂的界面，对于实现如 FTP、HTTP 等应用层协议没有很大的现实意义。因此，在嵌入式设备

中往往自定义应用层协议。

4.4 CAN-以太网网关协议转换

CAN-以太网网关功能 是将CAN总线与以太网连接起来，以实现现场设备网与信息网的无缝连接。为实现总线和网络中数据通信，CAN-以太网网关协议转换主要应完成两部分工作：协议转换和路由表的维护。

4.4.1 协议转换

当位于CAN总线中的设备 需要将数据发往位于以太网的设备时，前者先将数据按CAN协议格式进行封装，并送到协议网关；协议网关从CAN接口接收CAN协议报文后，将报文数据部分按系统规定的应用层格式进行封装；然后将应用层数据存入以太网接口发送缓冲区，由该接口在数据前面依次添加TCP或UDP头、IP头、以太帧头，最后封装成以太帧后通过以太网接 口发往以太网中相应设备。反过来，当以太网中的设备 向CAN总线中的设备发送数据时，按原路返回，在经过协议网关时，依次去掉以太帧头、IP头、TCP或UDP头和应用层头，再将应用层数据按CAN报文格式封装好后存入CAN接口发送缓冲区，由CAN接口发往CAN网络中相应设备。若应用层数据长度大于8，则还需将数据以8字节为一组进行拆分。

由于协议网关是CAN总线与远程 服务器相连的必经之路，在通信高峰期间，

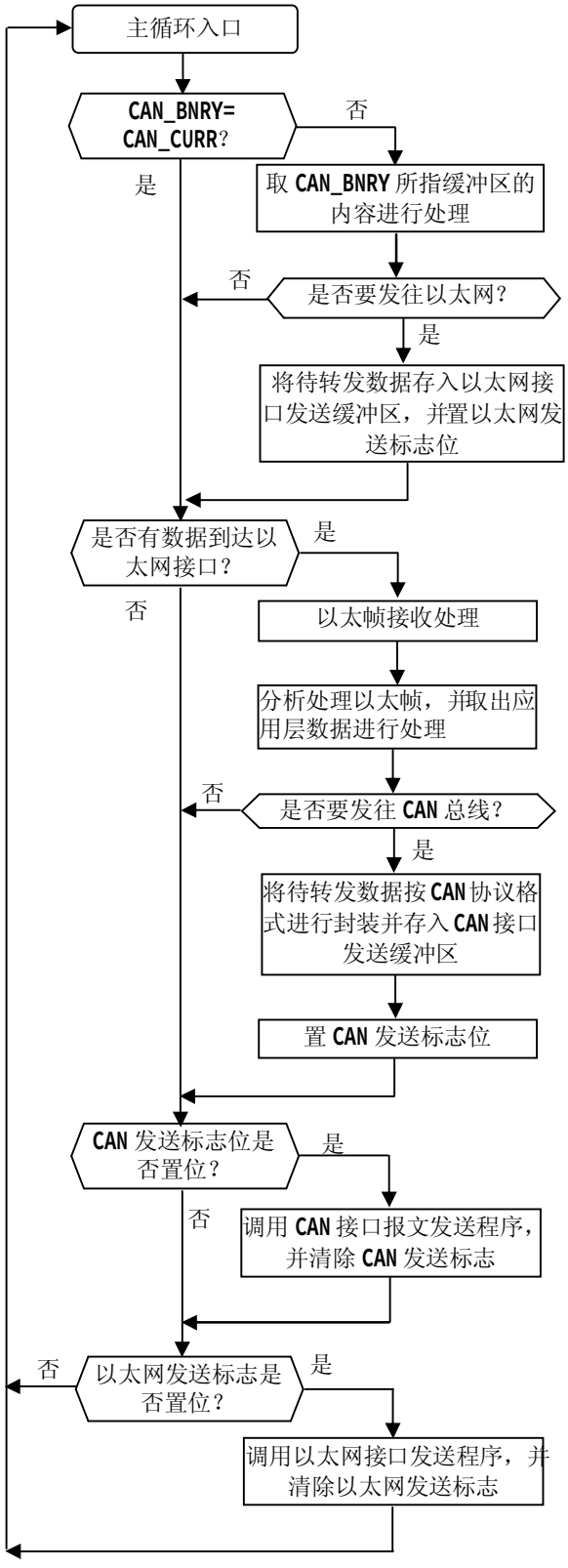


图 4-13 协议网关处理流程

向协议网关发送数据的CAN节点将会很多，虽然在某一时刻只能有一个节点获得发送权，但是，如果协议网关的对报文的处理速度跟不上CAN总线传输速度，仍然有可能会导致CAN接口的报文接收缓冲区的数据因来不及处理而被后续的报文覆盖；而协议网关与以太网中通信的节点很少，通常只有PC服务器，因此，以太帧被覆盖的机率很小。针对这种情况，在协议网关的软件设计中，CAN接收采用中断机制，并且建立一个接收缓冲区队列对接收的数据进行缓冲，该队列采用与RTL8019AS的接收缓冲区相同的的维护机制，即设置一个边界指针CANR_BNRY和一个当前指针CANR_CURR，通过修改这两个指针来维护CAN接收缓冲区。缓冲区的大小和队列长度依具体情况而定。当产生CAN接收中断时，在中断处理子程序中将CAN报文存入CANR_CURR所指的接收缓冲区，并修改CANR_CURR指针值。在主程序中CANR_BNRY与CANR_CURR是否相等，若不等，表示接收到新的CAN报文；取CANR_CURR指向的报文进行处理，处理完后，修改CANR_BNRY指针值。

网关主程序采用查询工作方式，各个处理模块通过置标志位的方式与其他模块协同工作。此处的标志位类似Windows下的消息。其工作流程如图4-13所示。

4.4.2 CAN节点路由表的维护

在实现CAN总线与以太网系统互连时， 需在协议网关中维护一张采集器编号与CAN协议标识符的对应表，即CAN节点的路由表。若没有路由表，网关在将数据发往CAN总线时，不知道该给数据添加什么样的报文标识符，从而目的节点无法接收到网关发出的数据。

当CAN报文到达网关时，在报文分析处理子程序中，先取出报文标识符和采集器编号。然后在CAN节点路由表中查找该报文标识符，若找到，则用当前采集器编号更新原来的值；若没找到，则将标识符和设备号添加到路由表中。CAN路由表格式如表4-7所示。

表 4-7 CAN 路由表格式

序号	ID 号	采集器编号
1	00000001,000	1
2	00000010,000	2
…	…	…

第五章 应用系统设计与实现

前面实现了CAN总线与以太网互连系统的软硬件设计，本章以苏州瑞萨稼动率采集系统为例，较为详细地介绍了互连系统在远程数据采集的实现过程，以实际应用检验了互连系统在现代企业生产中的可行性；同时对工业以太网在远程数据采集中的应用也作了一定的探讨，并分析了现场总线与工业以太网实现远程数据采集的区别。

5.1 稼动率采集系统概述

由于现代生产设备投资大，折旧快，只有保证它的很高的稼动率，才能最高限度的发挥它的使用效率。稼动率(设备正常生产率)的高低决定了投资回收期的长短，企业管理层必须实时了解现场每台设备的稼动率，以便作出正确的决策及调整。原有获取设备稼动率的方法主要通过手工抄写后再计算。由于不同车间之间、相同车间的不同设备之间都是相互孤立的，因而信息分散。这种现状给管理带来非常不便，且效率低下，与现代化生产很不相适。因此，实现稼动率的自动化采集、提高工作效率是当前急需解决的问题。

在稼动率采集系统中，主要是实现反映生产设备当前状态的指示灯的采集。车间内每一台生产设备都有一个状态灯管，灯管中包含三盏小灯，分别为红灯、黄灯、绿灯，三盏小灯通过“亮”、“暗”、“闪”实时反映当前设备的工作情况。稼动率采集系统就是实时地采集生产设备的三灯状态，先通过现场总线，再通过以太网将灯状态数据发往车间内的PC服务器；由PC服务器将数据入库，建立有关查询和报表，供管理层分析、查询、打印及远程访问等。

5.2 稼动率采集系统总体设计

图5-1是苏州瑞萨半导体公司稼动率采集系统的组成示意图。该系统需完成若干个车间内最多80台生产设备的稼动率的采集。每个车间内的数据采集PC需对相应车间内的多台设备的运行状态进行实时监控。由于车间的生产设备均为大功率电机，运转时电磁干扰非常严重，而传输的数据量较小，设备到远程服务器的数据为三盏灯的状态(亮、暗、闪)，PC到设备的数据为稼动率、故障率等少量计算后的数据，因此，为保证数据传输的高可靠性及实时性，在车间内部，通过CAN总线进行通信，而在车间

与管理层之间，则通过以太网进行通信；CAN总线中的数据通过嵌入式协议网关进入以太网并到达数据库服务器；同时，在车间内部也可以放一台PC机，协议网关通过SCI与该PC机相连；当协议网关不连到以太网中时，也可通过RJ45与PC机相连。

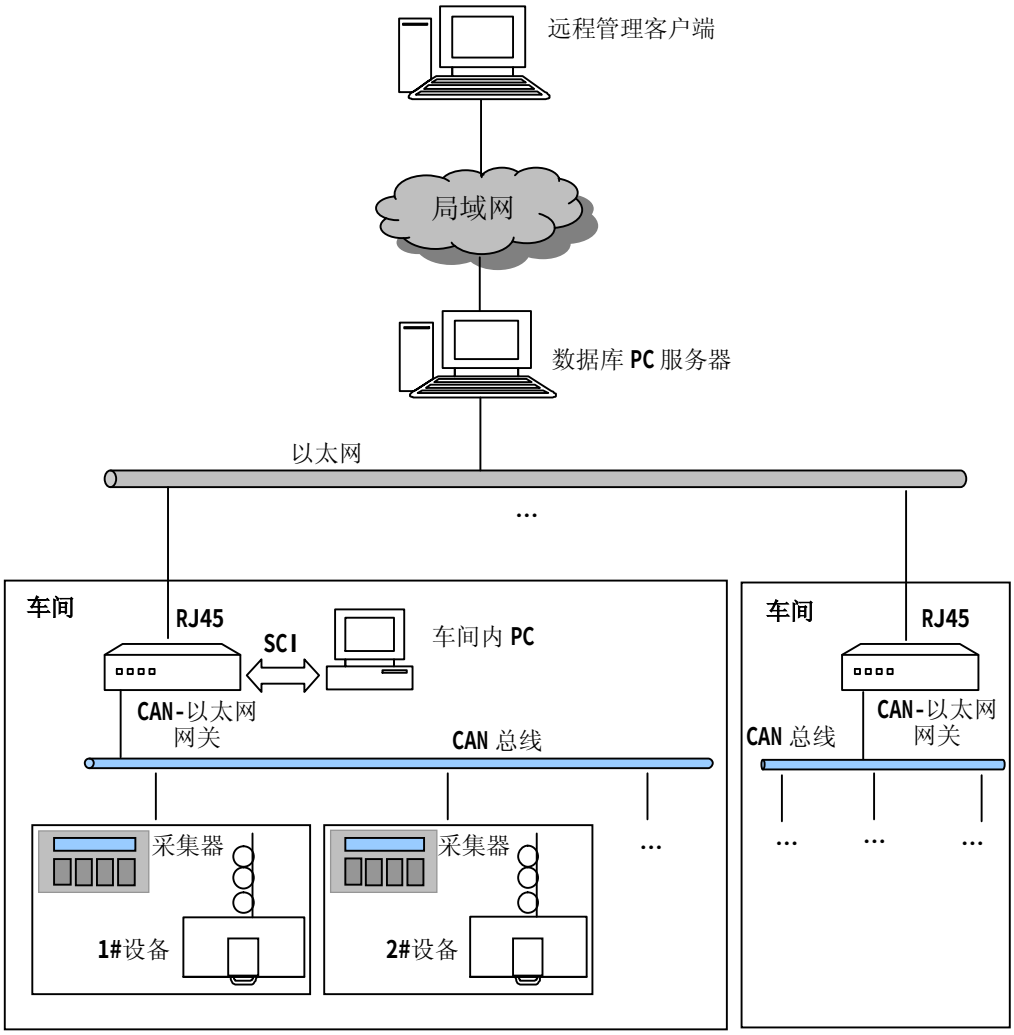


图 5-1 稼动率采集系统组成示意图

为实现稼动率采集，需为现场每一台设备系统配置一块数据采集器。采集器最基本的功能是完成三个状态灯的当前状态采集。而为了将状态发送到数据库服务器上，采集器还必须具备通信功能，同时，采集器应该提供一个友好的人机交互界面，以实现操作员特定的控制功能。根据上述功能需求分析，采集器需完成以下几部分功能：

数据采集：完成三灯亮、暗、闪状态的判定；

数据通信：实现现场设备数据到远程数据库服务器的上传以及远程数据库服务器数据到现场设备的下载；

键盘输入：完成现场操作员设备编号的输入等有关交互操作；

液晶显示：显示当前设备运行状态或键盘操作状态。

为实现现场设备与远程数据库服务器之间的数据通信，需为每个车间配备一台或几台协议网关，用于将现场总线数据转换成以太网数据或相反。同时，为提高灵活性，网关还提供了SCI接口。网关主要完成不同协议数据之间相互转换，以实现设备与PC机的相互连接。

5.3 稼动率采集说明

稼动率系统需采集的数据是采集红、黄、绿三色状态灯的亮、暗、闪状态。虽然三灯状态是由生产设备的信号控制的，但由于不同设备的信号电平不相同，有的24V，有的高达110V，而一般嵌入式应用系统都是弱电系统，像MC68HC908GZ60的供电电压为5V，在这种情况下，生产设备控制状态灯的信号是不能直接与单片机相连的。若采用电压转换模块，需要购买不同型号的转换模块，这将大大增加采集器的成本。此外，在采集器安装调试时，为了将状态灯的控制信号引出来与单片机相连，现场设备必须停产，这是生产方不能容忍的。因此，此处状态灯的数据采集是一个比较棘手的问题。

由于灯亮和暗时的光强度是有差别的，利用光敏电阻在不同光照强度下电阻不同的特性，在每盏灯旁边放置光敏电阻可将光信号转成弱电信号。这样在安装采集器时，丝毫不影响设备的生产。

将灯状态转成单片机能处理的信号后，单片机可以按开关量进行采集，也可按模拟量进行采集，这取决于实际需要。由于现场有的状态灯的亮暗强度差别非常小，即亮时不够亮，而暗时不够暗，这样一来，光敏电阻阻值变化相对比较小，若按开关量采集时不能正确区分出亮或者暗；而按模拟量采集时，则可以根据差值进行灯亮或暗的判断。

根据上述分析，此处采用MC68HC908GZ60的三路AD转换完成三盏灯的状态采集。单盏灯的采集原理如图5-2所示。灯状态由光信号转成电信号后，再通过AD转换可以由MCU读取。当灯由暗变亮时，光敏电阻的阻值会由大变小。实验中采用5539型号的光敏电阻进行测试时，灯暗时光敏电阻阻值为5K欧~6K欧，而灯亮时为600欧左右。在采集电路中，相应AD通道采集的是固定电阻的电压值。根据电路知识可知，灯亮时采集点的电压比灯暗时要高，因而MCU采集的AD值相应地也比灯暗时要大。为使电压变化尽可能明显，固定电阻应选取适当的阻值。

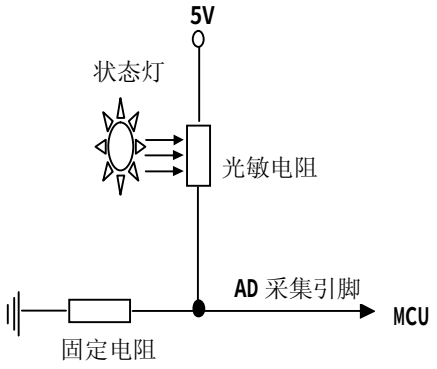


图 5-2 状态灯 AD 采集原理

只要亮暗AD差值超过一定范围，足以抵消AD采集的误差和外界的干扰，AD值就能正确反映灯的亮暗状态。经过在不同的外围光线强度环境中和现场多次测试，所有灯的亮暗AD差值均在100以上，最大时超过800。因此，测试的结果表明采用AD采集方案是可行的。

5.4 采集器硬件设计

采集器硬件功能模块见图5-3。

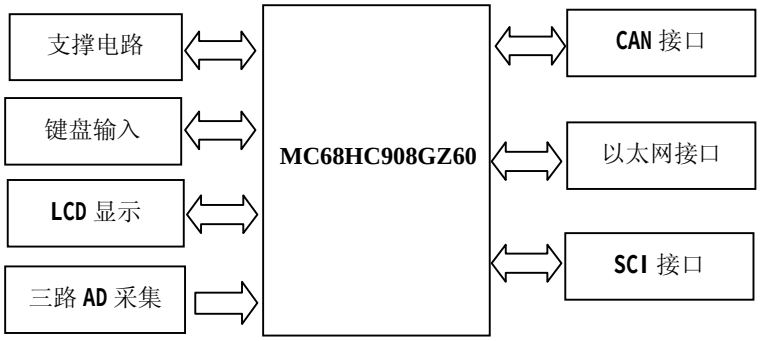


图 5-3 采集器硬件功能模块

在采集器的硬件设计中，除了采集、显示、键盘输入外，还提供了三种类型的通信接口，分别为串行通信接口、CAN总线通信接口和以太网通信接口，可以满足多种输出方式。由于提供了以太网接口和CAN接口，采集器也可以用作CAN总线与以太网互连的协议网关，从而不需要另外再设计协议网关。这样可节省成本，并降低整个应用系统的复杂性。

根据用户要求，采集器的LCD采用内嵌T6963C图形控制器的240*128点阵汉字液晶；键盘则采用18键(3*6)的薄膜键盘。

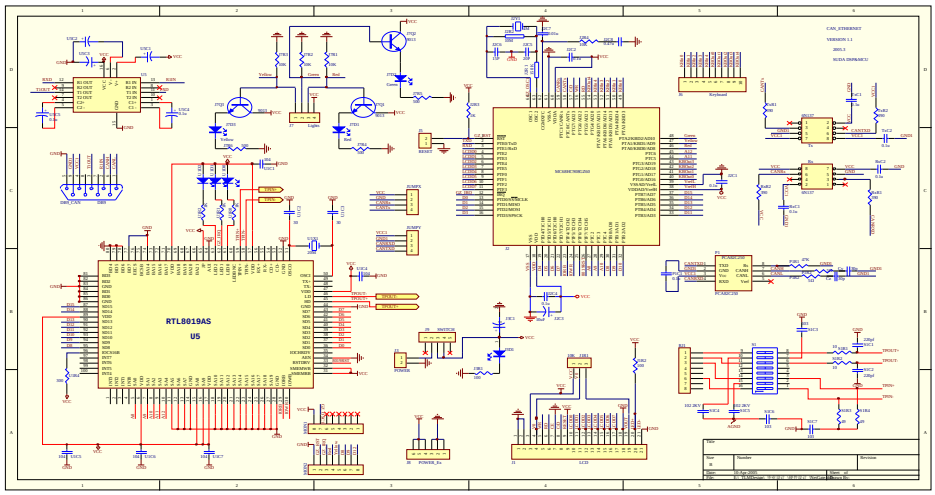


图 5-4 应用系统硬件原理图

采集器硬件原理图如图5-4所示。

5.5 采集器软件设计

采集器软件设计包括AD采集、汉字液晶显示、键盘输入、3种不同的通信接口等。

5.5.1 AD 数据采集及灯状态确定

灯状态采集除了要识别亮和暗状态之外，还需识别出灯的闪烁状态。在某个时刻灯的状态只能是暗或亮，因此，在判断闪烁状态时，必须在一段时间内才可确定。实际上，闪烁状态是由若干次亮暗的组合。在判定闪烁状态时，需要根据实际情况确定闪烁频率，不同的灯闪烁频率不相同，为使采集程序能适应所有状态灯，必须将最小闪烁频率作为编程依据。只有当三盏灯的状态(亮、暗、闪)都确定后，才能确定设备的某个运行状态。采用中断方式无法控制AD值的采集时刻，此处通过查询方式读取AD值。为保证闪烁状态的正确判定，必须保证在灯亮暗变化时间间隔内至少能读取一次所有AD通道。

一旦任何灯的状态发生变化，如由亮变暗或变闪，采集器必须确定三灯当前状态，并把确定后的状态数据发往远程服务器；而如果三灯状态长时间不变，则每隔一定时间(如4分钟)将灯状态数据上传。状态灯数据采集由AD采集、数据准备和状态判定三部分组成，采集流程见图5-5，其中数据准备和状态判定分别见图5-6和图5-7。

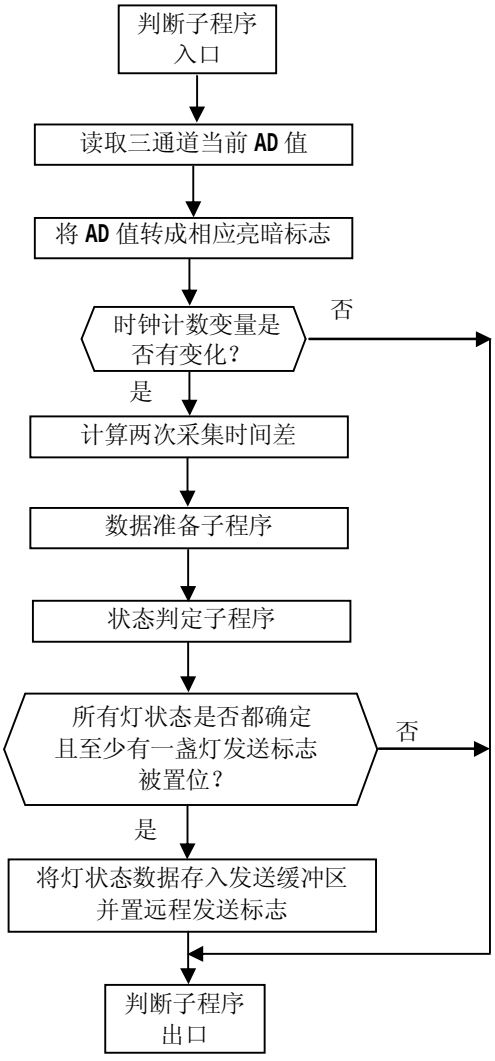


图 5-5 数据采集处理流程图

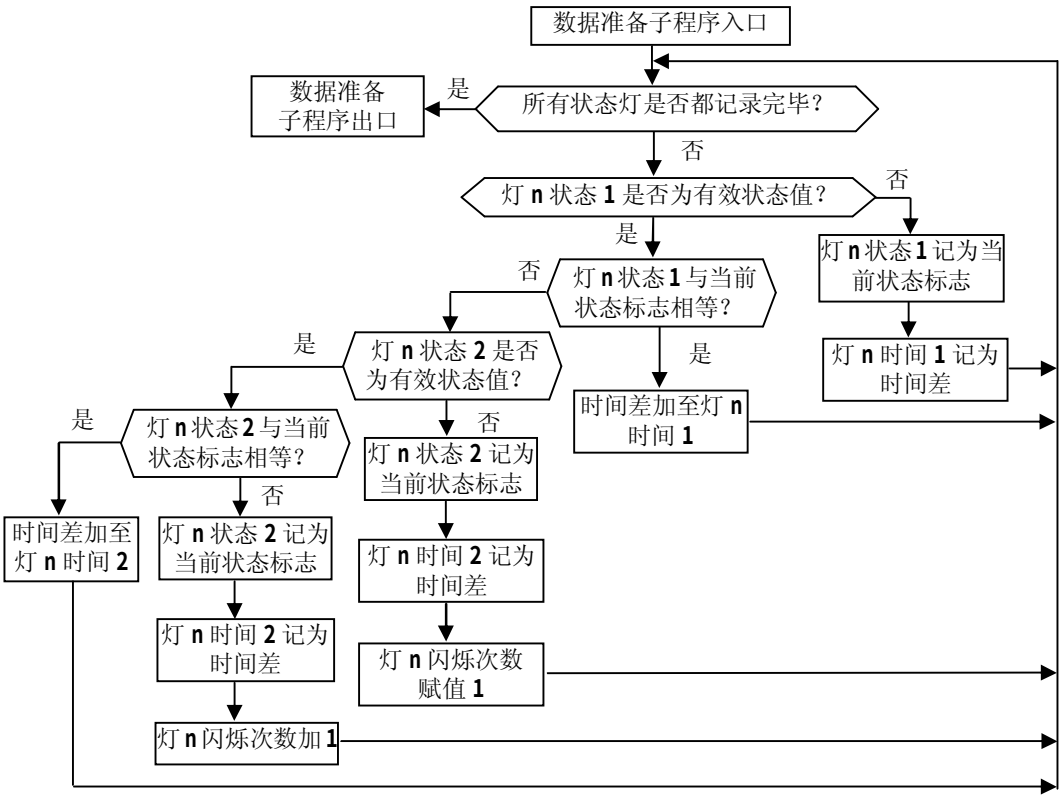


图 5-6 数据准备处理流程图

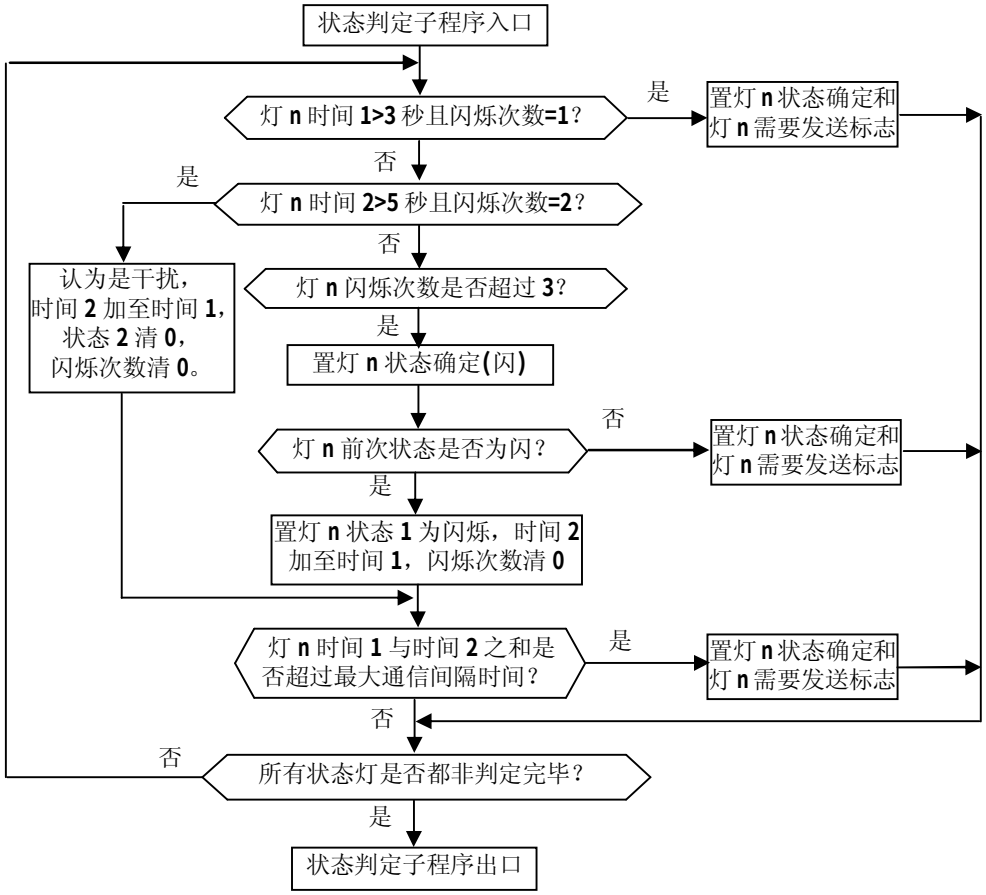


图 5-7 状态判定处理流程图

5.5.2 汉字液晶显示

在应用系统中，需要显示字符、汉字和图形。由于带汉字库的LCD价格昂贵，而此处要显示的汉字数量并不多，从成本和利用率角度考虑，采用只带ASCII字符库的240*128点阵的LCD进行显示。LCD支持文本和图形两种显示方式，ASCII码字符既可采用文本方式也可采用图形方式，而汉字或图形时则只能采用图形方式。以文本方式显示时，MCU只需发送字符在LCD字符发生器中的代号。采用图形显示方式时，MCU需将字符、汉字或图形的点阵发送LCD。

采集器主要有两种图形界面需要显示：一是非键盘操作时设备运行状态的显示，由界面框架和框架里面的内容组成，框架是固定的图形界面，内容为需要实时更新的稼动率(数字)、生产产品名称(英文)等数据；二是键盘操作时交互状态的显示，包括操作界面框架和当前按键(数字或英文)。框架数据以图形方式显示，由于MCU内部并不带汉字库和图形库，可先将框架数据固化在MCU的FLASH区，需要显示时，MCU将数据发往LCD；英文数字则按文本方式，根据实际情况随机刷新。

在存储框架数据时，由于一个点阵对应一位(bit)，故存储1屏(240*128)需要3840(240*128/8=3840)字节。考虑到显示界面中有多片连续空白或亮点，在固化到FLASH之前，可先对框架的点阵数据按行进行压缩。框架的点阵数据大小经压缩后不到压缩前的1/4，这大大节省了FLASH存储区。MCU预留20K FLASH用于存储若干屏压缩后的框架数据。在发往LCD显示前，MCU必须先对数据进行解压。解压规则如下：

```
// 对压缩后的数据进行分析
// 0B:图像宽
// 1B:图像高
// 2B以后为图像数据，以每个字节为单位
// 对当前字节的bit7-bit0按下列方法进行处理
// 若bit.0 = 0 未压缩，后面7位为7个像素数据
// 若bit.0 = 1 压缩，方法如下：
// 若bit.1 = 0 标准压缩，2位-6位为连续像素长度(<=32)
// 若bit.1 = 1 特殊压缩
// bit2-4 = 000 连续像素长度为剩下的行
//        = 001 连续像素长度为剩下的7/8行
//        = 010 连续像素长度为剩下的6/8行
//        = 011 连续像素长度为剩下的5/8行
//        = 100 连续像素长度为剩下的4/8行
```

// = 101 连续像素长度为剩下的3/8行

// = 110 连续像素长度为剩下的2/8行

// = 111 连续像素长度为剩下的1/8行

// bit.7为像素的颜色，1表示黑，0表示白

根据解压规则，相应的解压算法参见附录C。

5.5.3 键盘输入

键盘主要用于实现需要人机交互时的输入功能，例如采集器参数设置、手工输入设备运行信息等。当操作员在进行手工输入设备运行信息操作时，采集器进入手动采集状态，一旦键盘操作完成，将操作的结果上传远程服务器，并恢复自动采集。

5.6 远程数据采集的实现

采集器采集到数据后，由于与设备有关的信息存储在远程数据库服务器PC，故采集器生成的数据需要发往该PC机进行处理。采集器提供了CAN接口和以太网接口，下面介绍分别以这两种数据上传方式完成稼动率远程采集。

5.6.1 通过 CAN 接口上传

在这种上传方案中，现场数据先通过现场总线(此处为CAN总线)传至协议网关，再由协议网关将数据发往以太网，最后由以太网中的PC服务器负责接收处理并将处理结果按原路送回采集器。

表5-1 采集器通过CAN接口上传的数据格式

设备编号(1B)	数据类型(1B)	数据内容(6B)					
0~255	1	红灯状态(2B)		绿灯状态(2B)		黄灯状态(2B)	
		状态 1	状态 2	状态 1	状态 2	状态 1	状态 2
		1/2/3	1/2/3	1/2/3		1/2/3	1/2/3
	2	键盘功能代号					

说明：

1 整个数据长度为 8 字节(正好一个 CAN 数据帧最大数据长度)。

2 数据标志用于区分是灯状态数据还是键盘数据，1 为状态数据，2 为键盘操作数据。

3 如果是灯状态数据，在数据详细内容中包含三灯状态信息，每盏灯的状态都由 2 字节表示，状态 1 表示变化前的状态，状态 2 表示变化后的状态；其中，1 表示暗，2 表示亮，3 表示闪烁“灯状态”。若前后状态一致，则状态 1 和状态 2 相同。

4 所有灯都不变的情况下，节点每隔 250 秒发送一次三灯状态数据。

5 键盘数据为手工设置某功能后将功能代号发往远程服务器。

采集器将本地产生的数据存入CAN接口发送缓冲区，由CAN接口将本地报文标识符(11位ID)添加到数据前面，并置RTR为1、IDE为0，封装成CAN协议报文后送往CAN总

线，报文的数据部分格式见表5-1；CAN总线中的协议网关接收到该报文后，先去掉CAN协议头(即报文的仲裁域)，并按稼动率系统规定的应用层格式进行封装，再存入协议网关的以太网接口发送缓冲区，由该接口封装成以太帧后发往以太网，应用层格式见表5-2；位于以太网中的PC服务器负责接收并处理该以太帧。

表 5-2 协议网关发往远程服务器的数据格式

系统标志(2B)		版本(2B)		数据长度(2B)		数据内容(60B)	检验和(2B)
"J"	"D"	"1"	"0"	H	L	Data	check
稼动率采集应用系统标志		系统版本		高位	低位	从采集器收到的数据(不包括 CAN 标识符)	对全部数据检验

说明：
1 MCU 发往服务器的数据长度固定为 68 字节。
2 Data 为从采集器收到的数据，长度为 60 字节，不足部分补 0。
3 检验和为全部数据(68B)参与计算；若检验和不正确，不予接收，以提高数据传输过程中的可靠性。

远程服务器正确接收到协议网关发过来的数据后，先将数据入库，并更新相应设备的稼动率和故障率，再将新的稼动率和故障率通过协议网关送回采集器，由采集器更新本地LCD相关参数的显示。远程数据库送往协议网关的数据格式与网关送往服务器的数据格式相同，但数据内容部分不同，即协议网关反馈给采集器的数据不同，格式见表5-3。协议网关根据数据部分的设备号，找到相应的采集器的标识符后将它添加到数据部分的前面，并将封装好的报文存入协议网关的CAN发送缓冲区等待发送。

表5-3 采集器从协议网关接收的数据格式

设备编号(1B)	数据内容(7B)						
0~255	稼动率 x.y%(2B)		故障率 x.y%(2B)		服务器当前时间(3B)		
	整数 x	小数 y	整数 x	小数 y	时	分	秒

说明：
1 整个数据长度为 8 字节(正好一个 CAN 数据帧最大数据长度)。
2 稼动率和故障率均以百分数形式表示，保留两位小数。
3 服务器当前时间用于更新采集器当前时间。

为正确接收CAN总线上的数据，每个采集器都必须通过标识符(ID)寄存器和验收屏蔽寄存器配置好报文的验收机制。所有的采集器都只接收协议网关发出的数据，而协议网关可以接收所有采集器发出的数据，因此，可按如下方式设置协议网关与采集器CAN接口的验收机制：

网关：验收屏蔽寄存器0和1所有位全部取1，表示接收CAN总线上所有报文。

采集器：报文验收寄存器0和1在0b10000000,000~0b01111111,111中取值，不同采集器取值不同。验收屏蔽寄存器0和1分别取值0b00000000,0b000010111，表示只有当标识符与报文验收寄存器0和1中前11位的值均匹配的标准帧才会被接收。

5.6.2 通过以太网接口上传

通过以太网通信时，现场数据先按应用层格式进行封装，再通过采集器的以太网接口发到以太网中，由网络传输线负责将以太帧发往远程数据库服务器；远程数据库服务器发往采集器的数据则按原路返回。

通过以太网接口上传数据时，数据不再需要进行协议转换，因此协议网关便可省去。

5.6.3 两种远程采集方式的区别

两种远程采集方式各有优缺点，下面从通信速度、传输距离、供电方式等方面简要说明两者的区别。

1．通信速率

CAN总线最高通信速率可达1Mpbs(距离不超过40米)，在稼动率采集系统中，现场总线的通信速度通过软件设置设成400Kpbs；以太网采用10M以太网卡，实际网络通信速率可达500kbps。

2．传输距离

CAN总线在400Kbps的通信速率时，最大传输距离可在130米以上；而以太网协议(IEEE802.3协议)规定每一段双绞线(10Base-T)的长度不能超过100米。

3．响应时间

影响本系统响应时间的因素主要有2个方面：接口延时和网络传输延时。在互连系统中，采集器从发出数据到远程服务器接收数据，均需要协议网关转发，因而增加了两个接口的处理时间，导致响应时间被延长；而直接通过以太网通信，不需协议网关转发，因此可以有更快的响应时间。而在网络传输延时方面，虽然CAN总线通信速率不及以太网的通信速率，但CAN总线采用短帧结构，而以太帧最短长度为64字节，因此，网络传输的延时差别并不大。

4．抗干扰能力

CAN节点在严重受干扰时，能自动关闭总线，使总线上其他节点不受影响；另外，总线上的信号通过高速光耦隔离，具有很强的抗干扰能力；而以太网的隔离芯片不能完全将网络上各节点隔离开来，在强电磁干扰的现场，必要时需采用其他抗干扰措施。

5．成本

由于主控芯片GZ60内部集成了CAN协议控制器，在构建CAN节点时，不需再购买CAN协议控制器；而由于TCP/IP协议众多，目前并没有集成TCP/IP协议簇的8位机，需外接网络接口芯片，RTL8019AS市场零售价在20元左右，因此，实现CAN接口在成本上比

以太网接口要低。

6．适应环境能力

CAN接口采用锁紧机构的DB9连接网线，具有更好的抗振动、抗疲劳能力；设备通过总线供电，不用另外再提供电源，安装布线比较方便；以太网采用RJ45接口，容易从接口脱落和被损坏，且容易从接口脱落，通过定制带锁紧机构的网线接插件则相对比较麻烦。

在稼动率采集系统中，主要功能是实现远程数据采集，数据通信量小，而实时性要求相对并不是十分严格，且目前通信距离不超过40米，设备节点数在80台以下，因此两种方案都能较好地满足系统要求。

第六章 应用系统测试及体会

第五章比较详细地介绍了苏州瑞萨稼动率采集系统的各功能模块及实现过程，各个功能模块必须经过相应的测试才能集成在一起。而整个系统的软硬件功能最终是否能满足用户要求，只有经现场测试之后才能下结论。本章先简要叙述应用系统各功能模块独立测试及应用系统在现场的测试情况，接着分析了现场测试过程中出现的问题和采取的解决措施，最后较为详细地给出了本人对开发嵌入式应用系统的体会。

6.1 模块测试

软件功能模块设计完成后，应先单独测试该软件的正确性。只有在各功能模块正确运行的基础上，才有可能保证模块集成后整个应用系统的正确性。下面简要说明稼动率系统中主要软件功能模块的测试过程。

6.1.1 灯状态采集测试

在采集灯状态数据时，将光敏电阻安装在用户提供的测试灯旁边，再手工让不同的灯亮、暗或闪烁。MCU 识别状态后将状态数据通过 SCI 发往 PC 机，从而可以了解状态的判定是否与实际一致。状态数据在 PC 机的显示如图 6-1 所示。

图 6-1 灯状态采集测试程序界面

6.1.2 以太网通信接口测试

通过在 PC 方运行模拟的服务器程序，对以太网接口的通信速率、通信可靠性等进行测试。

在测试以太网接口的通信速率时，分别在 4M 和经过 PLL 编程倍频到 8M 总线频率下进行测试；在不同的总线频率下又分别将 RTL8019AS 设为 8 位模式和 16 位模式。4 种情况的测试结果见表 6-1。

表 6-1 以太网接口通信速率测试结果

测试模式 4M	、8 位	4M、16 位	8M、8 位	8M、16 位
速率(kpbs)	200	336	320	454

由于采用 UDP 通信，以太网接口通信可靠性测试主要是测试 UDP 通信丢包的概率。测试时，将带有以太网接口的采集器连到交换机上，并与连到同一交换机的另一台 PC 相互通信。采集器每隔 10 秒钟向 PC 发送 100 字节的 UDP 报文，当 PC 收到采集器的发过来的 UDP 报文时，立即回复一个 1450 字节的 UDP 报文。经过连续十几个小时的测试，结果表明采用 UDP 通信丢包次数和出错次数都极少。测试数据如图 6-2 所示。

图 6-2 以太网接口通信可靠性测试结果

6.1.3 CAN 总线通信接口测试

PC 机一般都不提供 CAN 接口，因而 CAN 通信接口的测试不如以太网接口测试那样直接。但 PC 机和 MCU 可通过 SCI 进行通信，因此，可采用将 CAN 总线上的数据通过 SCI 发往 PC 以检测 CAN 总线通信情况。

测试时，在 CAN 总线中接入 2 个 CAN 节点 A 和 B，节点 A 向节点 B 发送固定的数据(“Hello!”)，而节点 B 在接收到 A 的数据后将数据通过串口发往 PC 机，由 PC 机检测该数据是否预计的数据(“Hello!”)，同时，向 A 送出响应信息(“Welcome!”)，从而可确定 CAN 通信成功与否。

6.1.4 协议网关测试

要测试协议网关的数据转发功能时，在CAN 总线中挂接若干个采集器和 1 个网关(只完成协议转换的采集器)，并把网关接到以太网交换机上。每隔 10 秒钟，每个采集器向协议网关发送 1 个数据帧，协议网关收到数据后将通过以太网接口发往远程 PC 机。

针对实际应用系统的自身特点，CAN 总线中每个采集器从 1 开始编号(最大为 80)，相应 CAN 接口的本地 11 位滤波标识符与编号一致。当修改编号时，相应滤波报文标识符也一起修改。有了这一对应关系，协议网关可不需维护 CAN 路由表，而直接取报文数据域中的设备号作为 CAN 报文标识符即可。

每隔 10 秒钟，所有采集器通过 CAN 接口向协议网关发送 1 个数据帧。网关收到数据帧后将数据经以太网接口封装后发往以太网。为测试方便，网关接收到 CAN 报文后，将 CAN 报文经 SCI 发到 PC 机。

CAN 报文的标识符决定了其优先级，当多个采集器同时向网关发送数据时，编号小的采集器比编号大的优先获得 CAN 总线发送权，但由于采集器数据量小、上传数据周期长(实际应用中为 4 分钟上传一次)、对响应时间要求并不高，因此，编号大的采集器在一定时间内总能获得发送权；短暂的延迟对于稼动率的计算和显示可以忽略不计。

6.2 集成测试

在各功能子模块测试完成后，最终需将它们集成到一起进行测试。不管现场采用工业以太网还是现场总线与以太网互连方式实现远程数据采集，远程服务器的处理都是一样的，这使得服务器程序可独立于采集方式，从而提高了底层软件与上层服务软件的相互独立性。图 6-3 为运行在服务器端的测试程序。

图 6-3 稼动率系统状态灯远程采集测试程序

6.3 应用系统在现场测试中遇到的问题及应对措施

6.3.1 数据采集

1．出现的问题

在确定使用光敏电阻将光信号转成电信号后，由于用户提供的用于实验测试的灯亮暗强度差别比较大，同时，在灯罩内部安装光敏电阻也非常方便，因此，初始采用将光敏电阻放置灯罩内并按开关量进行采集。到现场安装时，发现有少数几盏冷光源的状态灯，亮暗差值非常小，无论怎么调整固定电阻的阻值，灯亮暗变化时 MCU 相应输入引脚的电压变化范围都非常小，以至无法区分高低电平。

2．应对措施

针对冷光源灯的亮暗差值小而导致按开关量采集无法正确识别灯的状态这一问题，经分析并测量后发现若通过 AD 方式采集时，亮暗变化时的 AD 差值至少在 100 以上，这样便可按模拟量进行采集。

6.3.2 现场干扰

1．出现的问题

上电启动后，若干台采集器液晶不能正常显示或刷新速度很慢，且后续功能不能正常执行，而将它们安装在其他地方，则一切功能运行正常；同时，将在其他地方运

行正常的采集器替换此处的采集器，也不能正常运行。但是，改用自带的网线，所有采集器运行正常。

2．应对措施

根据上述现象，先将现场环境与实验室环境进行了对比，推测是现场电磁干扰太强，初步认为是采集器硬件抗干扰能力不够，导致受环境的影响比较大。先检查硬件电路板上电源引脚、与外界通信的信号出入引脚附近是否有足够的抗干扰元件。在改进硬件电路板后，现象仍得到不到明显改善，因此，推测干扰源来自信号线。网线初始采用非屏蔽双绞线，由于其他原因不允许重新布线，但又必须在 2、3 天内解决，只能考虑先从软件着手解决。在调试过程中，发现如果有按键操作，则液晶刷新正常；而按键是采用中断方式处理，由此得到启发，程序有可能一直在响应某个中断，导致主程序运行缓慢。将所有外界中断源(此处只有键盘中断)关闭，采集器运行正常。至此，可以确定采集器不能正常工作是因为主程序频繁处理键盘中断。而导致频繁产生键盘输入中断的原因有可能是信号线进入采集器后影响到电路板上键盘输入信号。

6.4 体会

经过近三年对嵌入式技术的学习，本人对嵌入式技术应用系统的设计开发有了更深刻的认识。下面结合本人的毕业设计就如何开发嵌入式应用系统分别从硬件和软件两方面提出个人的几点体会，希望能提供给相关技术人员作参考。

1．必须明确用户的功能需求并制定合理的系统设计方案

在需求分析期间，必须尽量多地与用户沟通交流，明确用户最终需要什么样的产品。同时，开发人员应到现场去检验按用户的表述制定的系统设计方案是否可行。对于嵌入式应用系统开发来说，这两点尤为重要，因为接下来的硬件设计和软件设计都是在系统设计方案基础之上进行的。硬件设计需要进行原理设计分析，如果对用户的需求理解稍有偏差，得到的硬件设计原理就很有可能不正确，在错误的设计原理基础上得到的产品可能与用户的需求就差之千里了。另外，用户一般不懂实现的技术细节，因此，即使正确理解了用户的描述，如果没有去现场检验系统的可行性，也可能会导致最终产品不能满足用户实际要求。没有哪一个软件开发人员愿意听到用户说“你的

产品是我所要求的，但并不是我真正需要的产品”[39]。

2．必须保证硬件电路的正确性与合理性

在硬件设计时，必须在保证原理分析正确的基础上进行。系统各功能模块确定后，分析其控制原理，并确定主控芯片和控制方案。在制作PCB(印刷电路板)之前，应设计正确的电路原理图。布线时应先将所有元件以硬件功能模块为单位摆放到合适的位置，并使具有相同走线方向的元件最好能处于同一水平或垂直位置，这样可以使布线相对比较顺畅。元件位置摆放好后，通过自动布线，将不规范的走线调整过来，再手工重新连接。另外，PCB上除了系统要求硬件电路，还应预留测试点，以方便后续的软硬件调试。

3．尽可能提高系统电磁抗干扰能力

单片机系统是电子设备，而系统的运行环境本身也处于不同的电磁场中，因此，单片机系统都不同程度地存在着电磁干扰问题，其中以用于工业实时控制系统中的单片机系统的电磁干扰最为严重。如果没有采取合适的抗干扰措施，电磁干扰完全有可能影响到单片机系统无法正常运行。因此，在嵌入式产品尤其是工业产品的设计中，必须充分考虑电磁干扰问题。

提高电磁抗干扰能力时，硬件方面主要通过正确选用元件，采用接地、屏蔽、滤波和优化电路设计等技术以减少或消除干扰、破坏干扰信号的传输条件，从而提高整个系统的抗干扰能力及可靠性。例如，在选用元件时，应尽量选用可靠性高的元件；在绘制PCB板时，要适当地运用旁路和去耦技术，如在电源附近放置0.1u的滤波电容；在控制器的一些外部接口中，采用光电隔离元件进行隔离；通信介质采用屏蔽线，且将屏蔽端接地；等等。这些措施对提高系统的抗干扰能力是非常必要的。

对于单片机系统来说，有些干扰会侵入系统而引起系统非正常运行，因此，除了采用硬件抗干扰方法外，还应采取必要的软件抗干扰措施。如采取多次采样输入判断以提高输入的可靠性；重新初始化，强行恢复正常工作；查询中断源，防止干扰造成误中断；必要时对中断进行屏蔽；启用看门狗；增加冷热复位判断等。在单片机系统中，只要认真分析系统所处环境的干扰来源以及传播信息系统，采用硬件、软件相结合的抗干扰措施，就能保证系统长期稳定可靠地运行[40]。

4．软件设计时应按照合理的编程规范

好的程序不仅仅是体现在程序的正确性和运行效率上，良好的程序版式可以让阅读程序的人赏心悦目，因而更便于程序的阅读和维护。子程序的长短、文件的大小、行的缩进对齐、层次的嵌套、注释位置等等都是在编写程序过程中应该注意的地方。在进行软件设计时，适当的注释是非常有必要的。注释清晰的程序便于阅读，而光秃秃的代码让人不知所云，时间长了甚至编程者本人都不清楚该段代码的功能是什么。这样的程序使后期的调试和维护异常困难。当然注释也不是越多越好，注释太多反而使得程序拖沓冗长，结构不清晰。一般在注释子程序时，应在程序头说明程序名、功能、入口、出口、编程者、调用举例等内容。这样，使用者可以不必了解程序的具体实现细节，按说明调用即可。

软件系统往往并不是一个程序“一统到底”，合理地进行软件功能模块划分是非常必要的。为避免以后推倒重来，在设计的初期就要对整体做好把握，各部分功能进行合理的切割并尽量保证每个功能模块的独立性，明确各自的入口和出口，不能多个模块牵扯不清。当软件有错误时，能迅速定位，且不需“牵一发而动全身”。只有如此才能保证最终得到一件完美的软件产品。

5．测试要充分

测试是项目开发过程中重要的一环。嵌入式系统的测试包括硬件和软件两方面的测试。只有硬件稳定可靠，才能保证软件的可靠性。通常情况下，硬件和软件的测试是交互进行的。原理设计正确基本上能保证硬件正确工作，因此，测试过程中，主要的测试还在于软件。

软件测试应先进行各模块测试，再进行集成测试。编程者往往对自己的程序非常自信，代码一写完，几次测试后没发现问题便认为程序是正确的。而实际上，程序潜伏的错误因为没有经过充分测试而未能发现，这样的程序在集成到系统中之后一旦出现问题，便很难定位，因而使得后续的调试维护工作极其困难。

6．加强项目组开发人员间的沟通

一个大的项目往往是由多个开发人员共同完成的。在功能被分割成相对独立的子模块后，各人负责若干个子模块，子模块之间通过程序的入口和出口进行关联。在程

序的修改和完善过程中，必须及时与项目组其余开发人员沟通，在耦合度很高的模块之间，沟通尤显重要。若开发人员只低头编写自己的程序，程序出入口或功能被改变但并未及时通知其他开发人员，有可能导致他们的后续工作都是在做无用功，从而影响整个项目的开发效率。

第七章 总结与展望

7.1 总结

本文主要完成了以下工作：

(1) 详细剖析了 **CAN** 总线和以太网两种协议，分析了互连系统的可行性及其现实意义；

(2) 完成了互连系统的软硬件设计；

(3) 将该互连系统应用于稼动率远程采集系统中；

(4) 对工业以太网的应用进行了一定的探讨。

本文只是完成了 **CAN** 总线与以太网的数据通信，对于网络中数据安全、互连系统的传输速率、节点的响应时间、网关协议转换速率和机制等问题还有待作进一步深入的探讨。

7.2 展望

现场总线是应工业自动化而产生并迅速发展的，但由于各种总线之间协议不一，不同的自动化产品的互操作性有待加强。而以太网由于成本低、通信速度快以及开放性和兼容性，其应用越来越广泛，已经渗透到工业控制领域中而成为现场总线的一个重要发展趋势。但以太网不适用于易燃易爆以及环境条件恶劣、可靠性要求很高的应用场合。现场总线已有它的市场定位，工业以太网不可能完全替代现场总线，最有可能的是发展一种混合式控制系统。对工业以太网的广大用户更实际的好处在于如何利用现有网络为人们提供的功能。因此，现阶段最有效的方法是如何将工业以太网和现有的现场总线完善地结合起来，建立起完整的工业自动化网络体系[41]，从而实现底层生产与上层管理的紧密集成。这已成为企业信息化发展的趋势[42]。

致　谢

三年时间飞逝而过，在这即将结束研究生阶段学习的时刻，我只能用言语向我的导师王宜怀教授和他的夫人张建英老师表达我心中最诚挚的谢意。感谢他们在学习上和生活上对我的关心和帮助。在王老师的敦敦教诲下，我学会了如何独立思考问题、分析问题、解决问题。我所取得的每一个进步，都倾注了王老师大量的心血和汗水。已记不清有多少次，王老师坐在我的旁边，教我如何整理程序、如何发现并解决问题。王老师渊博的知识、严谨的治学态度、孜孜不倦、敢于突破的精神以及豁达的处世态度，都将让我终生受益。他把我领进了科研的大门，让我更明白今后努力的方向。

感谢我的师兄刘晓升，他娴熟的操作技能和对问题独到的见解让我受益匪浅，每次和他讨论，都会有新的收获，和他一起的学习生活让我终生难忘。感谢我的同学蒋建武，他勤奋好学的精神深深激励着我，他乐于助人的品德让我深受感动。感谢我的同学徐丽华，她的善良和宽容让我的人生中又多了一个永远的朋友。感谢我的师弟周海发、陈帅等，师妹刘雪兰、郑洪静等，他们在我的课题研究过程中给予了很大的帮助。同时，也感谢和我在同一实验室的林霞，她的安慰、鼓励和帮助让我紧张烦燥的心情得以缓解，让我能静下心来进行论文的撰写。感谢我们实验室全体成员，是他们让实验室变成一个团结协作、和谐融洽的学习生活大家庭。

感谢我的父母，这篇论文能够完成与他们的宽容和支持是分不开的，是他们提供给我一个良好的生活环境，使得我能全身心地投入到我的学习和工作中。

最后，感谢所有关心和鼓励我的老师、亲人、朋友和同学，谢谢你们对我的支持与帮助!

参考文献

[1] 柴雅静 ,向继东 .LonWorks 的嵌入式以太网关设计 [J].测控技术 ,2003,22(10):31-32,40.

[2] 梁泰文,阳宪惠.CAN/Internet 嵌入式网关的一种设计方案及实现[J].冶金自动化,2004,(2):5-10.

[3] 邬宽明 .CAN 总线原理和应用系统设计 [M].北京 :北京航空航天大学出版社,1996.

[4] 张荣跃,刘琳章.几种典型现场总线及特点[J].太原科技,2001,(3):14-16.

[5] 刘建昌 ,左云 ,钱晓友 ,陈智峰 ,冯立 .现场总线概述 [J].基础自动化,2003,7(6):1-5.

[6] 杨宇祥.基于 CAN 总线的电力远程监测系统研究[D].湖南:湖南大学,2002.

[7] 王博,蒋云峰,刘杰.基于 CAN 总线的网络监控系统[J].微计算机信息(测控自动化),2004,20(3):16-17.

[8] 陈粤.以现场总线为底层的企业测控网络系统[D].天津:天津工业大学,2001.

[9] 基于 CAN-bus 和以太网的区域信息管理系统[EB/OL].广州周立功单片机发展有限公司,www.zlgmcu.com,2003-11-20.

[10] 李正军.现场总线及其应用技术[M].北京:机械工业出版社,2005.

[11] 袁学文 ,黄天戍 ,庞辉 .CAN 总线与以太网 互连 系统设计 [J].电子技术应用,29(11):26-28.

[12] 白成杰 ,白成林 ,韩纪广 .计算机通信网络技术及应用 [M].北京:人民邮电出版社,2002.

[13] 华蓓,钱翔,刘永.计算机网络原理与技术[M].北京:科学出版社,2002.

[14] TCP/IP 详解[Z].卷 1:协议.2004.

[15] James F.Kurose, Keith W.Ross(美).计算机网络---自顶向下方法与 Internet 特色(影印版)COMPUTER NETWORKING(A Top-Down Approach Featuring the Internet)[M].北京:高等教育出版社,2001.

[16] James F.Kurose Keith W.Ross(美).计算机网络[M].北京 :清华大学出版社,2003.

[17] 吴进时.嵌入式网关的设计与实现[D].东北:东北大学,2002.

[18] 马志强.嵌入式以太网技术研究与应用[D].中国海洋大学,2003.

[19] 陈武,雷航.基于精简 TCP/IP 协议栈的信息家电网络服务器 [EB/OL].http://www.21ic.com/news/n1319c68.aspx,2004-12.

[20] 广州周 立功单片机发展有限公司 .CAN-bus 规范 V2.0 版本 [Z].http://www.zlgmcu.com,2004.

[21] BOSCH.CAN Specification 2.0[S].1991-09.

[22] 饶动涛,邹继军,郑勇芸.现场总线 CAN 原理与应用技术[M].北京:北京航空航天大学出版社,2003.

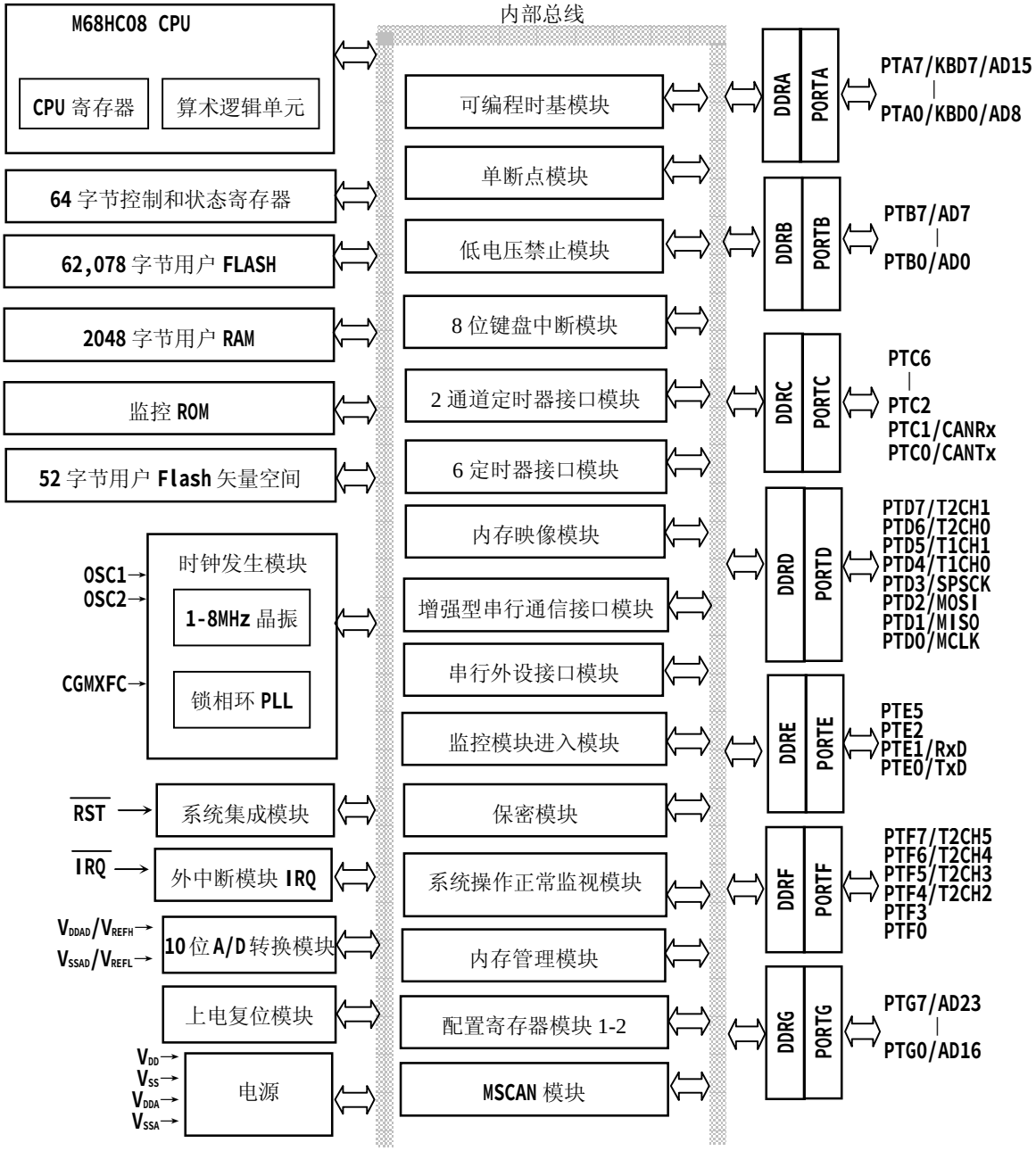
[23] 史久根,张培仁,陈真勇.CAN 现场总线系统设计技术 [M].北京:国防工业出版社,2004.

[24] 差分信号[EB/OL]. http: // www. latticesemi. com.cn/ support/ faq/ faq_isppac30_10.htm, 2004-3-20.

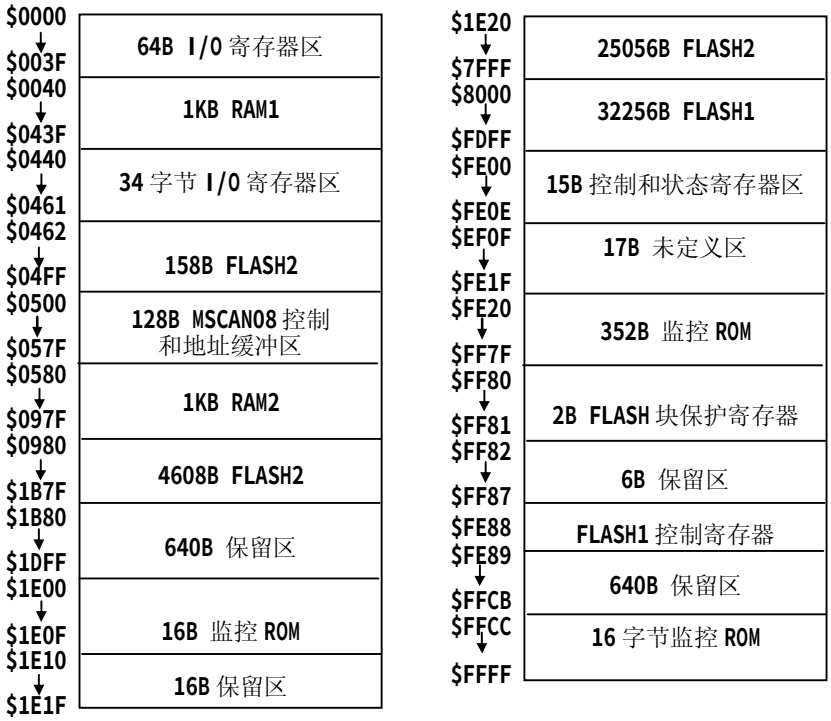
[25] 王宜怀.单片机原理及其嵌入式应用教程[M].北京:北京希望电子出版社,2002.
[26] Data Sheet MC68HC908GZ60-MC68HC908GZ48-MC68HC908GZ32Rev.1.0, Freescale Semiconductor,Inc[EB/OL]. http://www.freescale.com/,2003.
[27] RTL8019AS Data Sheet[Z].REALTEK SEMI-CONDUCTOR CO.LTD,2000.
[28] 金立田.CAN总线Internet接入服务器的研究与实现[D].江苏:东南大学,2002.
[29] 求是科技.单片机系统应用系统实例开发导航[M].北京:人民邮电出版社,2003.
[30] 刘晓升.基于8位MCU的嵌入式Internet设计与实现[M].苏州大学研究生论文,2004.
[31] PHILIPS Semiconductors – PCA82C250 CAN Controller interface data sheet[Z].1997.
[32] 李晋华,邱春玲,田地,凌振宝.基于CAN总线数据采集系统的设计与实现[J].吉林大学学报(信息科学版),2004,22(2):134-137.
[33] 黎新亮,马慧,马君璞.基于SJA1000的CAN总线通信模块的设计[J].沈阳工业大学学报,2004,26(1):99-102.
[34] 邵贝贝.单片机嵌入式应用的在线开发方法[M].北京:清华大学出版社,2004.
[35] skyeye硬件模块平台[EB/OL]. http: // download. gro. clinux. 0rg /skyeye /skyeyeinternal-0.6.8.pdf, 2005-3-20.
[36] 陈建恩.温室数据采集系统远程通信接口设计研究[J].农业工程学报,2003,19(4):259-263.
[37] 陈向群等译.嵌入式系统Web服务器TCP/IP Lean[M].北京:机械工业出版社,2003.
[38] 李勇,戴瑜兴.基于UDP协议的实时监控系统[J].电子技术,2003,(11):37-40.
[39] 袁兆山等译.软件工程Java语言实现(Stephen Schach)(美)[M].北京:机械工业出版社,1999.
[40] 刘光斌,刘冬,姚志成.单片机系统实用抗干扰技术[M].北京:人民邮电出版社,2003.
[41] 段永康.浅谈工业以太网的服务和应用[J].自动化博览,2004,(4):40-42.
[42] 邹益仁,马增良,蒲维.现场总线控制系统的设计和开发[M].北京:国防工业出版社,2003.

附录 A MC68HC908GZ60 相关资料

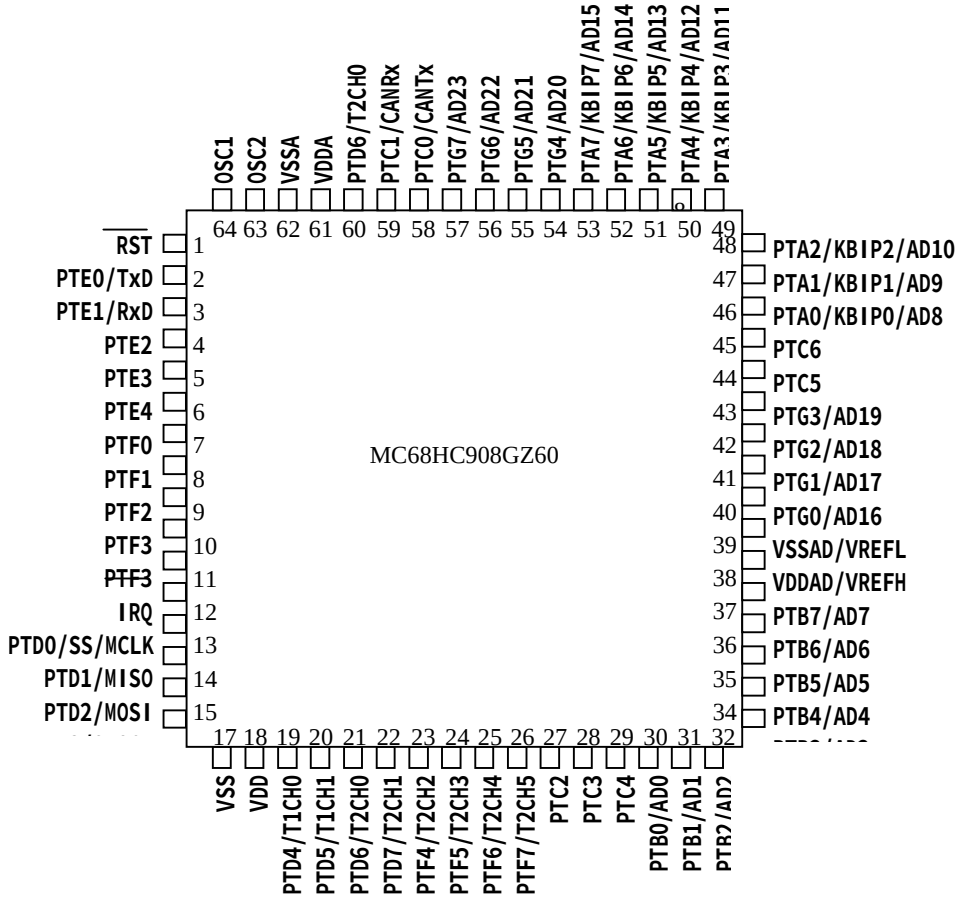
A.1　功能结构框图



A.2 存储器映像图



A.3 引脚封装图

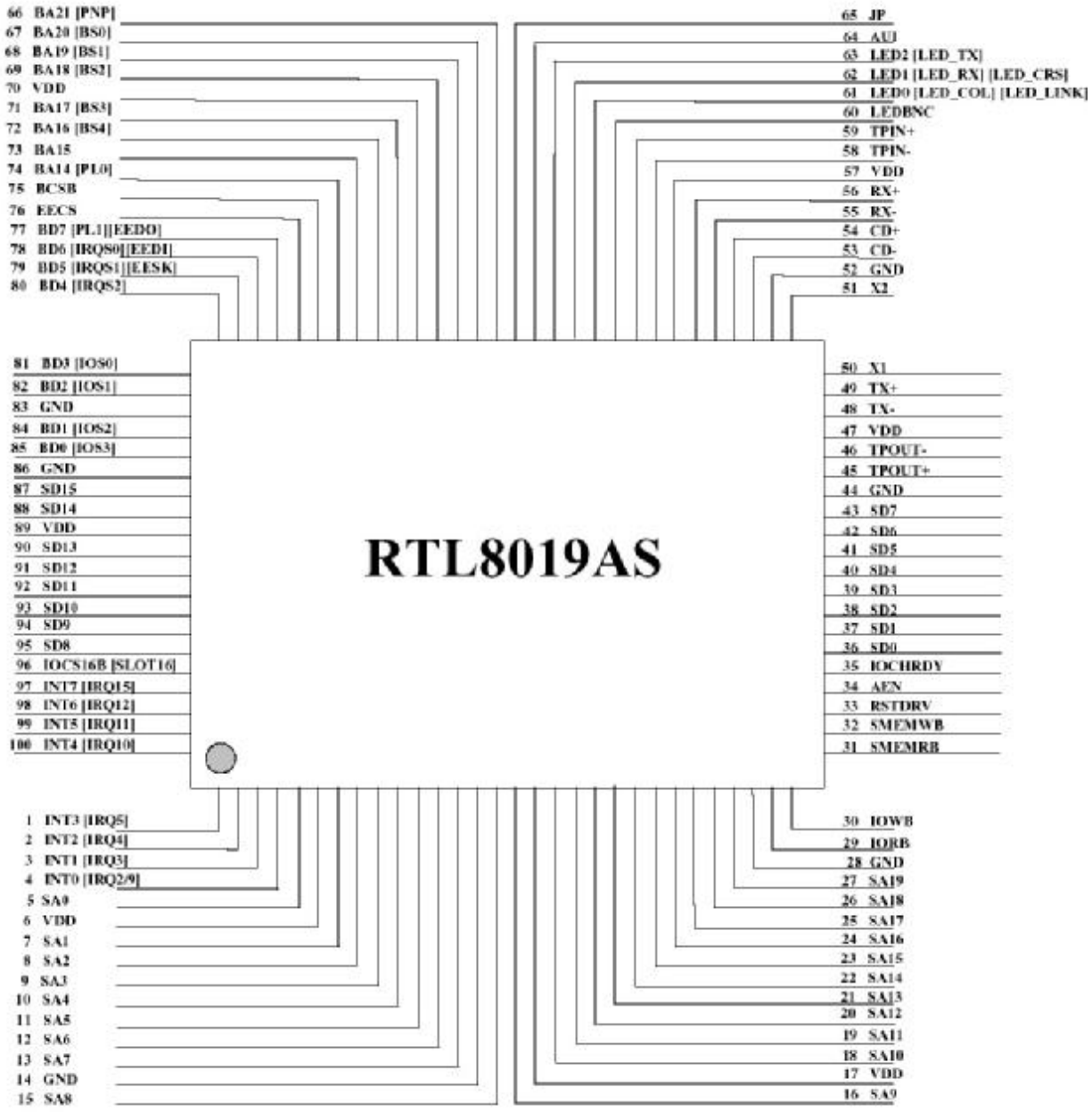


附录 A.4　MSCAN08 控制寄存器列表

地址	寄存器名		Bit7	6	5	4	3	2	1	Bit
$0500	CMCR0	R	0	0	0	SYNCH	TLNKEN	SLPAK	SLPRQ	SFTRES
		W								
$0501	CMCR1	R	0	0	0	0	0	LOOP	WUPM	CLKSRC
		W								
$0502	CBTR0	R	SJW1	SJW0	BPR5	BPR4	BPR3	BPR2	BPR1	BPR0
		W								
$0503	CBTR1	R	SAMP	TSEG22	TSEG21	TSEG20	TSEG13	TSEG12	TSEG11	TSEG10
		W								
$0504	CRFLG	R	WUPIF	RWRNIF	TWRNIF	RERRIF	TERRIF	BOFFIF	OVRIF	RXF
		W								
$0505	CRIER	R	WUPIE	RWRNIE	TWRNIE	RERRIE	TERRIE	BOFFIE	OVRIE	RXFIE
		W								
$0506	CTFLG	R	0	ABTAK2	ABTAK1	ABTAK0	0	TXE2	TXE1	TXE0
		W								
$0507	CTCR	R	0	ABTRQ2	ABTRQ1	ABTRQ0	0	TXEIE2	TXEIE1	TXEIE0
		W								
$0508	CIDAC	R	0	0	IDAM1	IDAM0	0	0	IDHIT1	IDHIT0
		W								
$0509	Reserved	R	R	R	R	R	R	R	R	R
		W								
$050E	CREXCRR	R	RXERR7	RXERR6	RXERR5	RXERR4	RXERR3	RXERR2	RXERR1	RXERR0
		W								
$050F	CTXRR	R	TXERR7	TXERR6	TXERR5	TXERR4	TXERR3	TXERR2	TXERR1	TXERR0
		W								
$0510	CIDAR0	R	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
		W								
$0511	CIDAR1	R	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
		W								
$0512	CIDAR2	R	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
		W								
$0513	CIDAR3	R	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
		W								
$0514	CIDMR0	R	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
		W								
$0515	CIDMR1	R	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
		W								
$0516	CIDMR2	R	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
		W								
$0517	CIDMR3	R	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
		W								

附录 B RTL8019AS 相关资料

B.1 RTL8019AS 的管脚图



附录 B.2 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						

附录C LCD解压算法

```
//-------------------------------------------------------------------------//
//程序名：unsigned char * LineUnzip(unsigned char *ZipData,unsigned char
//           *UnzipData,unsigned char width)
//功能：解压一行
//入口：unsigned char *ZipData -- 待解压数据首地址
//      unsigned char width -- 行宽度(以像素点而不是字节为单位)
//出口：unsigned char *UnzipData  -- 解压后数据首地址
//      unsigned char * LineUnzip -- 返回下一行第一个需处理的字节地址
//-------------------------------------------------------------------------//
unsigned char * LineUnzip(unsigned char *ZipData,unsigned char *UnzipData,
      unsigned char width)
{
    unsigned char i,j,t1,t2=0;  //循环变量
    unsigned char bits=0;       //当前需处理的点阵数(已经解压的点阵数)
    unsigned char curData=0;    //缓冲区当前位置(字节)的数据
    unsigned char posByte=0;    //当前bit位置(0-width)，一行为width点阵
    unsigned int  temp;         //临时变量,为计算乘法
    t2 = width / 8 ;             //一行解压后的字节缓冲区
    for (t1=0;t1<t2;t1++)  UnzipData[t1]=0;  //清存放解压数据的缓冲区
    while(posByte < width)
    {
        asm("sta $1800");
        t1=ZipData[i];          //当前需处理的字节(首次循环i=0)
        if((t1 & 0x01)==0)     //未压缩
        {
           for(j=0;j<7;j++)
          {
              if ((t1  & 0x80) == 0x00)     //最高位为0
              {
                 curData=Table1[posByte&0b00000111];//当前列应该放置的点阵
                 t2 =(posByte>>3)&0b00011111 ;      //高5位移到低5位(行号)
                 UnzipData[t2]|=curData;            //将当前点填入当前字节
              }
              posByte++,t1 = t1<<1;   //当前需处理的字节左移1位，处理下一位(bit)
          }
        }
        else   //压缩数据
        {
           if((t1 & 0x02)==0)   //普通压缩
              bits=(t1>>2)&0b00011111;//得到本次需处理的位数(bit2-6)，0或1由t1.7指定
           else                 //特殊压缩
```

```
        {
            curData =0x00;
            if ((t1&0b00000100)!=0)
                curData=curData|0b00000100;  //bit.2 = curData.2
            if ((t1&0b00001000)!=0)
                curData=curData|0b00000010;  //bit.3 = curData.1
            if ((t1 & 0b00010000)!=0)
                curData=curData|0b00000001;  //bit.4 = curData.0
            //此时当前行剩下的点数x中有curData/8 *x同为t1.7所表示的位
            t2 = width - posByte;          //当前行还剩下的点数
            //根据公式bits=(8-curData)*t2)得到本次需处理的t1.7的位数
            curData = 8- curData,temp = (curData * t2) / 8;
            bits = (unsigned char)temp;  //将计算后得到的本次需处理的位数
                                         //存入bits中,是0或1由t1.7指定
        }
          //已经得到当前需处理字节中包含t1.7的个数为bits；若t1.7=0，是亮点，
          //需要进一步的处理；若为1，表示不亮，则点阵指针直接加bits即可
          if (t1<128)    //小于128，表示t1.7=0
          {
              curData= (posByte & 0b00000111);  //分离出当前列号
              t2= ((posByte >>3 )& 0b00011111); //分离出当前行号
              //----以下是补充上一次最后字节的零头  begin----//
              curData = 8-curData;          //得到当前需补充的列数
              bits = bits-curData;          //剩余的列数
              posByte =posByte + curData;     //当前点阵指针加上刚补充的列数
              curData=Table2[curData];        //得到当前实际应填充的点阵
              UnzipData[t2] = UnzipData[t2]|curData;  //将当前点阵填充到当前字节
             //----以下是补充上一次最后字节的零头  end----//
             curData=(posByte>>3)&0b00011111;//得到补充点阵后的新的行号
             while ( bits >= 8)               //剩下的点阵数够1字节，则进入循环
              {
                   UnzipData[curData]=0xff;  //用1填充当前字节的所有位
                   curData++;
                   posByte += 8, bits -= 8;  //当前点阵位置加8，剩余点阵数减8
              }
              UnzipData[curData]=Table3[bits];//本次剩下未足8位的点阵数
          }
          posByte = posByte + bits;    //当前点阵位移加上已经处理的点阵数
      }
      i++;       //欲处理的字节的指针加1,以便处理下一未处理的数据
  }
  return (&ZipData[i]);    //返回下一行第一个需处理的字节地址
}
```

攻读学位期间公开发表的论文及参与的鉴定项目

[1] 汤龙梅、王宜怀，基于非接触式 IC 卡的会议签到系统的设计和实现，军民两用技术与产品，2003 年第 12 期。

[2] 郑春芳、汤龙梅，基于Freescale 新型 MCU 的小型发电机组控制屏的研制，现代电子技术，2005 年第 4 期。

[3] 参与《嵌入式网关通用接口》项目，该项目已于 2004 年 12 月 17 日通过江苏省科学技术厅鉴定。

[4] 参与王宜怀、刘晓升编著的《嵌入式应用技术基础教程》第 15 章 CAN 总线通信接口的撰写，该书将于 2005 年 7 月在清华大学出版社出版。