2009 年度专业硕士学位论文　　学校代码：10269

学　　号：63061500249

華東師範大學

DeviceNet 总线技术研究及其在嵌入式系统中应用

院　　系：软件学院

类　　别：工程硕士

领　　域：软件工程

指导教师：郭 建 副教授

申 请 人：宋 富

2009 年 4 月完成

2009 Professional Master's Degree Thesis University Code：10269

Student ID：63061500249

East China Normal University

DeviceNet Research And Its Application In Embedded System

Department： Software Engineering Institute

Type： Master of Engineering

Domain： Software Engineering

Supervisor： Jian Guo Asso. Prof.

Applicant ： Fu Song

2009.4

学位论文独创性声明

本人所呈交的学位论文是我在导师的指导下进行的研究工作及取得的研究成果。据我所知，除文中已经注明引用的内容外，本论文不包含其他个人已经发表或撰写过的研究成果。对本文的研究做出重要贡献的个人和集体，均已在文中作了明确说明并表示谢意。

作者签名： 宋富 **日期：** _____

学位论文授权使用声明

本人完全了解华东师范大学有关保留、使用学位论文的规定，学校有权保留学位论文并向国家主管部门或其指定机构送交论文的电子版和纸质版。有权将学位论文用于非赢利目的的少量复制并允许论文进入学校图书馆被查阅。有权将学位论文的内容编入有关数据库进行检索。有权将学位论文的标题和摘要汇编出版。保密的学位论文在解密后适用本规定。

学位论文作者签名： 宋富 **导师签名：** 郭建

日期： _____ **日期：** _____

宋 富 硕士学位论文答辩委员会成员名单

姓名	职称	单位	备注
贾益刚	高 工	上海环保局信息中心	
姜宁康	副教授	华东师范大学软件学院	
王长波	副教授	华东师范大学软件学院	
全红艳	副教授	华东师范大学软件学院	
琚小明	副教授	华东师范大学软件学院	

摘　要

现场总线作为工业数据总线，是自动化领域中计算机通信体系最低层的网络。根据国际电工委员会(IEC)标准和现场总线基金会(FF)的定义：“现场总线是连接智能现场设备和自动化系统的数字、双向传输多分支结构的通信网络”。目前，现场总线成为自动化技术发展的一个热点。

DeviceNet 现场总线是 Rockwell 公司提出的一种现场总线，具有开放、高效和低价等特点，特别适合于制造业、工业控制和电力系统等应用，对于低压开关设备和控制设备，具有非常广阔的发展前景。在中国，ODVA(Open DeviceNet Vendor Association)建立了 ODVA China，并于 2003 年成为国家标准。

本课题的研究有志于进一步提高国内工业自动化技术的应用水平，提高效率并降低成本，同时对国内相关领域技术水平与国外先进水平保持同步也具有重要意义。本文主要工作如下。

首先，概要介绍了现场总线的概念、国内外的发展现状和特点，并介绍了 DeviceNet 现场总线的由来和技术特点，从而论述了本文的意义。

其次，深入研究分析 DeviceNet 现场总线通讯协议；按照该协议，参与设计开发主站硬件平台，并在该平台上独立设计开发底层 CAN 驱动、主站 DeviceNet 应用层协议及 DeviceNet 网络管理与数据采集平台；并与从站一起组成了一个典型的主/从网络，定制应用于环境检测，从而实现提供了一种易实现、高可靠、低成本的现场总线解决方案。

最后，对本课题的研究进行总结，并就目前 DeviceNet 总线产品开发的趋势和热点给出了一些看法；对 DeviceNet 总线的数据采集平台在嵌入式系统中的实现进行了尝试，希望对国内总线产品的开发能起到抛砖引玉的作用。

【关键词】：现场总线， DeviceNet， 主站， 主/从网络

【论文类型】：应用基础与应用技术研究

Abstract

FieldBus is the bottom network of Computer Communication System in the field of automation.According to the International Electrotechnical Commission standards and Fieldbus Foundation definition:'Fieldbus is a way to connect instruments in a manufacturing plant. Fieldbus works on a network structure which typically allows daisy-chain, star, ring, branch, and tree network topologies'.Nowadays, the Fieldbus is becoming a focus of the automation technological development.

DeviceNet Fieldbus invented by Rockwell is one of the Fieldbus. It`s open, efficient, low-cost, and suitable for manufacturing sector, industrial control and power systems, and it has very wide prospects for development of Low-voltage switchgear and control equipment. In China, ODVA (Open DeviceNet Vendor Association) had established the ODVA China, and became the national standard in 2003.

The purpose of this paper is to improve the applications of the domestic industrial automation technology, to improve the efficiency and to reduce the costs. It is of great significance in keeping synchronization of technology between home and abroad. The main tasks of the paper are listed as follows.

First introduce the main concept, characteristics, development status of Fieldbus.Then introduce the origin and characteristics of the DeviceNet Fieldbus. Consequently, discusse the significance of the paper.

And then, research and analysis the DeviceNet Fieldbus protocol. According to the protocol, participated in designing and developing the hardware platforms, and designed and developed the Master`s CAN driver, the protocol of the DeviceNet application layer and the DeviceNet network management and application systems on my own. With the Slave constructed a typical Master/Slave network, thereby an easy, high reliable and low-cost fieldbus solution achieved.

Finally, show the conclusion of the study, the prospect in future, and some authors` opinions on DeviceNet FieldBus products.

【KEY WORD】: Fieldbus, DeviceNet, Master, Master/Slave Network

【THESIS TYPE】: Application infrastructure and technology research

目 录

第1章 绪 论

1.1 引言

在现代工业生产中，自动化技术是保证工业生产高质、高效、安全运行的重要手段。八十年代末九十年代初国际上发展形成用于过程自动化、制造自动化、智能楼宇的现场总线(Fieldbus)，正逐步取代七八十年代的DCS(Distributed Control System)并推动着现代工业控制技术的又一次飞跃[1]。

DeviceNet是基于CAN(Controller Area Network)总线的一种高效、可靠、易用、廉价的现场总线，具有互换性和互操作性，为用户提供了完整的设备级诊断功能。在许多控制领域获得广泛的认同，已成为事实上的工业自动化领域的标准网络。

1.2 现场总线简介

1.2.1现场总线的概念

现场总线是应用在生产现场、智能现场设备之间实现双向串行多节点数字通讯的低成本底层控制网络，已成为当今自动化领域的技术热点之一，它的出现标志着工业控制技术领域又一个新时代的开始，将对现代工业控制领域产生重要影响。

现场总线的标准实质上并没有统一过，所以对现场总线的定义也有很多种，以下是几种具有一定代表性的定义：

- 根据国际电工委员会IEC/ISA的定义，现场总线是指连接测量、控制仪表和设备，如传感器、执行器和控制设备的全数字化、串行、双向的通信系统。
- 根据Standard of Practice50对现场总线的定义，现场总线是一种串行的数字数据通信链路，它沟通了过程控制领域的基本控制设备(现场级设备)之间以及更高层次自动化控制领域的自动化控制设备(高级控制层)之间的联系。
- 现场总线是应用在生产现场、在微机化测量控制设备之间实现双向串行多节数字通信的系统，也被称为开放式、数字化、多点通信的底层控制网络。

现场总线的定义各不相同，但可归纳为现场总线是应用于工业底层的控制和

测量设备、实现了现场设备之间以及其高层自动化控制设备之间互连、数字化、串行、多点通讯的数据总线[2]。

1.2.2现场总线的发展现状

现场总线于80年代中期始于欧洲，90年代初期形成了几种较有影响的标准，如ProfiBus，FIP，ISP，ISA/SP50。另外还有一些公司自己推出的现场总线产品，形成了事实上的标准，影响较大的有CAN，LonWorks和HART。1994年ISP与FIP的北美和欧洲分会宣布合并，成立FF(Foundation Fieldbus)，许多国际知名的仪表和控制系统公司如Honeywell，Rosemount等加盟，形成了与ProfiBus相抗衡的两大现场总线阵营。

现场总线技术发展至今，大大小小已有40余种，形成优势的是FF和ProfiBus两大阵营，另外CAN、LonWorks、HART等一批实力雄厚的产品也在大量的应用中深入人心，但还没有统一的国际标准，这并没有影响它的飞速发展，因为该技术本身的优越性正是支持它发展的强大动力[2][3][4][5]。

除了以上叙述的常见的现场总线产品外，其它还有很多，如Phenix Contact公司的InterBus，丹麦Process-DataSikeborg Aps公司的P-NET，英国的国家标准ERA，挪威的国家标准FINT等。

国内现场总线发展起步较晚，目前有周立功单片机有限公司研发的iCAN，主要用于教学设备和高校研究。

现场总线应用范围已含盖了工业生产和社会生活的各个领域，每种现场总线技术都有其自身的技术特点和传统的应用领域，如CAN源于汽车工业，LonWorks大量应用于智能楼宇。同时每一类现场总线技术均以国际大公司为背景和依托，如Siemens主推ProfiBus，Fisher Rosemount主推FF。正是由于每种技术都与其背后大公司的利益息息相关，每种技术的成败得失都影响着巨额利润的最终分配，所以各类现场总线技术合纵连横，如同一场没有硝烟的战争。在竞争的同时，各种现场总线技术也在不断的交叉与融合，如Fisher Rosemount Alstom Yokogawa正在开发ProfiBus的产品；Smar的产品不仅支持FF、HART协议，同时也支持ProfiBus协议；Siemens的系统不仅支持ProfiBus，同时也支持DeviceNet、CAN等。所以这也是未来一段时间内现场总线技术的发展方向，各类技术将取长补短，共同生存。

现存的各类现场总线技术都包含着大量的资金投入，都有大批的仪器仪表和软硬件资源，所以大家都希望成为未来的国际标准，但目前现场总线群雄并起的局面还将存在一段时间。现实存在的大量传统模拟仪表要逐步被智能数字仪表取代，旧的现场总线仪表也将被新的仪表取代，传统的DCS将向FCS转化，而在过渡时期，混合控制系统将占主导地位。

1.2.3现场总线的特点

现场总线作为新型的底层控制网络，具有以下多种特点。

数字化的信号传输：无论是现场底层传感器、执行器、控制器之间的信号传输，还是与上层工作站及高速网之间的信息交换，全部使用数字信号。在网络通信中采用信息防撞与纠错技术，实现了高速及双向多点之间的可靠通信。与传统的DCS相比，在通信质量和连线方式上都有重大的突破。

分散的系统结构：废除传统DCS中采用的“操作站-控制站-现场仪表”三层主/从结构模式，把I/O及控制功能分散到现场智能仪表中，它们独立完成测量、校正、调节、诊断等功能，同时用网络协议把它们连接在一起统筹工作。任何一个节点出现故障只影响本身而不会危及全局，这种彻底的分散控制体系使系统更加可靠。

方便的互操作性：现场总线技术强调“互联”和“互操作性”。也就是说，不同厂商的产品可以异构的组成统一的系统，可以相互操作，统一组态，打破了传统DCS产品互不兼容的缺点，极大的方便了用户。

开放的互联网络：现场总线技术及标准是全开放式的。从总线标准、产品检验到信息发布都是公开的，面向所有的产品制造商和用户。通信网络可以和其它系统或高速网络相连接，用户可共享网络资源。

多种传输媒介和拓扑结构：现场总线技术由于采用数字通信方式，因此可采用多种传输介质进行通信。根据控制系统中节点的空间分布情况，可采用多种网络拓扑结构。这种多样性给自动化系统的施工带来了极大的灵活性。

简化结构降低费用：现场总线技术由于采用智能仪表和总线传输方式，可以减少大量的隔离器、端子框及电缆等，从而降低系统复杂程度，节省施工费用[4]。

1.3 DeviceNet 总线简介

从OSI网络模型的角度来看，现场总线网络一般只实现了第1层(物理层)、第

2层(数据链路层)和第7层(应用层)。因为现场总线通常只包括一个网段，因此不需要第3层(传输层)和第4层(网络层)，也不需要第5层(会话层)和第6层(描述层)的作用。

CAN现场总线仅仅定义了第1层和第2层(见ISO11898标准)，实际设计中，这两层完全由硬件实现，设计人员无需再为此开发相关软件(Software)或硬件(Firmware)；而没有规定应用层，其本身并不完整，需要一个高层协议来定义CAN报文中的11/29位标识符、8字节数据域的使用。CAN总线在工业自动化应用中越来越需要一个开放的、标准化的高层协议，这个协议支持各种CAN总线厂商设备的互用性、互换性，能够在CAN总线网络中提供标准的、统一的系统通讯模式、提供设备功能描述方式和执行网络管理功能。

基于这种需求，提出了多种基于CAN总线的高层协议，如CiA(CAN-in-Automation)定义的标准CANOpen，美国汽车工程师协会——卡车和公共汽车电气电子委员会下的卡车和公共汽车控制和通讯网络分委员会制定的高层CAN网络通讯协议SAE J1939，最初有美国的Rockwell自动化公司开发应用、目前由ODVA组织管理和推广的DeviceNet。本文将进一步对DeviceNet协议深入研究[3][5]6]。

DeviceNet是一个开放式的协议，只要付出象征性的资金获得ODVA的一个许可证，就可以得到DeviceNet协议的详细内容。DeviceNet属于CIP(Control and Information Protocal)网络的范畴，CIP网络有下列特点：

- 报文的传输类型有I/O、互索、配置、程序上下载；
- 是一个面向连接的协议，必须要先建立，才能通信；
- 采用生产者/消费者通信模型；
- 可以支持主从，多主，对等，或者三种模式的任意组合；
- 面向对象编程等特点[3]。

DeviceNet以CIP协议为基础，沿用了CAN协议标准所规定的ISO参考模型中物理层和数据链路层的一部分，并补充了不同报文的传送格式、总线访问仲裁规则及故障检测和隔离的发法。DeviceNet协议增加了传输介质的协议规范，每个网段最多只允许有64个节点，采用干线与支线方式进行网络拓扑；5线制总线结构(2信号线，2电源线，1屏蔽线)，总线支持125kbps、250kbps、500kbps三种波

特率；最大传输距离为500米，支持总线供电，又由于它采用5线制，在实际的应用中接错的可能性更大，所以要求节点能够承受由于任意的2条线误接而产生的电压，所以在这个原因下，要求在实际的应用中的收发器一定要符合DeviceNet规范的规约，要求收发器能承受2条信号线的误接线产生的电压。DeviceNet有节点的接地和隔离，即任一设备必须要有隔离栅，以及节点的误接线保护电路[3]。

1.4 研究内容及意义

本文受上海市科委“登山行动计划——高可信芯片设计前端平台与工业控制芯片设计应用”资助，研究基于ARM9微处理器的DeviceNet网络远程数据采集与控制的关键技术，解决相应网络通信和数据传输关键技术，为实现远程设备设置、远程数据采集、远程故障信号传递和远程控制奠定技术基础。本次课题研究内容包括以下几个：

- 深入研究DeviceNet协议，分析DeviceNet协议体系结构；
- 根据研究成果，以三星公司的S3C2440 ARM9微处理器、MCP2510 CAN控制器和Windows CE.Net4.2(RTOS)为开发平台，设计开发DeviceNet协议主站；
- 在主站此基础上设计开发DeivceNet网络管理与数据采集平台；
- 结合从站完成系统的组态，并将系统应用于实际环境检测中，实现从站配置管理、网络扫描列表管理、从站数据采集、数据的图形化显示、数据异常处理、最新若干组数据实时保存至内存和SD卡存储器，同时在主站上扩展实现视频图像的采集。

目前现场总线正朝着开放统一的方向发展，然而在不同的领域有着各自的特点。本文以ARM9微处理器为DeviceNet主站平台，实现DeviceNet网络管理与数据采集是一次全新的尝试，在一定程度上推动了DeviceNet总线在嵌入式系统中的应用和发展。

1.5本文创新点

本文具有下述几个创新点：

自成体系的现场总线组网方式：目前比较常用的CAN总线网络有a)、b）两种网构方式，而本文将在嵌入式系统中实现主站、开发扫描软件，既降低了网络费用，又能提高系统灵活性。

a) PC(Personal Computer)+组网软件+PLC (Programmable Logic Controller);

b) PC+组网软件＋PC卡;

数据采集扩展支持视频图像采集：支持多路视频图像数据采集，通过总线供电、视频图像信号线分离、DMA直接存储方式加快视频图像处理。

易于移植：本文实现的是一个网络管理与数据采集平台，可根据实际应用需要，定制不同的数据处理模块，增强了系统的可重用性。

1.6 本文结构

第一章介绍了现场总线的概念、现状、发展和特点，列举了目前典型的几种现场总线；简单介绍了DeviceNet总线技术的由来和特点；阐述了本文研究的内容、意义与本文结构。

第二章介绍DeviceNet协议原理和技术规范；分别从物理层、数据链路层和应用层进行论述，同时介绍了DeviceNet总线设备分类和预定义主/从连接组。

第三章DeviceNet主站整体设计；搭建开发平台，主站硬件平台设计与实现，CAN总线和视频硬件模块的结构与功能。

第四章DeviceNet主站软件设计实现；CAN和视频驱动的设计实现，主站软件需求分析，对系统软件功能进行UML建模，设计实现各模块功能。

第五章DeviceNet主站的网络管理与数据采集平台应用于环境检测。

第六章结论与展望。

第 2 章 DeviceNet 协议规范

CAN 总线是 20 世纪 80 年代中后期发展起来的，已成为开放的国际标准通信协议(ISO11898)，并应用于众多领域。许多著名的半导体制造商生产 CAN 控制器或集成 CAN 控制器的芯片，为 CAN 技术的应用打下了坚实的技术[7]。

然而CAN总线技术并未定义流量控制、数据分割、节点地址分配、通讯控制等具体内容，正因为CAN总线技术没有定义这些技术的具体内容，增强了CAN总线技术应用的灵活性，在基于CAN总线技术的高层协议中，各厂商可以根据自己的需要自行开发。DeviceNet协议是20世纪90年代中期发展起来的一种基于CAN2.0A总线协议的高层协议[9]。

2.1 DeviceNet 的物理层

DeviceNet协议是基于CAN总线技术的，而CAN总线只采用了ISO/OSI网络模型的物理层和数据链路层，DeviceNet在CAN的基础上增加了应用层，扩充了物理层的连接单元接口规范、媒体连接和媒体规范[4][9]。

DeviceNet物理层的主要功能是利用物理传输介质为数据链路层提供物理连接，主要包括收发器、连接器、错线保护电路、电压调节器、传输介质和可选的隔离线路，如图2-1所示，DeviceNet物理层具有如下技术特点[10]。

- 主线-分支混合结构；
- 同网段最多支持64个节点；
- 同时支持网络供电和设备自供电；
- 支持开放式和密封式连接器；
- 连线错误保护；
- 提供125kbps、250kbps和500kbps三种可选波特率，对不同波特率传输距离不同，如表2-1所示：计算公式为：

  lthick+5×lthin<=500m　　125kbit/s

  lthick+2.5×lthin<=250m　　250kbit/s

  lthick+lthin<=100m　　500kbit/s

  这里lthick是粗缆长度，lthin是细缆长度[11]。

- 电源可配置，链路电源的额定电压为直流 24V，粗缆干线部分可以支持 8A 电流，细缆干线部分可以支持 3A 电流，也可以使用多路电源供电；

最大支线电流受下面方程式的约束：I=4.5/l，这里：I 是允许的最大支线电流(A)，l 是支线长度(m)；

- 内置超载保护；
- 电源分流器支持多路转接；
- 支持设备热插拔[12][13]。

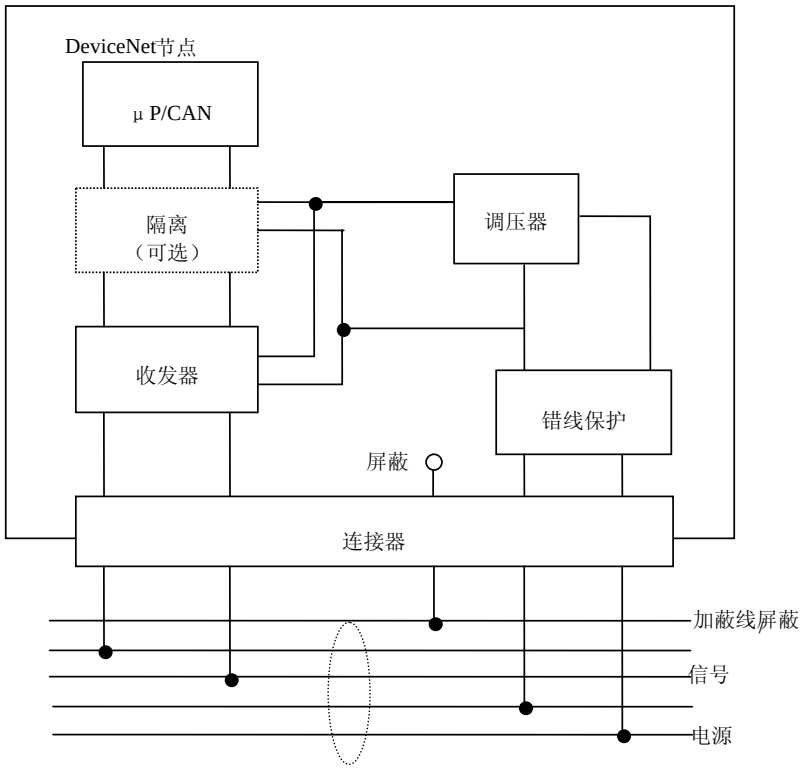

图 2-1 DeviceNet 物理层结构图

表2-1 波特率与传输距离关系表

波特率	主线最大距离	分支距离	
		最大	累计
125kbps	500m	6m	156m
250kbps	250m		78m
500kbps	100m		39m

2.2 DeviceNet 的数据链路层

DeviceNet数据链路层完全遵循CAN协议规范，并通过CAN控制器实现。CAN数据链路层分逻辑链路控制层和媒体访问控制层。逻辑链路控制层主要功能是报文过滤、报文处理和提供应用程序接口等；媒体访问控制层是数据链路层的核心，

主要功能是定义了帧结构、仲裁和检错等传送规则[14]。

DeviceNet是基于CAN2.0A规范的协议，因此DeivceNet数据帧标识符是CAN2.0A定义的11位标识符，每个节点根据这个标识符来确定是否接收这个数据帧，这些标识符定义于数据帧的仲裁域[13]。

DeviceNet仅支持CAN定义的四种帧结构中的数据帧、超载帧和错误帧，不支持远程帧。数据帧的结构如图2-2所示[12]。

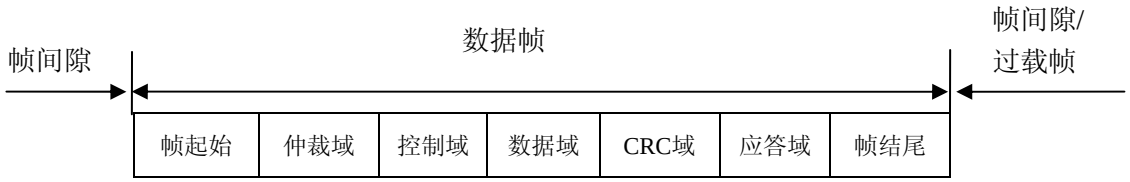

图2-2数据帧结构图

- 帧起始：它标志数据帧和远程帧的起始，由一个单独的“显性”位组成。
- 仲裁域：仲裁域包括标识符(ID)和远程发送请求位(RTR)。标识符(ID)的长度为11位，这些位的发送顺序是从ID-10到ID-0，最低位是ID-0，最高的7位(ID-10到ID-4)必须不能全为“隐性”。RTR位在数据帧里必须为“显性”，而在远程帧里必须为“隐性”。仲裁域主要用于表示信息内容及信息优先级。
- 控制域：控制域由6位组成，包括数据长度代码和两个将来作为扩展用的保留位。所发送的保留位必须为“显性”。数据长度代码指示了数据域中字节数量，长度为4位。
- CRC域：CRC域包括CRC序列，其后是CRC界定符，它包含一个单独的“隐性”位。
- 应答域：应答域长度为2位，包含应答间隙和应答界定符。在应答域里，发送站发送两个“隐性”位。当接收器正确地接收到有效的报文，接收器就会在应答间隙期间发送ACK信号，向发送器发送一“显性”的位以示应答。
- 帧结尾：每一个数据帧或远程帧均由一标志序列界定。这个标志序列由7个“隐性”位组成。

CAN节点发送的数据可以被总线上任何节点监听和应答，同一时刻只能有一个节点发送数据，若出现同时发送，则通过仲裁域无损逐位仲裁算法解决。

CAN提供了位错误、应答错误、CRC错误、填充错误和格式错误五种错误类型，提供了一种故障界定状态机制，有错误激活、错误认可和离线三种错误状态。

2.3 DeviceNet 的应用层

2.3.1DeviceNet标识符分组

DeviceNet是基于CAN总线的高层协议，对CAN2.0A规范的11位标识符含义进行了扩充，除了原有的总线仲裁和信息标识外，将11位标识符划分成MAC ID(Media Access Control Identity)表示媒体访问控制标识符和报文标识符信息ID，并定义了如图2-3四个报文组[16][18][19]。

<table>
<tr><td colspan="11">标识位</td><td rowspan="2">十六进制
范围</td><td rowspan="2">使用组别</td></tr>
<tr><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td></td><td colspan="4">组 1 信息 ID</td><td colspan="6">源 MAC ID</td><td>000-3ff</td><td>信息组 1</td></tr>
<tr><td>1</td><td>0</td><td colspan="6">MAC ID</td><td colspan="2">组 2 信息 ID</td><td>400-5ff</td><td>信息组 2</td></tr>
<tr><td>1</td><td>1</td><td colspan="3">组 3 信息 ID</td><td colspan="5">源 MAC ID</td><td>600-7bf</td><td>信息组 3</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td colspan="5">组 4 信息 ID（0－2f）</td><td>7c0-7et</td><td>信息组 4</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>×</td><td>×</td><td>×</td><td>×</td><td>7f0-7ff</td><td>无效的 CAN 标识</td></tr>
<tr><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td><td></td><td></td></tr>
</table>

图2-3 DeviceNet报文组

将11位标识符划分成四组后，报文优先级不再仅限于节点的MAC ID，报文优先级机制是标识符小的反而优先级高。

2.3.2DeviceNet报文类型及格式

CAN帧数据域可放0~8字节的数据，DeviceNet定义显式报文和I/O报文两种报文类型，又根据前后报文数据之间的关系定义了分段报文和未分段报文，用于传输不同用途的数据。

2.3.2.1显式报文

显式报文一般用于两个设备之间多用途的信息交换，如节点配置、故障情况和故障诊断，是典型的请求—响应通信方式，一般被赋予较低优先级。DeviceNet定义了一组公共服务显式报文，如读写属性、打开关闭连接和出错应答等。

显式报文根据分段和不分段定义了两种报文数据域格式，如图2-4(a)(b)所示，分段协议如图2-5所示。

字节偏移	7	6	5	4	3	2	1	0
0	Frag	XID	MAC ID					
1~7	I/O 数据							

（a）未分段

字节偏移	7	6	5	4	3	2	1	0
0	Frag	XID	MAC ID					
1	分段协议							
2~7	信息本体分段							

（b）分段

图2-4 显式报文数据域格式

字节偏移	7	6	5	4	3	2	1	0
1	分段类型	分段计数器						

图2-5 分段协议

- XID：是显式报文的事务处理标识位，当主站在某一段时间内未收到先前发送的显式请求报文应答时，会将XID置1，并重新发送该显式请求报文；
- MAC ID：包括源或目的MAC ID，接收一个显式信息时，要检查信息头中的MAC ID区。如果在CAN标识符ID中指定的是目的MAC ID，那么将在信息头中应指定另一个终点节点的源MAC ID。如果在CAN标识符ID中指定的是源MAC ID，那么将在信息头中指定接收模块的MAC ID。上述条件都不成立时，应放弃该报文；
- Frag：是报文的分段标识位，若为0则表示当前报文是未分段报文；若是1则是分段报文，此时需设置分段类型和分段计数器；
- 分段类型：指出该段报文是第一段还是某中间分段，或是最后分段，见表2-2。
- 分段计数器：标记每个独立信息段，这样接收者可以判定是否丢失了某些信息段。如果分段类型为第一段，该区代表一个特殊的含义(如表2-2中所示)。对于系列信息中的每个顺序信息段，信息段计数器的值递增1并且在溢出(信息段计数器值＝(信息段计数值＋1)/64)时复位0[15]。

表2-2分段类型

值	含义
0	第一分段，分段计数区的值应该为 0 或 3F。[1)]
1	某中间段。[2)]
2	最后一段。[3)]
3	分段应答。[4)]

表注：

1） 如果信息段计数器的值为 0，那么该段信息是一系列信息段中的第一个。如果信息段计数的值为 3F，表示它也是系列中的最后一个传送。

2） 该信息段既不是系列中的第一段，也不是最后一段。

3） 标志这是最后一个信息段。

4） 分段信息的接收者用该值来确认一个信息段的接收。

2.3.2.2 I/O报文

I/O报文用于传送实时信息，它格式简单、数据传送快和数据传送量大。DeviceNet定义了多种传送规则，有位-选通、轮询、状态改变和循环，用户可以根据实际应用需要选用合适的通信方式，I/O报文可以选择有应答或无应答方式，无应答方式可以节省时间，但降低可靠性，I/O报文支持点对点和点对多传送，I/O报文一般采用较高优先级ID。I/O报文数据域格式如图2-6(a)(b)所示[15]。

I/O报文分段协议与显式报文分段协议相同，不在此累述。

字节偏移	7	6	5	4	3	2	1	0
0~7	I/O 数据							

（a）未分段

字节偏移	7	6	5	4	3	2	1	0
0	分段协议							
1~7	分段 I/O 数据							

（b）分段

图2-6 I/O报文格式

2.3.3DeviceNet对象模型

DeviceNet协议采用面向对象方法学建模，总线上每个节点通信模型化为对象的集合，节点内部各对象的相互作用结果产生总线行为，而节点内每个对象具有特定的功能和运行方式，同时对象有自己的属性、服务和行为。如图2-7所示DeviceNet对象模型图，标识对象、路由对象、连接对象和DeviceNet对象属于通

信类对象是必须对象，其他如应用对象、参数对象等属于可选对象[18]。

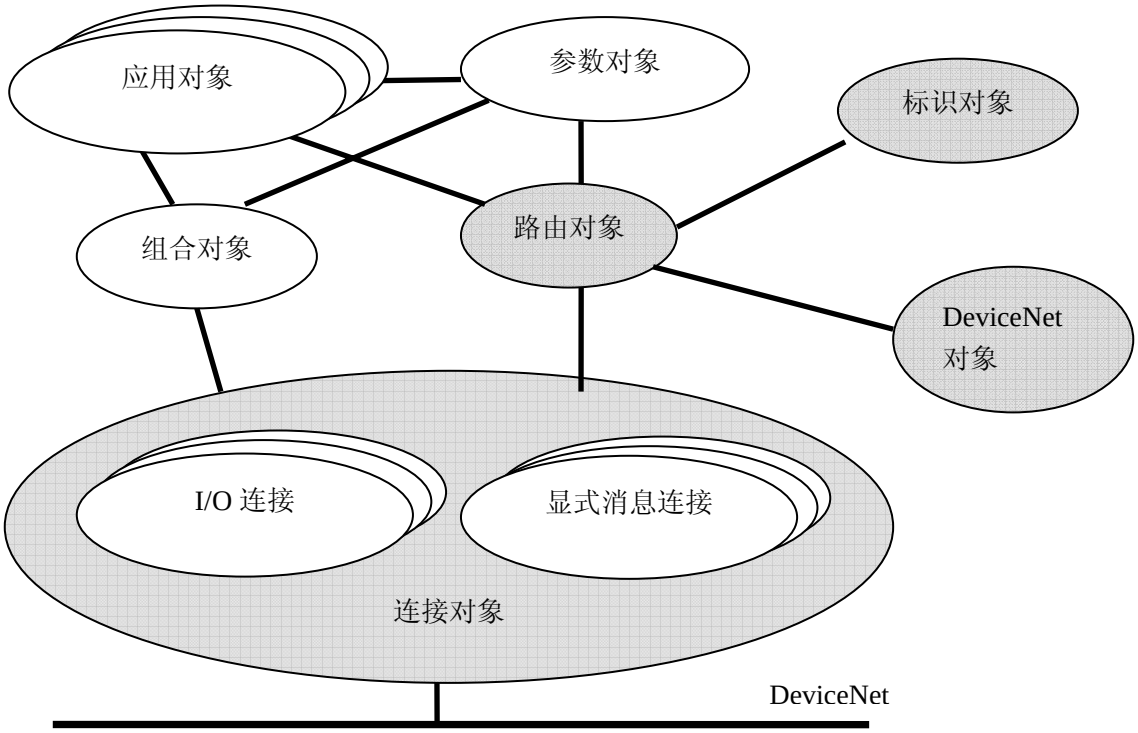

图2-7 DeviceNet对象模型

- 标识对象(Identity)：识别和提供关于设备的一般信息，如生产厂商 ID、产品类型等信息；
- 连接对象(Connection)：管理与提供运行时的信息交换，包含 I/O 连接和显式消息连接两种连接对象，每个连接对象代表 DeviceNet 网络上逻辑连接的一个端点；
- DeviceNet 对象(DeviceNet)：描述节点设备的配置信息和状态，如 MAC ID、波特率等信息；
- 路由对象(Meaasge Route)：向适当的对象发送显式请求消息，它是外部不可视的；
- 组合对象(Assembly)：处理 I/O 连接对象接收发送报文；
- 应用对象(Application)：完成企望的生产目的[18]。

2.3.4DeviceNet 对象编址

2.3.3节简要描述了DeviceNet对象模型，为便于总线访问设备内部对象，DeviceNet定义了这些对象的编址方法，由此将设备之间的信息通信转化为设备内部对象间的通信。基于显式连接未分段报文编址格式如图2-8所示。

字节偏移	7	6	5	4	3	2	1	0
0	分段标志[0]	XID	MAC ID					
1	R/R [0]	服务代码						
	分类 ID							
	实例 ID							
	服务数据[可选]							

图2-8 显式连接未分段报文编址格式

- MAC ID：节点设备介质访问控制标识符，用于区分总线上的节点设备；
- 分类ID：节点设备内部对象标识符，它是分配给节点外部可访问对象类的标识符，用于区分设备内对象类；
- 实例ID：是对象类分配给对象实例的标识符，用于区分对象类的所有实例；
- 服务代码：特定对象类或对象实例功能标识符，用于区分对象类或对象实例所有功能。

2.3.5DeviceNet设备状态机

DeviceNet节点从上电启动到在线建立连接有一个状态转换过程，其状态转换图如图2-9所示，上电或重启后，发送重复MAC ID检查请求报文，进入等待响应状态；此时如果接收到任何重复MAC ID请求或响应消息，就进入通信故障状态，如果在等待重复MAC ID检查响应状态超时，则根据发送次数是否到2次执行不同操作，如果是第一次超时则再次发送重复MAC ID检查请求，计数器加1，并进入等待重复MAC ID检查响应状态；两次发送重复MAC ID检查请求都没有接收到响应且CAN总线正常时，就进入在线状态；在线状态接收到重复MAC ID检查响应也会进入通信故障状态，如果在线状态接收到重复MAC ID检查请求，则发送重复MAC ID检查响应消息[19]。

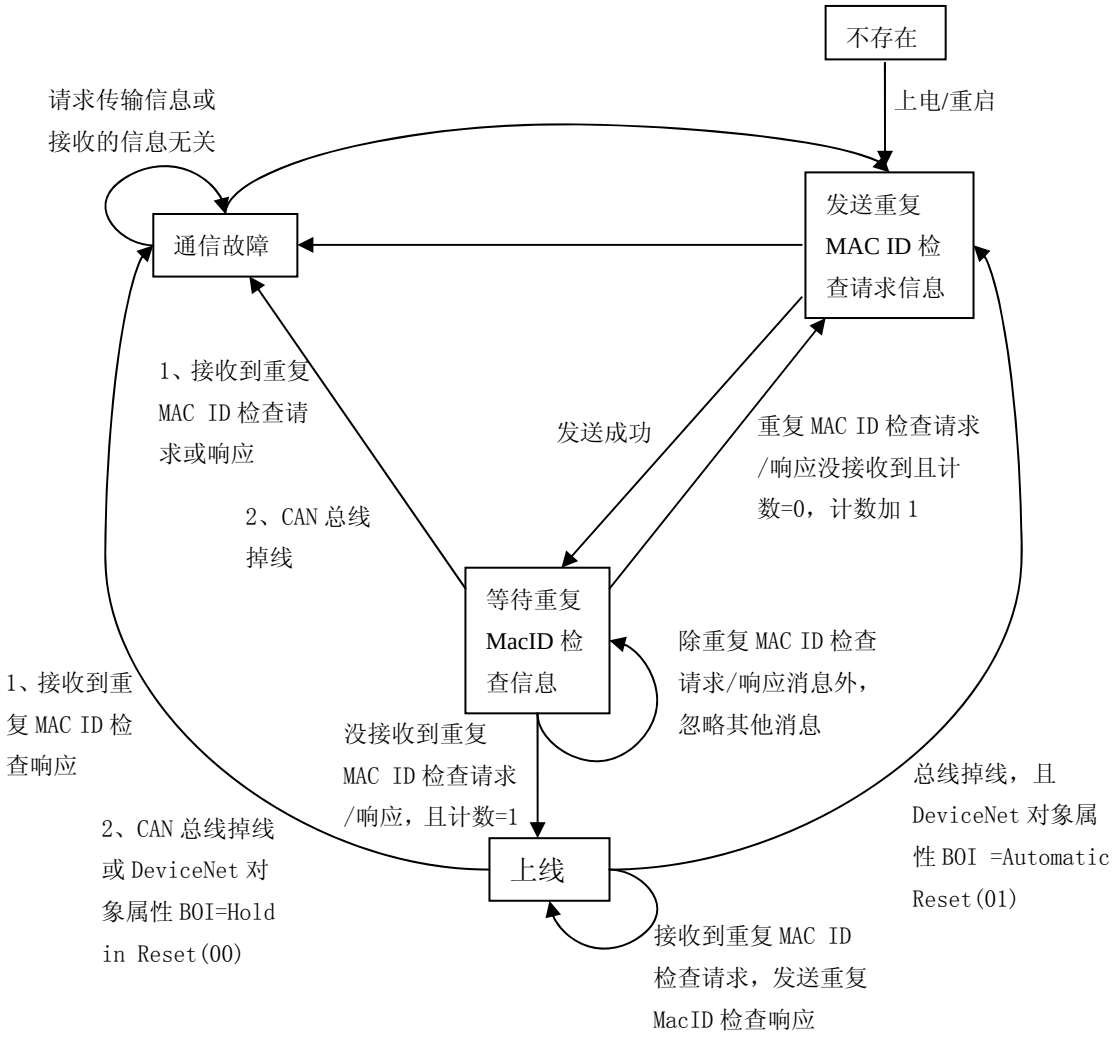

图2-9 DeviceNet节点状态转换图

2.4 预定义主/从连接组

预定义主/从连接组是能方便主/从关系中的常见连接通信，预定义好各报文组内一些ID的功能，省略了创建和配置应用与应用之间连接的许多步骤，减少了网络和设备资源来创建一个通信环境，表2-3是预定义主/从连接组ID分配表[20]。

主站(Master)是指过程控制器收集分配I/O数据的设备。从站(Slave)是指主站从该处收集I/O数据及向它分配数据的设备。而DeviceNet主站和DeivceNet从站分别是Master和Slave的一个应用类型。预定义主/从连接组预定义了一组报文传送机制，其相应的连接对象为：位-选通连接，轮询连接，状态变化/循环连接，多点轮询连接。

表2-3 预定义主/从连接组ID分配表

标 识 位											标识用法	十六进制范围
10	9	8	7	6	5	4	3	2	1	0		
0	组 1 信息 ID				源 MAC ID						组 1 信息	000－3FF
0	1	1	0	1	源 MAC ID						从站 I/O 状态改变或循环信息	
0	1	1	1	0	源 MAC ID						从站 I/O 位-选通响应信息	
0	1	1	1	1	源 MAC ID						从站 I/O 轮询信息或状态/循环应答信息	
1	0	MAC ID						组 2 信息 ID			组 2 信息	400－5FF
1	0	源 MAC ID						0	0	0	主站 I/O 位-选通命令信息	
1	0	多点通讯 MAC ID						0	0	1	主站 I/O 多点轮询命令信息	
1	0	目的 MAC ID						0	1	0	主站状态改变或循环应答信息	
1	0	源 MAC ID						0	1	1	从站显式响应信息	
1	0	目的 MAC ID						1	0	0	从站显式响应信息	
1	0	目的 MAC ID						1	0	1	主站 I/O 轮询命令/状态变化/循环信息	
1	0	目的 MAC ID						1	1	0	仅限组 2 未连接显式请求信息	
1	0	目的 MAC ID						1	1	1	重复 MAC ID 检查信息	

每一个实际存在的连接对象都被赋予了连接实例ID，以此标识连接分类中的多个连接对象，预定义主/从连接组对象实例ID如表2-4所示。

表2-4 预定义主/从连接的新连接实例ID

连接实例 ID#	**描述**
1	标识进入服务器的显式信息连接
2	标识轮询 I/O 连接
3	标识位-选通 I/O 连接
4	标识从站的状态变化或循环 I/O 连接
5	标识多点循环 I/O 连接

对于组2服务器(组2从站)，可通过UCMM先建立显式报文连接，再通过显式报文连接来分配预定义主/从连接组；对于仅限组2服务器，它不支持UCMM功能，只能通过预留的仅限组2未连接显式信息请求ID和响应ID来分配或删除预定义主/从连接组。

第 3 章 DeviceNet 主站整体设计

3.1 引言

DeviceNet现场总线作为底层设备网络，可支持点对点通信、多主或主/从通信，其网络构成方式十分灵活。为了确定整个网络的节点数量、通信波特率、I/O报文的通信方式等参数，在进行网络通信前，需对网络进行组态。目前比较常用的有以下两种网络构成方式：

- PC+组网软件+PLC；
- PC+PC 卡+组网软件。

如图3-1所示是PC+组网软件+PLC网络构成方式，必须有PC机和支持Device-Net的组网软件及相应PLC，通过转接器连接现场总线网络。这种网络构成方式存在系统内构件兼容问题，且成本高，网络结构复杂，给用户带来很大不便。

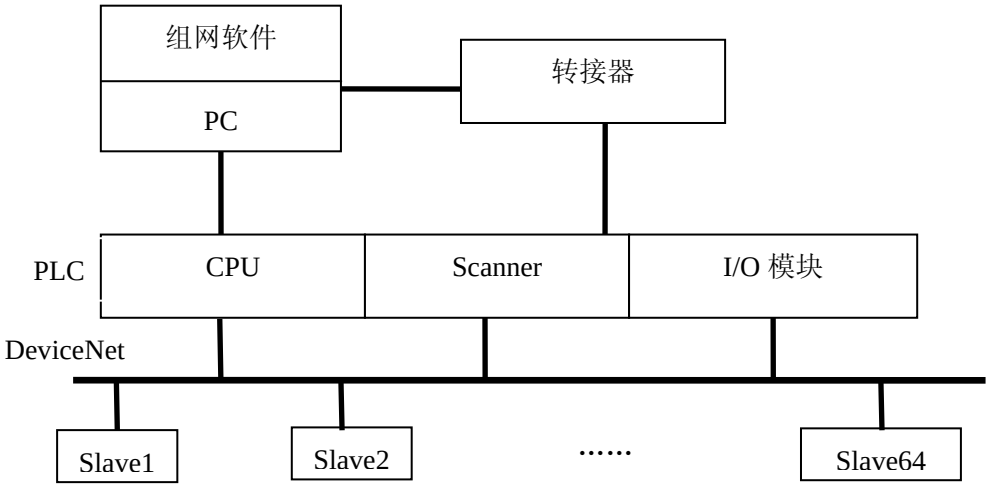

图 3-1 PC+组网软件+PLC 网络构成方式

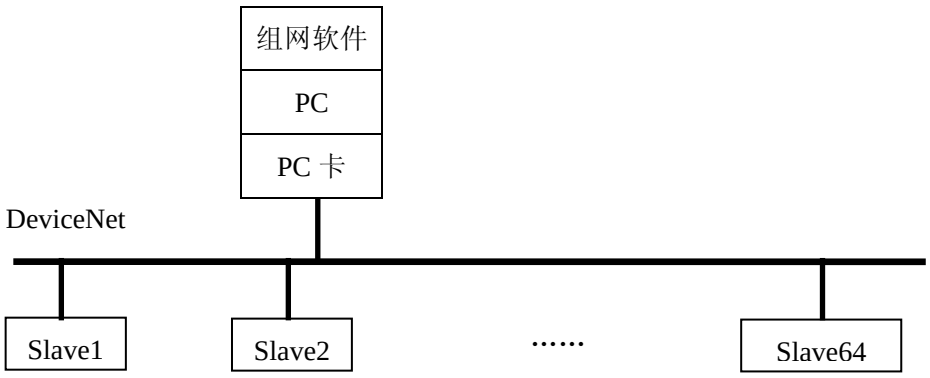

图 3-2 PC+PC 卡+组网软件网络构成方式

如图 3-2 所示是 PC+PC 卡+组网软件网络构成方式，PC 内必须插入支持

DeviceNet 的现场总线接口卡，结合其自带的组网软件构建网络，这种方式必须以 PC 为基础，而有些 PC 卡和软件只支持其自身作为主站，无法构成多主网络，降低了网络的灵活性。

这两种网络构成方式都是基于PC或者带有DeviceNet扫描器的PLC，需要使用通用或专用组网软件，给网络的构建带来很大的不便，增加了组网的成本。本文以嵌入式系统构建DeviceNet网络，在嵌入式系统上开发主站及扫描软件，既降低了网络费用，又能提高系统灵活性[21]。

本文以 ARM9 微处理器和 Windows CE.Net 操作系统为平台，支持 LCD 屏显示，实现 DeviceNet 协议，在 DeivceNet 协议基础上开发 DeviceNet 网络管理与数据采集平台。网络管理支持手动配置网络和导入 EDS 配置文件方式配置网络，在网络管理中可以编辑主站和从站的属性，编辑扫描列表；数据采集平台主要实现总线各节点传感器数据采集和视频图像采集；所开发的网络管理与数据采集平台可应用于多种场合，如工业控制、汽车电子、环境检测等等，本文最后将该平台应用于实际的环境检测。

3.2 开发平台设计

在系统设计开发之前，搭建如图3-3所示开发平台[22]。

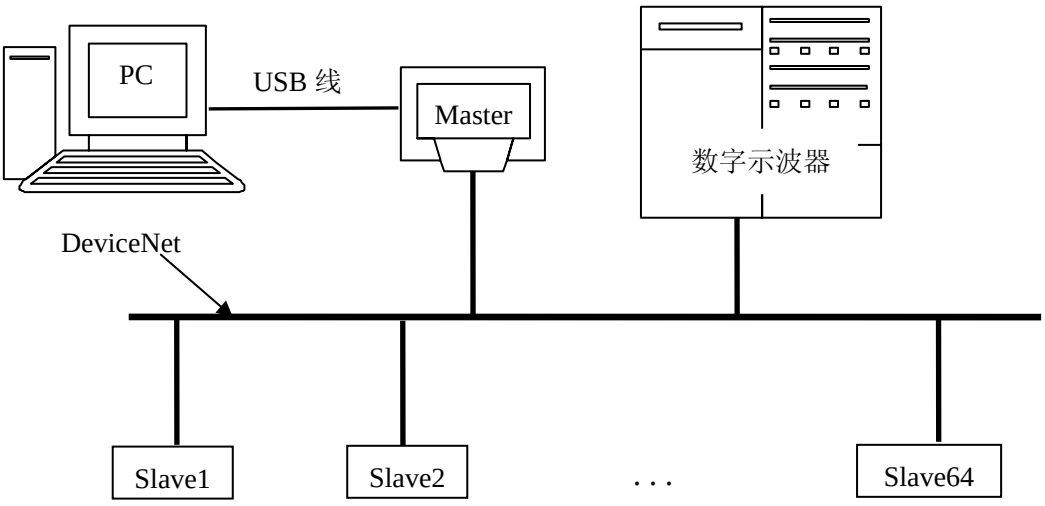

图3-3系统开发平台

本文将在PC上定制主站操作系统Windows CE.Net4.2、开发CAN控制器驱动程序、在Microsoft eMbedded Visual C++4.0开发环境上设计开发主站DeviceNet应用层协议，实现基于DeviceNet协议的网络管理与数据采集平台，定制的Windows

CE.Net4.2内核及应用程序通过USB接口下载至Master硬件平台。

逻辑分析仪用来捕获CAN总线报文、分析报文协议，主要用于测试与调试。Slave(从站)负责数据采集，响应主站I/O获取请求，将数据发送至DeviceNet总线[23]。

3.3 从站设计

DeviceNet 从站采用集成 CAN 控制器的 Pic18F458 芯片，但其 CAN 信号是 TTL 信号，不符合 DeviceNet 协议传输总线差分信号要求，故用 82C250 芯片处理 CAN 总线电平，Pic18F458 最多支持 8 路模拟 I/O，从站结构如图 3-4 所示，通过模拟 I/O 采集模拟信号，再进行 AD 转换成数字信号并编码；通过数字 I/O 采集数字信号与开关量；电源采用总线供电方式，总线电压为直流 24V，而除烟雾传感器是 15V 供电外，其他传感器和 Pic18F458 芯片都用 5V 供电，故用单独的电源处理模块来进行电压转换[5][24][25]。

从站DeviceNet应用层协议用MPLAB IDE开发环境，采用C语言实现DeviceNet应用软件[26]。

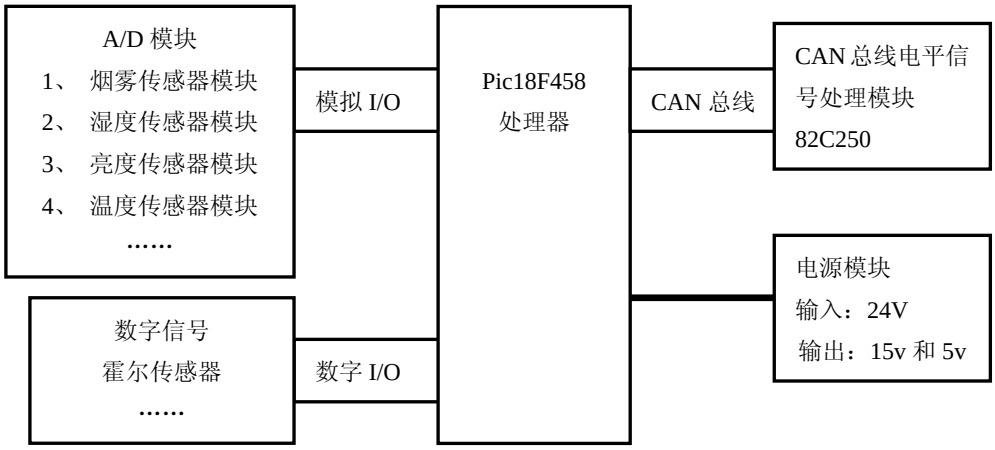

图3-4 从站结构

3.4 主站硬件平台设计与实现

系统硬件平台支持视频采集、CAN总线接口、LCD液晶屏显示及扬声器，存储器用Nor flash存放引导程序、Nand flash做数据和程序存储器、SDRAM做内存，扩展SD卡接口为外部扩展存储器；根据上述需求，设计系统硬件整体结构如图3-5所示[11][27]。

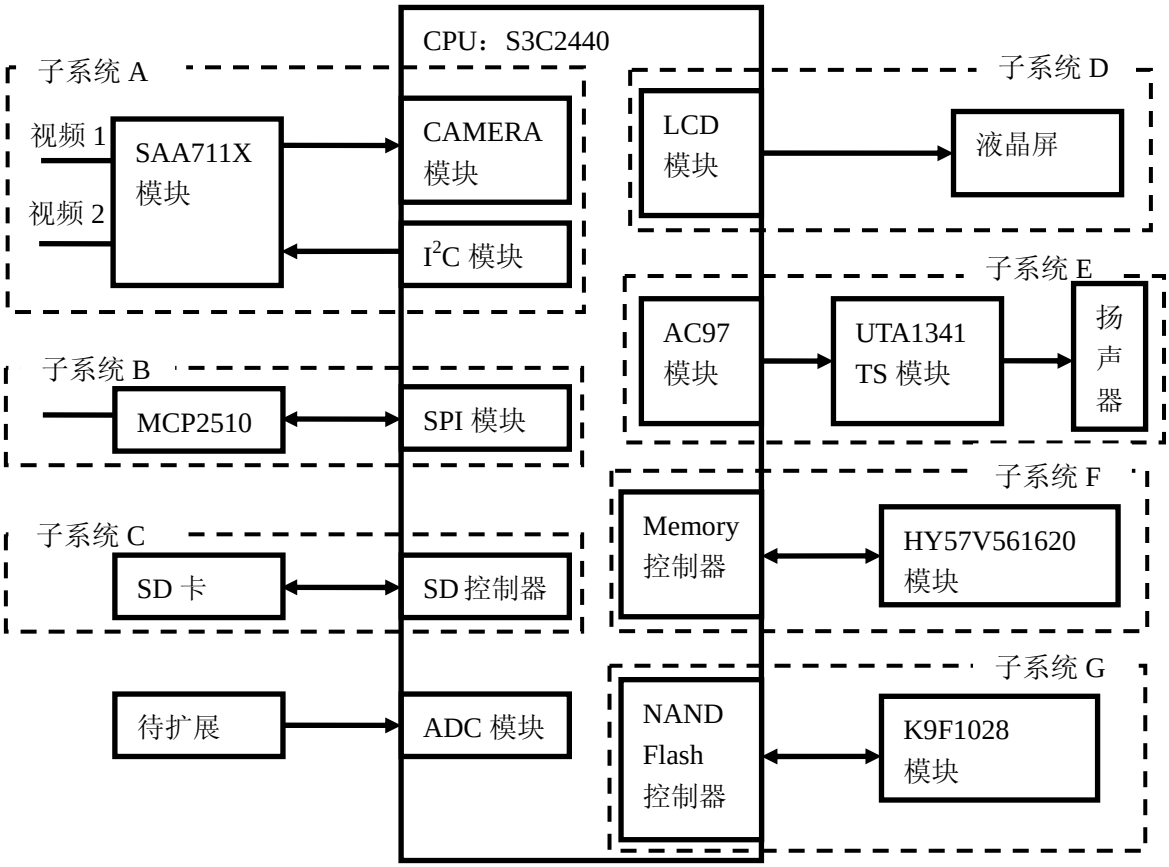

图3-5 系统总体硬件平台

子系统A是视频采集模块：通过SAA711X实现视频捕获，采集两路模拟视频信号，转换为数字的RGB信号，保存到内存中。

子系统B是CAN总线模块：通过MCP2510实现主站与CAN总线的通讯。

子系统C SD卡存储器模块：SD存储器最大可接2Gb容量的SD卡，保存系统信息。

子系统D LCD显示模块：读书内存中的视频信号，显示到TFT LCD上。

子系统E AC97音频模块：根据系统具体要求播放相应音频信息。

子系统F SDRAM模块：两块SDRAM芯片HY57V561620，做处理器内存使用。

子系统G NAND flash模块：一块NAND FLASH芯片K9F1208，存储系统文件。

3.5 CAN 总线模块的结构与功能

由于 S3C2440 芯片没有集成 CAN 控制器，故本文采用 MCP2510 作为外部 CAN 控制器，82C25X 为收发器。S3C2440 通过 SPI 接口连接 MCP2510 控制器，SPIMISO0 、SPIMISI0、SPISCK0 分别连接 MCP2510 控制器的输出、输入和时钟输入；MCP2510 的中断信号连接 S3C2440 处理器的 I/O 口，作为 S3C2440 的

外部中断信号源，通过该中断通知处理器已接收到 CAN 帧；82C25X 连接 MCP2510 的发送接收接口，CAN 控制器通过 82C25X 实现与 CAN 总线的连接，通过该收发器实现与 CAN 总线的交互。接收 CAN 总线模块的结构如图 3-6 所示[28]。

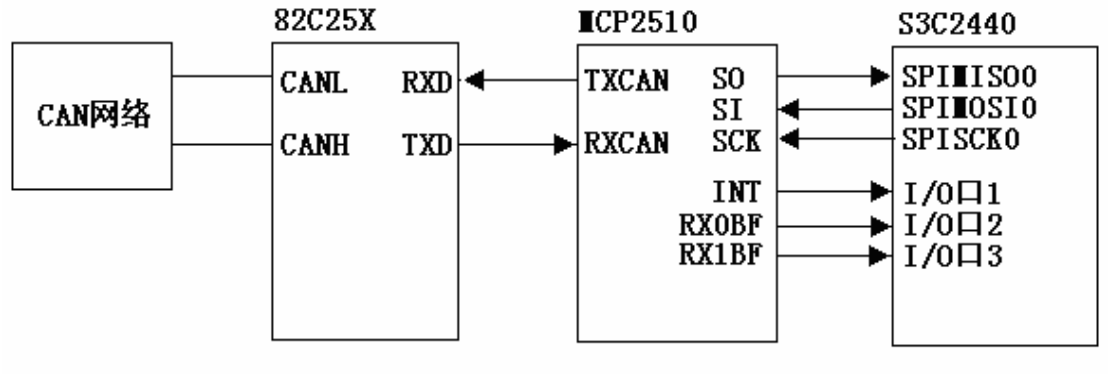

图 3-6 CAN 总线模块的结构

3.6 视频模块的结构与功能

S3C2440 芯片集成了数字视频控制器，支持 ITU601 和 ITU656 数字视频输入，支持 Preview path 和 codec path 两路视频传输通道。本文采用 ITU656 数字视频接口，减少接口电路信号线，便于接口电路的设计；由于 Preview path 是以 RGB 视频格式存放在内存中的，而本系统是以 RGB 视频信息格式才能正确显示在 LCD 屏上，故采用 Preview path 视频传输通道。视频信号处理由 CAMIF 模块实现，完成视频图像的缩放和裁剪功能。视频图像采用 DMA 直接存储方式存储到内存，独立于 CPU 完成数据块快速传输功能，实现将大的图像数据块传输到虚拟内存中。

视频模块结构如图3-7所示，采用SAA7113视频采集解码，SAA7113支持多路视频采集，本文采用2路视频采集，S3C2440通过I^2C接口控制SAA7113芯片，视频切换也可通过该接口实现。SAA7113采集两路模拟视频信号，转换为ITU-R BT.601/656数字视频信号输出给S3C2440，S3C2440数字视频控制器将接收到的数字视频信号保存至内存，应用程序可以将该内存视频信号输出到LCD显示视频图像。

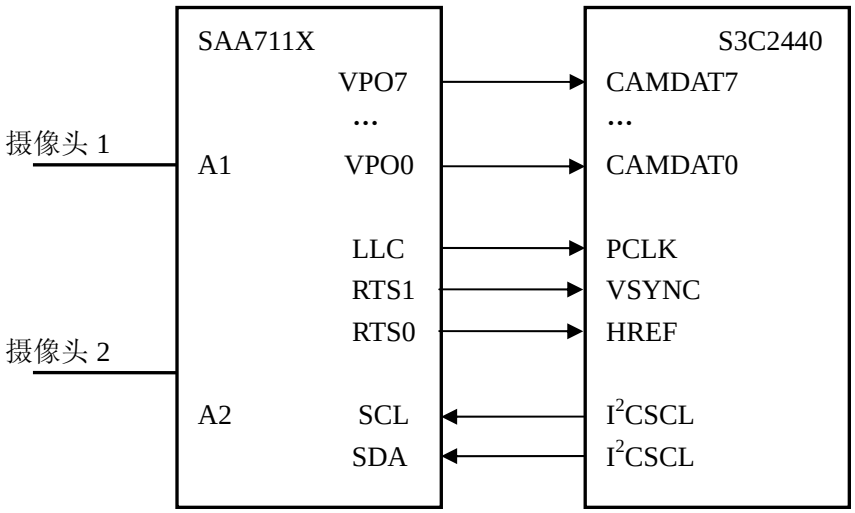

图3-7 视频采集硬件模块

第 4 章 DeviceNet 主站软件设计与实现

4.1 网络管理与数据采集平台需求分析

本文要实现的系统需求分功能性需求和非功能性需求两方面论述。系统功能性需求由网络管理、数据采集和退出系统三部分组成。

网络管理：网络管理包括主站管理和从站管理。

主站管理：主站管理包括打开主站、属性编辑和关闭主站。

打开主站：打开主站支持直接打开主站和导入配置文件两种方式，先加载打开 CAN 驱动程序，再按照 DeviceNet 协议进行重复 MAC ID 检查，检查未通过则上线失败，检查通过后则初始化从站管理列表。

属性编辑：采用直接打开主站方式时，可以在主站属性配置对话框中编辑主站属性；在采用导入配置文件方式打开主站时，可以预先在配置文件中编辑主站属性。主站属性有 MAC ID、BuadRate 和 BOI，MAC ID 取 0~63 的整数，BaudRate 支持 125kbps、250kbps 和 500kbps 三种，BOI 取 0 或 1，0 为 FALSE，1 为 TRUE。

关闭主站：关闭主站使系统恢复到系统刚启动后的状态。

从站管理：从站管理包括添加从站、删除从站、属性编辑、数据绑定和列表管理。

添加从站：添加从站主要有以下三种方式。手动添加，通过系统界面功能按钮手动添加从站至主站注册列表中；配置文件添加，在主站通过导入配置文件方式打开主站时，可以在主站配置文件中添加该从站及从站的配置文件名，导入主站配置文件后，从站会自动加入主站注册列表中；从站上线，在主站已经上线状态下，有从站上线，并且通过了重复 MAC ID 检查，那么系统自动会将该从站加入主站未注册列表内。

删除从站：将从站从主站列表中删除，不管是在注册列表还是在未注册列表内。

属性编辑：从站属性编辑主持两种方式，当通过导入配置文件方式打开主站时，用户可以在从站配置文件内设置从站属性；在网络管理界面从站属性编辑对话框内编辑选中的从站属性。

数据绑定：从站最多可以支持 8 字节每帧的 I/O 数据，用户可以将 I/O 数据的若干字节或某几位与从站下所挂器件绑定。

列表管理：实现从站在主站的注册列表和未注册列表之间移动。

数据采集：数据采集包括开始扫描和停止扫描。

开始扫描：开始扫描包括建立连接、请求 I/O 数据和处理 I/O 数据。

建立连接：与要扫描的从站建立预定义主从连接，连接方式支持通过显式连接和仅限组 2 非连接显式请求分配预定义主从连接组。

请求 I/O 数据：向已经建立连接的从站发送 I/O 报文，请求获取 I/O 数据。

处理 I/O 数据：处理从从站接收到的 I/O 数据，将数据保存到从站对象内，保存到对象内的数据具体如何处理，根据实际应用需要来扩展。

停止扫描：系统停止对注册列表内的从站进行扫描，即停止建立连接和请求 I/O 数据。

退出系统：退出系统释放系统占用的资源，关闭应用程序。

系统非功能性需求主要有以下几点：

- 健壮性：从站最新若干组 I/O 数据保存至内存和 SD 存储器。
- 可靠性：报文发送一次没有响应时，过 1 秒再发送一次来确定从站掉线。
- 易用性：支持 EDS 配置文件设置 DeviceNet 网络，避免重复配置网络。
- 清晰性：在 I/O 数据处理中，支持数据图形化显示。
- 安全性：从站重复上下线对主站运行无影响。
- 可扩展性：对于不同应用环境，易于将数据采集平台用于特定需求；系统主要参数可通过配置文件设置。
- 兼容性: 从站设备只要符合 DeviceNet 协议标准即可应用。
- 可移植性：可移植至其他支持 Windons CE.Net4.2 操作系统的微处理器上。

根据系统功能需求，采用 UML(Unified Modeling Language)进行系统建模，设计系统用例图等模型。

系统用例图如图 4-1 是系统主用例图，有 DeviceNet 网络管理，网络扫描和退出系统三部分。系统通过 ParseMessage 功能解析接收到的 CAN 总线报文，ReadBuffer 功能需要通过 SetReceivedFlag 获取 CAN 驱动数据已接收标志位，若数据接收标志位已置位，则读取 CAN 驱动内的帧，并通知 ParseMessage 解析该帧。

ManageDeviceNetNet 用例图如图 4-2 所示，网络管理功能分主站管理和从站管理两部分。

- ManageMaster：详细见图 4-4；
- ManageSlave：详细见图 4-5；

ScanNet 用例图如图 4-3 所示，扫描模块功能主要与扫描列表中的从站建立连接，获取从站 I/O 数据，支持异常数据的报警，数据范围可以在配置文件内设置。

- StartScan：开始扫描；
- StopScan：停止扫描；
- AlarmEnable：报警使能，报警使能后，数据超出其合理范围时发出蜂鸣声；
- AlarmDiasble：取消报警使能；
- StartScan：取消报警使能一段时间；时间可通过配置文件设置；
- Connect2Slvae：与从站建立连接；
- GetIOData：获取从站 I/O 数据。

ManageMaster 用例图如图 4-4 所示，主站管理主要有主站打开、关闭及主站属性编辑，属性有 MAC ID、BOI 和 BautRate。

- EditMacID：设置主站 MAC ID；
- EditBaudRate：设置主站波特率；
- EditBOI：设置 BOI(Bus-Off Interrupt)，语义请查看 DeviceNet Specification 第 1 卷 5-5.3.3；
- OpenMaster：打开主站，主站上线;
- CloseMaster：关闭主站，主站下线。

ManageSlaver 用例图如图 4-5 所示，从站管理主要有从站添加、从站删除、从站属性编辑和从站在注册列表与未注册列表之间的移动，从站属性有 I/O 连接方式、MAC ID、BOI 和 BaudRate。

- AddSlave：添加从站至主站未注册列表；
- DeleteSlave：删除列表内选中的从站；
- Move2Registed：将从站从未注册列表移至注册列表；

- Move2UnRegisted：将从站从注册列表移至未注册列表；
- EditSlaveMacID：编辑从站 MAC ID；
- EditSlaveBaudRate：编辑从站波特率；
- EditSlaveBOI：编辑从站 BOI 属性；
- EditConnectionMethod：编辑从站连接方式，连接方式有：Poll、StateofChange、Bit_Strobe、Cyclic、MulticastPoll；
- EditAcknowledged：设置是否有应答。

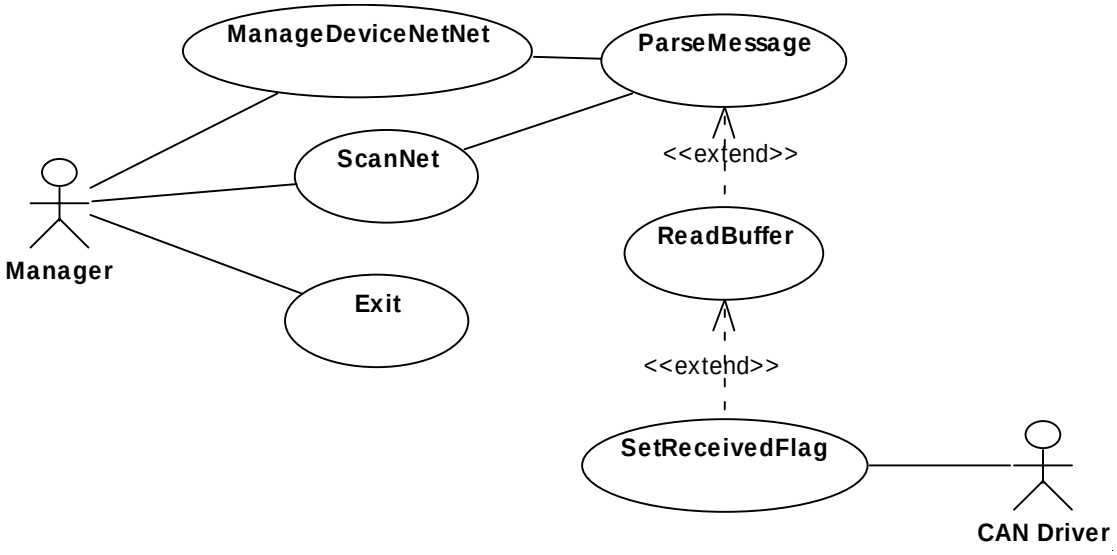

图 4-1 系统用例图

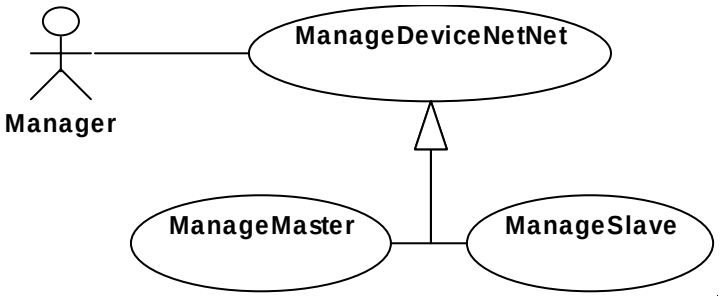

图 4-2 ManageDeviceNetNet 用例图

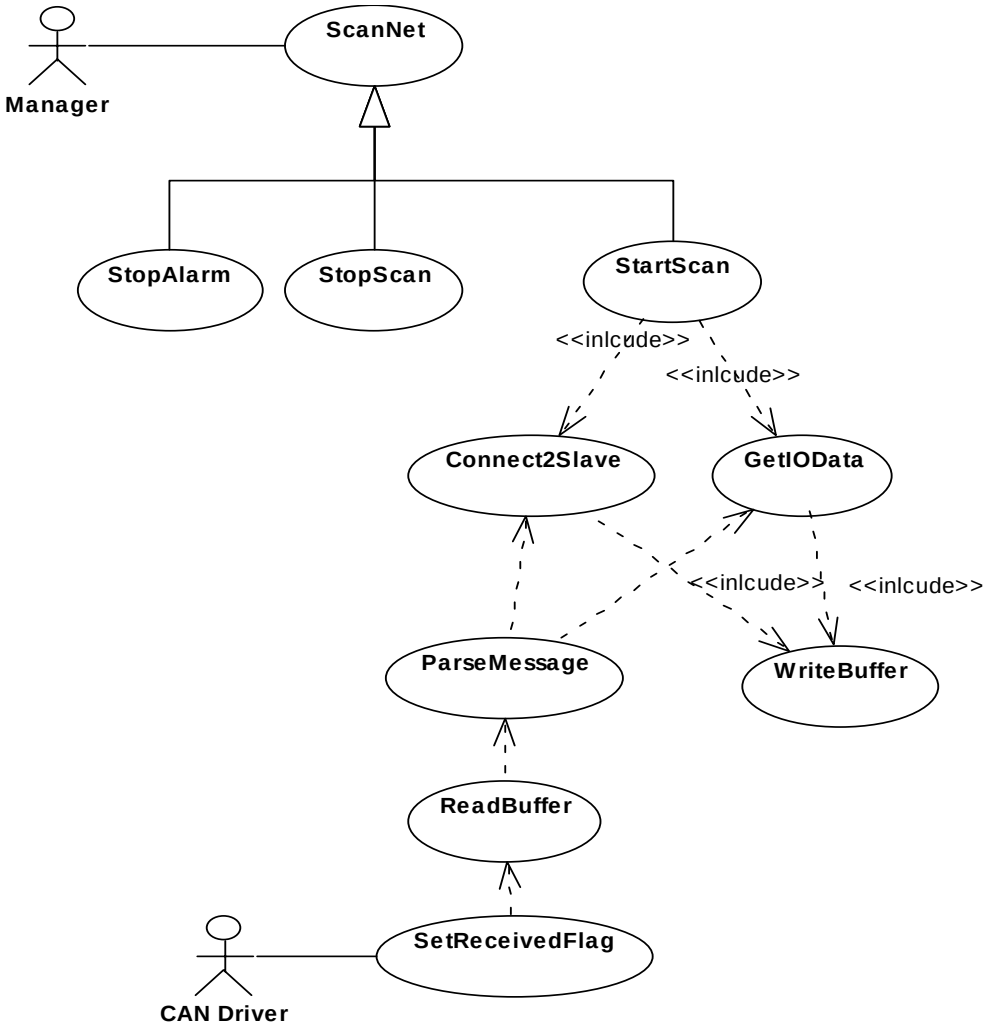

图 4-3 ScanNet 用例图

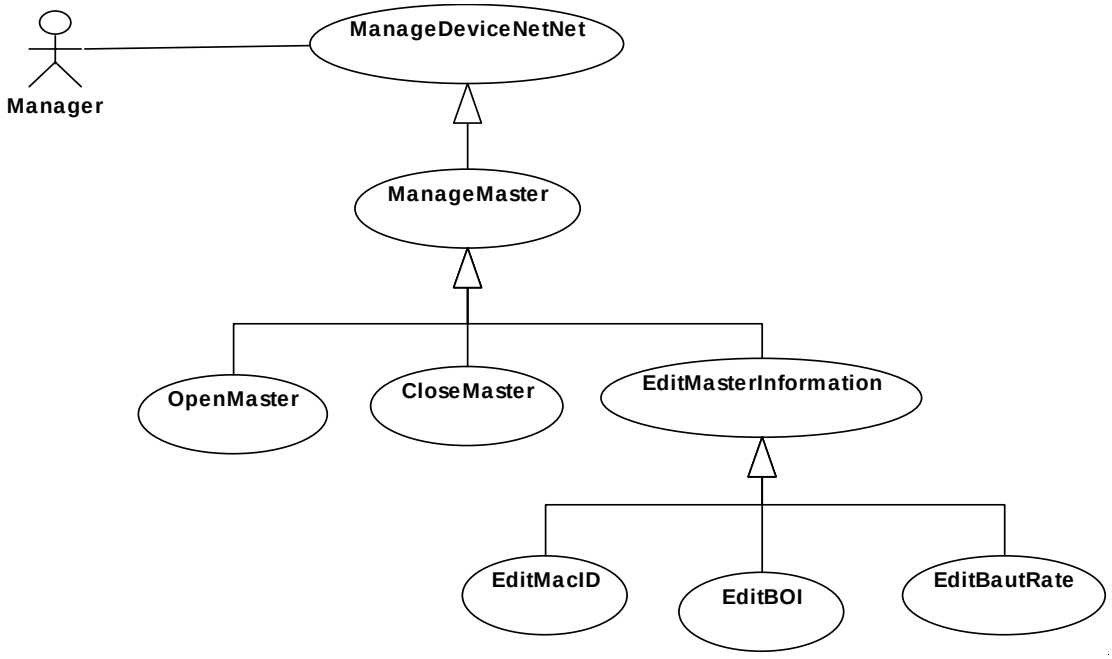

图 4-4 ManageMaster 用例图

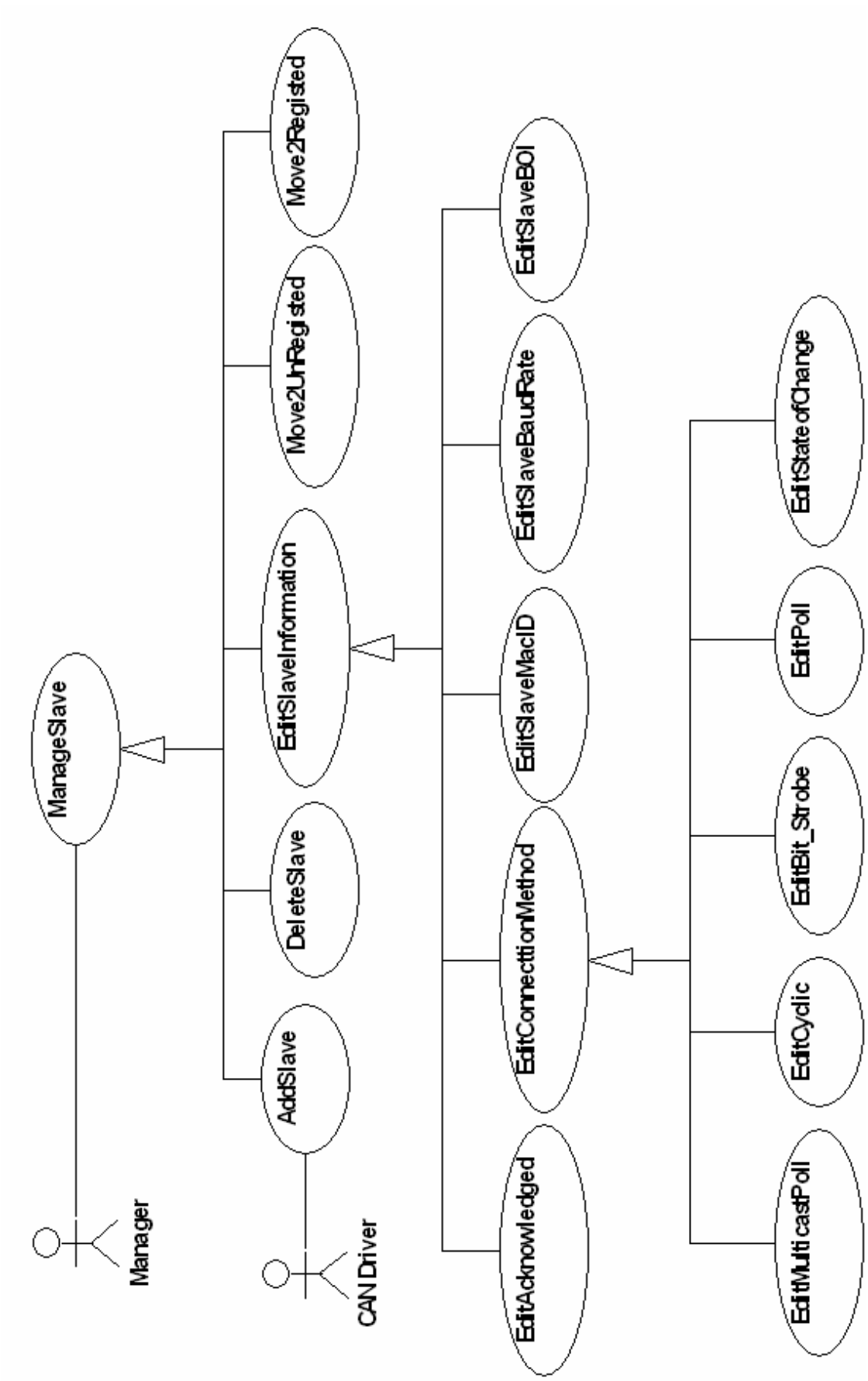

图 4-5 ManageSlaver 用例图

4.2 驱动程序设计实现

在 Windows CE.Net4.2 中，最简单的一个驱动程序莫过于一个内置(Built-in)设备的流接口驱动。对于一个不支持热拔插的设备，最快捷的方法就是为其实现一个内置的流接口的驱动。

对于这样一类驱动程序，我们只需要按一种特定的规则实现一个动态库，其中实现对所有的硬件功能的调用，再将这个动态库加入系统中，然后设置相关的注册表项，使得在系统启动时设备管理器能识别并且加载这个设备即可。本文要实现的 CAN 驱动程序是视频驱动程序都使用这种方式开发。

此动态链接库与应用程序层所用的库并没有很大差别，源文件可以是 C、C++、甚至汇编，只要实现 Windows CE.Net4.2 标准 I/O 系统函数。

本文将实现 CAN 驱动程序和视频驱动程序。

4.2.1 驱动程序入口点

流接口驱动程序入口点主要有几个函数。

DllEntry(HINSTANCE DllInstance, INT Reason, LPVOID Reserved)。

参数：DllInstance 是 DLL(Dynamic link library)的句柄，与一个 EXE(Executable extension)文件的句柄功能类似，一般可以通过它得到 DLL 中的一些资源，例如对话框；Reason 是 DLL_PROCESS_ATTACH 或 DLL_PROCESS_ DETACH, 等于前者是动态库被加载，等于后者是动态库被释放。

DWORD XXX_Init(LPCTSTR pContext,LPCVOID lpvBusContext)。

参数：pContext 是系统传入的注册表键，通过它可以得到我们在注册表中设置的配置信息；lpvBusContext 一般不用。实际上，很多程序中将这个函数写成了 DWORD XXX_Init(DWORD pContext)，只需要将 pContext 转化成 LPCTSTR 即可。

DWORD XXX_Open(DWORD hDeviceContext,DWORD dwAccess, DWORD dwShareMode)。

参数：hDeviceContext XXX_Init()函数返回给上层的值，也就是我们在 XXX_Init()函数中分配的用来记录驱动程序信息的那个结构体的指针，可以在这个函数中直接转化成所定义的结构，从而获取驱动程序的信息；dwAccess 是上层所要求的访问方式，可以是读或者写，或者是 0，即不读也不写；dwShareMode

上层程序所请求的共享模式，可以是共享读、共享写这两个值的逻辑或，或者是0，即独占式访问。系统层对设备文件的存取权限及共享方法已经做了处理，所以在驱动程序中这两个参数不用理会。这个函数一般不用做太多处理，可以直接返回 hDeviceContext 表示成功，对于一个不支持多个用户的硬件，在设备已经打开后，应该总是返回 0 以告失败，则 CreateFile()函数调用不成功。

DWORD XXX_Close(DWORD hDeviceContext)。

参数：hDeviceContext 为 XXX_Open()函数返回给上层的那个值。这个函数应该做与 XXX_Open()函数相反的事情。

DWORD XXX_Deinit(DWORD hDeviceContext)。

参数： hDeviceContext XXX_Init()函数返回给上层的那个句柄。

BOOL XXX_IOControl(DWORD hDeviceContext,DWORD dwCode, PBYTE pBufIn, DWORD dwLenIn, PBYTE pBufOut, DWORD dwLenOut, PDWORD pdwActualOut)。

参数：hDeviceContext XXX_Open()函数返回给上层的那个句柄，即自已定义的,用来存放程序所有信息的一个结构。dwCode 是 IO 操作码,如果是 Windows CE.Net4.2 已经支持的设备类，就用它已经定义好的码值，否则就可以自已定义。pBufIn 是传入的 Buffer，每个 IO 操作码都会定义自已的 Buffer 结构；dwLenIn pBufIn 以字节记的大小；pBufOut、dwLenOut 分别为传出的 Buffer 及其字节的数，pdwActualOut 驱动程序实际在 pBufOut 中填入的数据字节数。其中，前两个参数是必须的，其它的任何一个都有可能是 NULL 或 0。所以，当给 pdwActualOut 赋值时应该先判断它是否为一个有效的地址[29]。

4.2.2CAN 驱动程序实现

按照操作系统的标准入口函数编写 CAN 驱动程序，其中 XXX 在 CAN 驱动程序中用 BUS 字符串。

DllEntry()函数根据 dwReason 参数判断是加载还是卸载，加载就调用 DisableThreadLibraryCalls 函数用来屏蔽 DLL_THREAD_ATTACH 和 DLL_THREAD_DETACH 消息,，返回 1，否则返回 0。

BUS_Init()函数中分配内存并进行相关寄存器内存映射、初始化 IO 端口、初始化 SPI 控制器。

BUS_Deinit()函数中释放 BUS_Init()函数中分配的内存。

BUS_Open()函数清 CAN 控制器中断标志位、设置发送引脚、设置滤波和屏蔽为接收所有报文、设置接收寄存器、创建中断事件、向 OS 注册该中断事件、打开中断处理线程、设置中断处理线程优先级、返回 1。

BUS_Close()函数关闭中断事件、中断处理现场、返回 1。

BUS_Read()函数将接收到的 CAN 报文放入 pBuffer 中，清中断标志位。

BUS_Write()函数将 pSourceBytes 内数据发送到 CAN 控制器发送寄存器中，并返回 1。

BUS_IOControl()函数中实现 GET_SHARE_ADD、SET_CAN_BAUD- RATE、SET_IDENTIFER 三个 IO 控制码，分别实现应用程序读取 CAN 接收数据标志位、应用程序设置 CAN 控制器的波特率、应用程序设置 CAN 标识区，应用程序可以通过这些 IO 控制码设置 CAN 控制器。

IST_CANRxThreadFunc()函数是 CAN 驱动程序的中断处理函数，当 CAN 控制器接收到 CAN 报文时，产生外部中断通知 CPU 处理数据，此时 OS 调用该中断处理函数，在中断处理函数中设置数据接收标志位。

CAN 控制器的掩码和滤波器都没有提供设置接口，因为本文所要用到的 CAN 标识会有很多种，如果通过设置硬件滤波效率会比较低，故采用软件程序实现滤波，详细见 4.4.2.3CAN 报文处理设计与实现。

4.2.3 视频驱动程序实现

视频驱动程序的编写类似 CAN 驱动程序，XXX 用 CIS 字符串代替。

DllEntry()函数与 CAN 驱动程序一样。

CIS_Init()函数中分配内存并进行相关寄存器内存映射、初始化视频控制器、返回 1。

CIS_Deinit()函数中关闭视频、释放内存、返回 1。

CIS_Open()函数与 CAN 驱动程序 BUS_Open()函数中类似。

CIS_Close()函数与 CAN 驱动程序 BUS_Close()函数中类似。

CIS_Read()函数和 CIS_Write()函数是空函数，返回 1。

CIS_IOControl()函数中实现以下 IO 控制码，应用程序可以通过这些 IO 控制码实现视频图像的处理。

- IOCTL_CAM_CONT控制码用于继续采集视频图像；
- IOCTL_DISPLAY_ONE用于显示视频图像1;
- IOCTL_DISPLAY_TWO用于显示视频图像2；
- IOCTL_CAM_STOP用于停止视频图像采集[30][31]。

CameraCaptureThread()函数是视频驱动程序的中断处理函数，根据选在的视频通道，读取视频的 RGB 数据，将该数据拷贝至 LCD 显示图像对应内存 FRAMEBUF_BASE 中即可显示采集到的视频图像。

4.2.4 内核定制

前面 3 节仅仅实现了 CAN 和视频的驱动程序，本节将把这两个驱动程序加入到内核。本文采用修改注册表将驱动程序库加入 Windows CE.Net4.2 内核中，注册表修改方法如下。

- 在 platform/BSP/ FILES/Platform.reg 注册表中添加如下项目。

  [HKEY_LOCAL_MACHINE\Drivers\BuiltIn\ SampleDev]

  "Prefix"="XXX"

  "Dll"="MyDev.Dll"

  "Order"=dword:1

注：SampleDev 为任意与其它项目不重名的字符串，在 CAN 驱动程序和视频驱动程序中分别为 Camera 和 CANSPI；XXX 在 CAN 驱动程序和视频驱动程序中分别为 BUS 和 CIS；MyDev 为要生成的动态库的库名，在 CAN 驱动程序和视频驱动程序中分别为 CAN 和 CAMERA。

- 在 platform/BSP/ FILES/Platform.bib 文件中添加如下项目。

  MyDev.dll　　$(_FLATRELEASEDIR)\MyDev.dll　　NK　SH

  MyDev 分别为上文说讲的动态库的库名。

完成上述步骤以后，在 platform/BSP/ drivers 目录中新建 CAN 和 CAMERA 两个目录，然后将刚才编写的 CAN 和视频的驱动程序放入对应目录中，然后在 platform/BSP/ drivers/dirs 文件中加入 CAN 和 CAMERA 目录名。再在 CAN 很 CAMERA 目录中新建名称分别为 sources, makefile, mydev.def 的文件，其内容如下：

- makefile: 只需要这样一行：

!INCLUDE $(_MAKEENVROOT)\makefile.def

- mydriver.def 文件定义一些驱动程序中指定的入口函数，应用程序通过操作系统的 IO 文件系统间接这些函数。格式为：

LIBRARY MyDev

EXPORTS

BUS_ Init /CIS_ Init

BUS_Deinit/CIS _Deinit

BUS_Open CIS _Open

BUS_Close/CIS _Close

BUS_IOControl/CIS _IOControl

注：这个字符串分别为 CAN 和 CAMERA。“/”前面是用于 CAN 驱动程序，“/”用于视频驱动程序。

- Sources：这个文件很重要，最基本的该有如下内容：

TARGETNAME= MyDev（驱动程序的库名，本文分别为 CAN 和 CAMERA）

TARGETTYPE=DYNLINK（指定要生成的是一个动态库）

（下面两项指定需要与哪些动态库链接，一般要第一项就足够了）

TARGETLIBS=$(_COMMONSDKROOT)\lib\$(_CPUINDPATH)\coredll.lib

SOURCELIBS= $(_COMMONOAKROOT)\lib\$(_CPUINDPATH)\ceddk.lib \

DEFFILE=MyDev.def (mydriver.def 文件名)

DLLENTRY=DllEntry（指定动态库的入口函数）

SOURCES=（写上你所有源文件的名字，它们将会被编译）

按照上述步骤完成 CAN 驱动程序，应用程序中就可以通过 Windows CE.Net 4.2 提供的 IO 文件系统与驱动程序进行交互[23][27][32][33][34]。

4.3 网络管理与数据采集平台设计与实现

根据系统功能需求分析，设计 DeviceNet 主站网络管理与数据采集平台结构图如图 4-6 所示，应用程序由网络管理、数据采集、数据显示、实时保存和 DeviceNet 协议处理五个模块组成。

网络管理和数据采集与 DeviceNet 协议处理模块交互，DeviceNet 协议处理模块交互同 CAN 驱动交互，实现硬件抽象；配置文件用于配置 DeviceNet 总线网络节点，包括配置主站和从站；数据显示模块包括从站 I/O 数据显示和视频图像

显示，用户可以通过外部按钮切换显示画面，从站 I/O 数据处理包括在 LCD 屏上图像化显示数据，并根据配置文件判断数据是否异常及报警；实时保存主要分两部分，一实时保存于系统内存中，采用循环队列保存最新若干组数据，一实时保存于存储器中，采用操作系统的文件系统及文件 IO 系统，进行写文件。

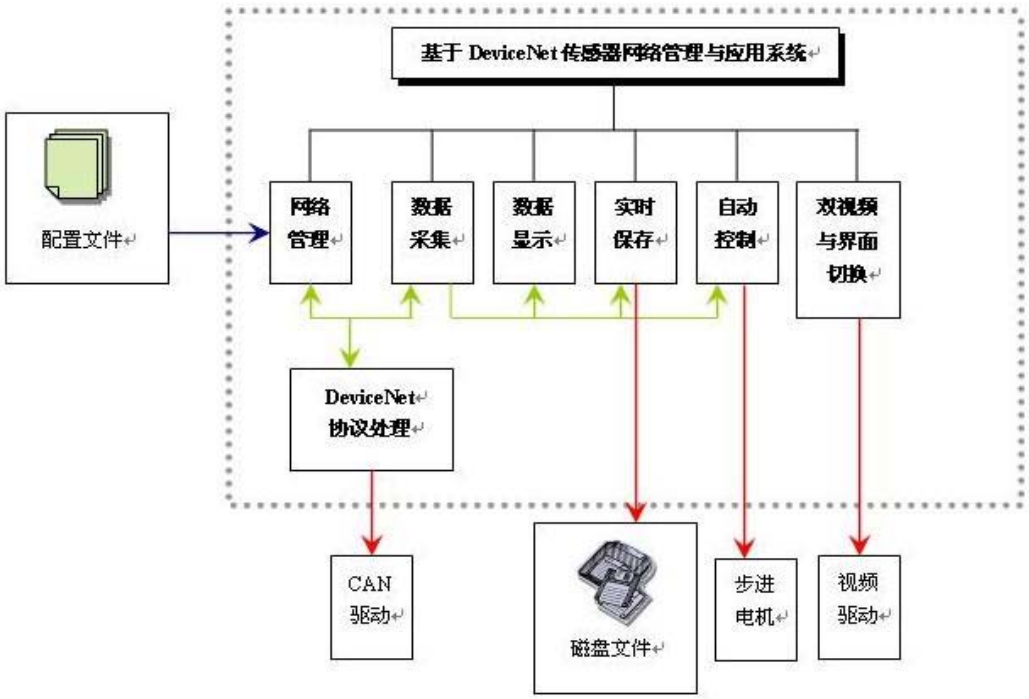

图 4-6 DeviceNet 主站网络管理与数据采集平台结构图

DevieNet 协议处理模块如图 4-7 所示，将上层发送的帧通过 IO 文件系统写入 CAN 驱动程序；同时应用程序用线程扫描 CAN 驱动数据接收标志位，接收到报文后，DevieNet 协议处理模块读取 CAN 驱动程序内的报文，根据 DeviceNet 协议解析报文，并通知相应对象处理该报文。

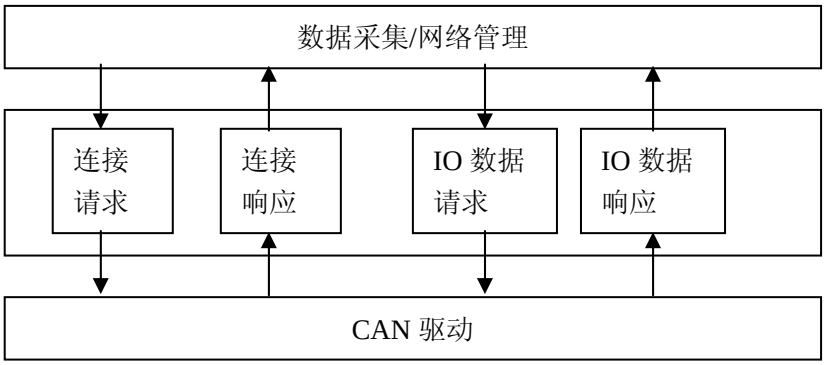

图 4-7 DevieNet 协议处理模块

设计系统类图如图 4-8 所示，类图说明如下。

- System：系统架构类，下挂有 NetManage 类和 Scan 类，分别用于网络管理模块和数据采集模块；
- Scan：负责从站扫描列表扫描，通过分配预定义主/从连接组，请求 I/O 数据；可扩展 I/O 数据显示、报警使能、实时保存 I/O 数据、异常报警和自动控制等功能；
- NetManage：用于网络管理，提供主站、从站属性设置接口，从站注册列表和未注册列表管理；
- CArray：固定长度用于存放从站的数组，系统有两个数组，一个是注册列表扫描时作为扫描列表，一个是未注册列表；
- Slave：管理从站的属性、I/O 连接方式和 I/O 数据及绑定的传感器集合等；
- Sensor：传感器类，管理传感器属性和对应 I/O 数据等；
- Myqueue：保存 Sensor 数据，同时每次入队时将数据写入对应文件，这样每次接收到数据时不会把前面的数据覆盖掉，在系统掉电时最新若干组数据也不会丢失，增加数据的可回溯性；
- Master：实现 DeviceNet 协议规定的主站功能，管理其下的 Timer、Identity、Connection、UCMM、MessageRoute 和 DeviceNet 等对象；
- Timer：系统定时器；
- DeviceNet：负责预定义主/从连接组，MAC ID 检查等；
- Identity：负责管理主站产品类型序列号、生产号、制造商号等信息；
- UCMM：负责显式消息连接功能；
- MessageRoute：负责消息路由，分析接收到的 CAN 报文，发送给对应对象；
- Connection：负责网络连接包括消息发送和消息接收；
- Consumer：负责 CAN 报文接收；
- Producer：负责 CAN 报文发送；
- DuplicateMacID：负责重复 MAC ID 检查，包括主站上线及从站上线时的重复 MAC ID 检查；

- CAN：提供底层 CAN 驱动接口，负责与 CAN 总线交互。

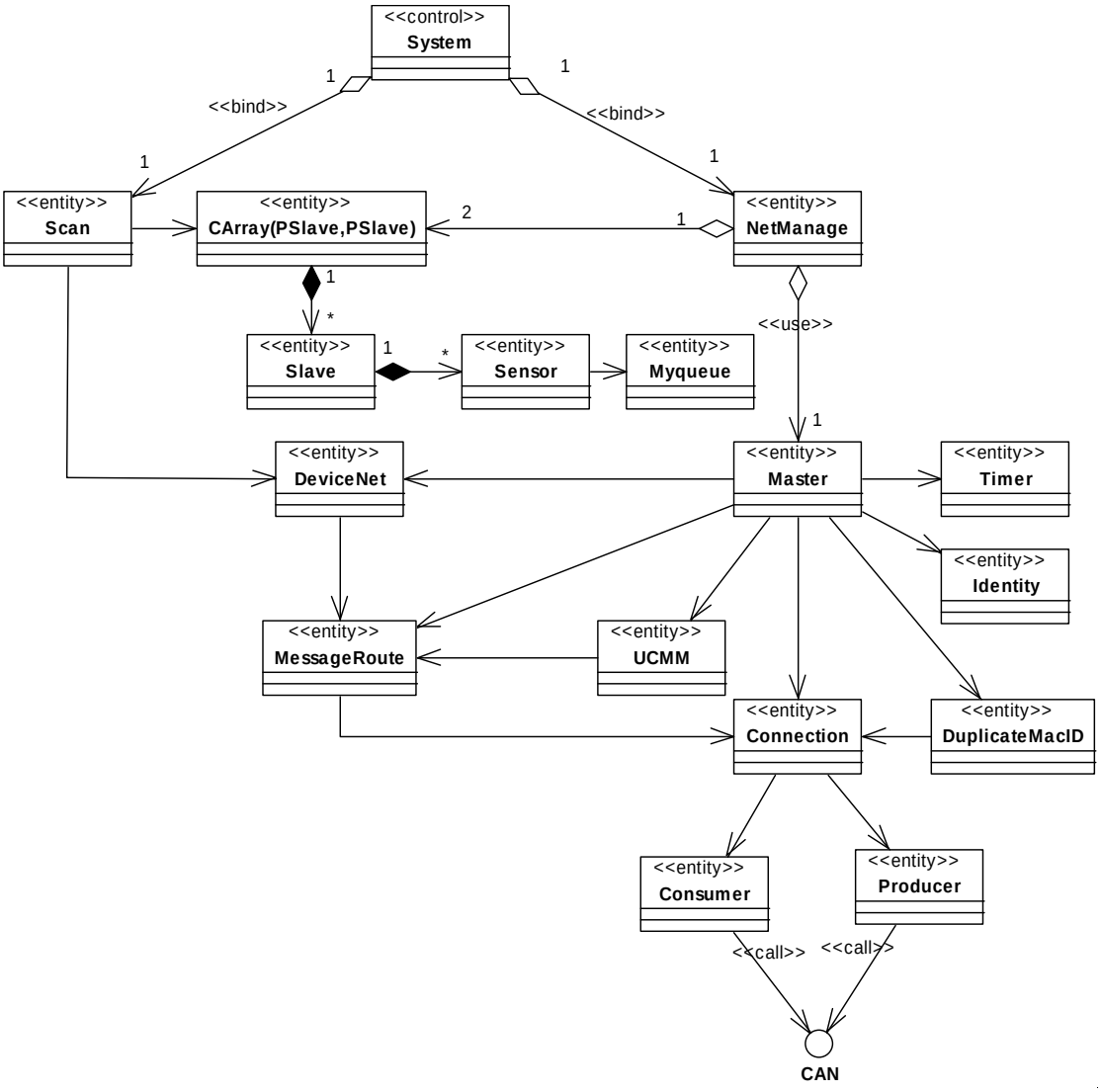

图 4-8 系统类图

4.3.1 网络管理模块设计与实现

网络管理主要是扫描列表管理、主站上线重复 MAC ID 检查和从站上线处理，主站上线支持直接打开主站和导入配置文件两种方式。

4.3.1.1配置文件格式与解析

配置文件分主站配置文件和从站配置文件，在配置文件中可以设置主站或从站的 MAC ID、BuadRate、BOI 属性，主站配置文件中还可以设置总线网络上个从站的 WatchDog 时间和主站下要挂载的从站及从站配置文件名；从站配置文件

中可以设置从站的类型、支持的连接方式、是否支持应答，还可以在从站下挂载器件，绑定到从站 I/O 数据对应字节或位。

主站配置文件格式如下所示，详细文件编写格式见附件二。

系统导入主站配置文件时，按以下方式解析配置文件。系统调用 NetManage 类的 parsefile 方法，动态产生 Master 对象，设置 Master 属性；读到主站配置文件中挂载从站时，自动读取对应从站配置文件，系统调用 Master 对象的 parsefile 方法解析从站配置文件，动态产生 Slave 对象，设置从站属性，把从站加入到 Master 注册列表中；读到从站配置文件中 I/O 数据绑定时，调用 Slave 对象的 parsefile 方法，动态产生 Sensor 对象，将 Sensor 对象数据与 Slave I/O 数据特定位或字节绑定，并初始化 Sensor 的最新若干组数据保存队列 Myqueue 对象。

4.3.1.2扫描列表设计与实现

网络管理主要是管理系统扫描列表，本文采用两个数组分别管理注册列表与未注册列表，两个列表最大长度为 64，列表元素是从站对象 Slave；采集从站数据时，注册列表成为扫描列表，系统循环从扫描列表中取从站对象，建立连接、获取从站 I/O 数据。

系统支持直接打开和导入配置文件两种方式打开主站，当主站通过导入配置文件方式打开时，会将主站配置文件中挂载的从站加入到注册列表，除此之外还支持以下两种从站添加方式，主站在上线状态时，如果有从站上线，并且该从站通过了重复 MAC ID 检查，那么系统自动将该从站加入未注册列表；通过 AddSlave 功能手动添加从站到未注册列表中；从站属性设置通过 ListView 实现，双击需要编辑的从站，ListView 中会显示该从站属性，用户可以双击 ListView 中属性来编辑该属性。

在界面提供按钮“=>”和“<=”实现从站在两个列表之间移动；用户也可以通过 DeleteSlave 按钮将从站从列表中删除。

4.3.1.3主站上线设计与实现

系统提供导入配置文件和直接打开两种方式打开主站，用户可以在主站打开方式选择对话框中选择打开方式，直接打开主站方式比较简单，初始化主站各对象、初始化注册列表和未注册列表及重复MAC ID检查（注册列表和未注册列表详见4.4.1.2节扫描列表设计与实现），通过重复MAC ID检查后主站上线；通过导

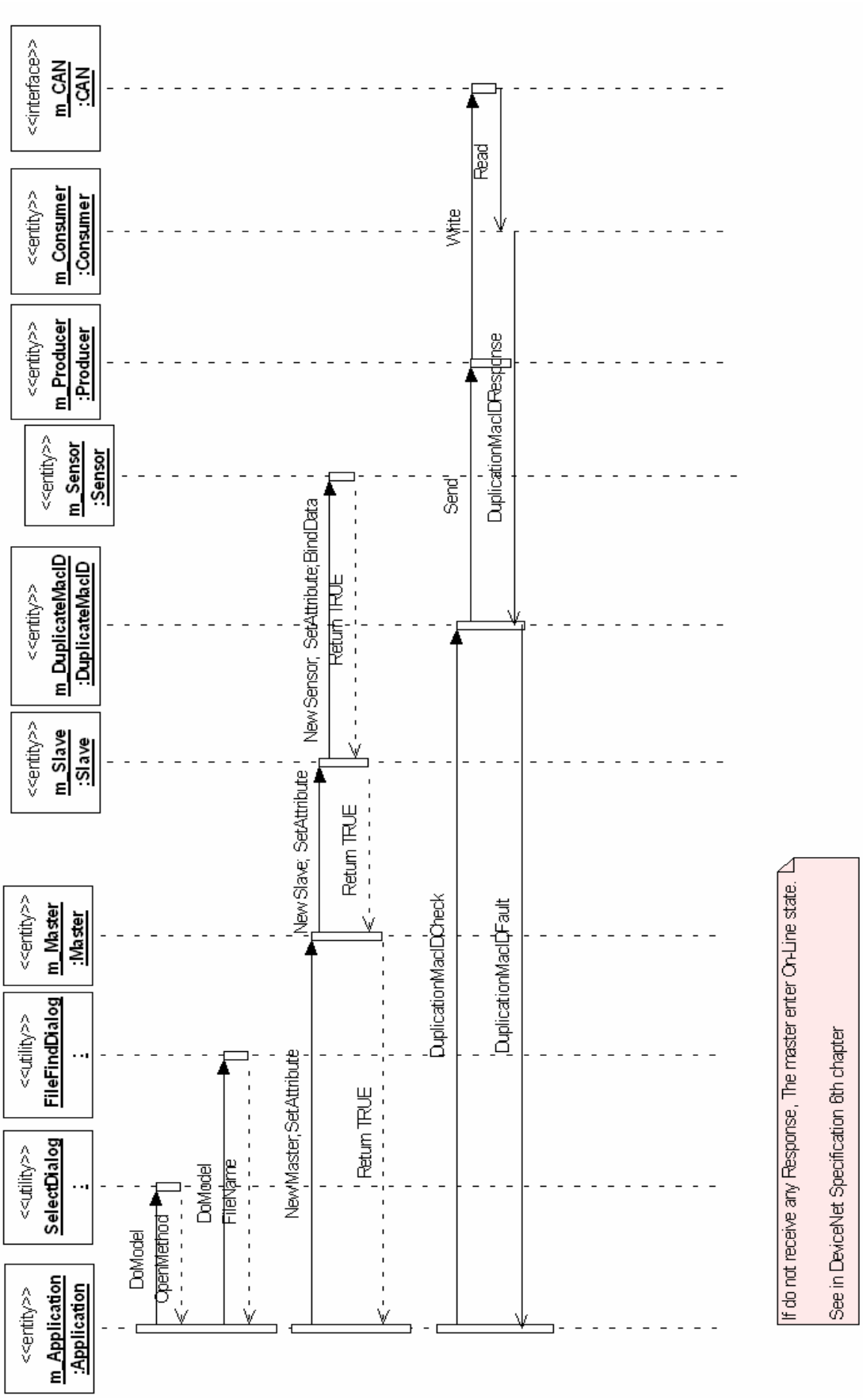

图4-9主站上线导入配置文件方式执行顺序图

入主站配置文件方式打开主站的执行顺序图如图4-9所示，用户通过文件选择对话框选择导入文件，系统根据配置文件动态生成主站和从站、设置主站各属性、配置注册列表、初始化未注册列表，然后进行重复MAC ID检查[35]。

不管采用哪种方式使DeviceNet主站上线，最后都会进行重复MAC ID检查，检查主站MAC ID是否跟总线上其他节点相同。

主站或从站上线都由类DuplicateMacID处理。主站上线调用DuplicateMAC-ID::SendDuplicateMacID方法，按照DeviceNet协议发送重复MAC ID检查请求报文，详细见2.3.5节的图2-10 DeviceNet节点状态图，而报文的发送与接收是不同线程处理的，本文利用操作系统提供的事件机制实现同步。

重复MAC ID检查CAN标识符是组2信息ID为7，目的MAC ID为本节点的MAC ID，CAN标识符格式为：

1	0	MAC ID	1	1	1

重复MAC ID检查CAN帧数据域格式如图4-10所示，第0字节第7位R/R是请求/响应标志位，0表示请求，1表示响应，第0~6位是节点物理端口；第1~2字节是16位制造商ID，每个设备生产商都有唯一的制造商ID，低字节放在第1字节，高字节放在第2字节；第3~6字节是存放32位产品序列号，每个制造商都会给设备一个唯一的序列号，分别由低位到高位存于第3~6字节，第7字节不用为0。

OffSet	7	6	5	4	3	2	1	0
0	R/R	物理端口						
1	制造商ID							
2	（1低字节，2高字节）							
3	产品序列好							
4	（3、4、5、6字节有底到高）							
5								
6								

图4-10重复MAC ID检查数据域格式

重复MAC ID检查发出请求报文后等待响应1秒时间，等待超时后再发送一次重复MAC ID检查请求报文，并等待响应1秒，如果再次超，说明总线网络中无重

复MAC ID，主站就上线，如果在等待过程中接收到重复MAC ID检查报文，主站就上线失败，进入通信故障状态，详细过程见2.3.5节图2-10 DeviceNet节点状态转换图，实现伪代码如下[13][36]。

```
……
设置主站状态为WaitingState;
malloc报文内存；
初次化重复MAC ID检查报文发送次数i=0；
While(i<=1) {
    设置重复MAC ID检查报文；
    发送报文;
    等待响应、超时事件；
    if  接收到响应{
        设置主站状态为通信故障；
        回收重复MAC ID检查报文内存；
        return FALSE；}
    else i++；
    fi } // end while
```

4.3.2 数据采集平台模块设计与实现

数据采集模块新建线程，按照网络管理模块配置好的注册列表，与列表中元素 Slave 对象建立连接，然后获取 I/O 数据，处理获取的 I/O 数据。

数据采集过程活动图如图 4-11 所示，实现过程见顺序图 4-12 所示，图 4-11 中 Connect2Slave 方法实现过程将在后文讲述。数据采集时先获取扫描列表中从站 Slvae 对象，判断该 Slvae 对象是否已经建立连接，如果已经建立连接则请求获取 IO 数据；如果没有建立连接则请求与该从站建立连接，连接过程详见 4.3.2.1 预定义主/从连接组设计与实现，连接成功后设置从站属性再请求获取 IO 数据；如果获取 IO 数据失败则设置该从站连接标志位为未连接，获取 IO 数据成功则设置该从站连接标志位为已连接，并通知数据处理模块处理 IO 数据，详细处理过程见 4.3.2.4 I/O 数据处理设计与实现[23][37]。

数据采集时从主站注册列表中获取的从站 Slave 对象与网络管理时主站注册

列表中的从站对象是同一组对象，因此需要设置从站对象访问权限，网络管理属性编辑和数据采集不能同时访问同一个从站 Slave 对象，本文在 Slave 对象中增加状态标志位，该标志位在网络管理编辑状态时，不能进行数据采集，同样在数据采集是不能在网络管理中编辑该从站 Slave 对象的属性。

扫描线程建立和线程函数用 CreateThread API 函数创建新线程，ScanThreadFunc 为线程函数，创建创建成功后，执行线程函数，其伪代码如下。

```
DWORD Scan::ScanThreadFunc(LPVOID lparam)          // 线程函数
{
    lparam 参数类型转换;
    for( MAC ID=0; MACID%64; MACID++){
       获取 Slave 对象指针;
       if 对象指针不为 NULL        //表示 MAC ID 从站在注册列表内;
            if Slave 对象不在属性编辑状态
                设置对象状态标志位为数据采集;    //此时不能编辑该 Slave。
                if Slave 对象已 I/O 建立连接;
                    请求获取 I/O 数据;
                    if 请求获取 I/O 数据失败;
                        获取 I/O 数据失败，设置 I/O 连接标志位为未连接;
                        发送消息; //通知数据处理线路即主线程,
                                  //获取 I/O 数据失败,
                                  //保存空数据，将空数据压入队列。
                    fi
                else Slave 对象未建立 I/O 连接
                    请求通过仅限组 2 非连接显式请求消息分配 Slave 的
                    预定义主/从连接组;
                    if  分配成功
                        请求获取 I/O 数据;
                        if  获取 I/O 数据成功
                            设置连接成功标志位;
```

```
                else  获取 I/O 数据失败
                    设置连接失败标志位;
                    发送获取 I/O 数据失败消息;
                    发送消息通知数据处理模块;
                fi
            else 分配失败
                设置连接失败标志位;
                发送消息通知数据处理线程;
            fi
        fi
        设置 Slave 扫描标志位；//可以在网络管理中编辑该从站属性
    fi
    等待线程结束事件信号 20ms，等待结束继续循环;
    fi
    if  有结束信号
        退出线程函数;
    fi
    }     //end for
}     //end ScanThreadFunc
```

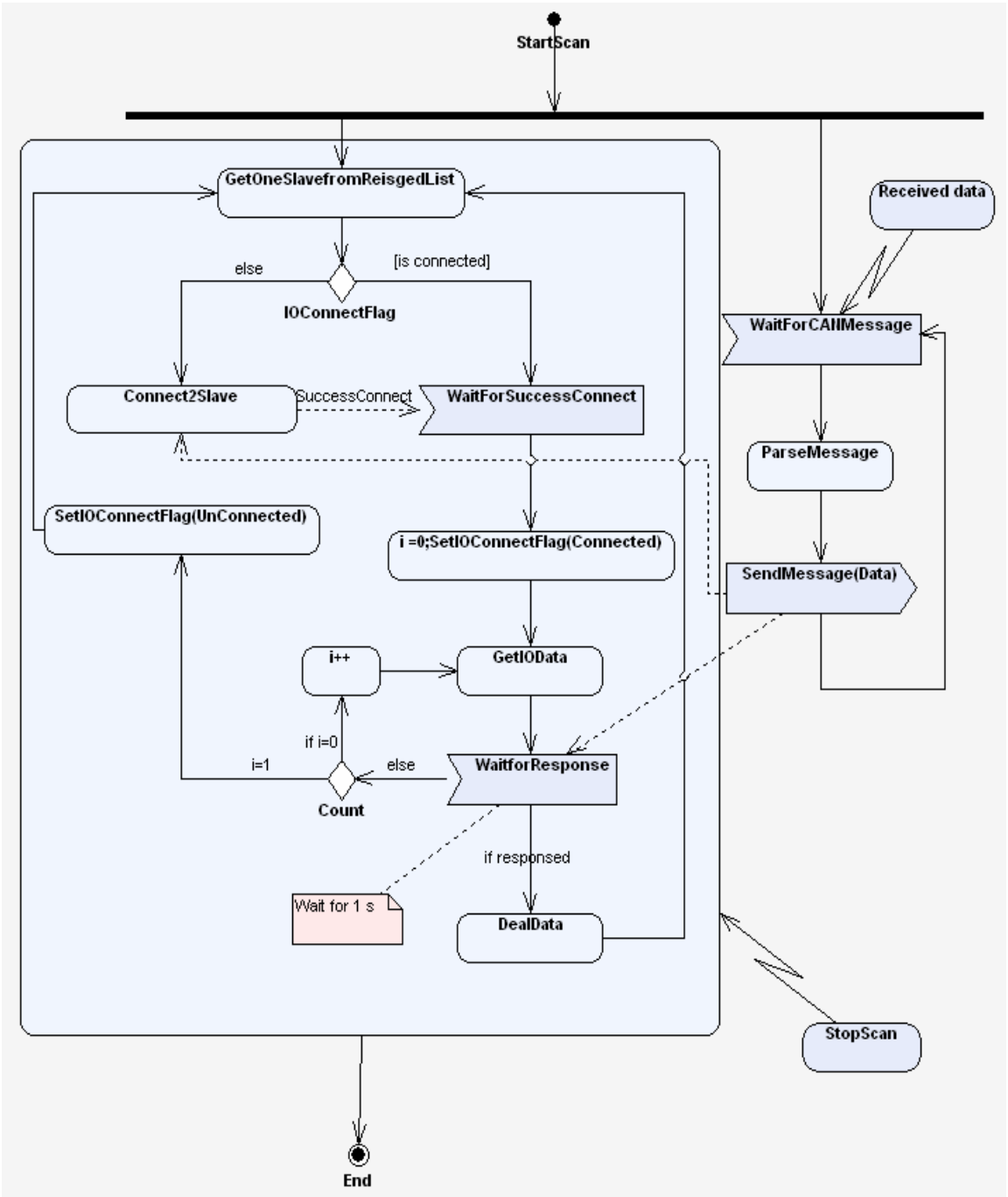

图 4-11 扫描过程活动图

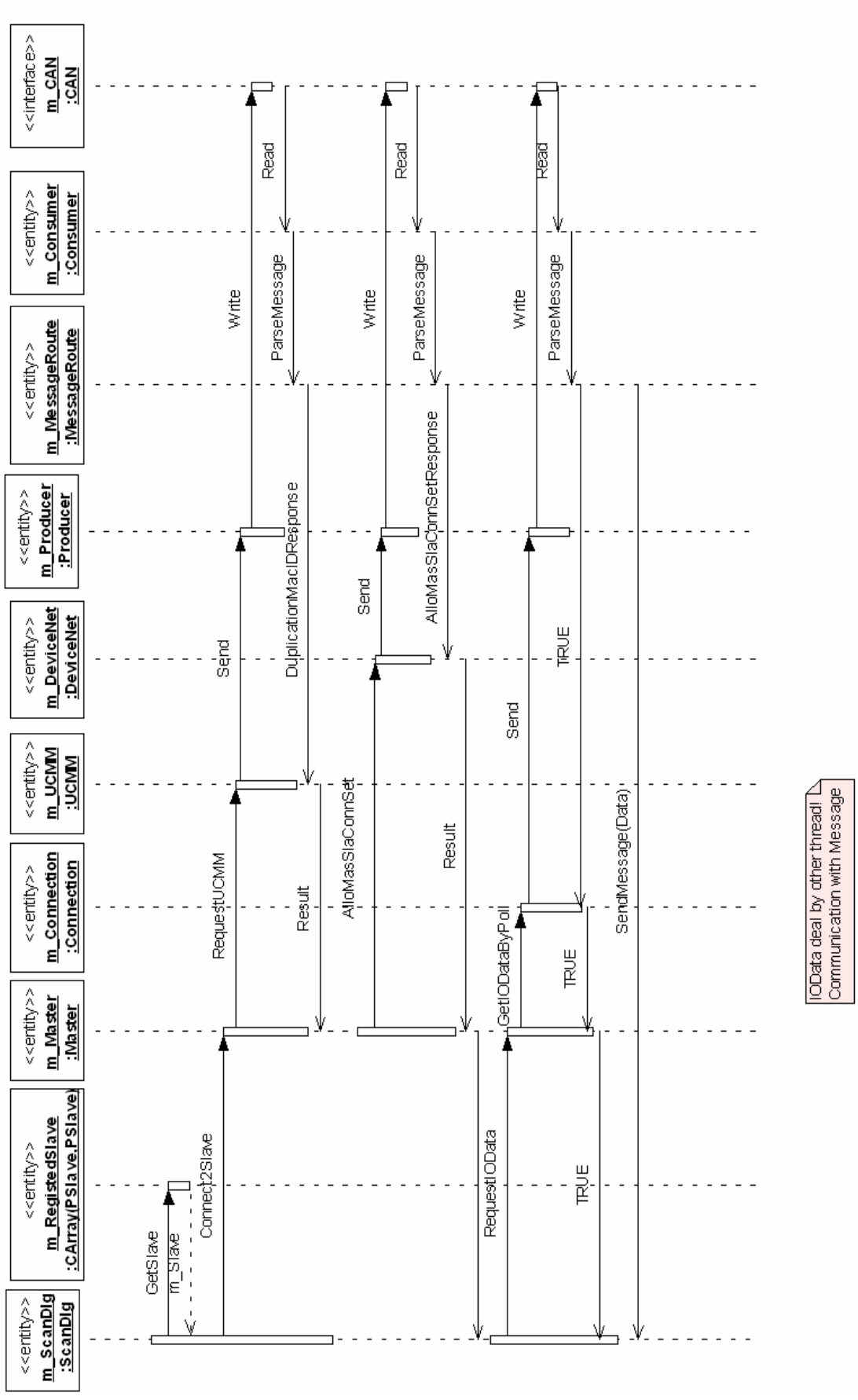

图 4-12 扫描过程顺序图

4.3.2.1预定义主/从连接组设计与实现

本文实现的DeviceNet主站网络管理与数据采集平台是针对预定义主/从连接组的，其连接过程活动图如图4-13所示，先请求通过UCMM建立显式连接，若显式连接成功，再请求通过建立的显式连接分配预定义主/从连接组；如果显式连接失败，则请求通过仅限组2非连接显式请求消息来分配预定义主/从连接组，主站按照DeviceNet协议发送报文。

显式消息连接CAN标识符格式如表4-1所示，组3信息ID5是显式消息连接响应帧头，组3信息ID6为显式消息连接请求帧头，源MAC ID是发送节点的MAC ID。

表4-1 建立显式消息连接CAN标识符格式

1	1	1	0	1	源MAC ID	UCMM Response Message
1	1	1	1	0	源MAC ID	UCMM Request Message

显示消息连接CAN帧数据域格式如图4-14所示，Frag为分帧标志位，0表示该帧是非分帧报文，1表示该帧是分帧报文，详细见2.3.2节；XID消息标记位，第一次发送是0，第二次发送是1；MAC ID是目的节点的MAC ID；R/R标志位0表示是请求报文，1表示响应报文；服务代码4B表示新建一个显式消息连接服务；消息格式指定基于该显式消息连接发送数据的格式，详细见参考文献[19]第一卷第四章表4.4；保留位是将来扩展用的，此处全设置为0；选择组和源信息ID表示建立的该显式消息连接将通过该选择组的这个源信息ID来通信；连接实例ID是指该连接对象Connectiong的第几个实例[17] [16][40]。

Offset	7	6	5	4	2	1	0
0	Frag[0]	XID	MAC ID				
1	R/R[0]	服务代码[4B]					
2	保留位(全为0)				请求信息格式		
3	选择组				源信息ID		
4							
5							

(a)请求数据域格式

Offset	7	6	5	4	3	2	1	0
0	Frag[0]	XID	MAC ID					
1	R/R[1]	服务代码[4B]						
2	保留位(全为0)						实际信息格式	
3	目标信息ID						源信息ID	
4	连接实例ID							
5								

(b)响应数据域格式

图4-14建立显式消息连接CAN数据域格式

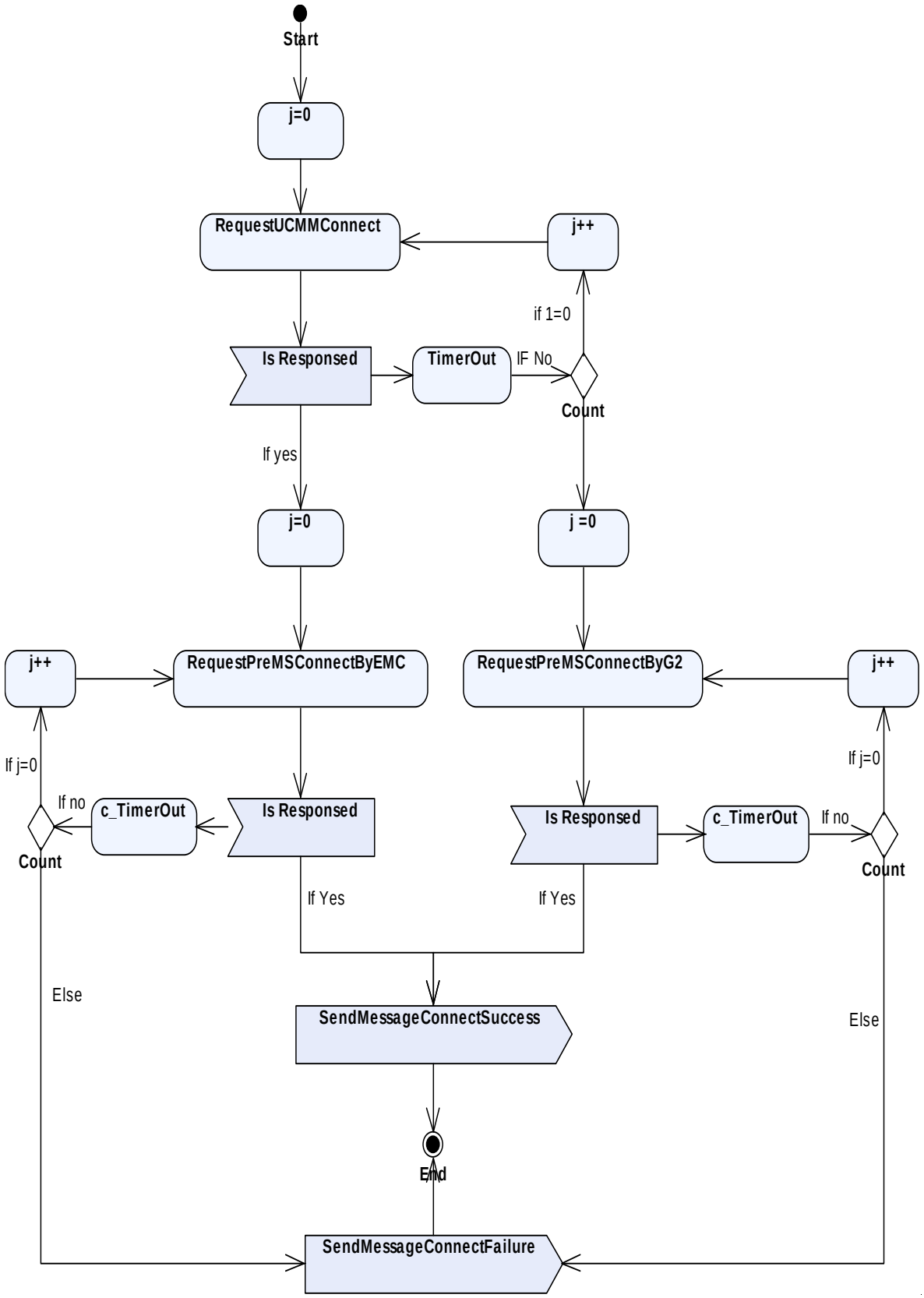

图 4-13Connect2Slave 过程活动图

仅限组2非连接显式请求消息分配预定义主/从连接组CAN帧数据域格式如图4-15所示，请求和响应报文的CAN标识区分别采用组2信息ID6和组2信息ID3；Frag、XID、R/R和服务代码4B与之前介绍的一样，MAC ID是本节点的MAC ID，即源MAC ID分类ID是Connection类的ID为3；实例ID为Connection类的实例ID，因为是第一个Connection实例，故是01；分配选择如表4-2所示，Acknowledge Suppression设置是否有应答方式，Cyclic位表示I/O连接触发方式是循环，Change位表示I/O连接触发方式是状态改变，Muliticast位表示I/O连接触发方式是广播方式，Bit Strboed位表示I/O连接触发方式是位-选通，Polled位表示I/O连接触发方式是轮询，Explicit Message位表示建立显式消息连接；分配的MAC ID是源MAC ID。

Offset	7	6	5	4	2	1	0
0	Frag[0]	XID	MAC ID				
1	R/R[0]	服务代码[4B]					
2	分类ID[3]						
3	实例ID[01]						
4	分配选择						
5	0	0	分配的MACID				

(a)请求数据域格式

7	6	5	4	3	2	1	0
Frag[0]	XID	MAC ID					
R/R[1]	服务代码[4B]						
保留位(全为0)					实际信息格式		

(b)响应数据域格式

图4-15分配预定义主/从连接组数据域格式

表4-2分配选择字节含义

7	6	5	4	3	2	1	0
保留	Acknowledge Suppression	Cyclic	Change Of State	Muliticast Polling	Bit Strboed	Polled	Explicit Message

分配预定义主/从连接组实现过程如下，Connect2Slave是连接方法，pSlave参数是要建立连接的从站对象指针。

```
DWORD Master::Connect2Slave(PSlave pSlave)
{
    请求通过UCMM建立显式消息连接；
    if  建立显式消息连接成功
```

```
        设置显式连接标识位；
        请求通过建立的显式连接来分配预定义主/从连接组；
        if 分配失败
            return FALSE;
        fi
    else  建立显式消息连接失败
        设置显式连接标识位；
        设置主站状态标识位；
        请求通过仅限组2非连接显式消息分配预定义主/从连接组；
        if 请求分配预定义主/从连接组失败
            return FALSE；
        fi
    fi
    获取分配选择字节中的各标志位；
    if 支持显示连接
        设置从站显示连接属性；
        if 设置失败
            return FALSE;
        fi
    fi
    …… // 设置从站其他属性
    return TRUE；
}  //end Connect2Slave
```

4.3.2.2轮询I/O连接设计与实现

本文主站I/O连接触发方式仅实现轮询I/O连接。轮询I/O连接CAN标识区请求与响应消息分别用组2信息ID5和组1信息ID15，详细示例如图4-16所示[17]。

主站根据从站的MAC ID计算连接ID(CID)即CAN标识区，如主站MAC ID为01，从站MAC ID为09，轮询I/O消息请求CAN标识区ID计算见式4-1，响应CAN标识区ID计算见式4-2，“{}”为拼接运算符。

{10，从站MACID，组2信息ID} =10001001101 =44D hex　　式4-1

{0，组1信息ID，源MACID} =01111001001 =3C9 hex　　式4-2

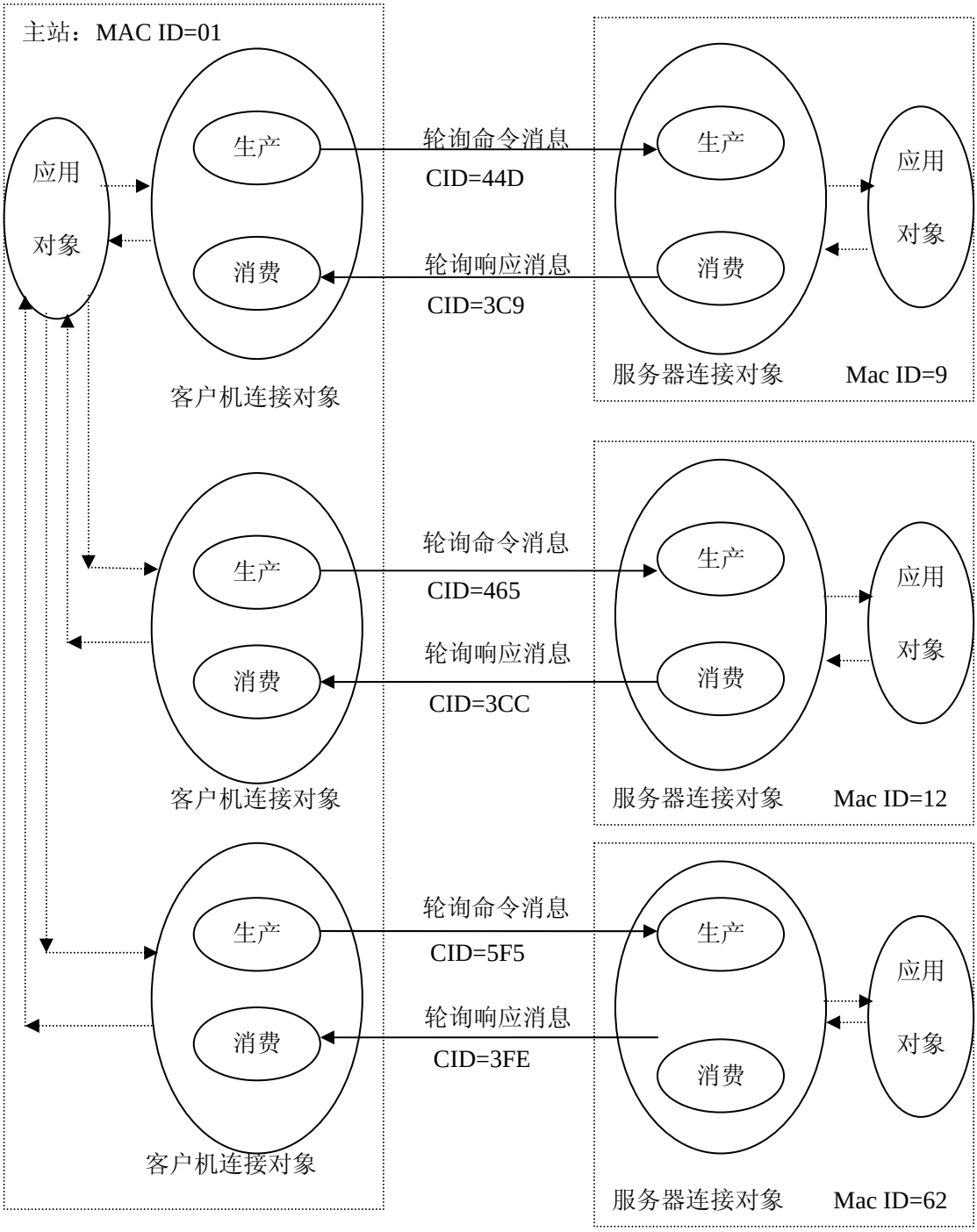

图4-16 轮询连接示例

4.3.2.3CAN报文处理设计与实现

本文采用多线程设计，CAN报文接收处理用单独线程处理，通过CAN驱动接口函数BUS_IOControl扫描CAN驱动数据接收标志位，如果标志位为1，表示

驱动已经接收到CAN报文，则通过IO文件系统的readfile读取数据，数据接收标志位自动清零然，判断读取的报文数据域数据长度len是否为0，len不为0就调用parseMessage解析报文，len为0不处理读取的报文。接收流程图如图4-17所示[22]。

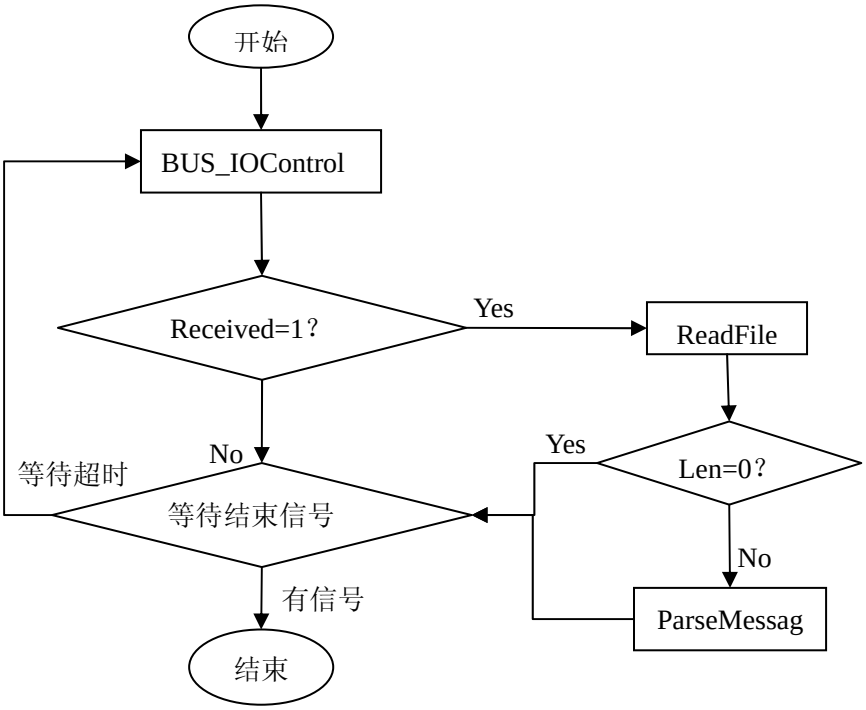

图4-17 CAN报文接收流程图

CAN报文数据读取伪代码如下：

```
……
while(1){
    读CAN驱动数据接收标志位；
    if  读取标志位失败；
        标识位清零；
    fi
    if  标志位已置位
        readfile读取CAN驱动内报文；
        if  读取失败
            continue;        //跳过剩余语句，不解析报文；
        fi
        if  报文数据长度=0
```

```
                continue;   //跳过剩余语句，不解析报文；
            fi
            调用ParseMessage解析报文；
        fi
        等待关闭信号2ms；
        if  有关闭信号
            break;      //有关闭信号，跳出循环
        fi   }   // end while
```

报文解析由MessageRoute类ParseMessage函数实现报，解析过程伪代码如下。

```
……
if   主站状态不在通信故障状态&接收到组4信息
    return   FALSE;
else if  组4报文
    处理该报文；
    return   TRUE;
 else if  组3报文
    switch  组3的信息ID{
        case Message ID 6:
            处理报文；
        case Message ID 5:
            处理UCMM信息；
            return   TRUE;
        case Message ID 4:
          ……                              ……
        case Message ID 0：
            处理报文；
            break;
        default:
            return   FALSE; } // end switch
```

```
        return    TRUE;
    ……          //其他组及各组信息ID解析类似上面
    ……
else if  组1报文
    switch  组1报文信息ID{
    case Message ID F:              //Polled响应信息ID
        if  主站在线状态
            调用G1MFIODataManage函数处理I/O数据；
            return    TRUE；
        fi
     case Message ID E:
        …… //处理其他  I/O报文}     //end switch
fi
```

4.3.2.4I/O数据处理设计与实现

在4.4.2.2节报文处理过程中，接收到I/O数据后，调用G1MFIODataManage函数处理组1信息IDF的I/O报文，获取报文源MAC ID，根据MAC ID获取注册列表中的从站Slave对象指针，使能事件通知请求I/O数据的线程已接收到响应，将接收到的报文数据逐字存入从站对象，再通过消息机制通知数据处理线程处理I/O数据。G1MFIODataManage函数伪代码如下[29][31][41]。

```
DWORD G1MFIODataManage(UCHAR* pMessage,LPVOID lparam)
{
    ……
    获取从站的MAC ID；
    根据MAC ID获取Slave对象指针m_pSlave；
    if m_pSlave为空
        return FALSE；
    fi
    if pSlave在等待I/O数据响应
        使能事件通知等待线程已接收到响应；
    fi
```

保存报文数据于m_pSlave;

发送消息；

} //end G1MFIODataManage

在主线程Scan类中系统重载WindowProc虚函数，根据MFC(Microsoft Foundation Classes)消息机制，可以在WindowProc中处理自定义消息。自定义消息从G1MFIODataManage函数处理完报文后发送或者请求I/O数据等待超后发送，WindowProc函数接收到消息后判断是不是自定义消息，如果不是自定义消息，则调用MFC结构默认的WindowProc函数去处理，如果是需要特定处理的自定义消息，按照自定义消息类型及参数执行相应语句，其实现过程如下：

```
LRESULT ScanDialog::WindowProc(UINT message, WPARAM wParam,
                                LPARAM lParam)
{   // TODO: Add your specialized code here and/or call the base class
    if 消息ID为自定义消息ID        //WM_MY_MESSAGE
        处理消息lParam参数，类型转换DOWRD;
        处理消息wParam参数，类型转换为Slvae;
        调用UpDataIODATA处理I/O数据
    else      调用CDialog::WindowProc处理其他消息；
    fi}
```

根据实现应用环境的不同，定制UpDataIODATA实现各种功能，比如数据图形化显示、实时保存、根据数据控制总线节点等。本文将在下一章DeviceNet主站网络管理与数据采集平台在环境检测中的应用中定制UpDataIODATA函数。

第 5 章 网络管理与数据采集平台在环境检测中的应用

前面讲述了 DeviceNet 网络管理和数据采集平台，可以根据实际应用需求定制 EDS 配置文件和 UpDataIODATA 函数。本文根据项目需求将该平台应用于实际环境检测中。

5.1 搭建应用系统

根据本文设计开发的DeviceNet主站网络管理与数据采集平台，将该平台定制应用于环境检测，总线节点布置如图5-1所示。

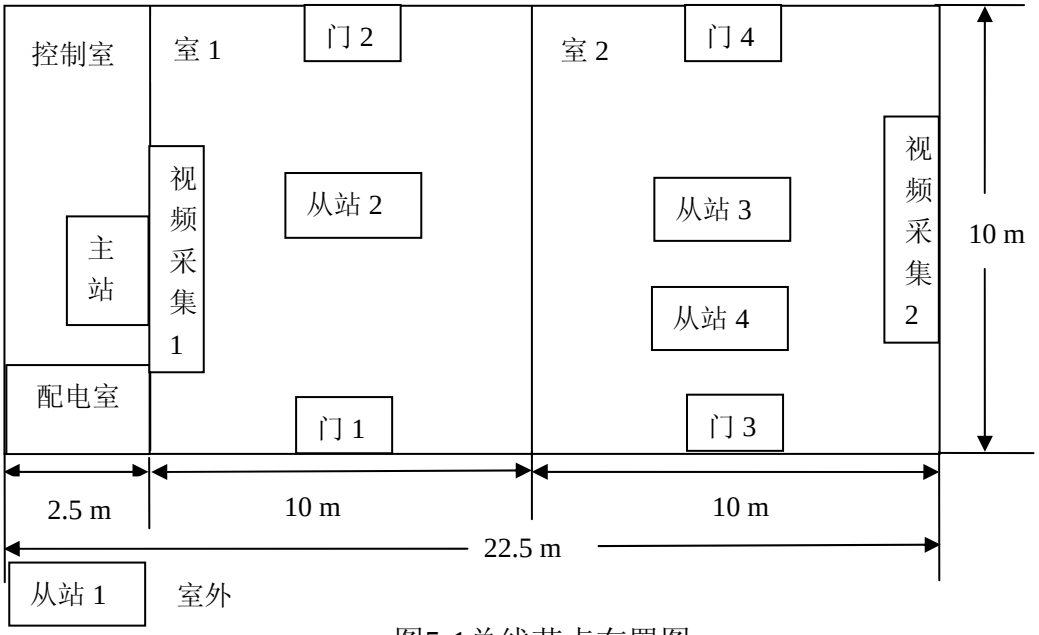

图5-1总线节点布置图

整个总线网络有5个节点，主站节点放置在控制室，其他四个节点从站1、从站2、从站3和从站4分别放置室外、室1和室2。总线各节点实现如下功能：

- 从站1：采集室外温度、湿度和检测系统电源；
- 从站2：采集室1内温度、湿度、亮度、门1和门2开关状态；
- 从站3：采集室2内温度、湿度、亮度，门3和门4开关状态
- 从站4：采集室2内的烟雾；
- 主站：定制扩展网络管理与数据采集平台实现从站配置、管理、数据采集、数据图形化显示、数据实时保存和数据异常报警，同时采集室1和室2内的两路视频图像，通过外部按钮可切换视频1、视频2与应用程序之间的界面。

注：因为温度、湿度和亮度传感器是直接安装在从站上的，而烟雾传感器会产生大量热量，导致从站温度比周围环境高很多，所以采集烟雾使用独立从站。

从站采用Pic18F458单片机，每个从站下面挂有传感器，各从站和两路摄像头由CAN总线供电，视频图像由单独信号线传送。根据总线节点布置图，定制主站I/O数据显示页面，根据各从站需求编写从站配置文件。

数据图形化显示用Microsoft eMbedded Visual C++4.2开发工具提供的控件实现，每个从站下面的传感器向下绑定一个显示控件，向上绑定对应从站I/O数据域。主站在接收到I/O数据后，按照4.3.2.3节所实现的流程执行至UpDataIODATA函数，UpDataIODATA在此需要重新定制，其伪代码如下[42]。

```
Scan::UpDataIODATA(PSlave m_pSlave,DWORD m_state)
{   获取从站所挂传感器个数Num;
    获取从站类型;
    for(int i=0;i<Num;i++){
        获取传感器Sensor对象指针;
        调用UpDataList处理传感器数据
    }  //end for
} //end UpDataIODATA
```

UpDataList (CString m_SlaveType, int m_State,PSensor m_Sensor)函数根据其参数从站类型m_SlaveType和传感器对象m_Sensor的name属性，得到数据图形化显示界面绑定的控件；m_State参数用于标识获取从站I/O数据是否成功，为0就表示获取该从站I/O失败，从站挂载的传感器绑定的控件显示灰色背景和“Off“；如为1表示获取I/O数据成功，则调用Draw(float m_State,int m_ID)函数图像化显示数据和调用SaveData函数保存数据到MyQueue队列和SD卡存储器中。

Draw函数用于图形化显示数据，并按照从站配置文件设置比较判断数据是否超出设定范围，如果超出设定范围就设置数据显示背景为红色并发出警报声(前提是报警已使能)，如果数据在设定范围内则正常显示数据，数据背景颜色为绿色；SaveData函数将数据压入传感器的MyQueue数据队列，数据队列保存最新若干组数据，同时将数据保存于SD卡存取器内文件，数据永久保存，文件名根据从站类型和传感器name自动生成。I/O数据处理活动图如图5-2所示[43]。

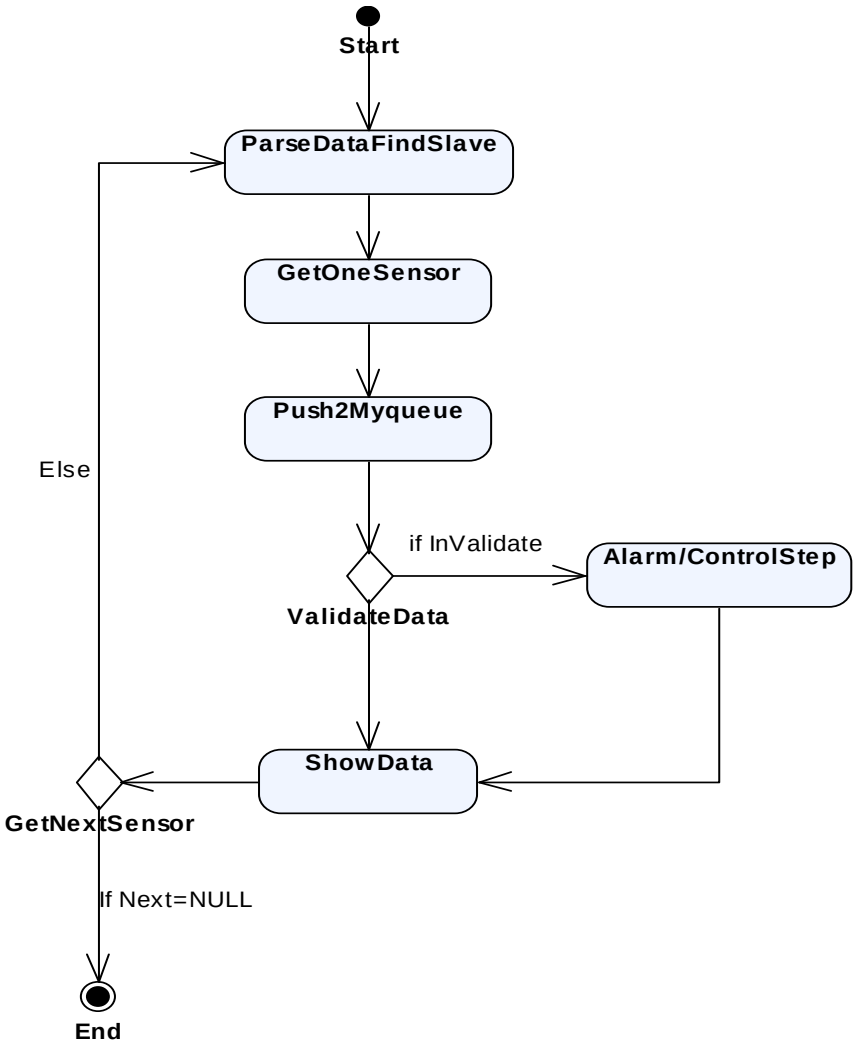

图5-2 I/O数据处理活动图

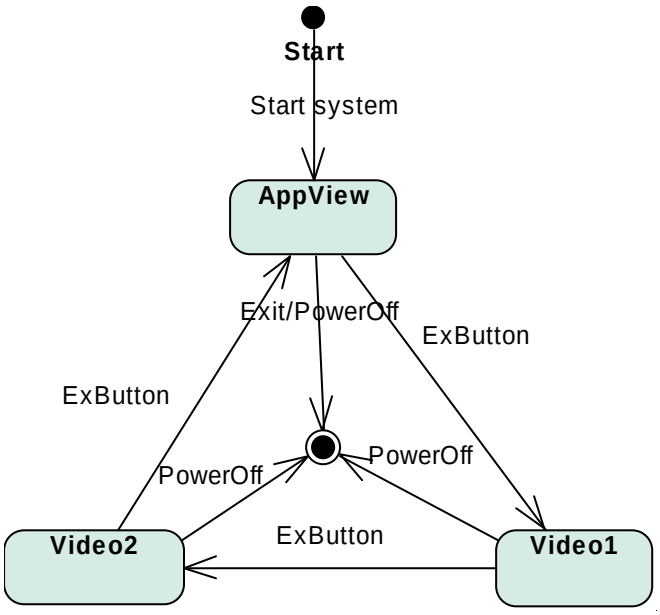

图5-3 视频与应用程序界面状态图

视频与应用程序界面切换通过外部按钮中断实现，外部按钮触发一次，视频1、视频2与应用程序界面之间切换一次；外部中断发生后，会在按钮驱动设置中断标志位，然后马上结束中断，应用程序单独一个线程通过EXB_IOControl扫描按钮驱动的中断标志位，标志位每置位一次，界面切换一次，其执行状态图如图5-3所示。

5.2 DeviceNet 主站应用系统运行结果

应用程序界面有两部分组成，网络管理界面如图5-4所示，网络扫描数据采集显示界面如图5-5所示。

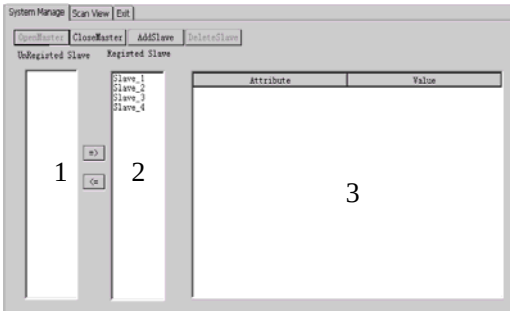

图5-4 网络管理界面

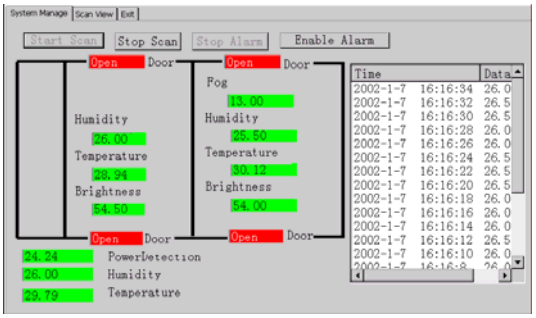

图5-5 数据采集显示界面

网络管理与数据采集界面之间通过左上角标签“System Manage”和“Scan View”按钮切换，“Exit”按钮释放资源，退出系统；图5-5中“1”、“2”控件分别是未注册列表和注册列表，“3”控件是从站属性编辑框；图5-6右边列表显示的是最新若干组保存的数据及数据获取的时间，供用户查询，见面中绿色或红色中显示的是各从站下传感器采集的数据，颜色表示数据与设定范围比较结果。

打开主站时用户可以选择导入配置文件或直接打开主站如图5-6：“Manual Edit”表示直接打开主站，“Import EDS file”表示导入配置文件方式打开主站。

系统完成了预期的所有功能，按照需求分析用例图对系统进行测试，编写了测试报告，所有功能正常运行，系统通过了上海市软件测评中心项目验收测试，并申请取得了系统的软件著作权。

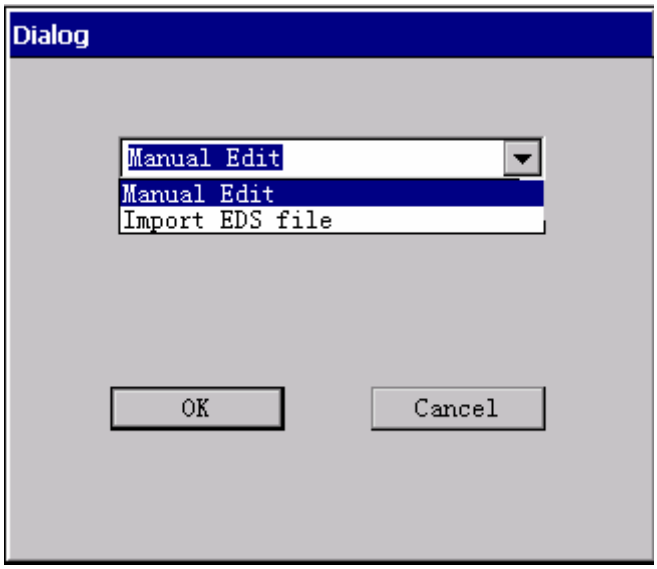

图5-6 配置方式选择窗口

第 6 章 总结与展望

现场总线技术在控制领域崛起和发展了二十几年，凭借其自身的种种技术优势，在全世界范围内都获得了迅速的发展。DeviceNet 作为一种高效、可靠、易用、廉价的现场总线协议，已成为事实上的现场总线标准。在我国有很多专家学者投入到现场总线技术的研究中，但现场总线在我国的发展还很薄弱。

本文先分析了现场总线与 DeviceNet 总线技术的研究现状和发展趋势，深入研究分析了 DeviceNet 协议及其通信模型、拓扑结构及应用层协议。参与设计实现基于 ARM9 的 DeviceNet 主站硬件平台；按照软件工程学思想独立分析设计 DeviceNet 主站需求模型、实现了主站功能，在此基础上设计开发 DeviceNet 网络管理与数据采集平台，最后将该平台定制应用于环境检测中，通过了上海市软件测评中心项目验收测试，申请取得了软件著作权。

由于 DeviceNet 协议本身的复杂性，本文只实现了 DeviceNet 预定义主/从连接组中的 Polled 连接方式和 DeviceNet 主站的核心功能，由于开发条件和时间的限制，对主站代理其下所挂从站的 UCMM 功能和分段报文等功能未实现，有待于以后继续研究开发实现。但本文在嵌入式系统中实现 DeviceNet 主站是一种全新的尝试，并取得了预期的成果。现场总线在我国还有很大的发展空间，同时要认识到同世界先进技术水平的差距，以及国外产品竞争对我国的影响。在本文研究的过程中，得到了以下几点体会：

- 我国总线产品各研究单位缺乏强有力的交流；
- 总线产品的研发没有形成特定产品链，如 DeviceNet 协议标准信号线国内没有一家企业生产，需要从日本或美国进口，间接导致国内 DeviceNet 总线应用成本增加；
- 目前我国企业总线产品更多偏重于应用，研发主要集中于高校，由于研究经费等限制，总线技术的发展与欧美先进技术有一定差距。

随着我国越来越多的专家学者投身于现场总线技术研究、国家对现场总线技术的重视和更多企事业单位的加入，我们有理由相信，不久的将来会出现属于自己的现场总线。

参考文献

[1] 王岩，张福恩：现场总线技术的现状与发展[J]，电子器件，2001, 24(1): 79~84

[2] Peter Neumann: Communication in industrial automation—What is going on? [J], Control Engineering Practice, 2007, 15: 1332~1347

[3] 王桂荣，钱剑敏：CAN总线和基于CAN总线的高层协议[J]，计算机测量与控制，2003, 11(5): 391~394

[4] 蔡忠永，高涵：现场总线发展概况与DeviceNet进入中国[J]，低压电器，2000, (2): 59~61

[5] Zhi Wang, Xingfa Shen, Jiming Chen, Yeqiong Song, Tianran Wang, Youxian Sun: Real-time performance evaluation of urgent aperiodic messages in FF communication and its improvement [J], Computer Standards & Interfaces, 2005, 27: 105~115

[6] XU Wei-hua, Wolfgang Foerster, Guenter Mueller: A Universal Portable Appliance for Stellarator W7-X Power Supply Controlling, Plasma Science & Technology, 2001, 3(3): 781~790

[7] Subrat Kaushik , K.K.M. Haneef, M.N. Jayaram and D.K. Lalsare: Advantages and Safety Features using Foundation Fieldbus-H1 based Instrumentation & Control for Cryo System in Accelerators [J], International Symposium on ”Vacuum Science and Technology”，Journal of Physics: Conference Series, 2008, 114: 1~4

[8] 奚培锋：一种DeviceNet模拟量远程I/O模块的设计[J]，通用低压电器，2007, 1: 30~32

[9] 北京博控自动化技术有限公司: CANopen 协议介绍

[10] 吴乃优等：DeviceNet 网络的通信原理[J]，电气传动，2000, 6: 51~53

[11] Nilufer Cenesiz, Murat Esin: Controller Area Network(CAN) for computer integrated manufactural systems [J],Journal of Intelligent Manufacturing, 2004, 15: 481~489

[12] Steve Biegacki, Dave VanGompel: The application of DeviceNet in process control [J], ISA Transactions, 1996, 35: 169~176

[13] 董爱华，舒国汀：DeviceNet总线技术应用探讨[J]，东华大学学报(自然科学版)，2002, 28(5): 75~59

[14] 阮于东：DeviceNet 总线技术(一)[J]，低压电器，2002, 3: 59~61

[15] 坂东卫持：DeviceNet现场总线的现状及今后的发展[J]，低压电器，2000, 5: 60~62

[16] 阳宪惠：现场总线技术及其应用[M], 北京，清华大学，1999

[17] BOSCH: CAN Specification Version2.0

[18] 邬宽明：CAN 总线原理与应用系统设计[M],北京，北京航空航天出版社，1996

[19] Open DeviceNet Vendor Associatiing on Inc., DeviceNet Specification, Release 2.0, including Errata 4, April 1, 2001

[20] 易建钢，刘光临，严开勇，陈奎生：DeviceNet现场总线及其在风机远程监控及故障诊断系统中的应用[J]，噪声与振动控制，2007, 1: 90~92

[21] 刘代飞，李劼，丁凤其，郑文波：基于DeviceNet现场总线智能从节点的开发[J]，计算机测量与控制，2007, 15(3): 384~386

[22] 顾祥，江颖，左小五，许伟明：DeviceNet网络的通信研究[J]，湖南工程学院学报，2007, 17(2): 67~70

[23] 陈维刚，费敏锐，边宁宁：基于嵌入式扫描器的DeviceNet网络[J]，自动化仪表, 2004, 25(8): 16~19

[24] 洪剑青，赵德安，孙月平，方政：DeviceNet 总线的应用[J]，通用低压，2007, 7, 21~24

[25] 王忠清，陈雪松，曹微言，戴晓飞，李荫涛：基于 DeviceNet 现场总线的从站接口卡的设计与实现[J]，信息化技术与控制，2007, 17(3): 13~16

[26] 曹茂虹，吴建新，杨芬：DeviceNet 智能通信接口设计[J]，工矿自动化，2007，3：110~112

[27] 高艳芳，胡颖：DeviceNet 现场总线主站通信接口的研制[J]，华东交通大学学报，2006, 23(1): 121~124

[28] 李国洪，王景芹，武壮，刘金梅：DeviceNet 在智能电器控制网络中的应用电测与仪表[J]，2006, 7: 55~59

[29] Tektronix: TDS5000B Series Digital Phosphor Oscillooscopes Quick Start User Manual, Beaverton

[30] Wootae Jeong, Shimon Y. Nof: Performance evaluation of wireless sensor

network protocols for industrial applications [J], Journal of Intelligent Manufacturing, 2008, 19: 335~345

[31] 姜波：Windows CE.Net 程序设计[M]，北京，机械工业出版社, 2007

[32] 浙江大学罗克韦尔自动化技术中心：可编程序控制器系统[M]，杭州，浙江大学出版社，2000

[33] Robert L.Kruse, Alexander J.Ryba: Data Structures And Program Design[M]，北京，高等教育出版社，1999

[34] H.M.Deitel, P.J.Deitel 著，薛万鹏等译：C++程序设计教程[M]，北京，机械工业出版社,2002

[35] 侯俊杰:深入浅出 MFC[M]，第二版，华中科技大学出版社，武汉，2001 年 1 月

[36] Jim Arlow, Ila Neustadt: UML and the Unified Process: Practical Object Oriented Analysis & Design [M], NJ, Pearson Education, 2002

[37] Microsoft Corporation: Microsoft Win32 程序员参考大全[M]，欣力等译：北京，清华大学出版社，1995

[38] 汪兵等：EVC 高级编程及其应用开发[M]，北京，中国水利水电出版社，2005

[39] 探矽工作室，嵌入式系统开发圣经[M]，北京，中国铁道出版社，2003

[40] Microsoft Corporation, 希望图书创作室译：Microsoft Windows CE User Interface Service Guide[M]，北京，北京希望电子出版社

[41] 杜春雷，ARM 体系结构与编程[M]，北京，清华大学出版社，2003

[42] Samsung: S3C2440 datasheet, 2004

[43] Microchip: MCP2510 datasheet

[44] PHILIPS: Saa711X datasheet

[45] 周立功单片机：单片机与嵌入式系统解决方案，2005 年 5 月

[46] 周立功：iCAN 现场总线原理与应用[M]。北京航空航天大学出版社,2007 年

附录一 攻读硕士学位期间发表的学术论文及所参与项目

科研项目经历：

1、2007 年 2 月至 2007 年 10 月，本人参加嵌入式研究所“高可信芯片设计前端平台与工业控制芯片设计应用”项目。在金乃咏老师和倪韬雍学长的指导下，参与面向 IP 的 PSL 规范集成设计工具的部分功能模块的设计与实现。

2、2007 年 11 月至 2008 年 9 月，本人参加嵌入式研究所“高可信芯片设计前端平台与工业控制芯片设计应用”项目。在金乃咏老师的指导下，检索阅读了相关科技文献，提出了自己的创新点与算法，设计实现基于 DeviceNet 协议的网络管理与应用系统，并通过上海市软件测评中心项目验收测评，申请获得该软件著作权。

附录二 EDS 配置文件格式

从站配置文件格式

从站配置文件可以配置从站属性和从站 8 字节数据段。配置语句格式如：name =value；以分号表示一个配置语句。

从站属性

从站可配置属性有整数型 MAC ID 支持数据范围是 0-64，整数型 BaudRate 支持 125Bps、250 Bps 和 500 Bps 三种、BOOL 型 BOI 支持 0 或 1、字符串型 Type，支持的 IO 触发方式有 Acknowledged，StateChange，Poll，Bit_Strobe，MulticastPoll，Cyclic，触发方式标志位是 BOOL 类型的支持 0 或 1，目前 1.0 版本触发方式只实现了 Poll，具体例子如下。

```
MacID =1;
BaudRate =125;
BOI =0;
Type = Outdoors;
Acknowledged =0;
StateChange =0;
Poll =1;
Bit_Strobe=0;
MulticastPoll=0;
Cyclic=0;
```

从站数据段配置

从站数据段最多有 8 字节，可以将传感器数据与数据段的 1 字节、多字节、或者 1 位绑定。而传感器的属性可以在绑定位置之后设定。以下是至少 1 字节情况：

```
Sensor_2:
        Name =Humidity;
        ByteOrder =2;
```

Length =8;

MinValue =10;

MaxValue =200;

Factor =2.56;

TempKeepLenth =20;

Sensor_2 是“Sensor“ +”_“ + ”id“，Sensor 是固定格式，表示开始配置传感器，以“:“结尾，“id”是传感器标示号；

Name 是传感器的类型名字，表示这个是什么传感器，以“;“结尾。

ByteOrder 表示传感器的对应从站 8 字节数据段的起始地址，地址是从 0 开始。

Length 表示数据位数，比如 1 字节这 Length=8；2 字节就是 Length =16；若是 1 表示 1 位。

MinValue，MaxValue 表示传感器的数据合理范围，若超出范围，则显示异常，相应图标显示红色，而正常是绿色，未在线是灰色。

Factor 是比率因子，接收到的传感器数据按照比率因子及传感器特性得到实际的值，比如温度传感器接收到数据是 86，比率因子是 1.5，那么实际温度时就(86-56)/1.5，为 20 摄氏度(计算公式和 56 是传感器类型决定的)。

TempKeepLenth 表示传感器数据最新的多少组数据保存于内存中。

如果传感器是 BOOL 型的只需要 1 位，以下是 1 位情况的配置文件格式：

Sensor_8:

Name = Door4;

ByteOrder =5;

Length =1;

BitOrder=3;

Factor =1;

TempKeepLenth =20;

与多字节传感器相比，不同之处在于 BitOrder 和 ByteOrder，ByteOrder 表示从站 8 字节数据段的第几位开始，而 BitOrder 表示 ByteOrder 指定字节的哪一位；不必设置 MinValue 和 MaxValue 值；所以如果传感器是 1 位的数据，那么在同一

ByteOrder 的不同位可以配置多个 BOOL 型传感器。

实际例子如下有 1 字节和 1 位传感器的从站配置文件：

```
MacID =2;
BaudRate =125;
BOI =0;
Type =Captivity;
Acknowledged =0;
StateChange =0;
Poll =1;
Bit_Strobe=0;
MulticastPoll=0;
Cyclic=0;
Sensor_2:
    Name =Humidity;
    ByteOrder =2;
    Length =8;
    MinValue =10;
    MaxValue =200;
    Factor =0.96;
    TempKeepLenth =20;
Sensor_5:
    Name =Door1;
    ByteOrder =5;
    Length =1;
    BitOrder =0;
    Factor =1;
    TempKeepLenth =20;
    TempKeepLenth =20;
```

主站配置文件格式

主站配置属性有 MACI D，BaudRate，BOI，AlarmTime，IntervalTime。MacID，BaudRate，BOI 三个属性与从站属性配置格式一样；AlarmTime 表示警报停止时间，在发出警报声情况下，点击 AlarmTime 可以使警报声暂停一段时间，时间单位为秒。

注：从站的 BaudRate 必须与主站 BaudRate 一样，否则不能正常通讯。所有站点包括主站和从站的 MACID 不能重复，否则重复 MACID 的站点最多只有一个能正常工作。IntervalTime 表示主站设置从站 ExpectedPacketRate 属性值。ExpectedPacketRate 属性含义请查看 DeviecNet 规范。

具体如下所示例子：

MacID =0;

BaudRate =125;

BOI =0;

AlarmTime =5;

IntervalTime =10000;

Slave_1 =Slave_1.EDS;

Slave_2 =Slave_2.EDS;

Slave_3 =Slave_3.EDS;

Slave_4 =Slave_4.EDS;

上面例子里的“Slave_1 =Slave_1.EDS;“表示配置主站下所挂从站及从站配置文件名，”Slave”表示主站下所挂从站”1”表示从站 MACI D 为 1，与从站配置文件里的 MAC ID 同一个值；”=“后面的”Slave_1.EDS “是该从站的配置文件，其格式在上面已经定义，”;”表示语句结束。

致 谢

在此我首先感谢金乃咏和郭建两位导师给予我的悉心指导和关怀。在我攻读硕士学位的时间里，在学习、生活和工作中得到了导师关怀和教诲，使我的理论水平和工作能力有了较大提高。同时，导师那渊博的知识、精湛的学术造诣、严谨的治学作风、孜孜不倦的工作精神使我终生受益。导师一丝不苟的工作态度，勇于创新、与时俱进的精神将永远激励着我努力工作和学习。值此论文成稿之际，谨对尊敬的导师表示崇高的敬意和衷心的感谢！

我还要特别感谢我的同事林敏。感谢他在我完成本论文期间学习和工作上的巨大帮助。从他身上，我学到很多很多。还要特别感谢我的同事陈俊、韩菲及师兄倪韬雍等，和你们在一起生活、学习是一段非常宝贵的经历。

与此同时，我还要对上海嵌入式系统研究所的同事们表示感谢，从你们身上学到很多很多，在工作和学习上给于我巨大支持，非常感谢你们给予我的帮助。

感谢华东师范大学软件学院的师生给予我许多帮助和热情鼓励，感谢上海市科委登“高可信芯片设计前端平台与工业控制芯片设计应用“的资助。

在我多年的求学过程中，得到了家人始终如一的全力支持和无私奉献，他们的关怀、照顾和鼓励使我得以坚持完成学业，在此表示深深的谢意。

感谢各位评阅老师对本论文的审阅和提出的宝贵意见。

最后，谨以此文献给所有教导和关心过我的人。

衷心的谢谢你们！