分 类 号 TP393　　学　　号 2009005008

密　　级

西安石油大学

Xi'an Shiyou University

硕 士 学 位 论 文

题　　目	CANopen 协议 在 RTU 通信控制模块中的应用研究
作 者 姓 名	胡欣欣
指导教师姓名、职称	曹庆年 教授
学科(专业)名称	计算机应用技术
提交论文日期	2012 年 5 月 20 日

学位论文创新性声明

本人声明所呈交的学位论文是我个人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除了文中特别加以标注和致谢中所罗列的内容以外，论文中不包含其他人已经发表或撰写过的研究成果；也不包含为获得西安石油大学或其它教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中做了明确的说明并表示了谢意。

申请学位论文与资料若有不实之处，本人承担一切相关责任。

论文作者签名：_____ 日期：_____

学位论文使用授权的说明

本人完全了解西安石油大学有关保留和使用学位论文的规定，即：研究生在校攻读学位期间论文工作的知识产权单位属西安石油大学。学校享有以任何方法发表、复制、公开阅览、借阅以及申请专利等权利。本人离校后发表或使用学位论文或与该论文直接相关的学术论文或成果时，署名单位仍然为西安石油大学。

论文作者签名：_____ 日期：_____

导　师　签名：_____ 日期：_____

论文题目：CANopen 协议在 RTU 通信控制模块中的应用研究
专　　业：计算机应用技术
硕 士 生：胡欣欣（签名）_____
指导教师：曹庆年（签名）_____

摘　要

CANopen 协议是一种开放的、标准的 CAN 现场总线的应用层协议，它的规范全集除了包括应用层和通信协议之外，还包括各种不同的框架、建议，以及标准的设备规范、接口协议与应用技术规范，支持不同制造商的设备互操作和互换。RTU 远程测控终端，负责对现场数据采集、工业设备监测和控制，较适合于恶劣的工业现场环境中。本文以油田 RTU 的应用为背景，将 CANopen 协议应用于 RTU 上，不仅能对现场设备数据进行采集、传输，而且实现了对现场设备进行实时监控。

本文首先提出了一种基于 CANopen 协议的 RTU 模块的监控系统，给出了以 AT91SAM7X256 微处理器为主控制器的嵌入式 RTU 模块的硬件设计；其次详细剖析了 CAN 总线以及 CANopen 协议，并对 CANopen 协议中的对象字典、PDO 对象、SDO 对象、NMT 对象、特殊功能对象等进行了详细阐述。为了增加嵌入式 RTU 系统的可靠性和实时性，本监控系统选取了源码开放的μC/OS-Ⅱ嵌入式实时操作系统。

最后，根据本监控系统的功能需求，在系统的主从 RTU 上实现 CANopen 协议，并进行测试。测试结果表明，基于 CANopen 协议的主从 RTU 节点通信正常，具有良好的可靠性和实时性。

关键字：　CAN；CANopen；通信协议；远程测控终端
论文类型：应用研究

（本研究得到 RTU 联合实验室基金的资助）

Subject： Applied Research of CANopen Protocol in RTU Communication Control Module

Specialty： Computer Application Technology

Name： Hu Xinxin(signature)_____

Instructor：Cao Qingnian(signature)_____

ABSTRACT

CANopen protocol is one kind of an open standard application layer protocol for CAN bus,CANopen specification complete works in addition to the application layer and communication protocols,including a variety of framwork,suggestions,as well as the standard equipment specifications,interfaces and application technical specifications. And it supports equipments interoperability and interchangeability which are made of different manufacturers. RTU is responsible for the field data collection,monitoring and control of industrial equipments,it is very suitable for harsh working environment. According to the background of oil field,CANopen protocol used in RTU not only could gather and transmit data,but also could achieve real-time monitoring of devieces.

This paper first presents a monitoring system based on the CANopen protocol which is used in RTU,and gives the hardware design of the RTU which takes AT91SAM7X256 microprocessor as main controller.Secondly,CAN bus is described in detail,as well as object dictionary,PDO object,SDO object,NMT object,special features object and so on of CANopen protocol. In order to increases the reliability and real-time of embedded RTU system,selectes μC/OS-Ⅱ which is open-source,real-time embedded operation system.

Finally,according to the functional requirements of the monitoring system, the CANopen protocol has been implemented in RTU and tested. The experimental results show that the communications among master and slave RTU nodes based on CANopen protocol work well，and have the characteristics of reliability and real-time.

Key words：Controller Area Network;CANopen;Communication Protocol;RTU

Thesis： Application Study

（The dissertation is supported by the RTU Joint Laboratory Fund）

目 录

第一章　绪论

1.1 课题研究的背景与来源

1.1.1 课题研究的背景

随着工业生产规模的不断扩大和对生产过程自动化要求的不断提高，利用数据转换技术作为数据测量、采集处理和过程控制的基本手段，并与计算机技术、通信技术相结合的分布式监控系统，成为控制系统硬件发展的趋势。二十世纪八十年代美国石油天然气行业提出了远程测控终端（RTU）的产品概念，作为体现“测控分散、管理集中”思路的产品最早产生于美国，应用于北美，并随着天然气、电力行业的自动化发展而被普遍应用。

RTU 是 SCADA 系统的基本组成单元，其主要负责对现场信号、工业设备的监测和控制。SCADA 是 Supevisory Control and Data Acquisition 的缩写，是应用于分布距离远、生产单位分散的生产系统的一种数据采集与监视控制系统，在油气田、环保、热网、水文水利、电力、交通等的监控系统中得到广泛应用。RTU 至少具有数据采集及处理、数据传输（网络通信）功能，许多 RTU 还具备 PID 控制功能或逻辑控制、流量累计等功能。RTU 具有很多优点：灵活互相兼容的开放式接口；良好的、可靠的通信能力，通信距离长；大容量存储；便于扩展的模块化结构；不仅可以应用于室内场合，而且可以用于室外控制装置中，甚至于环境条件比较恶劣的场合。RTU 在远端数据采集转换及通信方面具有明显的优势，在广域分布式监控系统中得到越来越多的应用。

现场总线系统是应用在生产现场的，在测量控制设备之间实现双向、串行、多点通信的数字通信系统，也被称为开放式、数字化、多点通信的底层控制网络[1]。把现场总线应用在现场设备中，采用统一规范的通信协议以及通信介质，便可构成针对于不同应用的自动化控制系统。现场总线控制系统（Fieldbus Control System，FCS）是一项集嵌入式系统、控制、计算机、数字通信、网络为一体的综合技术。

CAN（Controller Area Nerwork）控制器局域网络，是国际上应用最广泛的现场总线之一，最初是由德国 BOSCH 公司为汽车应用而开发的一种能有效支持分布式控制实时控制的串行通信网络[2]。其总线规范现已被 ISO 国际标准组织制订为国际标准（ISO 11898）。CAN 总线传输时间短，受干扰概率低，检错能力强，最大通信距离可达 10km，最大通信波特率可达 1Mbps，最多可挂接 110 个设备；采用非破坏性位仲裁总线结构机制，能有效地支持具有很高安全等级的分布式实时监控系统。CAN 的应用范围从高速的网络到低价位的多路配线都可以使用，目前其应用范围已不再局限于汽车行业，而向自动控制、航空航天、航海、过程工业等领域发展。

现场总线网络一般只实现了 OSI 网络模型的第 1 层（物理层）、第 2 层（数据链路层）、第 7 层（应用层），而 CAN 只定义了物理层和数据链路层，没有规定应用层，本身并不完整，需要一个高层协议来定义 CAN 报文中的标识符和数据的使用。而且，在

基于 CAN 总线的工业自动化应用中，越来越需要开放的、标准的高层协议，要求能支持不同制造商的设备互操作和互换。CANopen 是一种由 CiA（CAN-in-Automation，自动化 CAN 用户和制造商协会）定义的 CAN 的应用层协议，采用面向对象的思想设计，具有很好的模块化特性和很高的适应性，具有相对较广的应用范围。在 1995 年，CiA 发表了完整版的 CANopen 通信子协议，仅仅用了 5 年的时间，它已成为全欧洲最重要的嵌入式网络标准[3]。CANopen 不仅定义了应用层和通信子协议，而且为可编程系统、不同器件、接口、应用子协议定义了大量的行规，遵循这些行规开发出的 CANopen 设备将能够实现不同公司产品间的互操作性。

1.1.2 课题来源

本课题来源于西安石油大学-北京安控科技发展有限公司 RTU 联合实验室合作项目。

1.2 国内外研究现状与发展前景

1.2.1 国内外研究现状

从上个世纪八十年代，RTU 作为体现“测控分散、管理集中”思路的产品在国内外得到广泛的应用，并逐步从集中式控制结构向模块化、分散式、开放性的系统控制结构发展[4]。近年来，众多国际知名自动化巨头都参与到 RTU 领域的开发生产中，并加大在 RTU 方面的研发速度。艾默生具有 ROC800 等系列 RTU，在 2006 年收购了知名 RTU 厂商 Bristol Babcock（BB）公司；Honeywell 在 2008 年之前以外购 RTU 为主，2008 年开始有了自己 RC500 系列 RTU；Schneider 在 2009 年收购了加拿大 RTU 专业厂商 Control Microsystems 公司，使其由以 PLC 为主的工业应用控制系统扩展到 RTU 应用领域；ABB 公司近两年推出自己的 RTU560 系列，进一步扩展了 RTU 应用领域；Sixnet 公司是一直从事研发生产 RTU 的专业公司，近两年重点为知名自动化公司做贴牌业务；日本横河主要以研发生产 DCS、PLC 而知名的公司，最近也推出了一款 RTU 产品；而在欧洲主要有一些遵循 PROFIBUS、CANopen 等通讯标准的西门子、施耐德等品牌的 RTU 设备。国内一些厂商也在研发 RTU 产品，国内应用较多并具有代表性的 RTU 产品主要包括解放军南京工程兵学院微机测控研究所的 893-IDCB 和北京安控科技发展有限公司的 SuperE。

在国外，CANopen 协议应用在了很多领域，CANopen 协议在欧洲的应用范围已从医疗装置发展到保安控制系统中；CANopen 协议在美国不仅在嵌入式系统中得到了大量应用，而且是一些运输设备等的标准。目前国外的许多大公司开发了 CANopen 软硬件产品，如：Northhampton 公司的 CANopen 的开发工具，Downers Grove 公司的 CANopen 控制模块，Elkhart 公司的 CANopen 开发工具和软件代码。还有一些公司开发了 CANopen 协议的组态软件和配套的硬件下载工具，比如 Microcontrol 公司的μCAN.Open.er 和 Philips 公司的 CANopenIA Developer’s Kit[5]。另外一些大公司则开发出性能和兼容性良好的 CANopen 协议栈源代码，并且提供支持。由公司开发的盈利性 CANopen 源代码昂

贵，在免费 CANopen 开源代码方面，主要包括 CANopenNode、CanFestival 和 MicroCANopen[6]。

在国内，CAN 总线除了汽车外，已在工业现场控制系统、各种控制设备、交通工具、医疗仪器以及建筑、环境控制等领域得到广泛的应用，但是普遍停留在比较低层次的应用上。而且，许多 CAN 用户采用的是自己定义的应用层协议，这些设计并没有很好地体现 CAN 总线可以多主机、优先级仲裁等优点。CANopen 协议的应用产品的开发在国内尚处在初级阶段，需要进一步推广。北京中国单片机实验室从 1999 年开始对 CANopen、SAE J1939 协议进行研究，一些公司也开发出若干具有自主产权的 CANopen 产品。

1.2.2 发展前景

随着中国工业化和信息化水平的不断提高，SCADA 系统技术的应用领域也越来越广泛，随着低碳经济、绿色环保方面的需求不断增加，使得对产品性能的要求也越来越高，进而推动了各个行业的技术改造和升级。由于 RTU 具有可靠性高、通信距离远和不受现场环境限制的优点，并且作为一种嵌入式系统设备，在石油、天然气、煤层气、城市燃气、煤炭、水利、环保、供水、气象、交通等恶劣场所都可应用。CAN 总线协议从颁布至今已有几十年，仍在发展完善，应用领域不断扩大。CAN 总线的应用层协议 CANopen，在欧洲被认为是基于 CAN 的工业系统中占主导地位的标准。大多数重要的设备类型，例如数字和模拟的输入/输出模块、驱动设备等，都在 CANopen 的设备子协议中进行了描述；设备子协议中定义了不同类型的标准设备及其相应的功能。来自不同厂家的设备依靠 CANopen 协议的支持，可通过 CAN 总线来配置实现相互通信。

1.3 课题研究的意义

CAN 现场总线已被公认为几种最有前途的现场总线之一，是建立在国际标准组织的开放系统互连模型基础上，其只定义了物理层和数据链路层，在实际设计中，这两层完全由硬件实现，设计人员无需在为此开发相关软件或固件。而在基于 CAN 总线的工业应用中，开放的分布式系统标准的主要代表是 CANopen 和 DeviceNet。CANopen 的通信机制较简单，物理层全部采用 CAN 总线标准，可直接建立在 CAN 总线上，而 DeviceNet 具有严格的物理层标准。CANopen 不仅定义了应用层和通信方案，也制定了可编程系统、不同设备、接口和应用方案的结构，在保证网络节点互用性的同时，允许节点的功能随意扩展，或简单或复杂，具有很好的模块化、很高的适应性和和很好的扩展性。CANopen 协议能够在 CAN 网络中提供标准的、统一的系统通信模式、提供设备功能描述方式，并执行网络管理功能。CANopen 协议在工业自动化系统中的应用，不仅能够实现工业自动化嵌入式系统设备的功能，而且能够很好解决系统后期设备的互联，以及不同厂家设计的 CAN 节点的兼容。由于市场竞争激烈，伴随着模块化、可重构、可扩充的软硬件开放式控制系统的兴起，CANopen 协议无疑具有巨大的发展空间。CANopen 协议被认可为最具有发展前景的 CAN 应用层协议，因此对 CANopen 的研究具有较大的现实意义。

1.4 课题研究内容及篇章结构

本文采用理论与实践相结合，通过对 CANopen 协议的通信规范和设备规范的深入分析研究，以油田油井为应用背景，设计实现将 CANopen 协议应用于 RTU 中组成现场监控网络系统。本文的主要研究内容包括：

- 学习研究 CAN 总线和高层协议 CANopen，主要分析研究 CANopen 协议的通信子协议和设备子协议。
- 根据监控系统的应用要求和特点，提出 CANopen 监控网络系统方案，完成 ARM7 嵌入式系统主从 RTU 节点硬件平台的设计。
- 研究μC/OS-Ⅱ嵌入式操作系统，并移植μC/OS-Ⅱ嵌入式操作系统到各个 RTU 上。在此基础上，CANopen 协议应用于 RTU 主从节点上，完成数据通信、网络管理和其他的系统监控的功能。
- 对 CANopen 协议在主从 RTU 节点上应用的功能测试。

根据课题的研究内容，具体章节安排如下：

第一章：绪论。介绍课题研究的背景与来源、论述国内外现状与发展前景、阐述课题研究的意义，最后给出本文研究的内容以及篇章结构等。

第二章：系统硬件平台设计。介绍以油田油井为应用背景，构建的以 RTU 为组成单元的监控系统网络结构，详细描述了系统中主从 RTU 节点的具体硬件电路设计，主要包括 ARM7 主控制单元模块、ARM 系统外围电路和 CAN 通信控制模块电路。

第三章：嵌入式操作系统μC/OS-Ⅱ的移植。研究μC/OS-Ⅱ嵌入式操作系统，并移植μC/OS-Ⅱ嵌入式操作系统到各个 RTU 节点上。

第四章：CANopen 应用层协议分析。将 CAN 总线协议进行分析，重点分析本文的核心协议 CANopen 的内容。

第五章：CANopen 协议在 RTU 模块中的编程与调试。根据系统设计需要完成底层 CAN 通信和应用层 CANopen 协议的功能实现，并对系统通信进行测试。

第六章：结束语。对研究内容进行了总结，分析了系统设计的不足，并对基于 CANopen 协议 RTU 的发展进行了展望。

第二章　系统硬件平台设计

2.1 系统总体结构

随着“数字油田”这种新的石油行业信息化理念被普遍接受，油田的全面信息智能化已成必然。油田中油井大多都分布比较分散，对油井工作状态的监测和控制，主要是对油田油井生产的井口设备的工作状况、技术数据监测和数据管理[7]。所以，为实现数字化油井作业和降低油田生产成本，提出了以 RTU 设备为组成单元的一种油井监控系统，如图 2-1 所示。

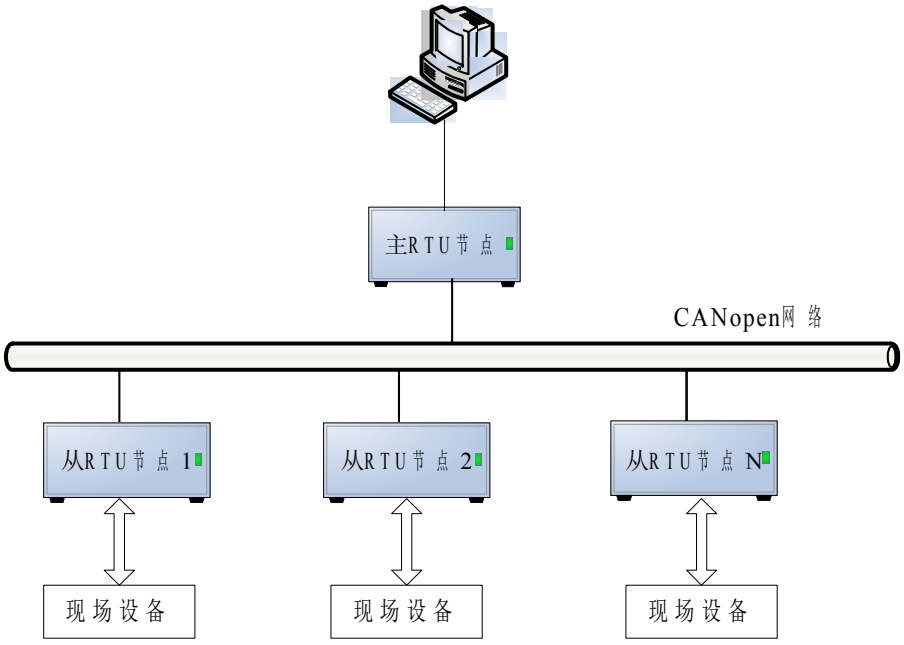


图 2-1 系统网络结构图

监控系统的主、从 RTU 节点分别为通信控制模块和数据测控模块，它们通过 CANopen 网络进行通信。通信控制模块具有网络管理、实时数据传输和错误处理功能，并通过串口与上位机 PC 进行通信。数据测控模块一方面监测现场设备的状态、传输实时数据，另一方面执行上位机发送过来的控制命令，控制现场设备。监控系统不仅能够独立完成实时数据采集和传输功能，而且能对现场设备的状态进行监控和管理。

由于该监控系统采用 CAN 总线主从方式进行工作，主从 RTU 节点需要较强的实时处理能力，负责数据监测和传输，还具有存储、转发和设置功能，同时具备灵活性、可靠性和稳定性，所以系统 RTU 模块选用内部集成 CAN 控制器的 AT91SAM7X256 嵌入式微处理器作为核心处理器。在本设计中深入研究μC/OS-Ⅱ嵌入式实时操作系统、CAN 总线和 CANopen 协议的基础上，将它们应用于 RTU 模块上，使得主从 RTU 节点间基于 CANopen 协议进行通信，达到监控现场设备的目的。

2.2 系统硬件设计

2.2.1 ARM 嵌入式系统简介

嵌入式系统（Embedded System）就是可以嵌入到其他系统中的微处理器应用系统，在组成上，嵌入式系统以微处理器及软件为核心部分。由于嵌入式系统是应用于特定环境下，针对特定用途来设计的系统，所以不同于通用计算机系统，它是针对具体应用设计的“专用系统”。其具有以下显著特点：面向特定任务，是专用的计算机系统；比通用PC 系统资源少；功耗低、体积小、集成度高、成本低；具有系统测试和可靠性评估体系，保证嵌入式系统高效、可靠、稳定地工作；具有较长的生命周期；具有固化在非易失性存储器中的代码；使用 RTOS（嵌入式实时操作系统），实时性较好；包含专用调试电路；是技术密集、资金密集、高度分散、不断创新的知识集成系统。由于嵌入式系统具有体积小、性能好、可靠性高以及面向行业应用的突出特征，其广泛应用于军事国防、消费电子、信息家电、网络通信、工业控制等领域[8]。

以 ARM 为核心的微处理器是目前嵌入式系统行业非常流行的处理器芯片，ARM 微处理器及其技术的应用已遍及安保、存储、工业控制等各类产品市场，几乎已经深入到国民生产的各个领域。目前，ARM 处理器系列有 ARM7 系列、ARM9 系列、ARM9E 系列、ARM10E 系列、ARM11 系列、SecurCore 系列、Intel 的 Xscale、Intel 的 StrongARM，设计者可根据应用需求进行选择。ARM7 系列微处理器为低功耗 32 位 RISC 处理器，最适合用于对价位和功耗要求较高的产品，其具有以下特点：

- 嵌入式的 ICE-RT 逻辑，方便调试开发；
- 功耗极低，适合对功耗要求较高的应用；
- 能够提供 0.9MIPS/MHz 的三级流水线，并基于冯·诺依曼结构；
- 代码密度高并兼容 16 位的 Thumb 指令集；
- 支持较多的操作系统，包括 Window CE、Linux、Palm OS、μC/OS、VxWorks 等；
- 具有可达 130MIPS 高速主频；
- 指令系统与多种 ARM 系列处理器兼容，便于用户对产品进行升级换代。

ARM7 系列微处理器主要应用于工业控制、Internet 设备、网络和调制解调器、移动电话等多种多媒体和嵌入式产品[9]。

2.2.2 RTU 硬件设计

由于系统分为主从 RTU 节点，因此系统硬件设计主要由主 RTU 和从 RTU 硬件设计两部分组成，主从 RTU 节点的硬件平台采用基于 ARM 的嵌入式系统设计。

从 RTU 节点与现场设备相连，通过 CAN 总线传输现场设备上的实时数据给主 RTU 节点，并执行上位机发送过来的命令控制现场设备。因此本系统中从 RTU 节点选用内部集成 CAN 控制器的 AT91SAM7X256 微处理器作为控制 MCU，TJA1050 收发器作为 CAN

收发器。此外从 RTU 节点还配备了电源模块、复位电路、EEROM 电路、JTAG 仿真器接口和 I/O 量接口。从 RTU 节点硬件结构如图 2-2 所示。

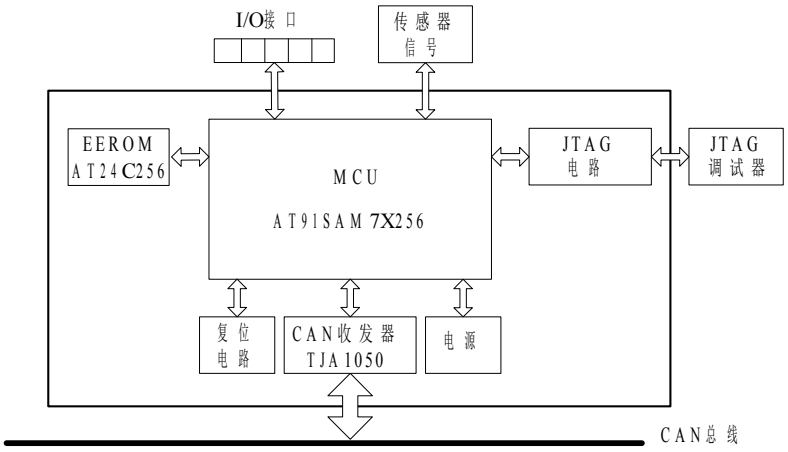


图 2-2 从 RTU 节点硬件结构图

主 RTU 节点通过串口与上位机 PC 进行通信，管理和监控网络中各个从节点状态。而且，主 RTU 节点通过 CAN 总线接收各个从节点发送过来的实时数据，并发送上位机的控制命令。主 RTU 节点需要具有较强的处理能力，因此选用内部集成 CAN 控制器的 AT91SAM7X256 微处理器作为控制 MCU，TJA1050 收发器作为 CAN 收发器。此外主 RTU 节点还配备了电源模块、复位电路、EEROM 电路、JTAG 仿真器接口以及 RS-232 串口电路。主 RTU 节点硬件结构如图 2-3 所示。

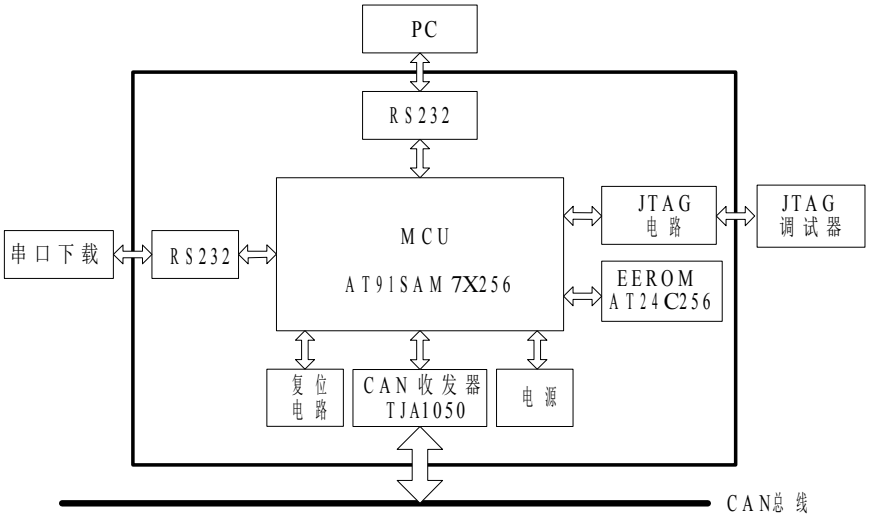


图 2-3 主 RTU 节点硬件结构图

2.2.3 AT91SAM7X256 微处理器介绍

AT91 系列微控制器是 ATMEL 公司基于 ARM7TDMI 嵌入式微处理器的 16/32 位微处理器。ARM7TDMI 处理器是 ARM7 处理器系列成员之一，是目前应用很广泛的 32 位

高性能嵌入式 RISC 处理器，不仅功耗很低，而且内部的工作寄存器很多，非常适合于实时控制应用。AT91SAM7X256 采用 ATMEL 公司的高密度 CMOS 技术，把 ARM7TDMI 和较多的 Flash 程序存储器、片内 RAM 以及各种外围功能模块集成在了一个芯片上。AT91 系列微控制器使用了基于先进微控制器总线结构 AMBA（Advanced Microcontroller Bus Architecture）的模块化设计方法，具有综合、快速和高性能价格比的特点[10]。

AT91SAM7X256 是 ATMEL 公司于 2005 年推出的基于 ARM7TDMI 处理器的工业级芯片，集成有 256K 字节的高速 Flash、64K 字节的 SRAM 和包括 ADC、SPI、SSC、TWI、USART、CAN 控制器、10/100M 以太网 MAC、定时器/计数器、RTT、ADC、USB2.0、5V 兼容的 I/O 在内的全套外围设备[11]。其高性能、低功耗及非易失性掉电检测功能非常适合应用在工业控制现场，特别是在一些要用到 Ethernet 网络、CAN 总线和 Zigbee 无线网络通讯的领域。

AT91SAM7X256 处理器结构[12]如图 2-4 所示。

2.2.4 电源管理控制器 PMC

电源管理控制器 PMC 通过控制系统和用户外设时钟来优化功耗。PMC 可以启用或禁用许多外设和 ARM 处理器的时钟输入。电源管理控制器提供以下时钟：

- 主时钟 MCK，可通过编程控制使其从几百赫兹到设备的最大操作频率不等；
- 处理器时钟 PCK，当处理器进入空闲模式时关闭，以在等待中断时降低能耗；
- 外设时钟，通过外设时钟可分别控制嵌入式外设；
- USB 时钟 UDPCK；
- 四个可编程时钟输出。

2.2.5 先进中断控制器 AIC

先进中断控制器（AIC）是一个有 8 个优先级，单独屏蔽向量中断控制器，它可处理最多 32 个中断源，将中断处理的程序及时间开销大大降低。AIC 驱动 ARM 处理器 nFIQ（快速中断请求）和 nIRQ（标准中断请求）输入，8 优先级控制器允许用户定义每个中断源中断级别（7~0）。级别 7 为最高级别，级别 0 为最低级别。如果 AIC 同时接受到多个中断，则具有最高优先级的中断首先得到服务。内部中断源为可编程的电平触发或边沿触发，外部中断为可编程的上升、下降沿触发或高、低电平触发。中断源 0 保留为快速中断输入 FIQ，中断源 1 保留为系统外设（RTT，PIT，EFC），其他的中断源控制外设或外部中断。

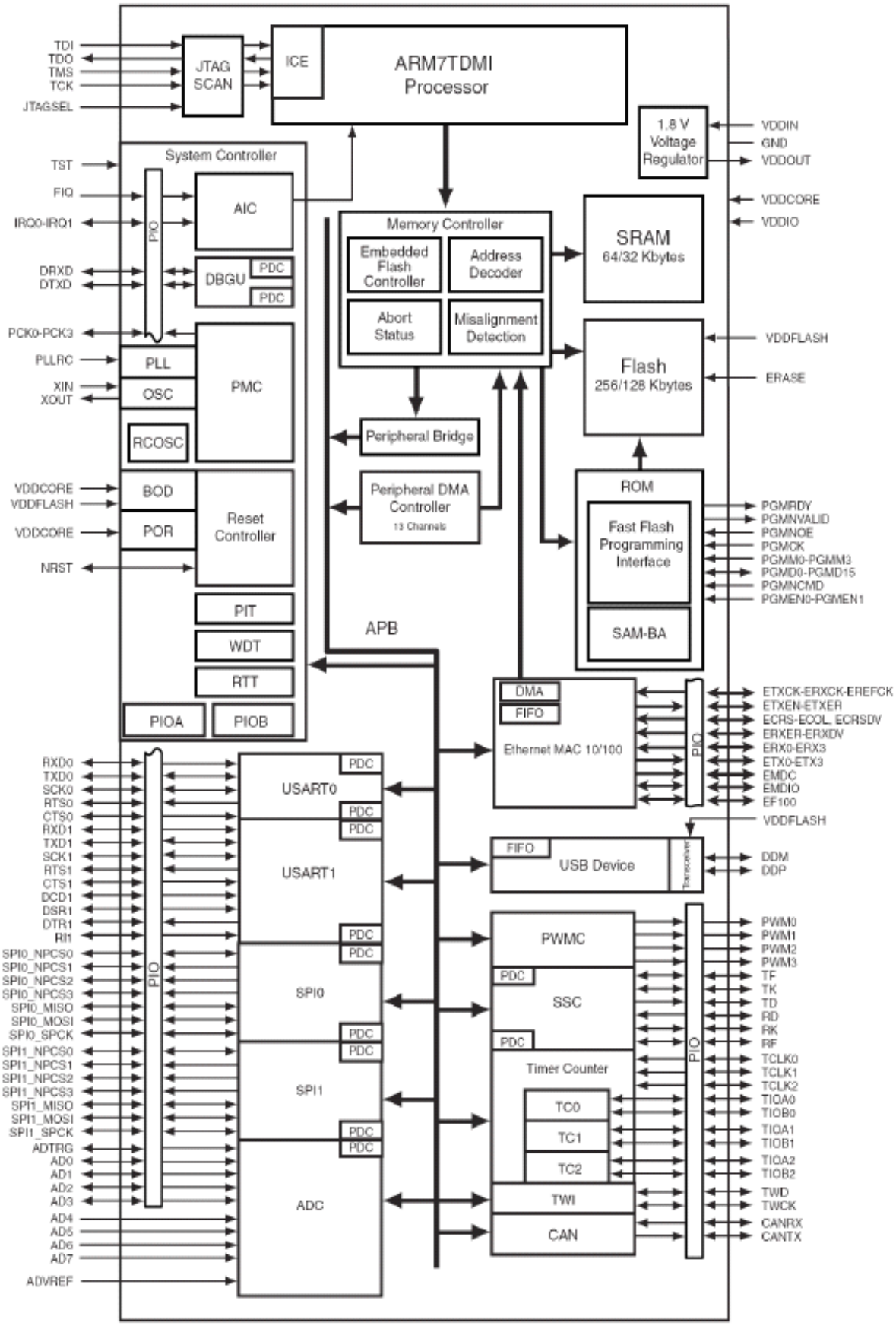


图 2-4 AT91SAM7X256 处理器结构图

2.2.6 ARM 系统外围电路设计

1. 电源电路

AT91SAM7X256 需要提供核心电压、I/O 电压、ADC 参考电压、Flash 电压和 PLL 电压等，分为 3.3V 和 1.8V 两种电压等级。其中有六种类型的供电引脚，并集成有一个电压调节器。AT91SAM7X256 电源提供片上 1.8V 稳压器，可以为内核及外部组件提供高达 100mA 的电流，3.3V VDDIO 提供 I/O 线电源，独立的 3.3V VDDFLASH 提供 flash 电源和 USB 电源，具有掉电检测的 1.8V VDDCORE 提供内核电源，3.3V VDDIN 提供电压调节器电源和 ADC 电源，因此对 AT91SAM7X256 处理器仅提供一种电压即可。由于系统供电电源为 5V 电压，所以采用线性稳压芯片 LM1117 将外部提供的 5V 电源转换成 3.3V 提供给处理器。其电源电路如图 2-5 所示。

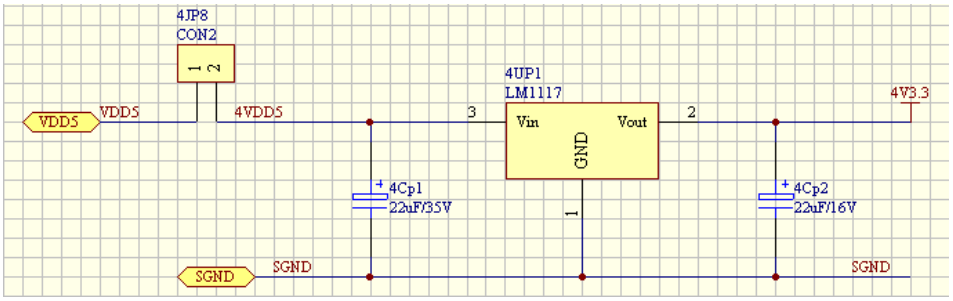


图 2-5 电源电路图

2. 时钟电路

AT91SAM7X256 内部时钟发生器包含一个低功耗 RC 振荡器、一个主振荡器和一个 PLL。它的主要特点如下：RC 振荡器的范围为 22KHz 到 42KHz 之间；主振荡器频率为 3MHz 到 20MHz 之间；主振荡器可以被旁路。本系统设计采用外部 18.432MHz 晶振作为时钟输入连接到 AT91SAM7X256 的 XOUT 和 XIN/PGMCK 管脚上，为工作提供所需的时钟信号，并且设计了滤波电路消除干扰。时钟电路如图 2-6 所示。

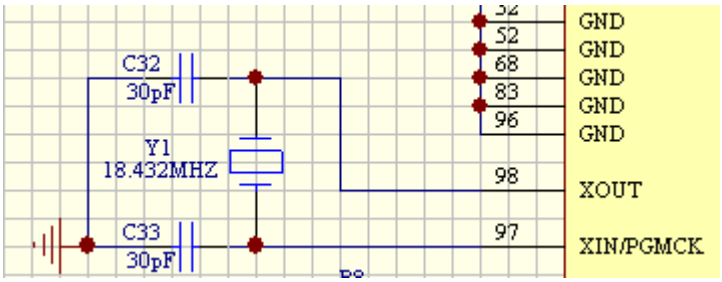


图 2-6 时钟电路图

3. 复位电路

AT91SAM7X256 的复位引脚（NRST）是一个带一个开漏输出缓存的双向引脚，其由片上复位控制器控制，可以向外部组件发送一个复位信号或者外部低电平复位微控制器。复位电路采用上电复位的形式，在系统上电时提供一个复位信号，在整个系统的初

始复位阶段，保证整个系统能够正确启动，直到系统正常工作，复位信号才撤销。如果在系统正常工作阶段出现异常，若工作电源电压低于门槛电压（此处为 3.3V），并且低电源状态持续时间超过一定时间，则对 ARM 系统进行复位。

复位电路如图 2-7 所示。电路采用 SP809EK 上电复位芯片为电路核心芯片，其复位管脚与 AT91SAM7X256 处理器的 NRST 管脚相连，可实现对系统的复位功能，提高系统可靠性。

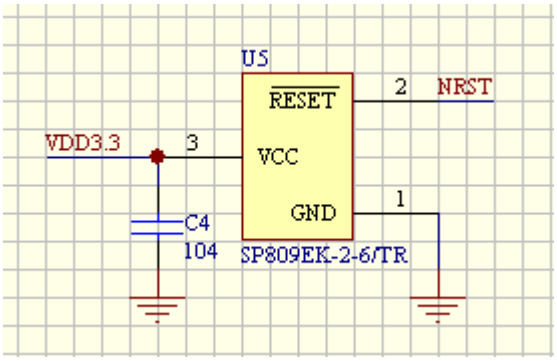


图 2-7　复位电路图

4.　EEPROM 电路

AT91SAM7X256 片内有一个高速 64K 字节 SRAM 和一个 256K 字节的 Flash，但是当停机或者断电时，其存储的数据就会丢失。因此，增加 EEROM 存储电路模块，其存储的数据在停机或断电后不会丢失，保证了系统数据的可靠性和安全性。EEPROM 电路如图 2-8 所示。电路采用 ATMEL 公司的 AT24C256，其是一种采用低功耗 CMOS 技术、兼容 1MHz I^2C 总线、存储容量为 256KB、内部可以组成 32K×8 存储单元的 EEPROM 芯片。具有硬件写保护和软件数据保护，滤波输入抑制噪声、高可靠，数据可保存 40 年的特性[13]。

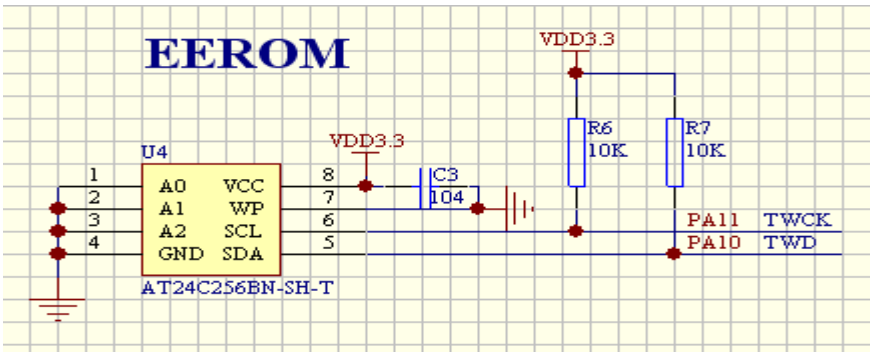


图 2-8 EEROM 电路图

5.　JTAG 调试接口电路

JTAG(Joint Test Action Group)是一种国际标准测试协议（IEEE 1149.1），主要用于芯片内部测试和系统进行仿真、调试。AT91SAM7X256 具有 JTAG 边界扫描口，JTAG 设备可通过此对系统进行调试，调试的时候不使用任何片上资源，也不需要任何存储器，也不占用目标系统的任何端口。

JTAG 的标准接口是 4 线：TMS、TCK、TDI、TDO，分别为模式选择、时钟、数据输入和数据输出线。在 JTAG 接口中，TMS、TDI 和 TCK 都是施密特触发型的输入引脚，TMS 和 TCK 与 5V 兼容。设计采用 20 针接口的 JTAG，用于 PCB 板的测试，以及系统的调试和仿真。JTAG 接口电路如图 2-9 所示。

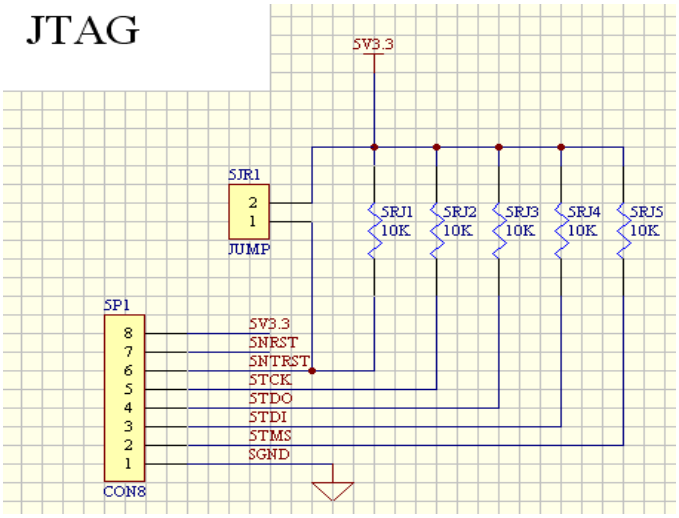


图 2-9 JTAG 调试接口电路图

6. 串口接口电路

AT91SAM7X256 集成有两个通用同步/异步收发器（USART0 和 USART1）和一个调试单元（DEBUG），它们具有可编程的波特发生器、可编程的通信协议、完善的错误检测机制、3 种回路测试方式、中断产生、多点模式的支持功能。由于系统中主 RTU 节点通过串口与上位机进行通信，因此，设计 USART0、USART1 作为 RTU 的串口 0、串口 1。标准 9 针 RS232 的电气特性在 RS232-C 中任何一条信号线的电压均为负逻辑关系，即逻辑“1”，-5~-15V；逻辑“0”，+5~+15V。而 AT91SAM7X256 系统的 TTL 的标准逻辑“1”，+2V～+3.3V ；逻辑“0”，+0V～+0.4V。设计采用 MAX3232 芯片实现 4 线串口接口电路，电路如图 2-10 所示。而且，在嵌入式系统中通过串口调试也是一种重要的调试手段。

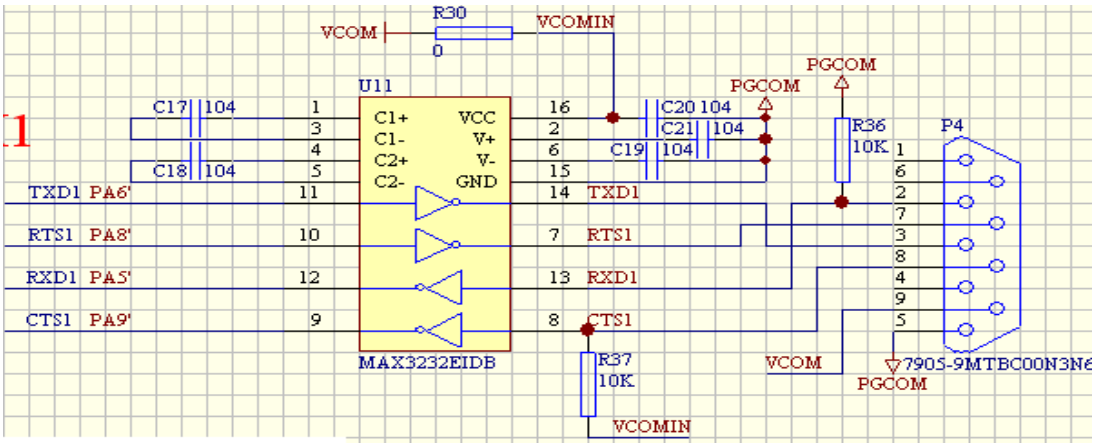


图 2-10 串口接口电路图

2.2.7 CAN 模块电路设计

1. CAN 控制器

AT91SAM7X256 内部集成的 CAN 控制器，提供实现 CAN 协议串行通信所需的功能，完全兼容 CAN 2.0A 和 2.0B。CAN 控制器能够处理所有 CAN 总线协议的帧类型（数据帧、远程帧、错误帧和过载帧），并自动处理 CAN 协议的数据链路层和部分物理层功能，位传输速率为 1Mbit/s[14]。处理器通过 CAN 控制器模块内的 8 个面向对象的邮箱实现对数据的读或写，其具有以下一些属性：都可编程为符合 CAN 2.0A 和 2.0B 标准的消息；每个邮箱都可独立的配置为接收或者发送模式；每个邮箱对应一个标识符；每个邮箱数据对象都可访问 32 位数据寄存器；在发送或接收消息时使用 16 位时间标签；用于时间标签和网络同步的 16 位时间标签；可对接收邮箱的接收缓存长度进行编程；发送邮箱的优先级管理；自动波特率和监听模式；低功耗模式，并且可编程为总线活动唤醒或应用程序唤醒，结束低功耗模式。CAN 控制器模块结构如图 2-11 所示。CAN 控制器时钟通过电源管理控制器（PMC）激活，以及通过先进中断控制器（AIC）使能 CAN 控制器的中断。其中 CAN 控制器是 AIC 的内部中断源之一，并不采用边缘触发模式。

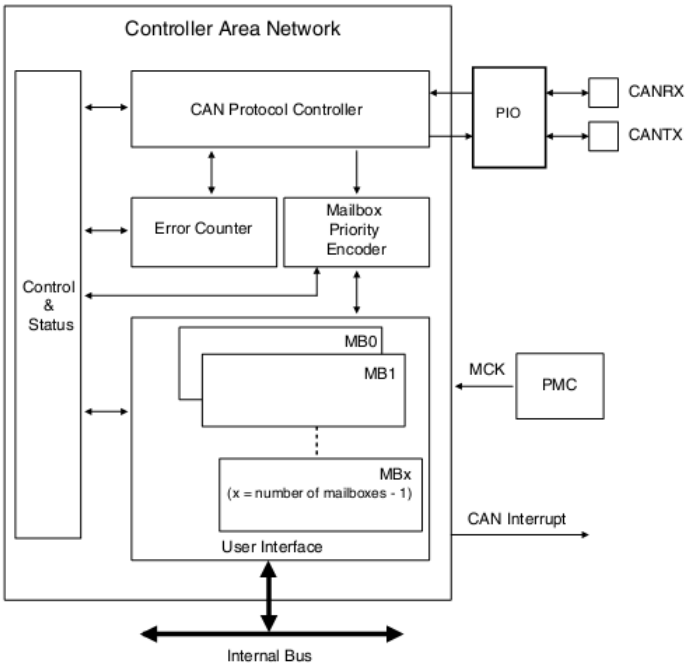


图 2-11 CAN 控制器模块结构图

2. CAN 收发器

CAN 总线的物理层被细分成 3 个子层。它们分别是：

- 物理信令：位编码定时和同步；
- 物理媒体连接：驱动器和接收器特性；
- 媒体相关接口：总线连接器。

数据链路层和物理信令层通常由 CAN 总线控制器来实现，物理传输媒体与 CAN 控

制器之间通过物理媒体连接层接口进行通信，设计采用TJA1050收发器实现物理连接层。TJA1050的主要特征为：完全符合ISO 11898标准；1Mbps的高速率；极低电磁辐射；高抗电磁干扰；节点不加电就不会扰动CAN总线；有防超时功能；保护引脚功能；兼容3.3V和5V的器件；输出驱动器受到温度保护。TJA1050有高速（正常）模式和静音（低功耗）模式两种工作模式都由引脚RS来控制[15]。CAN收发器接口电路如图2-12所示。

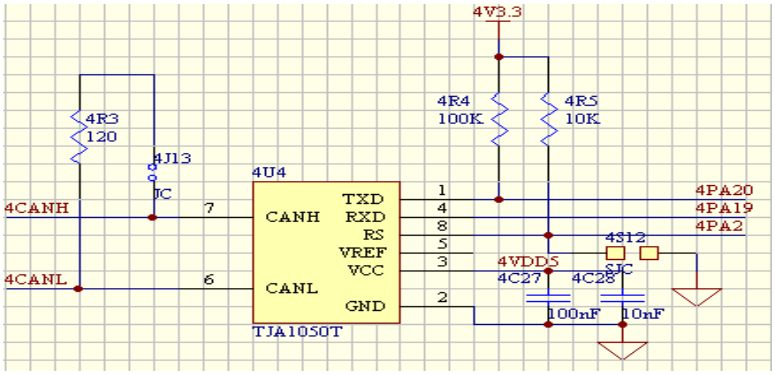


图2-12 CAN收发器接口电路图

第三章　嵌入式操作系统μC/OS-Ⅱ的移植

3.1 嵌入式操作系统概述

随着科学技术在网络通信、工业控制、芯片制造等领域的飞速发展，推动了嵌入式系统向着多功能、高速度、高密度、高可靠性、低功耗、低成本的方向不断地进步。由于应用系统的复杂程度增加，实时性要求提高，开发周期缩短等原因，嵌入式实时操作系统（Real Time Operating System，RTOS）为复杂嵌入式系统的设计提供了一种高效、快捷的解决方案，使用 RTOS 不仅增加了嵌入式应用系统的可靠性，而且使得实时应用程序的设计和扩展变得容易，无需大的改动就可以增加新的功能。

3.1.1 嵌入式操作系统的特点

嵌入式操作系统（Embedded Operating System，EOS）是一种支持嵌入式应用的操作系统软件，它是组成嵌入式系统的重要一个部分，一般包括底层驱动软件、系统内核、设备驱动接口、通信协议等内容。其具有通用操作系统的基本特点，具有负责全部软、硬件资源的分配、调度的能力，控制与协调并发事务的活动、完成任务调度、同步机制、中断处理、文件系统等功能。嵌入式操作系统与通用操作系统相比，具有的特点为：

- 可拆卸性：开放性、可伸缩性的体系结构；
- 实时操作性：EOS 实时性较强，一般可用于各种设备的控制；
- 环境适应性：嵌入式操作系统具有很好的可移植性和灵活性，能够用于不同的硬件平台和环境；
- 专用性：嵌入式操作系统一般比较小，对于不同的应用对象可进行修改，有比较明显的专用性；
- 精简性：根据嵌入式硬件平台的程序存储器、动态存储器、运算速度等硬件资源的有限性，因此嵌入式操作系统比较精炼；
- 稳定性：嵌入式系统一旦运行就不需要用户过多地干预，这就要负责系统管理的 EOS 具有很强的稳定性。嵌入式操作系统的用户接口一般不提供操作命令，它通过系统调用命令向应用程序提供服务。

常见的嵌入式系统有 Linux、μClinux、Windows CE、Palm OS、Symbian、eCos、μC/OS-II、VxWorks、pSOS、Nucleus、ThreadX、Rtems、QNX、INTEGRITY、OSE 以及 C Executive 等，其中 Linux、Windows CE、VxWorks、μClinux、μC/OS-Ⅱ、Nucleus 和 eCos 是目前流行的嵌入式操作系统。

3.1.2 μC/OS-Ⅱ操作系统

μC/OS-Ⅱ是一款源代码完全开放的实时嵌入式操作系统，而且是第一个公开内核实现机制的实时操作系统，它的全名是 Microcontroller Operating System Version 2。μC/OS-Ⅱ结构非常紧凑，使用比较简单。μC/OS-Ⅱ的特点如下[22]：

- 源代码完全开放，可被免费获得，即使商业性的应用也只收取少量的许可费用；

- 可移植性，μC/OS-Ⅱ源码绝大部分是用移植性很强的 ANSI C 写的，与微处理器硬件相关的部分是用汇编语言写的，移植过程简单；
- 可固化，μC/OS-Ⅱ可作为产品的一部分；
- 可裁剪，根据应用对象的不同，可只使用μC/OS-Ⅱ的部分服务，也可对其进行增减或修改；
- 可剥夺性，μC/OS-Ⅱ是完全可剥夺型的实时内核，即μC/OS-Ⅱ总是运行就绪条件下优先级最高的任务；
- 多任务，μC/OS-Ⅱ可管理 64 个任务，支持 56 个用户任务；
- 可确定性，绝大多数的函数调用和服务的执行时间具有可确定性；
- 任务栈，μC/OS-Ⅱ每个任务都有自己单独的栈；
- 系统服务，提供很多系统服务，包括任务管理，任务间的通信，进程调度，内存管理等；
- 中断管理，每个任务都有优先级，优先级高的任务在中断嵌套全部退出后立即执行；
- 稳定性和可靠性，μC/OS-Ⅱ是基于μC/OS 的，μC/OS 自 1992 年以来已经在数百个商业中应用，稳定性和可靠性得到了认证。

μC/OS-Ⅱ作为免费型、透明性的嵌入式操作系统具有上述诸多优点，使得它非常适合实时性强的场合，因此选用μC/OS-Ⅱ为 RTU 嵌入式系统的实时操作系统。μC/OS-Ⅱ的体系结构以及它与硬件的关系如图 3-1 所示。

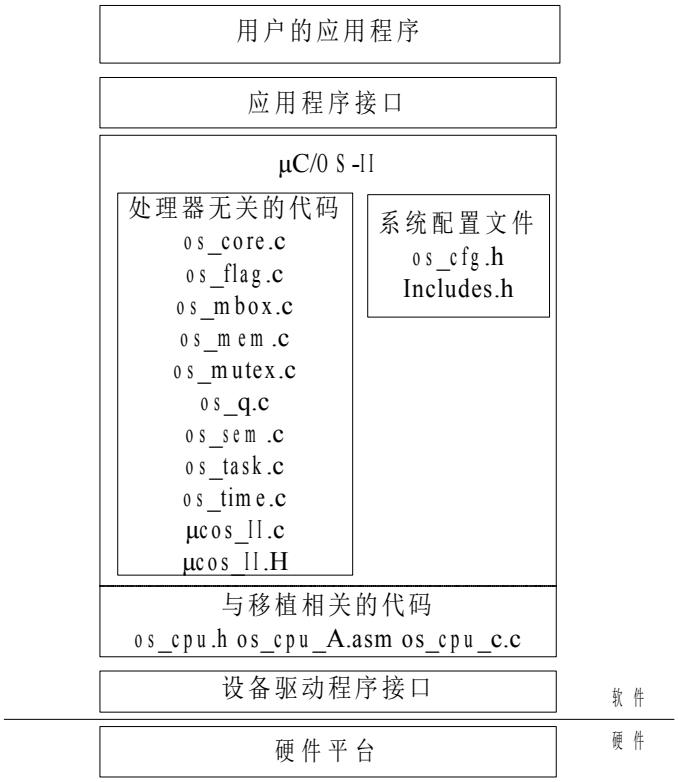


图 3-1 μC/OS-Ⅱ的体系结构

μC/OS-Ⅱ的内核主要分为 3 个部分：处理器无关的代码、处理器相关的代码和系统配置代码。处理器无关的代码提供了μC/OS-Ⅱ大部分的资源，实现了操作系统必要的功能；处理器相关的代码是μC/OS-Ⅱ操作系统移植的重点，这些代码是μC/OS-Ⅱ与硬件平台的接口；系统配置代码根据用户的应用需求，定义了一些配置μC/OS-Ⅱ的选项。

3.2 μC/OS-Ⅱ的移植

μC/OS-Ⅱ并不是能够移植到所有微处理器上，μC/OS-Ⅱ要求目标处理器必须满足一定的条件：

- 处理器的 C 编译器能产生可重入代码；
- 支持使用 C 语言来打开和禁止中断；
- 处理器支持中断，并且能产生定时中断；
- 处理器支持能够容纳一定数量的硬件堆栈；
- 处理器有将堆栈指针和其他 CPU 寄存器读出并存储到堆栈或内存中的指令。

本文选定的 AT91SAM7X256 微处理器是基于 ARM7TDMI 处理器核完全满足上述 5 个条件的要求，所以能够把μC/OS-Ⅱ嵌入式操作系统移植到 RTU 的微处理器 AT91SAM7X256 上。此外移植μC/OS-Ⅱ需要标准的 C 语言交叉编译器，同时此编译器还应支持汇编语言，ARM 公司开发推出的新一代集成开发工具 ADS 集成了 ARM 的 C 编译器 armcc、Thumb 的 C 编译器 tcc、汇编器 armasm、链接器 armlink 以及符号调试器 armsd 等应用软件的开发工具，因此本文选择 ADS 为系统的编译器。

μC/OS-Ⅱ的移植工作主要集中在多任务的实现上，涉及到 OS_CPU.H、OS_CPU_C.C 和 OS_CPU_A.ASM 这三个文件上，具体的内容包括[23]：

- OS_CPU.H 是与处理器相关的数据类型的定义，3 个宏定义（中断开关、堆栈属性和任务切换）；
- OS_CPU_C.C 主要是任务堆栈初始化和μC/OS-Ⅱ功能扩展等函数；
- OS_CPU_A.ASM 编写 4 个汇编程序，完成任务执行、任务切换、Tick 时钟、ISR 的相关处理。

3.2.1 OS_CPU.H 的编写

OS_CPU.H 文件定义了与处理器相关的一些常量、宏和数据类型。

1. 数据类型定义

针对 ARM 以及根据 ADS 编译器的特性，找到对应于μC/OS-Ⅱ的标准 C 数据类型，做出相应的修改。此外，用户必须将任务堆栈的数据类型告诉给μC/OS-Ⅱ。这个过程是通过为 OS_STK 声明正确的 C 数据类型来完成的，所有的任务堆栈都必须用 OS_STK 来声明数据类型，由于 ARM7 处理器上的堆栈是 32 位的，所以 OS_STK 声明为无符号的 32 位整型，其编写如下。

```
typedef unsigned char      BOOLEAN;
```

```
typedef unsigned char      INT8U;
typedef signed char        INT8S;
typedef unsigned short     INT16U;
typedef signed short       INT16S;
typedef unsigned int       INT32U;
typedef signed int         INT32S;
typedef float              FP32;
typedef double             FP64;
typedef unsigned int       OS_STK;
typedef unsigned int       OS_CPU_SR;
```

2. 堆栈增长方向

μC/OS-Ⅱ使用常量OS_STK_GROWTH来定义堆栈的增长方向。定义OS_STK_GROWTH为0，表示堆栈从低往高长；定义OS_STK_GROWTH为1，表示堆栈从高往低长。由于ARM的堆栈是从上往下长的，所以编写如下。

```
#define    OS_STK_GROWTH      1                  /* ARM 的堆栈从高往低长*/
```

3. 开关中断的宏定义

与所有的实时内核一样，μC/OS-Ⅱ需要先禁止中断，再访问代码的临界段，并且在访问完毕后重新允许中断。这使得μC/OS-Ⅱ能够保护临界段代码免受多任务或中断服务例程（ISRs）的破坏。μC/OS-Ⅱ定义了两个宏来禁止和允许中断：OS_ENTER_CRITICAL()和OS_EXIT_CRITICAL()，以屏蔽编译器提供的实现方法的细节，从而允许从C程序中直接方便地执行中断的控制操作。OS_ENTER_CRITICAL()和OS_EXIT_CRITICAL()可以使用3种不同的方式实现，通过在OS_CPU.H文件中定义OS_CRITICAL_METHOD常数。本文采用在C函数的局部变量中保存中断的开/关状态的第三种方式，其实现语句如下。

```
#define    OS_CRITICAL_METHOD       3
#if          OS_CRITICAL_METHOD = = 3
#define    OS_ENTER_CRITICAL()    (cpu_sr = OS_CPU_SaveSR())    /*关中断*/
#define    OS_EXIT_CRITICAL()     (OS_CPU_RestoreSR(cpu_sr))    /* 开中断*/
#endif
```

ARM处理器是通过PSR寄存器来控制中断的开、关状态，在进入中断临界段时，读取PSR寄存器的值到C函数的局部变量中的语句，退出临界段时需要从局部变量中恢复PSR寄存器的值。

4. 宏定义 OS_TASK_SW()

宏定义OS_TASK_SW()指向了μC/OS-Ⅱ在从低优先级任务切换到最高优先级任务时所调用的处理程序。任务切换也就是将当前处理器寄存器保存到将被挂起的任务的堆

栈中，并且将更高优先级的任务从堆栈中恢复出来。如果有更高优先级任务处于就绪状态时，由 OSIntExit()函数执行任务切换功能。

OS_TASK_SW()是由宏任务级切换函数 OSCtxSx()来完成。OSCtxSx()通过软中断或是陷阱（TRAP）指令产生中断，使处于就绪态的任务的堆栈结构恢复到对应的处理器寄存器中。

```
#define   OS_TASK_SW()              OSCtxSw()
```

3.2.2 OS_CPU_C.C 的编写

在 OS_CPU_C.C 文件中，在任务建立的时候 OSTaskStkInit()函数用来初始化任务的堆栈结构，它也是移植的重点。OS_CPU_C.C 文件中的函数包括：OSTaskStkInit()；OSTaskCreatHook()；OSTaskDelHook()；OSTaskSwHook()；OSTaskIdleHook()；OSTaskStatHook()；OSTaskTickHook()；OSTaskHookBegin()；OSTaskHookEnd()；OSTCBInitHook()。只有当 OS_CFG.H 文件中的 OS_CPU_HOOKS_EN 被设置为 1 时才会产生 Hook 代码。

OSTaskCreate()函数和 OSTaskCreateExt()函数通过调用 OSTaskStkInit()函数来初始化任务的堆栈结构，并返回新的堆栈指针 stk。在 ARM 中，通常 R15 用做程序计数器（PC），R14 用做链接寄存器（LR），R13 用做堆栈指针，CPSR 是程序状态寄存器，SPSR 是用来保存进入当前处理器模式前的上一个模式下的 CPSR 值。ARM 堆栈结构如图 3-2 所示。

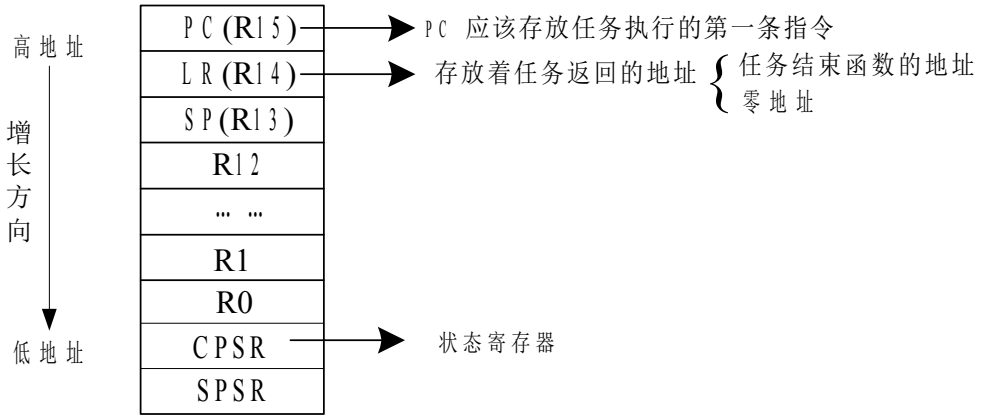


图 3-2 ARM 堆栈结构图

OSTaskStkInit()函数的修改代码如下所示：

```
OS_STK *OSTaskStkInit (void (*task)(void *pd),void *pdata,OS_STK *ptos,INT16U
opt)
{
    OS_STK *stk;
    opt         = opt;                          /* 'opt' is not used, prevent warning */
    stk         = ptos;                         /* Load stack pointer */
```

```
        *(stk)     = (OS_STK)task;                    /* Entry Point*/
        *(--stk) = (INT32U)0x14141414L;               /* LR*/
        *(--stk) = (INT32U)0x12121212L;               /* R12*/
        *(--stk) = (INT32U)0x11111111L;               /* R11*/
        *(--stk) = (INT32U)0x10101010L;               /* R10*/
        *(--stk) = (INT32U)0x09090909L;               /* R9*/
        *(--stk) = (INT32U)0x08080808L;               /* R8*/
        *(--stk) = (INT32U)0x07070707L;               /* R7 */
        *(--stk) = (INT32U)0x06060606L;               /* R6 */
        *(--stk) = (INT32U)0x05050505L;               /* R5 */
        *(--stk) = (INT32U)0x04040404L;               /* R4 */
        *(--stk) = (INT32U)0x03030303L;               /* R3*/
        *(--stk) = (INT32U)0x02020202L;               /* R2*/
        *(--stk) = (INT32U)0x01010101L;               /* R1 */
        *(--stk) = (INT32U)pdata;                     /* R0 : argument */
        *(--stk) = (INT32U)ARM_SVC_MODE;      /* CPSR    (Enable both IRQ and FIQ
interrupts) */
        *(--stk) = (INT32U)ARM_SVC_MODE;              /* SPSR*/
      return (stk);
    }
```

3.2.3 OS_CPU_A.ASM 的编写

OS_CPU_A.ASM 文件的汇编程序是μC/OS-Ⅱ移植过程的重点，这里需要编写 4 个汇编语言函数：OSStartHighRdy()、OSCtxSw()、OSIntCtxSw()、OSTickISR()。

1. OSStartHighRdy()函数

OSStartHighRdy()函数被操作系统启动函数 OSStart()调用，来启动在 OSstart()函数运行之前已经处于就绪状态的、具有最高优先级的任务。OSStartHighRdy()函数的主要工作是从任务堆栈中恢复出最高优先级的任务对应的所有处理器寄存器的值，并且返回中断。μC/OS-Ⅱ使用变量 OSTCBHighRdy，其指向最高优先级任务的任务控制块（TCB），而堆栈地址在 TCB 的起始位置，所以 OSStartHighRdy()函数可以通过这个变量获取最高优先级任务的堆栈地址。此外，OSStartHighRdy()函数还需要调用 OSTaskSwHook()函数，并且调用之后，在优先级最高的任务启动之前，将变量 OSRunning 设为 True，标示μC/OS-Ⅱ在运行。函数基本运行流程如图 3-3 所示。

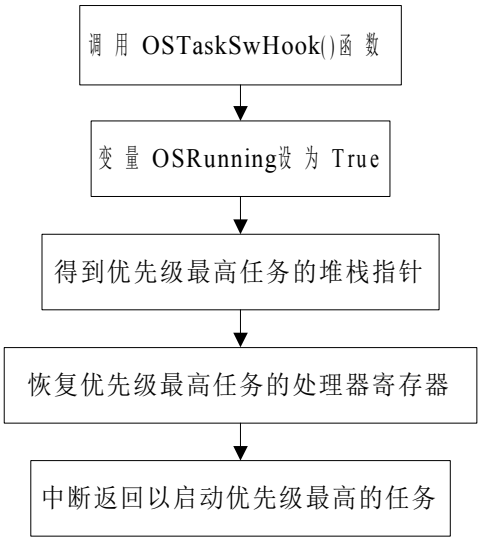


图 3-3 OSStartHighRdy()函数基本运行流程图

2. OSCtxSw()函数

OSCtxSw()是一个任务级的任务切换函数，在任务需要进行切换时被 OS_CPU.H 文件中的宏定义 OS_TASK_SW()调用，此时的任务切换是在非异常模式下进行的，它的工作是先将当前任务的 CPU 现场保存到该任务堆栈中，然后获得最高优先级任务的堆栈指针，从该堆栈中恢复此任务的 CPU 现场，使之继续。函数基本运行流程如图 3-4 所示。

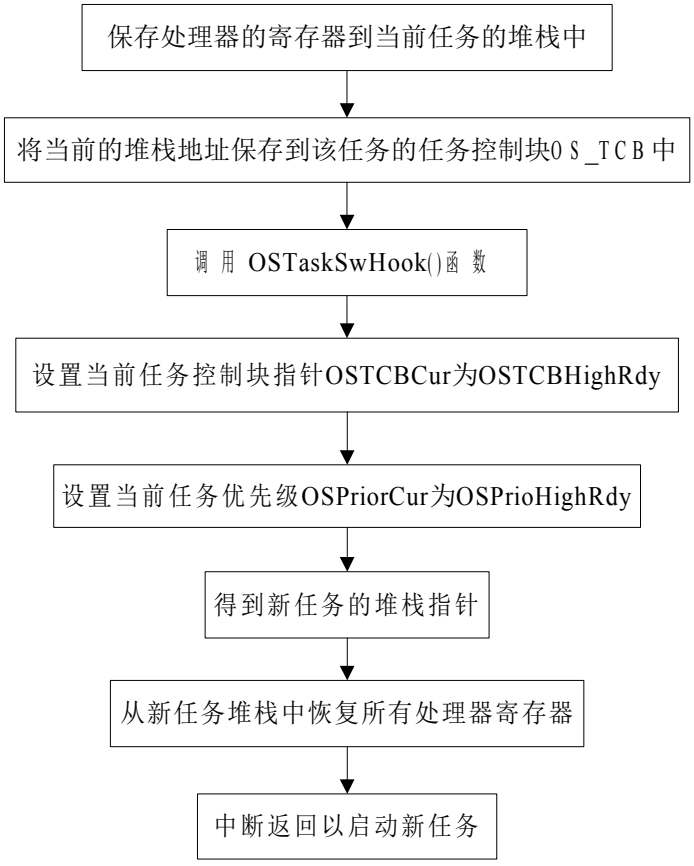


图 3-4 OSCtxSw()函数基本运行流程图

3. OSIntCtxSw()函数

OSIntCtxSw()是一个中断级的任务切换函数负责在中断服务程序退出时完成必要的任务切换。OSIntCtxSw()函数的原理基本与任务级的切换函数 OSCtxSw()相同，但是由于进入中断时已经保存过了被中断任务的 CPU 现场，因此不必再进行类似的操作，只需要调整相应的堆栈指针，对由于函数的嵌套所引起的多余压栈信息进行清理。在调用 OSIntCtxSw()函数的时候，堆栈的情况如图 3-5 所示。

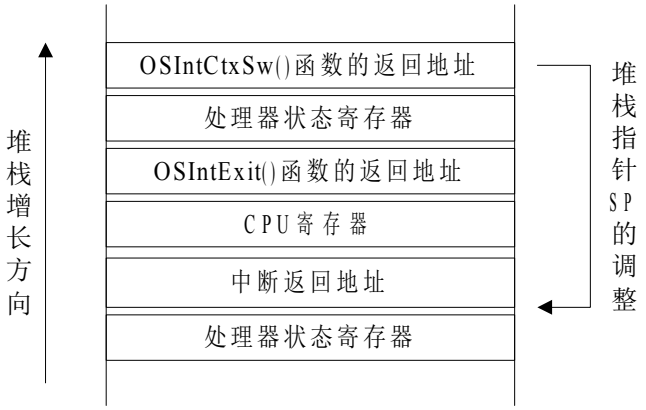


图 3-5 调用 OSIntCtxSw()函数的堆栈情况图

OSIntCtxSw()函数的基本运行流程如图 3-6 所示。

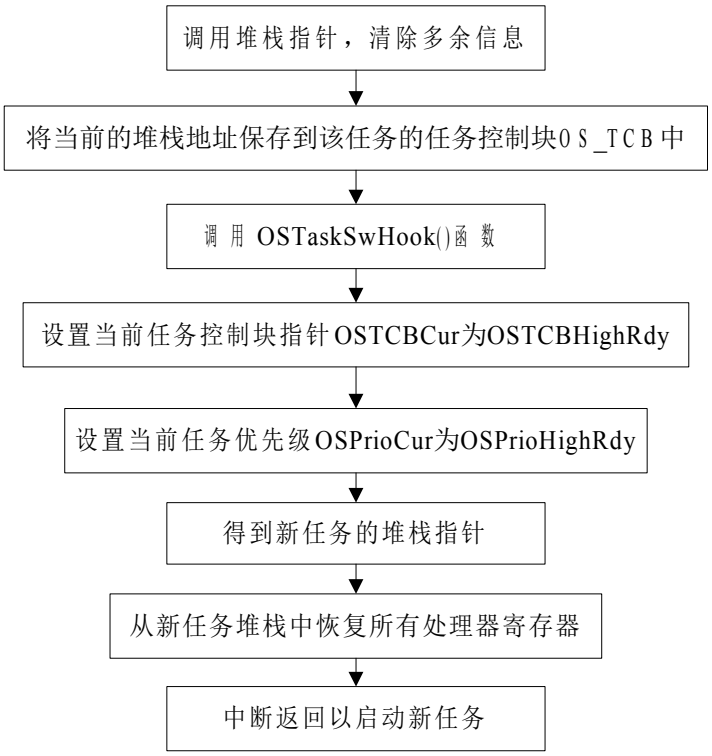


图 3-6 OSIntCtxSw()函数基本运行流程图

4. OSTickISR()函数

μC/OS-Ⅱ为了实现系统的延时和运行控制任务使用时钟节拍（Tick），一般由处理器内部的硬件时钟实现。时钟节拍中断处理函数 OSTickISR()，不仅负责处理系统的时

钟中断，而且能够调用 OSTimeTick 函数，完成任务的运行控制，同时为完成必要的任务切换调用 OSIntExit()函数。OSTickISR()函数的基本运行流程如图 3-7 所示。

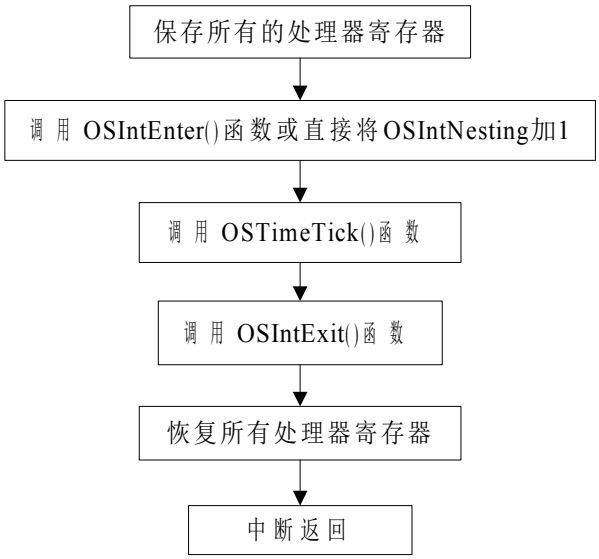


图 3-7 OSTickISR()函数基本流程图

通过对上述文件的修改，并经过编译器的编译、测试成功，μC/OS-Ⅱ嵌入式操作系统能够在AT91SAM7X256微处理器上正常运行，完成了μC/OS-Ⅱ嵌入式操作系统在RTU模块上的移植工作。

第四章　CANopen 应用层协议分析

4.1 CANopen 的层次结构

所有标准的工业通信系统均必须符合国际标准化组织（ISO）所定义的 OSI 开放协议标准。CANopen 以涵盖物理层和数据链路层功能的串行总线系统 CAN 为基础，所有 CANopen 功能均被映射到一个或多个 CAN 报文。在 OSI 模型中，CAN 标准、CANopen 协议之间的关系，即 CANopen 的层次结构，如图 4-1 所示。

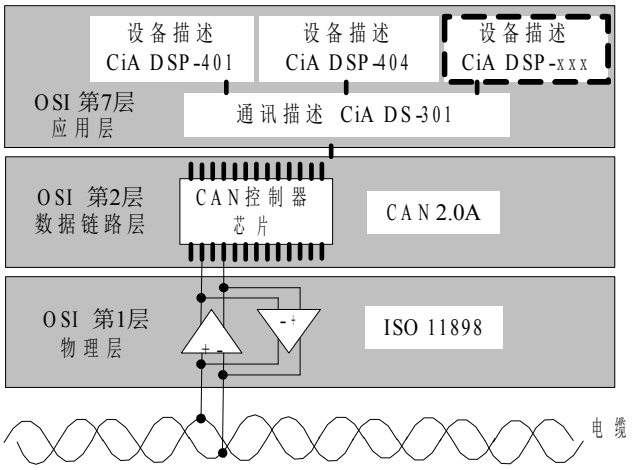


图 4-1 CANopen 的层次结构图

CAN 控制器采用的 CAN 协议符合 ISO11898-1 规范标准。它包含的子层有 LLC（逻辑链路控制）、MAC（介质访问控制）以及 PLS（物理信号）。预定义 CANopen 消息使用的是标准的报文格式。物理层包含的子层有 MAU（介质访问单元）和 PMA（物理层链接），两者均符合高速传输标准 ISO11898-2 规范的要求，相应的驱动模块也必须符合该规范的要求。在 CANopen 中仅需要一部分网络层、传输层、会话层或表示层的功能，CANopen 应用层对此进行了具体描述，而且 CANopen 的通讯描述提供了配置设备、通信数据的含义，定义数据通信方式；设备描述定义了不同类型的标准设备及其相应的功能。

4.2 CAN 总线协议

4.2.1 CAN 总线概述

CAN 总线具有极高的可靠性、数据传输速率高、传输距离长、实时性强，特别适合工业现场监控设备的互联。其特点可概括如下[1]：

- 采用通信数据块编码，可实现多主工作方式，数据收发灵活；
- 采用基于优先级非破坏性的总线仲裁技术；
- 信号传输用短帧结构（8 个字节），传输时间短，受干扰率低；
- CAN 的每帧信息都有 CRC 校验及其他检错措施，具有很好的检错效果；

- 在总线不关闭的情况下可任意连接或拆除节点，具有灵活性和可扩展性；
- CAN 上的节点分成不同的优先级，可满足不同的实时要求，优先级高的数据最多可在 134μs 内得到传输；
- 通信介质选择灵活。

为使设计透明和执行灵活，遵循 ISO/OSI 标准模型，CAN 分为数据链路层和物理层。CAN 的分层结构如图 4-2 所示。

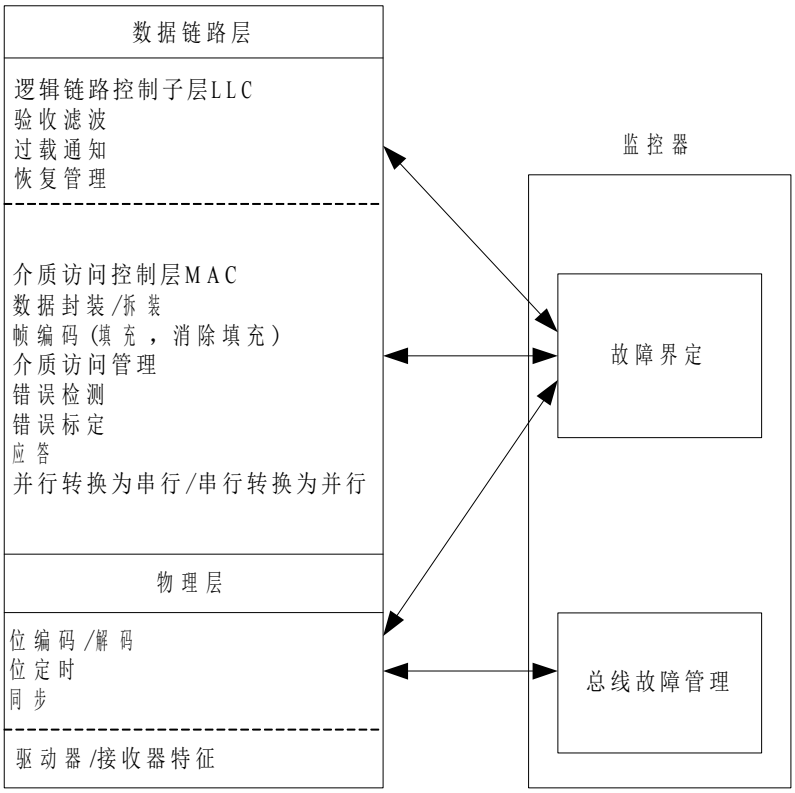


图 4-2 CAN 的分层结构图

物理层分为物理信令（PLS）、物理媒介附件（PMA）与媒体接口（MDI）三部分，完成电气连接、实现驱动器/接收器特性、定时、同步、位编码解码。数据链路层的 MAC 子层是 CAN 协议的核心，它描述由 LLC 子层接收到得报文和对 LLC 子层发送的认可报文。MAC 子层可响应报文帧、仲裁、应答、错误检测和标定。LLC 子层的主要功能是报文滤波、超载通知和恢复管理。

CAN 网络采用多主的 CSMA/CA 模式，其总线电平分为两种：显性电平和隐性电平，显性电平覆盖隐性电平。当来自同一 CAN 网络的各个不同总线设备同时发送显性电平和隐性电平时，总线上只会出现显性电平；仅当网络上的所有总线设备同时发送隐性电平时，总线上才会出现隐性电平。CAN 规范中定义，隐性电平的逻辑值为 1，显性电平的逻辑值为 0。

4.2.2 CAN 报文格式

CAN 总线支持两种报文格式：含有 11 位标识符的帧为标准帧；含有 29 位标识符的

帧称为扩展帧。CAN 总线报文有以下 4 种不同的帧类型：

- 数据帧：数据帧携带数据从发送器至接收器。
- 远程帧：由总线单元发出，请求发送具有同一标识符的数据帧。
- 错误帧：任何单元检测出总线错误即发送此类帧。
- 超载帧：用于提供当前和后续的数据帧（或远程帧）的附加延迟。

数据帧和远程帧借助帧空间与当前帧分开，帧间空间（ITM）由 3 个隐性电平的位组成。CANopen 协议使用的是 CAN 总线的标准帧格式，并不支持扩展帧格式。

1. 数据帧

数据帧由帧起始（SOF）、仲裁域、控制域、数据域、CRC、应答域（ACK）和帧结尾（EOF）7 个不同的域（Bit Field）组成，标准格式的数据帧结构如图 4-3 所示。

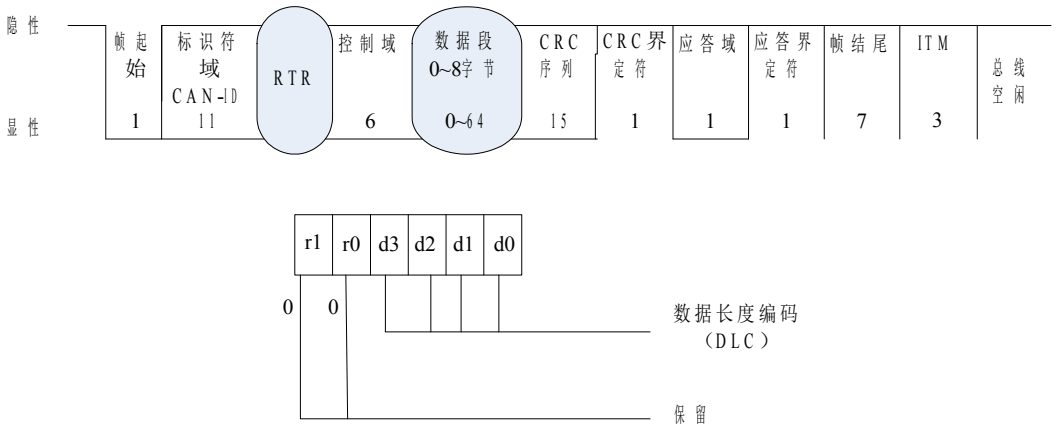


图 4-3 CAN 标准格式数据帧结构

帧起始（SOF）用来指示一个数据帧或远程帧的开始。仲裁域由标识符域和远程发送请求位组成（RTR）：标识符由 11 位组成，用于表示确定的待传输消息，其数值也作为待传输消息的优先级；RTR 用来区分数据帧和远程帧，数据帧中 RTR 为显性。控制域中保留位必须为显性，DLC（数据长度代码）表示此帧在数据段中的传输字节数。数据段表示数据帧中的发送数据，可以为 0~8 个字节。CRC 由 15 位 CRC 序列和 1 位 CRC 界定符组成。应答域由 1 个应答间隙位和 1 个应答界定符位组成，发送器在应答间隙位传输一个隐性电平，在正确接收了完整的消息之后，接收器便发送一个显性电平以进行确认，这种应答机制只可用来检测网络的消息响应故障。帧结尾由 7 个隐性的位组成，表示一帧的结束。

2. 远程帧

远程帧可用来请求有效数据。当一个网络设备的接收器收到一个远程帧时，该设备的发送器就会发送一个用于应答的数据帧。接收到的远程帧与发送出的数据帧应当使用同一个 CAN-ID 标识符。远程帧与数据帧的格式相同，但远程帧的 RTR 位永远被置为隐性电平，且远程帧没有数据域，远程帧的仲裁优先级低于数据帧。

3. 错误帧

错误帧由网络中各个参与设备所叠加的错误标志和一个错误界定符组成。错误标志用来发出一组违反总线协议的故障信号，有两种形式的错误标志：6 个连续显性位组成的主动错误标志；6 个连续隐性位组成的被动错误标志。错误标志符由 8 个隐性位组成，用于结束错误帧。

4. 超载帧

超载帧通常由尚未处理完上一帧消息的 CAN 控制器发出，可以用于延迟网络中其他设备发送下一条帧消息，它由网络中各个参与设备所叠加的超载标志和一个超载界定符组成。6 个显性位组成超载标志，且必须在帧间空间 ITM 的前两个位之内开始。8 个隐性位组成超载界定符，用于结束超载帧。超载帧不会影响错误计数器的读数，一个数据帧或远程帧之后最多可以跟随两个超载帧。

4.3 CAL 协议

CANopen 是一个基于 CAL 的子协议，使用了 CAL 通信和服务协议子集，CAL（Can Application Layer）为基于 CAN 的分布式系统的运作提供了应用独立、面向对象的环境，且提供了通信的对象和服务、标识符的分配、网络的管理。

CAL 提供了以下 4 种应用层服务功能：

1. **CMS（CAN-based Message Specificaiton）**[16]

CMS 提供基于变量、时间、域类型的对象，以设计和规定一个设备（节点）的功能如何被访问，提供开放的、面向对象的用户应用环境。

2. **NMT（Network ManagemenT）**[17]

提供采用主/从通信方式（只有一个 NMT 主机）来实现网络管理服务。

3. **DBT（DistriBuTor）**[18]

提供采用主/从通信方式来实现的动态分配 CAN-ID 服务，CAN-ID 更名为 COB-ID（Communication Object Identifier）通信对象标识符。

4. **LMT（Layer ManagemenT）**[19]

提供 LMT 主节点可以设置 LMT 从节点的某层参数的服务，如改变一个节点的 NMT 地址或改变 CAN 接口的位定时和波特率。

CAL 提供了所有的网络管理服务和报文传送协议，但并没有定义 CMS 对象的内容或者正在通信的对象类型。而 CANopen 协议不仅定义了应用层和通信子协议，而且为各种不同类型的设备定义了大量的行规，描述了 CMS 对象的内容和正在通信的对象类型。

4.4 CANopen 应用层协议

4.4.1 CANopen 协议结构模型

CANopen 协议中包含了标准应用层规范和通信规范。在 CANopen 的应用层，设备间通过相互交换通信对象进行通信，良好的分层和面向对象的设计思想带给用户一个清

晰的通信模型。通信模型如图 4-4 所示。

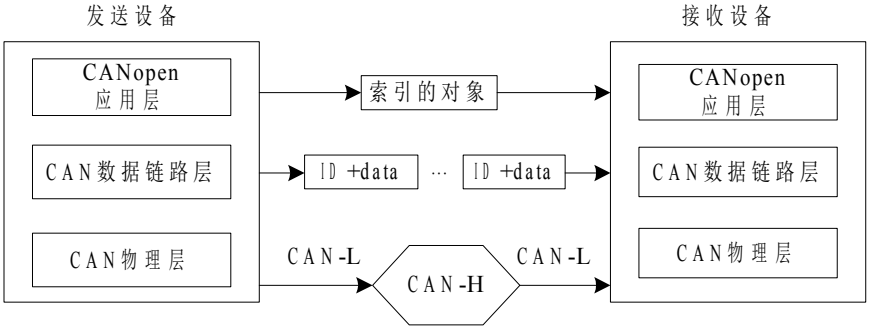


图 4-4 CANopen 协议通信模型

由于 CAN 只对物理层和数据链路层进行了定义，因此为了能让设备之间通过 CAN 网络进行通信，用户还需要进行一些与应用相关的定义。首先，将网络中可用的 CAN 标识符分配给每个设备，这才能知道消息的优先级高低，设备之间是否具有优先顺序，或 CAN 标识符中是否包含预设功能。其次，为了不让系统中出现功能不同但 CAN 报文相同的情况，用户还要做出一些相关的定义。如：CAN 标识符为 0x181 的 CAN 报文表示传输为模拟量。

除上述定义外，传输的数据内容也要定义，主要包括数据内容的传输格式以及数据读取规则。另外还有一个必不可少的条件，就是总线上的设备具有单个设备的控制机制和错误处理能力。以上这些定义全部包含在应用层，并且需要保证设备能够通过 CAN 总线提供统一的接口。

CANopen 是由一系列称为子协议的文档组成：通信子协议和各个设备子协议。以上所述功能在 CANopen 的通信子协议和设备子协议中都有规定和描述。CANopen 规定了应用层和通信子集（CiA DS-301），适用所有的 CANopen 设备，既支持对设备参数的直接存取，又支持对时间苛求的过程数据通信[20]；而针对各种不同类型的工业专用设备描述，都在相应的设备子协议（CiA DSP-4XX）里有具体的规定，可根据应用需要选择和查看相应的设备子协议。

一个 CANopen 设备模型分为通信部分、应用部分和对象字典 3 个部分，如图 4-5 所示。

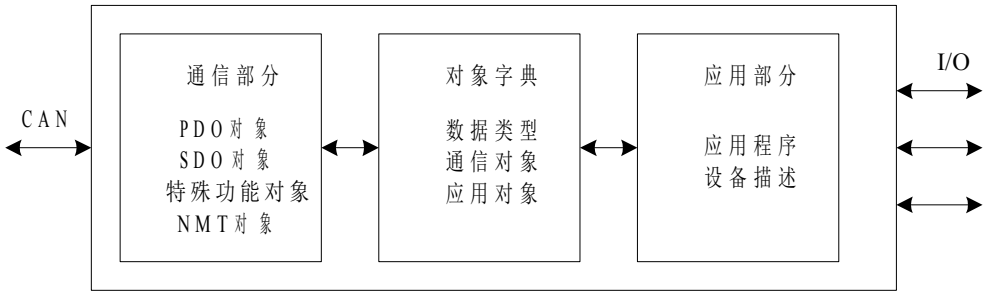


图 4-5 CANopen 设备模型

通信部分由 CAN 收发器、CAN 控制器以及 CANopen 协议栈组成。协议栈中包括实

现通信的通信对象（如过程数据对象（PDO）和服务数据对象（SDO））和状态机。通信部分提供数据传输所需的所有机制和通信对象，符合 CANopen 规范的数据可以利用这些机制通过 CAN 接口进行传输。

在 CANopen 设备的应用部分中，对设备的基本功能进行定义或描述，由用户根据应用要求编写和实现。如：在 I/O 设备中，可以访问设备的数字或模拟输入输出接口。

对象字典是应用部分和通信部分之间的接口，实际上是设备的所有参数列表，应用部分和通信部分都可访问这个参数列表，是一个 CANopen 设备的核心部分。对象字典中的词目（对象或参数）通过一个 16 位索引和一个 8 位子索引进行识别或定位，用户可对词目进行读或写。如：为通信对象配置不同的 CAN 标识符。

4.4.2 通信对象

CANopen 网络的通信和管理都是通过不同的通信对象来完成的，CANopen 应用层详细描述了各种不同类型的通信对象（COB），这些通信对象都是由一个或多个 CAN 报文来实现的。通信对象分为以下 4 种类型：

- 过程数据对象（PDO 消息），用来传输实时数据；
- 服务数据对象（SDO 服务器消息和 SDO 客户端消息），用来读/写其他 CANopen 设备的对象字典；
- 网络管理对象（NMT），用来控制 NMT 状态机（NMT 消息）和监测设备（心跳、启动报文）。
- 特殊功能对象，包括同步对象（SYNC）、时间标记对象（TIME）和紧急状态对象（EMCY）。

1. 过程数据对象 PDO（Process Data Object）

在 CANopen 中，过程数据被分为几个单独的段，每个段最多为 8 个字节，这些段就是过程数据对象（PDO），由一个 CAN 报文帧组成，即被映射到单一的 CAN 帧中，PDO 的优先级由对应的 CAN-ID 标识符决定，优先级较高。PDO 的通信方式可由生产者/消费者模式来描述，每个 PDO 有一个唯一的标识符，过程数据对象可从一个设备发送另一个设备或许多其他的设备同时接收，所以 PDO 是无确认模式的传输。作为广播对象它的上层没有附加协议，执行 PDO 的传输没有协议的开销。对 PDO 有两种类型的使用，数据发送的 PDO（TPDO）和数据接收的 PDO（RPDO），由生产者发送的 PDO 为 TPDO，由消费者接收的 PDO 为 RPDO，相应的对象字典条目的索引通过以下公式计算：

RPDO 通信参数索引=1400h+RPDO_编号-1

TPDO 通信参数索引=1800h+TPDO_编号-1

RPDO 映射参数索引=1600h+RPDO_编号-1

TPDO 映射参数索引=1A00h+TPDO_编号-1

PDO 通信是基于生产者/消费者关系模式，即推/拉模式包括一个生产者和 0 个或多

个消费者。PDO 写请求服务（Push 模式）不需要确认，PDO 读请求服务（Pull 模式）需要确认，如图 4-6 所示。

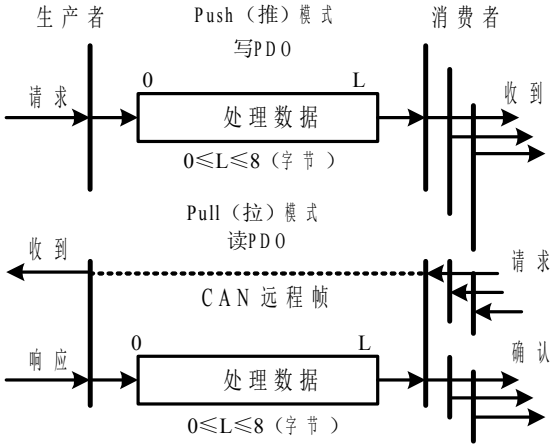


图 4-6 PDO 生产者/消费者模式

PDO 传输模式有两种为同步传输和异步传输：非周期同步由远程帧预触发传送，或通过设备子协议中规定的对象事件预触发传送，周期同步传送在每 1 到 240 个 SYNC 消息后触发；异步的触发传送方式同上。同步报文传输，可实现在网络范围内协调数据的获取和发送，异步报文传输，可以在任何时间传输。

有 3 种不同的报文触发 TPDO 传输以及接收 RPDO 方式：

- 事件触发：报文的传输通过一个对象特定事件的出现触发。对于同步的 PDO，通过接受 SYNC 对象同步终止一个规定的传输周期。为了非循环同步 PDO 和异步 PDO，报文传输的触发是在设备文档中指定的一个设备_特殊事件。
- 时间触发：报文传输既可由设备特定事件的发生触发，也可由事件没有在一个规定时间内发生触发。
- 远程请求触发：一个异步 PDO 发送是通过接收到来自其他设备（PDO 消费者）的远程请求。

表 4-1 给出了由传输类型定义的不同 PDO 传输模式，8 位无符号整数定义了 PDO 的传输类型。

表 4-1 PDO 传输类型与触发方式对应表

传输类型	触发 PDO 条件（B=both need　O=one or both）			PDO 传输
	SYNC	RTR	EVENT	
0	B	-	B	同步，非循环
1~240	O	-	-	同步，循环
241~251	-	-	-	保留
252	B	B	-	同步，在 RTR 之后

续表 4-1 PDO 传输类型与触发方式对应表

传输类型	触发 PDO 条件（B=both need　O=one or both）			PDO 传输
	SYNC	RTR	EVENT	
253	-	O	-	异步，在 RTR 之后
254	-	O	O	异步，厂商特定事件
255	-	O	O	异步，设备子协议特定事件
说明： SYNC ——接收到 SYNC-object。 RTR ——接收到远程帧。 Event ——事件类型，如数值改变或者定时器中断。 传输类型为 1 到 240 时，该数字代表两个 PDO 之间的 SYNC 对象的数目。				

2. 服务数据对象 SDO（Service Data Object）

服务数据对象可传输大于 8 字节的配置信息，即 SDO 传送协议允许传送任意长度的对象。传输 SDO 用于对对象字典的读/写访问，它的优先级较低，但可以实现可靠的数据传输。发送和接收者间将建立点对点的通信，接收者将确认收到的每个段信息，通信模式为客户机/服务器模式。SDO 通过传输对象字典的索引和子索引，可以定位相应的对象字典入口，以实现对节点参数的设置、下载程序、定义 PDO 通信类型和数据的格式等功能。它由两个 CAN 对象在两个网络节点之间通过点对点的通信来实现。通过 SDO 传送的报文可以不受长度的限制，但传送 SDO 报文需要额外的协议开销。

SDO 的基本结构如表 4-2 所示。第一段内的第 1 字节 SDO 命令字包含下载/上传、请求/应答、分段/加速传送、CAN 帧数据字节长度、用于后续每个分段的交替清零和置位的触发位；第 2~4 字节包含要读或写的对象字典的索引和子索引；最后 4 字节是要传输的数据。有同样 CAN 标识符的第二段以及其后继段包含控制字节和多达 7 字节的数据。

表 4-2 SDO 基本结构

字节 0	字节 1~2	字节 3	字节 4~7
SDO 命令字	对象索引	对象子索引	数据

SDO 通信方式是基于客户端/服务器模式，这是单一客户端和单一服务器之间的关系。客户端发出一个请求（上传/下载），这样触发服务器去执行某个任务，在结束任务后服务器回应这个请求，如图 4-7 所示。

服务器 SDO（SSDO）和客户端 SDO（CSDO）的性能在 SDO 通信参数中描述，相应的对象字典条目的索引计算通过以下公式计算：

SSDO 通信参数索引=1200h+SSDO_编号-1

CSDO 通信参数索引=1280h+CSDO_编号-1

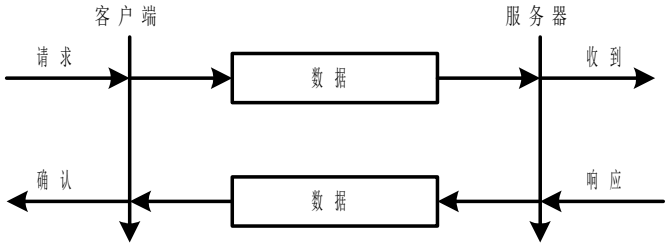


图 4-7 客户端/服务器模式

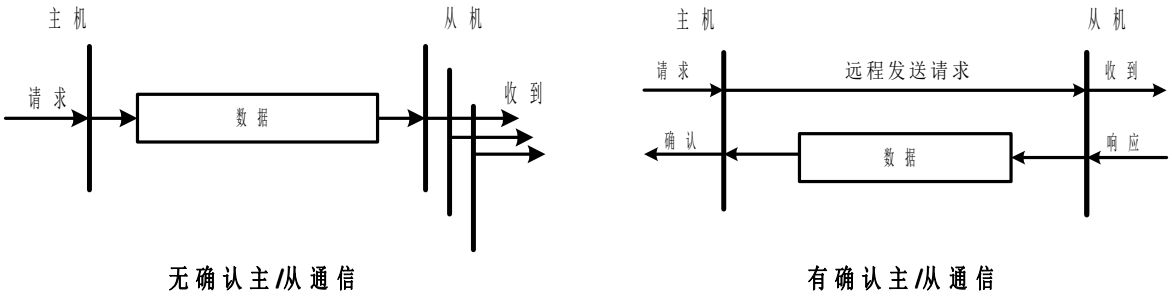
用服务数据对象（SDO）来访问对象字典，因为这些条目可以包含有任意大小的数据和数据类型，SDO 可传输多个数据集从客户端到服务器，反之同样。客户可通过多索引（对象字典的索引和子索引）控制服务器对象字典中的哪些数据集被传输。

SDO 有 3 种传输类型：加速 SDO 传输、分段传输、块传输。加速 SDO 传输最多传输数据长度为 4 字节数据，其适用于对象字典中的大多数对象，因为这些对象的数据长度都在 4 字节以内，所以整个加速 SDO 传输只需要交换 2 条 CAN 报文即可。分段传输长度超过 4 字节的数据，传输的数据会被分成好几个小于 7 字节的数据段，对发送的每一段都要进行一次确认，因此对于较长的数据对象来说效率不是很高，不仅会占用许多网络资源，也会浪费许多时间。分块传输数据超过 4 字节的数据，将数据划分成几个单一的包，在连续的请求或应答中逐块传输这些包，相当于分段传输中的多个数据段传输只需 1 个确认报文应答，因此增加了总线的吞吐量，传输效率更高、速度更快、数据量更大。

SDO 和 PDO 是 CANopen 的基本传输机制。PDO 对于小型数据进行高速传输；通过 SDO 服务对对象字典进行访问，主要用在设备配置过程中传递参数以及大块数据。

3. 网络管理对象 NMT（Network Management Object）

网络通信对象 NMT，包括节点监护（Node Guarding）和 NMT 对象，网络管理命令 NMT 控制从站状态。CANopen 网络管理是基于节点的并采用主/从结构，主/从结构分为无确认通信和有确认通信，如图 4-8 所示。



无确认主/从通信 **有确认主/从通信**

图 4-8 主/从结构通信模式

在网络中，有一个设备履行 NMT 主机的功能，其他节点均是 NMT 从机，一个 NMT

从机用其唯一节点 ID 被识别。NMT 对象被用作执行 NMT 服务，通过 NMT 服务，节点被初始化、被启动和被监视、被复位或被停止。网络管理服务包括模块控制服务和错误控制服务。

- 模块控制服务

通过模块控制服务，NMT 主机控制 NMT 从机的状态。模块控制服务可由某个节点或所有节点同时执行。模块控制服务除了启动远程节点外，还可被本地服务程序启动。服务内容包括启动远程节点（Start Remote Node）、停止远程节点（Stop Remote Node）、进入预运行（Enter Pre-Operational）、复位节点（Reset Node）和复位通信（Reset Communication）。模块控制服务采用无确认主/从通信模式。

NMT 对象映射到单一的带有 2 字节数据长度的 CAN 帧，标识符为 0。第一字节包含命令说明符，第二字节包含必须执行此命令的设备的节点标识符（当节点标识符为 0 时，所有的节点必须执行此命令）。NMT 消息格式如表 4-3 所示。

表 4-3 NMT 消息格式

COB-ID	Byte0	Byte1
0x000	CS（命令字）	Node-ID

CS 命令字取值如表 4-4 所示。

表 4-4 CS 命令字取值表

命令字	NMT 服务
0x01	Start_Remot_Node
0x02	Stop_Remote_Node
0x80	Enter_Pre-operational_State
0x81	Reset_Node
0x82	Reset_Communication

- 错误控制服务

错误控制（监控设备）服务，用于检测网络中的设备是否在线和设备所处的状态。CANopen 网络管理系统提供用于设备监控的功能：生命迹象消息（心跳报文），是一种周期性地发送给一个或多个设备的消息，设备之间可以相互监督；NMT 从机监控（节点保护），NMT 主机通过远程帧周期性地监控从机的状态；NMT 主机监控（寿命保护），通过收到的用于监视从机的远程帧来间接监控 NMT 主机的状态。

节点/寿命保护（Node/Life Guarding）采用有确认的主/从通信模式。在监控过程中，NMT 主机根据其设置的监视时间，通过远程帧周期性地查询所有 NMT 从机，从机则会用包含当前设备状态的数据帧来应答主机。如果 NMT 主机在节点生命期内没有接到从节点的响应，或响应反应的状态和预期的不符，主节点会产生一个节点保护事件，从节点会产生一个寿命保护事件。

心跳报文（Heartbeat）机制，采用生产者/消费者的推模式不需要远程帧，这里的“心跳”指的是生产者消费者之间的一种通信。心跳生产者根据对象字典中的“生产者心跳时间间隔”参数中所设置的周期来发送心跳报文，一个或多个的心跳消费者接收这个报文。在心跳消费者期间内，心跳消费者监护心跳的接收，在此期间如果没有接收到心跳，就会产生一个心跳事件（Error）。

用户只能采用节点/寿命保护或心跳协议中的一种方法来进行设备监控。心跳协议能实现更加灵活的监控结构，而且不需要使用远程帧，因此一般采用心跳协议。

4. 特殊功能对象

CANopen 还定义了 3 个特定用于同步、紧急状态表示和时间标记传送的特殊功能对象：同步对象、紧急状态对象和时间标记对象。

同步对象（SYNC Object）由同步生产者向网络进行周期性的广播，该对象将提供基本的网络时钟。同步对象用于同步设备周期性地对所有应用设备广播 SYNC 对象，需要进行同步操作的设备可使用 SYNC 对象来同步本地时钟和同步设备的时钟。同步报文之间的时间由通信循环对象定义，可通过配置工具在引导过程周期间写入应用设备。SYNC 具有很高的优先级，SYNC 传输按照生产者/消费者的推模式。同步对象被映射到一个单一的带有标识符为 128 的帧，用缺省配置，同步对象不带任何数据，但可以具有多达 8 字节的用户专用数据。

时间标记对象（Time Stamp Object）将为应用设备提供公共的时间帧参考，包含一个时间和日期的值。时间标记对象的传输按照生产者/消费者的推模式，映射到 CAN 帧标识符为 256 和 6 字节长度的数据字段。

当设备发生严重的内部错误时，相关的一个紧急状态客户机将发送一个紧急状态对象（Emergency Object）。它是由设备内部出现致命错误来触发的，并从相关应用设备上的紧急客户以最高优先级发往其他设备，使得它很适合作为设备内部错误中断类型的报警信号。每个“错误时间”（Error Event）只能发送一次紧急状态对象，只要设备上不发生新的错误，就不得再发送紧急状态对象。紧急错误对象传输按照生产者/消费者模式，零个或多个紧急状态对象消费者可接收。CANopen 定义了紧急状态对象中要传送的若干个紧急错误代码，它是单一的具有 8 个数据字节的 CAN 帧，数据字节格式如表 4-5 所示。

表 4-5 紧急状态对象数据格式

字节	0	1	2	3	4	5	6	7
内容	紧急错误代码		错误寄存器 （对象 100h）	制造商特定错误域				

4.4.3 对象字典

对象字典（OD）是一个有序的对象组，是 CANopen 设备中的核心部分，是通信部分与应用部分之间的接口。CANopen 网络中每个节点都有一个对象字典。在对象字典中，

CANopen 设备的所有对象都是以标准化方式进行描述的。对象字典包括各种数据类型、通信协议和设备描述等内容，利用对象来描述 CANopen 设备的全部功能。CANopen 协议已经将对象字典进行了分配，用户可以通过一个 16 位索引和一个 8 位子索引获得所有设备中的通信对象，以及用于某种设备类别的对象。

CANopen 的设备、接口和应用规范（CiA DS4XX）定义了标准化的应用对象和基本功能。CANopen 网络管理服务简化了项目设计、系统集成和诊断。在每个分散的控制应用中都有各自所需的不同的通信对象。在 CANopen 中，所有这些通信对象都是标准化的，并在对象字典中进行了详尽的描述。

CANopen 对象字典结构如表 4-6 所示。

表 4-6 CANopen 对象字典结构

索引（十六进制）	对象
0000	未使用
0020 - 001F	静态数据类型（标准数据类型，如 Boolean，Integer16）
0020 - 003F	复杂数据类型 （预定义由简单类型组合成的结构如 PDO CommPar, SDO Parameter）
0040 - 005F	制造商规定的复杂数据类型
0060 - 007F	设备子协议规定的静态数据类型
0080 - 009F	设备子协议规定的复杂数据类型
00A0 - 0FFF	保留
1000 - 1FFF	通信子协议区域 （如设备类型，错误寄存器，支持的 PDO 数量）
2000 - 5FFF	制造商特定子协议区域
6000 - 9FFF	标准的设备子协议区域
A000 - BFFF	标准化接口规范区域
C000 - FFFF	保留

一个完整的对象字典结构包括 6 列，它们对于所有设备是共同的，如表 4-7 所示。

表 4-7 对象字典结构

Idex 索引	Object 对象名	Name 名称	Attrib 属性	Type 类型	M/O 强制/可选

索引列是表示在对象字典中对象的位置，作用就是一种地址，联系着想要访问的数据域，子索引用作反映在一个复杂对象中的数据域，如一个数组或记录，子索引在此无规定。对象列包含的是对象的名称（NULL，DOMAIN，DEFTYPE, DEFSTRUCT，VAR，ARRY，RECORD），用作表示哪一类对象在对象字典中某个特殊的索引位置。名称列提

供对哪个特定对象功能的简单文本描述。类型列给出这个对象类型的相关信息，包括以下预定义的类型：静态数据类型、复杂数据类型、PDO 通信参数和映射参数和可能的其他诸如制造商或设备的特征等。属性列定义了对对象字典内每个对象的访问权限的访问权限，包括读/写（rw），只读（ro）、只写（wo）、常量（const，只读）。M/O 列定义对象是强制型还是可选型，强制型对象必须是在设备上实现，可选型对象不需要在设备上实现。

4.4.4 网络初始化和系统启动

1. 初始化过程

网络初始化过程总体流程如图 4-9 所示，它由 NMT 主机应用程序或配置应用程序控制。

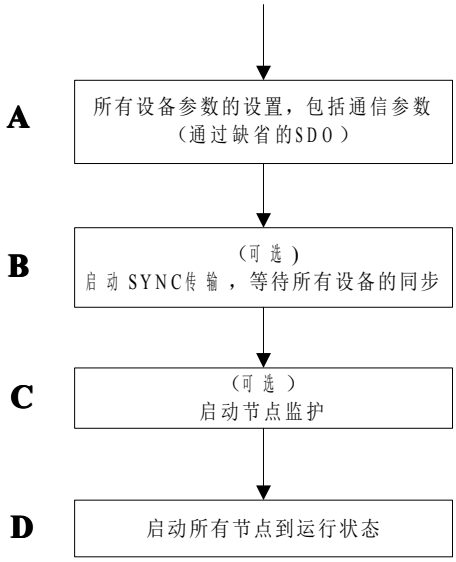
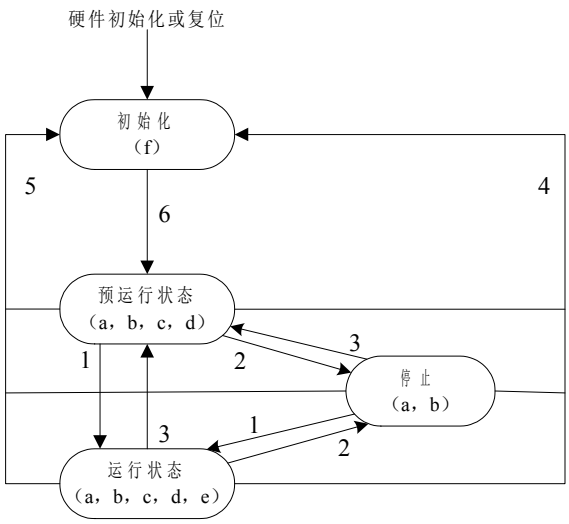


图 4-9 网络初始化整体流程图

2. CANopen boot-up 过程

所有 CANopen 设备都内置一个内部状态机，方便设备管理。内部状态机中，状态之间的转变通常由内部事件来触发（如设备启动、内部功能错误或内部复位），或由 NMT 主机在外部触发。NMT 状态机以及状态转换如图 4-10 所示，而且 NMT 状态转换图也表示设备的初始化过程。

NMT 从机任何时候都可接收 NMT 主机发送的模块控制报文，实现 NMT 主机控制 NMT 从机的状态转换。NMT 服务的 CAN 报文由 CAN 头（COB-ID=0）和两个字节数据组成；第一个字节表示请求的服务类型（NMT command specifier），第二个字节是节点 ID，或者 0（以广播方式将指令发送给所有设备）。设备在结束初始化后，直接进入预运行状态，在这个阶段可通过 SDO 进行参数配置和 ID 分配，然后设备就可直接进入运行状态。



注：
括号内的字母表示处于不同状态下，哪些对象可以使用：
a.NMT；b.Node Guarding；c.SDO；d.Emergency；
e.PDO；f.boot-up。
状态转换的触发点（1~5由NMT模块控制服务发送）：
1.启动远程节点指令（Start_Remote_node）；
2.停止远程节点指令（Stop_Remote_node）；
3.进入预操作状态指令（Enter_Pre-Operational_State）；
4.复位节点指令（Reset_Node）；
5.复位通信指令（Reset_Communication）；
6.初始化结束，自动进入预运行状态，发送 Boot-up状态。

图 4-10 NMT 状态机以及状态转换图

初始化状态分为以下三个子状态，目的是能够完全或部分地节点复位。初始化：这是在上电复位或硬件复位后设备进入的第一个子状态，在结束基本节点初始化后，设备自动进入复位应用程序状态。复位应用程序：在这个状态，制造商特定的描述和标准设备描述参数被设置为上电值，在上电设置后，自动进入复位通信状态。复位通信：在这个状态，通信描述的参数被设为上电值，之后初始化结束，通知设备执行一个写 boot-up 对象服务并进入预运行状态。

预运行状态可通过 SDO 通信，PDO 不存在，所以不允许 PDO 通信。PDO 的配置、设备参数和应用对象的分配（PDO 映射）可通过配置应用程序执行。

运行状态下，所有通信对象都可以起作用，转换到运行状态就创建全部的 PDO。

停止状态迫使全部停止通信除了节点监控和心跳监控外。

3. 预定义连接集

由于 CAN 是一种面向网络的通信对象（COB），在网络中每个 COB 有一个或多个关联的标识符，标识符表示节点设备的优先级，因此对 COB 的标识符的地址分配是系统设计中的一个主要方面。

为了简化配置工作，CANopen 网络定义了“预定义连接集”，即强制性的缺省标识符地址分配表。利用 CAN 控制器报文的标识符段，定义出 CANopen 的通信对象标识 COB-ID（Communication Object Idetifier），如图 4-11 所示。这种默认的 CAN 标识符分配方案，对于最多具有 127 个 CANopen 设备的简单网络而言，在不需要配置 COB-ID 参数的情况下，CANopen 设备就能进入正常工作状态。

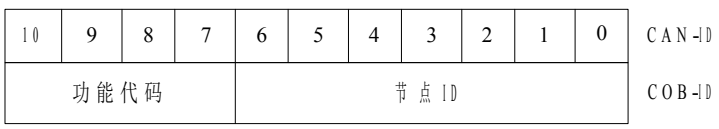


图 4-11 预定义连接集 COB-ID

预定义连接集的 11 位 CAN 标识符包含一个 4 位功能代码和一个 7 位节点 ID（设备

标识符）。功能代码用来设定服务类型，包括 4 个接收 PDO，4 个发送 PDO，1 个 SDO，1 个紧急对象和 1 个节点错误控制 ID，也支持不需确认的 NMT 模块控制服务、SYNC 和时间戳（Time Stamp）对象的广播；节点 ID 将消息明确地与设备节点对应起来，在 CANopen 网络中，所有的 CANopen 节点设备都必须配有一个明确的节点 ID。CANopen 预定义连接集 COB-ID 分配表如表 4-8 所示。

表 4-8 CANopen 预定义连接集 COB-ID 分配表

通信对象	功能代码	COB-ID	相应的对象字典
广播或组播对象			
NMT 模块控制	0000B	000H	—
SYNC	0001B	080H	1005H，1006H，1007H
Time Stamp	0010B	100H	1012H，1013H
点对点对象			
紧急	0001B	081H~0FFH	1024H，1015H
TPDO1	0011B	181H~1FFH	1800H
RPDO1	0100B	201H~27FH	1400H
TPDO2	0101B	281H~2FFH	1801H
RPDO2	0110B	301H~37FH	1401H
TPDO3	0111B	381H~3FFH	1802H
RPDO3	1000B	401H~47FH	1402H
TPDO4	1001B	481H~4FFH	1803H
RPDO4	1010B	501H~57FH	1403H
SDO（发送/服务器）	1011B	581H~5FFH	1200H
SDO（接收/客户）	1100B	601H~67FH	1200H
NMT 错误控制	1110B	701H~77FH	1016H，1017H

PDO/SDO 发送/接收是由（slave）CAN 节点观察的，NMT 错误控制包括节点保护（Node Guarding）、心跳报文（Heartbeat）和 Boot-up。

4.5 CANopen 网络的优势

由上述介绍可知对于使用 CANopen 协议工作的网络系统，存在 3 种工作模式，即主/从模式、客户端/服务器模式，以及生产者/消费者模式。每种工作模式都有各自的优缺点，在 CANopen 网络中使用了各种传输模式的优点来进行数据通信，从而达到网络系统的最优通信[21]。

主/从模式在 CANopen 网络中适用于网络管理（NMT），主机对从机设备进行管理。在同一个网络中只能有一个有效工作的 CANopen 主机，该主机拥有对所有从机的控制权。

客户端/服务器模式是一种可靠的数据通信模式，在传输数据时需要建立连接，并对数据传输进行确认应答。这种模式主要用于 CANopen 网络中设备参数配置，缺点是传输效率比较低。

生产者/消费者模式主要是针对实时数据的传输，在 CANopen 网络中 PDO 数据和紧急报文的传输采用这种方式。数据传输不需要接收确认，这样可以保证数据实时高效地进行传输，并且对 PDO 感兴趣的设备都可以进行接收，从而提高数据传输。

第五章　CANopen 协议在 RTU 模块中的编程与调试

本系统主从节点间的通信是通过 CAN 总线来完成的，CANopen 协议是 CAN 的应用层协议，CANopen 的实现是在 CAN 通信的基础上，因此首先需要完成底层 CAN 通信功能，其次再完成 CANopen 协议的功能。

5.1 CAN 通信程序设计

5.1.1 CAN 控制器初始化

CAN 总线协议的实现，包括各种帧的组织和发送，都集成在了 CAN 总线控制器的内部电路中实现。AT91SAM7X256 内部 CAN 控制器的接收数据引脚（CANRX）和发送数据引脚（CANTX）对应处理器芯片 I/O 引脚为 PA19 和 PA20，这两个引脚是 PIO 控制器和片内外围复用的。同时 CAN 接收器 TJA1050 的高速模式和静音模式通过其引脚 S 控制，引脚 RS 对应处理器芯片 I/O 复用引脚 PA2，因此初始化需要对 I/O 口和 TJA1050 进行设置。使得收发器 TJA1050 处于高速模式（正常工作模式），实现对 CAN 数据的接收和发送功能。

CAN 控制器由一系列寄存器组成，用户通过这些寄存器来实现 CAN 总线通信的功能。CAN 控制器的寄存器结构与功能如表 5-1 所示。

表 5-1 CAN 控制器的寄存器结构与功能

寄存器	名称	访问方式
模式寄存器	CAN_MR	Read-write
中断使能寄存器	CAN_IER	Write-only
中断禁用寄存器	CAN_IDR	Write-only
中断屏蔽寄存器	CAN_IMR	Read-only
状态寄存器	CAN_SR	Read-only
波特率寄存器	CAN_BR	Read-write
时钟寄存器	CAN_TIM	Read-only
时间戳寄存器	CAN_TIMESTP	Read-only
错误计数寄存器	CAN_ECR	Read-only
传送计数寄存器	CAN_TCR	Write-only
终止命令寄存器	CAN_ACR	Write-only
邮箱 0 模式寄存器	CAN_MMR0	Read-write
邮箱 0 接收屏蔽寄存器	CAN_MAM0	Read-write
邮箱 0 ID 寄存器	CAN_MID0	Read-write
邮箱 0 簇 ID 寄存器	CAN_MFID0	Read-only
邮箱 0 状态寄存器	CAN_MSR0	Read-only
邮箱 0 低位数据寄存器	CAN_MDL0	Read-write

续表 5-1 CAN 控制器的寄存器结构与功能

寄存器	名称	访问方式
邮箱 0 高位数据寄存器	CAN_MDH0	Read-write
邮箱 0 控制寄存器	CAN_MCR0	Write-only
邮箱 1 模式寄存器	CAN_MMR1	Read-write
邮箱 1 接收屏蔽寄存器	CAN_MAM1	Read-write
邮箱 1 ID 寄存器	CAN_MID1	Read-write
邮箱 1 簇 ID 寄存器	CAN_MFID1	Read-only
邮箱 1 状态寄存器	CAN_MSR1	Read-only
邮箱 1 低位数据寄存器	CAN_MDL1	Read-write
邮箱 1 高位数据寄存器	CAN_MDH1	Read-write
邮箱 1 控制寄存器	CAN_MCR1	Write-only

系统上电后，CAN 控制器是禁用的，控制器时钟必须通过电源管理控制器（PMC）激活，以及通过中断控制器（AIC）使能 CAN 控制器的中断。CAN 控制器由网络参数来初始化，在 CAN_BR 寄存器定义波特率，其决定了位时间周期的采样点，CAN_BR 寄存器必须在 CAN 控制器使能之前设置。CAN 控制器的使能通过 CAN_MR 寄存器中的 CANEN 标志位进行设置。只要 CAN 控制器为同步方式，CAN_SR 的 WAKUP 标志位自动设置为 1，退出低功耗模式。CAN 控制器在自动波特率模式下开始监听网络，这种情况下，错误寄存器被锁，邮箱可被配置成接收模式。通过扫描错误标志位，CAN_BR 寄存器的值与网络同步。一旦没有发现错误，自动波特率模式被禁止，同时 CAN_BR 寄存器的 ABM 域被清除。CAN 控制器初始化流程如图 5-1 所示。

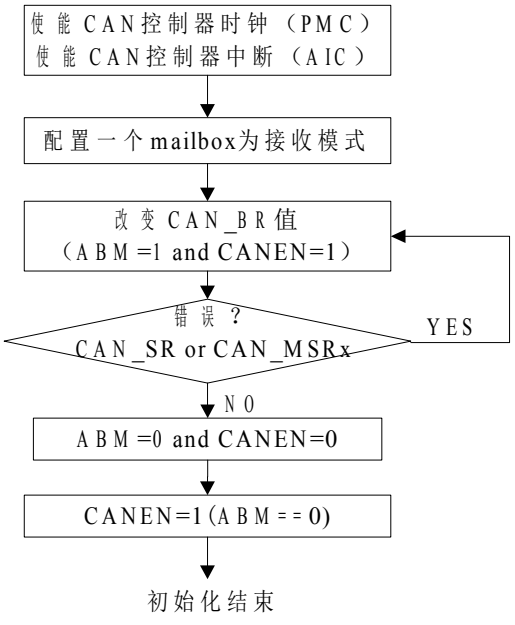


图 5-1 CAN 控制器初始化流程图

CAN 总线上的所有控制器必须具有相同的比特率和位长，在每个控制器的时钟频率的不同条件下，比特率根据控制器内的时间段来调整。CAN 控制器位长度由 BRP、PRORAG、PHASE1 和 PHASE2 四个参数确定。

$t_{BIT} = t_{CSC} + t_{PRS} + t_{PHS1} + t_{PHS2}$

各时间段由以下公式计算：

$t_{CSC} = (BRP+1)/MCK$

$t_{PRS} = t_{CSC} \times (PROPAG + 1)$

$t_{PHS1} = t_{CSC} \times (PHASE1 + 1)$

$t_{PHS2} = t_{CSC} \times (PHASE2 + 1)$

再同步跳转宽度（SJW）计算公式：

$t_{SJW} = t_{CSC} \times (SJW+1)$

本系统中MCK系统时钟为48MHz，取CAN总线波特率为1Mbit/s，因此位时间为1μs，同时规定位时间为8个TQ，驱动总线延时时间与电路接收延迟时间的总和为200ns，不计总线延迟时间。根据上述公式计算：

$t_{PRS} = 2 \times (200+0) = 400ns = 4\ t_{CSC}$ => PROPAG = 3；

$t_{SYNC} = 1t_{CSC}$ => $8t_{CSC} = 1t_{CSC} + 4t_{CSC} + t_{PHS1} + t_{PHS2}$；

$t_{PHS1} + t_{PHS2} = 3t_{CSC}$ => Phase Segment 2 = Max(IPT=2TQ,Phase Segment 1) = 2TQ

=> PHASE2 = 2-1 = 1 => PHASE1 = 0；

t_{SJW}= Min(4 TQ, Phase Segment 1) = 1TQ => SJW = 1-1 = 0。

因此配置CAN_BR寄存器的值为0x00050301，即BRP=0x05；PROPAG=0x03；PHASE2=0x01；PHASE1=0；SJW=0。

5.1.2 CAN 报文发送和接收

AT91SAM7X256 微处理器的 CAN 控制器有 8 个邮箱，通过设置 CAN_MMRx 寄存器的 MOT 位可将每个邮箱都可配置成发送或接收邮箱。CAN 控制器通过发送或接收邮箱来对 CAN 报文进行发送或接收，而且根据需要可通过设置 MOT 位把发送或接收邮箱设置成生产者或消费者模式。生产者模式邮箱收到一个远程帧后会自动发送报文，消费者模式邮箱在收到发送请求后自动发送报文。

设置 MOT 位把邮箱配置为发送邮箱用来发送报文，根据需要配置每个邮箱的优先级（CAN_MMRx 寄存器的 PRIOR 位）。当总线发送允许时，优先级最高的邮箱先发送报文。发送模式使能后，CAN_MSR 寄存器的 MRDY 标志自动置位直至第一个报文发送出去。当应用程序要发送数据时，分别将报文的 ID、数据长度值和数据内容写入发送邮箱的相应位，然后置位 CAN_MCRx 的发送请求标志位，发送数据。CAN 报文发送数据流程如图 5-2 所示。

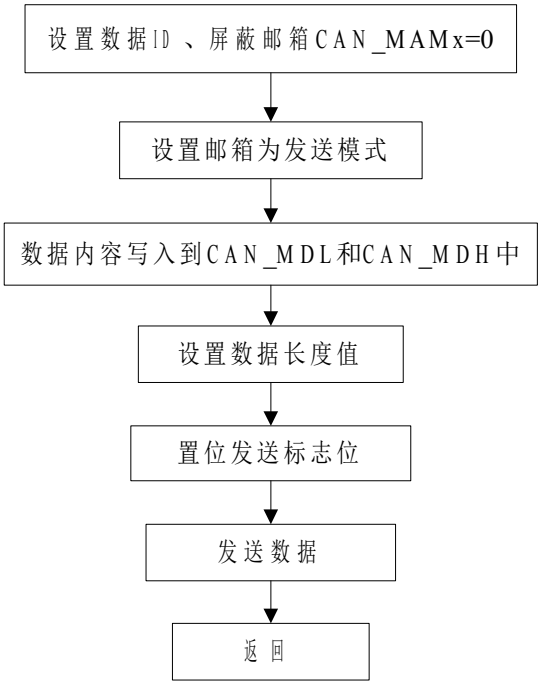


图 5-2 CAN 报文发送流程图

设置 MOT 位把邮箱配置为接收邮箱。接收模式使能后，CAN_MSR 寄存器的 MRDY 标志自动清 0 直至接收第一个报文。当控制器要接收报文时，把报文 ID 和接收邮箱的 ID 进行比较，符合的邮箱接收该报文，并把报文的数据内容和长度存储到邮箱内，然后将 MRDY 标志位置位。应用程序查询状态寄存器，如果有接收到报文，提取出报文的 ID、数据内容和数据长度，然后清空接收邮箱的寄存器，并设置邮箱的发送请求标志位，使邮箱能够接收新的报文。报文接收流程如图 5-3 所示。

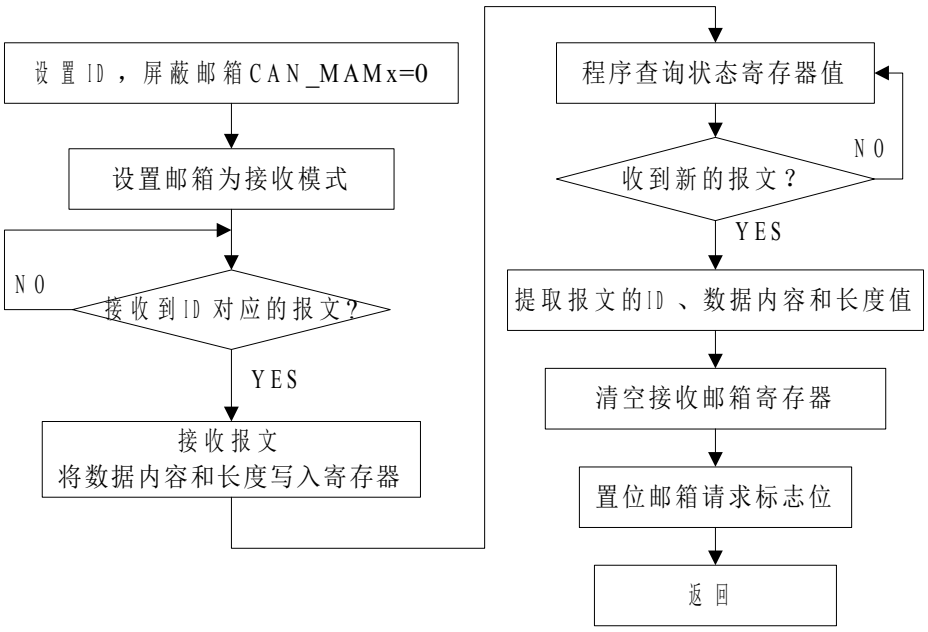


图 5-3 CAN 报文接收流程图

5.2 CANopen 协议的实现

本系统构建的是一个基本的 CANopen 监控网络，而基本的 CANopen 网络中只有一个 CANopen 主站和若干个 CANopen 从站，因此系统中的主 RTU 节点（RTU 通信控制模块）为 CANopen 的主站，从 RTU 节点（数据测控模块）为 CANopen 从站。CANopen 主从站在网络中拥有唯一的节点地址，节点地址范围为（1~127）。

CANopen 从站除了具有网络管理（NMT）从机的功能，还具备以下功能：（1）传输实时过程数据 PDO；（2）SDO 服务器功能；（4）节点/寿命保护或心跳报文、生产紧急报文。每个 CANopen 从站都需要有一个对象字典，描述从站所具有的通信参数和应用参数。

CANopen 主站具有网络管理（NMT）主机的功能，监控和管理整个网络中的从站，并支持以下功能：（1）支持 PDO、SDO 发送与接收；（2）支持 NMT 网络管理；（3）支持 PDO 通信类型并能够监控每一个 PDO 目标；（4）支持从站管理功能：类型与名称读取、对象字典读写；（5）紧急报文发送功能[24]。CANopen 主站也有自己的对象字典。

根据上述系统 CANopen 主从站所支持的功能，分别对各个功能部分进行实现。

5.2.1 对象字典

对象字典是 CANopen 协议中最核心的部分，它包含了描述本地设备和网络行为的所有参数。用户可通过访问 CANopen 设备的对象字典，获得该设备中的通信对象，以及用于该设备的对象（设备、应用或接口子协议）。由于用户是通过一个已知的 16 位索引识别对象字典中的对象，因此若将所有的索引都放在一个很大的表格中，则该表格中 2^{16} 个条目需要非常大的存储空间，如果把要用得到的条目写在一个表格中，这样就会存在空隙，用户就必须通过一种查找算法来找到合适的条目。本设计中采用如图 5-4 所示的一种 CANopen 结构，实现通过固定的索引/子索引对对象字典中的对象列表条目访问。

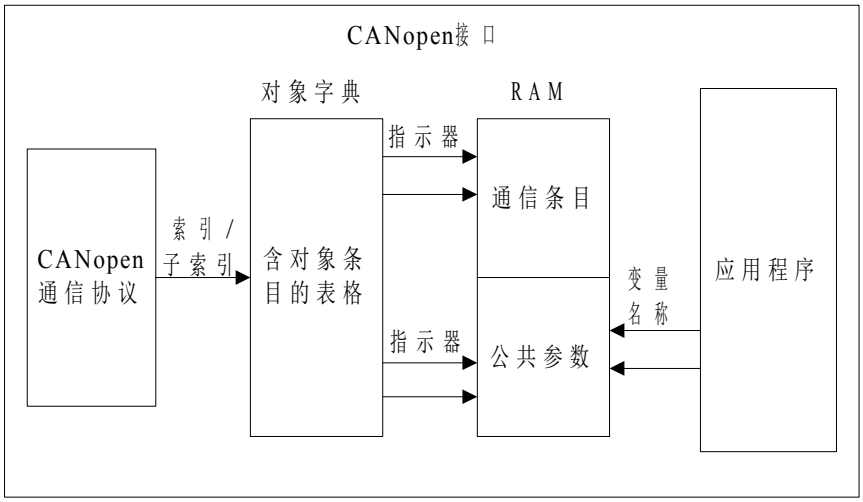


图 5-4 访问对象字典方式图

对象列表提供一个指向存储器中某个变量的指针，应用程序可直接通过变量名称访

问所需的条目。对象字典列表就构成了索引/子索引与对应变量名称之间的接口。对象字典的索引和子索引的编写如下：

```
typedef struct td_subindex
{
    u_char bAccessType;                 /*对象字典的访问类型*/
    u_char bDataType;                   /*对象字典的数据类型*/
    u_int   size;                       /*对象字典的变量长度*/
    void   *pObject;                    /*指针指向变量*/
} subindex;
typedef struct td_indextable
{
    subindex *pSubindex;                /*  子索引指针*/
    u_char     bSubCount;
} indextable;
```

CANopen网络中设备标识符的分配，既可采用缺省的默认值分配，也可采用动态配置方式。由第三章可知，CANopen协议为了减少对简单网络配置的工作，定义了一种强制型默认标识符分配方式，由于本系统实验所构建的监控网络比较简单，采用默认的标识符值分配完全能够满足网络的要求，所以采用预定义连接集的默认分配方案，运行期间标识符是固定的[25]。本系统构建的监控网络中的数据测控模块都是硬件设计一样的I/O模块，而且输入输出数据都为数字量，因此采用一个CANopen主站和一个CANopen从站进行实验，监控网络中的主从站Node-ID分配如表5-1所示。

表5-1 网络主从站Node-ID分配表

节点	主站	从站
Node-ID	0x01	0x03

由上述CANopen从站支持的功能可知，本系统的从站需要发送的对象有SDO（服务器）对象、TPDO1对象、心跳报文对象、Boot-up报文对象；需要接收的对象有SDO（客户端）对象、NMT对象、RPDO1对象。同时，与从站对应的主站需要发送的对象有SDO（客户端）对象、TPDO1对象、NMT对象，需要接收的对象为SDO（服务器）对象、RPDO1对象、从站心跳报文对象。因此必备的对象字典索引如表5-2所示。

表5-2 必备的对象字典索引表

索引	子索引	对象名称	数据类型	访问方式
1000H	0H	设备类型	UNSIGNED32	RO
1001H	0H	错误寄存器	UNSIGNED8	RO

续表5-2 必备的对象字典索引表

索引	子索引	对象名称	数据类型	访问方式
1018H	1H	制造商代码	UNSIGNED16	RO
1018H	2H	产品代码	UNSIGNED32	RO
1018H	3H	修正版号	UNSIGNED32	RO
1018H	4H	序列号	UNSIGNED32	RO
1016H	0H	消费者心跳周期	UNSIGNED32	RW
1017H	0H	生产者心跳周期	UNSIGNED16	RW
1200H	0H	条目数	UNSIGNED8	RW
1200H	1H	1STSDO COB-ID客户机→服务器	UNSIGNED32	RO
1200H	2H	1STSDO COB-ID服务器→客户机	UNSIGNED32	RO
1200H	3H	服务器节点号	UNSIGNED32	RO
1400H	0H	包含的条目数	UNSIGNED8	RO
1400H	1H	1STRPDO COB-ID	UNSIGNED32	RO
1400H	2H	发送类型	UNSIGNED32	RO
1600H	0H	RPDO映射对象的数目	UNSIGNED8	RW
1600H	1H	1STRPDO映射对象	UNSIGNED32	RW
1800H	0H	包含的条目数	UNSIGNED8	RO
1800H	1H	1STTPDO COB-ID	UNSIGNED32	RO
1800H	2H	发送类型	UNSIGNED32	RO
1A00H	0H	TPDO映射对象的数目	UNSIGNED8	RW
1A00H	1H	1STTPDO映射对象	UNSIGNED32	RW
6000H	0H	输入条目数	UNSIGNED8	RO
6000H	1H	8-BITs数字输入	UNSIGNED8	RW
6200H	0H	输出条目数	UNSIGNED8	RO
6200H	1H	8-BITs数字输出	UNSIGNED8	RW

本系统设计各个从站只能与主站进行通信，从站之间不能进行通信。因此从站只有自己本地的对象字典，主站作为监控枢纽与从站进行通信，因此其不仅具有自己本地的对象字典外，而且还集合了从站的对象字典。从站只可访问自己本地的对象字典，主站不仅能够访问自己本地的对象字典，而且通过SDO服务数据对象的通信机制访问从站的对象字典。本地对象字典的访问流程如图5-5所示。

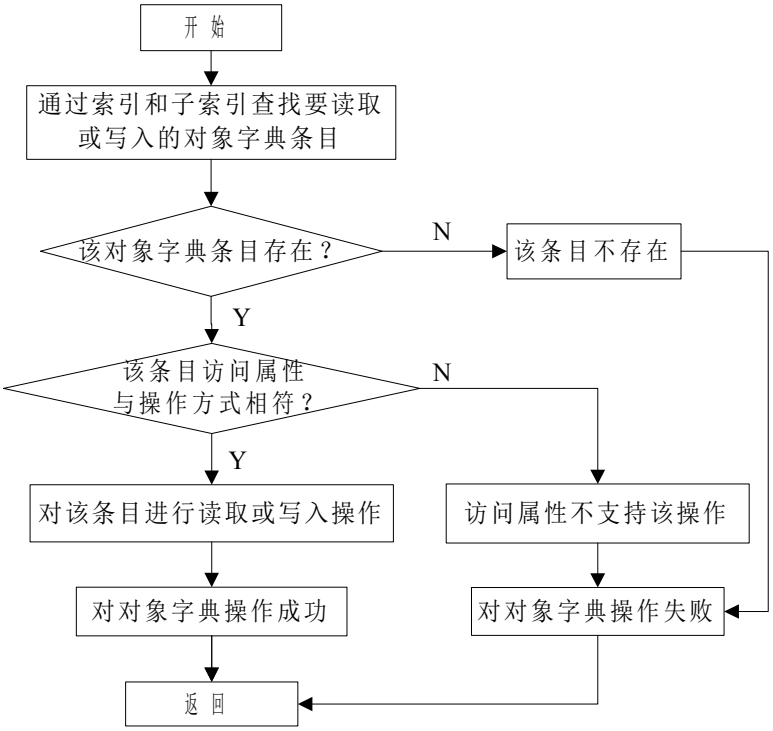


图5-5 本地对象字典访问流程图

5.2.2 NMT 网络管理

CANopen协议中的网络管理服务包括模块控制服务和错误控制服务。NMT模块控制报文只能由NMT主机发送，NMT模块控制服务被所有NMT从机支持。NMT从机任何时候都可接收NMT主机发送的模块控制报文，实现NMT主机控制NMT从机的状态转换。NMT错误控制包括节点/寿命保护、心跳报文协议和Boot-up，用来监测网络中站节点的状态。

所有CANopen设备都具有NMT从机功能，通常NMT从机都由NMT主机来启动、监控和重启。在CANopen网络中只允许有一个活动的NMT主机，其负责网络管理，用于控制所连接的网络设备和NMT从机的状态。因此系统设计CANopen主站为NMT主机，CANopen从站为NMT从机。

NMT主机上电启动后，采用自动方式启动自身内部的NMT状态机，自行从预运行状态转换到运行状态。NMT主机从预操作状态转换到运行状态之前，它需要启动所有NMT从机。NMT主机控制网络初始化节点启动流程如图5-6所示。

NMT从机在上电启动后，通过发送Boot-up报文告知NMT主机其已经从初始化状态转换成预操作状态，内部状态机的状态转换由NMT主机通过发送NMT模块控制报文控制。NMT从机的Boot-up报文格式如表5-3所示。

表5-3 NMT从机boot-up报文格式

COB-ID	字节0
0x700+从机ID	0

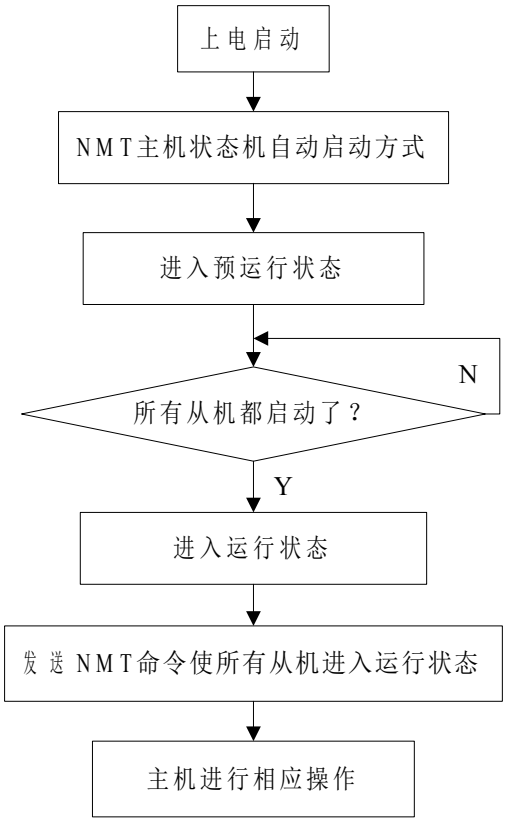


图5-6 NMT主机控制网络节点启动初始化流程图

NMT从机在网络中NMT主机管理控制下的流程如图5-7所示。

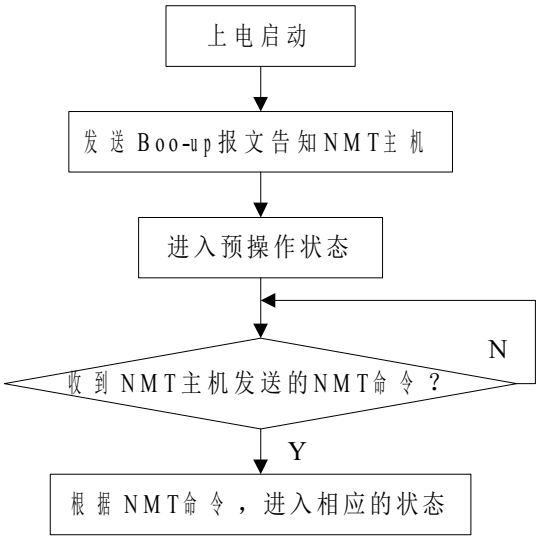


图5-7 NMT状态流程图

NMT错误控制包括节点/寿命保护和心跳报文协议。由于CANopen协议规定一个站节点只能采用节点/寿命保护或心跳协议中的一种方法来进行设备监控[26]。心跳报文协议能实现更加灵活的监控结构，而且不需要使用远程帧，因此本系统对站节点采用心跳报文协议的监控方式。心跳报文协议是基于生产者/消费者的模式，所以本系统设计NMT主机为心跳消费者，NMT从机为心跳生产者。在心跳时间周期内，NMT主机通过解析接

收到的NMT从机心跳报文，便可知NMT从机的实时状态。心跳报文格式如表5-4所示。

表5-4 心跳报文格式

COB-ID	字节0	
0x700+从机ID	状态	
	0	Boot-up初始化
	4	停止
	5	运行
	127	预运行

NMT从机（心跳生产者）的Boot-up报文是其第一个心跳报文，心跳报文发送流程如图5-8所示。

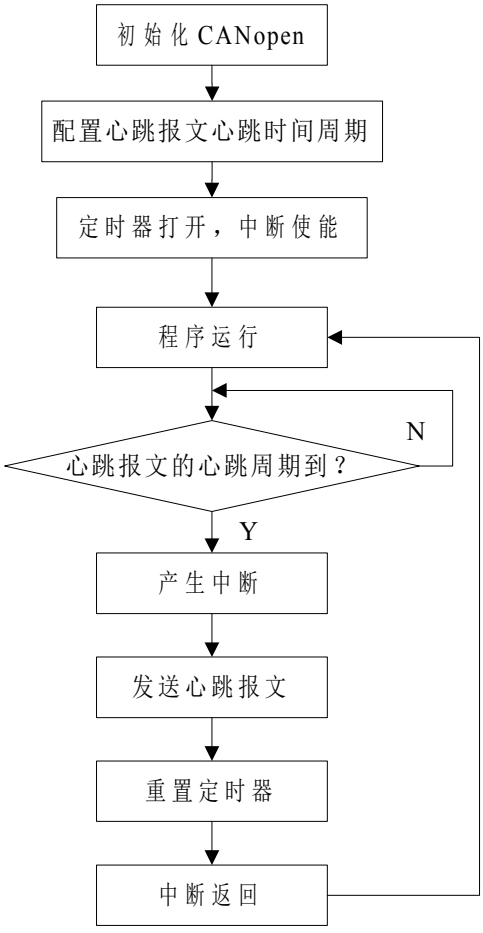


图 5-8 心跳报文发送流程图

5.2.3 SDO 服务数据

CANopen 设备对象字典的条目可通过 SDO 服务数据对象来访问，发起访问的设备称为 SDO 客户端（Client），被访问对象字典且提供所请求服务的 CANopen 设备称为默认的 SDO 服务器（Server）。CAN 标识符的默认分配方案建立了 NMT 主机与 CANopen 网络中其他设备之间默认的 SDO 通信。NMT 主机也是所有默认 SDO 服务器的客户端，

NMT 从机是默认 SDO 服务器。SDO 通信对象只能在设备状态机中的预运行和运行状态下使用。SDO 客户端可以对 SDO 服务器对象字典中的单个对象进行读/写，读操作成为“上传”，写操作成为“下载”。由上述可知，本系统中 CANopen 从站为 SDO 服务器，CANopen 主站为 SDO 客户端，利用 SDO 通信读/写对象字典条目如图 5-9 所示。

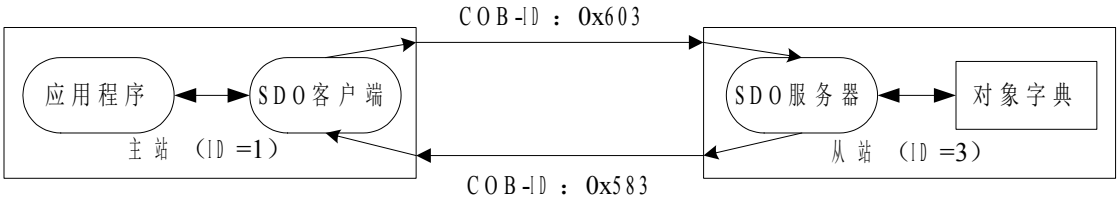


图 5-9 SDO 读/写对象字典条目图

由第三章可知，SDO 有三种传输类型：加速 SDO 传输、分段传输、块传输。本系统采用的是基本 CANopen 网络较简单，SDO 传输的数据量不大，因此采用加速 SDO 传输和分段传输方式即可[27]。与传输类型对应的 SDO 请求/应答报文有：启动域下载，域分段下载，启动域上传，域分段上传和域传送中止。它们在 SDO 的命令字中 5 至 7 位被定义如表 5-5 所示：

表 5-5 SDO 命令字（5 至 7 位）

报文类别	Client->Server（请求）	Server->Client（应答）
启动域下载	001B	011B
启动域上传	010B	010B
域分段下载	000B	001B
域分段上传	011B	000B
域传送中止	100B	

SDO 通信对象的数据结构定义如下：

```
struct struct_s_transfer {
    u_char              nodeId;
    u_char              whoami;
    u_char              state;
    u_char              toggle;
    u_int               abortCode;
    u_short             index;
    u_char              subIndex;
    u_int               count;
    u_int               offset;
    u_char              data[SDO_MAX_LENGTH_TRANSFERT];
#ifdef SDO_DYNAMIC_BUFFER_ALLOCATION
```

```
    u_char                *dynamicData;
    u_int                 dynamicDataSize;
#endif
    u_char                dataType;
    TIMER_HANDLE      timer;
    SDOCallback_t Callback;
};
```

SDO 传输主要分为发起传输、读或写数据和结束传输 3 步。发起传输时，SDO 客户端通知 SDO 服务器要访问对象字典（索引、子索引）中的哪一个条目，以及访问的类型（写或读）。在发起传输阶段，SDO 服务区会检查将要访问的对象是否存在以及是否允许访问。在读或写数据阶段，待传输的数据如果不大于 4 字节则采用加速传输方式，直接通过 CAN 报文来传输；如果大于 4 字节便采用分段传输方式，传输的数据会被分成几个 7 字节大小的段，然后通过 CAN 报文来传输。当特定的结束标识符传输完毕后，SDO 数据传输结束。此外，SDO 服务器与 SDO 客户端也可以随时通过中止消息结束传输。用于 SDO 传输的 CAN 报文长度通常为 8 字节，第 1 个字节通常为 SDO 命令字，其余字节为数值 0。

5.2.4 PDO 过程数据服务

PDO 过程数据对象来传输实时数据，分为发送 PDO（TPDO）和接收 PDO（RPDO），其只能在设备的运行状态下使用。由上述可知，CANopen 主从站分别有 TPDO 和 RPDO，CANopen 从站在 TPDO 中发送现场设备输入的数据，对于接收这个 TPDO 数据的 CANopen 主站来说，这个 TPDO 就是主站的 RPDO，反之亦然。在本系统中相互对应的关系如图 5-10 所示。

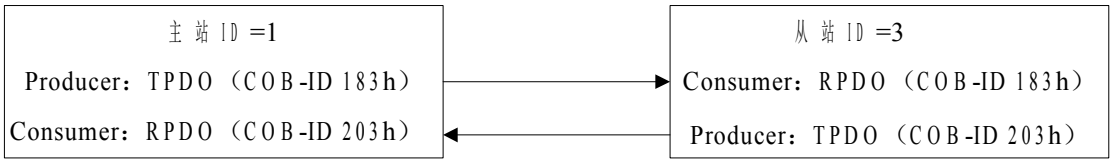


图 5-10 CANopen 主从站 PDO 对应关系图

每一个 PDO 都在它的对象字典中通过 PDO 通信参数和 PDO 映射参数来描述。通信参数用来描述 PDO 的特性，其按照定义好的地址（16 位索引和 8 位子索引）保存在设备对象字典中；映射参数包含指向 PDO 需要发送的过程数据的指针（利用索引和子索引表示）。PDO 通信参数和映射参数的数据结构如下：

```
typedef struct td_s_pdo_communication_parameter // Index: 0x20
    {
     u_char    count;
     u_int    cob_id;
```

```
    u_char    type;
    u_short    inhibit_time;
    u_char    reserved;
    u_short    event_timer;
    } s_pdo_communication_parameter;
#define COUNT_OF_PDO_MAPPING_PARAMETER    4
typedef struct td_s_pdo_mapping_parameter      // Index: 0x21
   {
    u_char count;
    u_int object[COUNT_OF_PDO_MAPPING_PARAMETER];
    }s_pdo_mapping_parameter;
```

根据本系统的具体设计，同时选择PDO报文的触发方式为异步触发，从站RPDO和TPDO的通信参数和映射参数的具体配置如表5-6，表5-7所示。主站根据从站相关参数的配置进行设置[28]。

表5-6 从站PDO对象通信参数配置表

PDO对象	条目数量		COB-ID	发送类型
RPDO	2		0x203	255（异步）
TPDO	2		0x183	255（异步）
相应的对象字典配置				
索引	访问类型	数据类型	变量长度	通信参数的域
1400h（RPDO）	RO	uint8	1	条目数
	RO	uint32	2	COB-ID
	RO	uint32	1	发送类型
1800h（TPDO）	RO	uint8	1	条目数
	RO	uint32	2	COB-ID
	RO	uint32	1	发送类型

表5-7 从站PDO对象映射参数配置表

	映射对象数目		被映射的对象	
RRDO	1		0x62000108（8位数字量输出）	
TPDO	1		0x60000108（8位数字量输入）	
相应的对象字典配置				
索引	访问类型	数据类型	变量长度	映射参数的域
1600h（RPDO）	RO	uint8	1	条目数
	RW	uint32	4	数字量输出1

续表5-7 从站PDO对象映射参数配置表

索引	访问类型	数据类型	变量长度	映射参数的域
1600h（RPDO）	RW	uint32	4	数字量输出2
	RW	uint32	4	数字量输出3
	RW	uint8	4	数字量输出4
1A00h（TPDO）	RO	uint8	1	条目数
	RW	uint32	4	数字量输入1
	RW	uint32	4	数字量输入2
	RW	uint32	4	数字量输入3
	RW	uint32	4	数字量输入4

当 CANopen 主从站要传输数据时，CANopen 主从站都需要工作在运行状态。当要发送数据时，把数据放到相应的对象字典中，然后根据通信参数和映射参数把数据封装成 PDO 报文发送出去。当要接收数据时，解析接收到的 PDO 报文，根据其 ID 查找到其在对象字典中的位置，将 PDO 报文中的数据写入到该位置中，完成 PDO 报文的接收。

5.3 CANopen 网络通信

由上述可知，根据系统的功能需要，这里只实现 CANopen 协议的部分内容，主要包括 CANopen 协议各个通信参数的初始化，相关对象字典的构建，SDO 通信对象、PDO 通信对象、NMT 通信对象和心跳报文几个部分。本文设计的主从 RTU 节点通过 CANopen 网络进行通信的整体流程如图 5-11 所示。其中，系统初始化包括μC/OS-Ⅱ操作系统初始化，整个外围电路的初始化，处理器芯片的初始化，CAN 控制器初始化，主从站的通信参数初始化等。

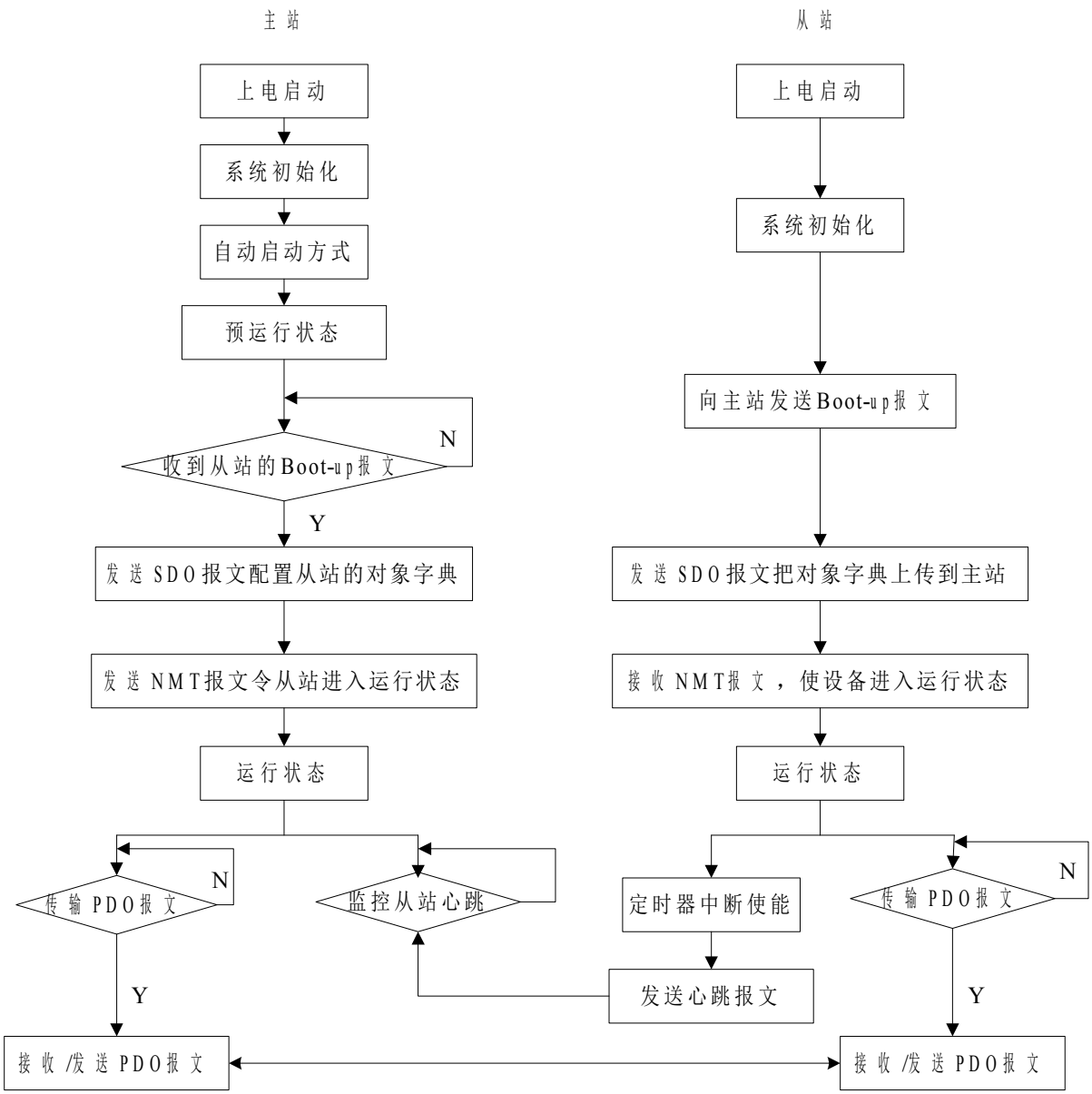


图 5-11 CANopen 网络通信流程图

为了直观的验证该监控网络系统的有效性，在与主 RTU 节点进行通信的上位机 PC 上，用组态软件设计了一个“CANopen 监控界面”，节点的报文信息通过串口输出到 PC 机上通过该“CANopen 监控界面”显示。根据上述 CANopen 网络通信整体流程，从节点初始化结束后向主节点发送的 Boot-up 报文如图 5-12 所示。随后主从节点进入预运行状态，其心跳报文如图 5-13 所示。当从节点进入预运行状态后，主节点通过发送 NMT 模块控制报文，控制从节点进入运行状态，这时从节点便可把实时数据发送到主节点，数据显示如图 5-14 所示。通过实验测试表明，基于 CANopen 协议的主从 RTU 节点间通信正常，构建的基于 CANopen 协议的监控网络系统运行正常。

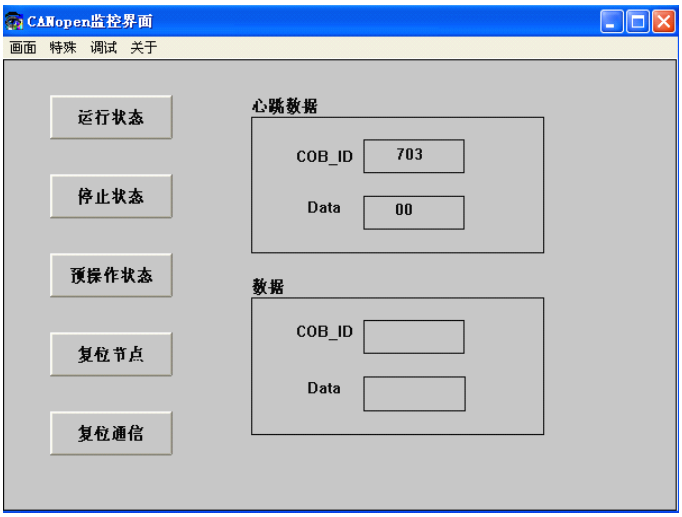


图 5-12 Boot-up 报文图

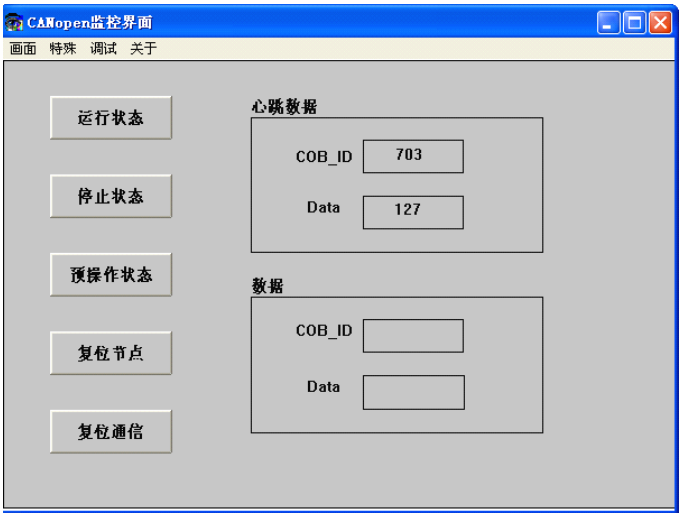


图 5-13 预运行状态的心跳报文图

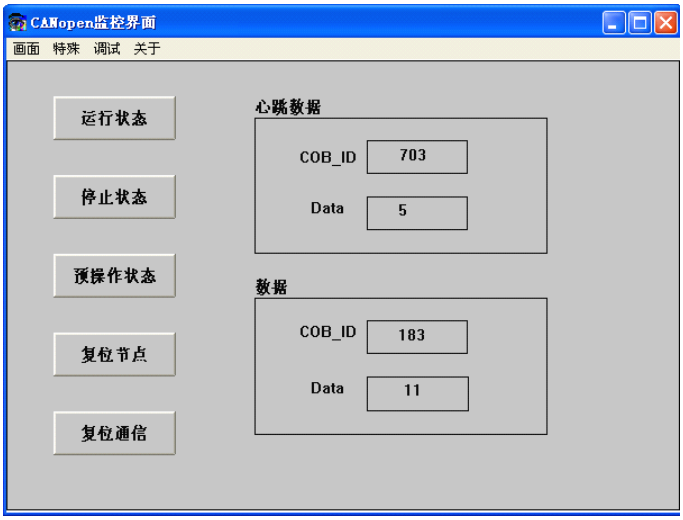


图 5-14 运行状态的数据报文图

第六章　结束语

CANopen 协议做为开放的、标准的 CAN 现场总线的应用层协议，支持不同制造商的设备互操作和互换，得到越来越多的应用。同时，RTU 在远程数据采集转换及通信方面具有明显的优势，在广域分布式监控系统中得到越来越多的应用。将 CANopen 协议应用于 RTU 上，使得 RTU 基于 CAN 现场总线进行通信，非常适合于工业现场远程监控。本文在对 CANopen 协议的通信规范和设备规范的深入分析研究的基础上，以油田油井为应用背景，设计了一种基于 CANopen 协议的监控网络系统，采用 AT91SAM7X256 为主控制器的主 RTU 通信控制模块和从 RTU 数据测控模块，实现通过 CANopen 协议进行通信，达到监控现场设备的目的。

6.1 论文完成的工作

主要的研究工作和研究成果包括以下几点：

（1）深入研究 CAN 总线和高层协议 CANopen，介绍了 CAN 总线通信的特性，重点深入研究了 CANopen 协议的通信子协议和设备子协议。

（2）根据油田油井的实际状况，结合 RTU 和 CANopen 协议的特点，提出了将 CANopen 协议应用于 RTU 中的一种监控系统。

（3）结合实际需要选用了 ATMEL 公司的 AT91SAM7X256 微处理器作为 RTU 模块的主控制器，设计了主从 RTU 节点的硬件电路。

（4）分析嵌入式操作系统的特点，选用μC/OS-Ⅱ操作系统作为 RTU 模块上的嵌入式操作系统，并把μC/OS-Ⅱ移植到 AT91SAM7X256 微处理器上，成功运行。

（5）深入研究 AT91SAM7X256 芯片，重点分析了其内置的 CAN 控制器的特性、各个寄存器功能设置和收发机制，并编写了 CAN 通信控制程序。

（6）根据本文构建的系统功能，实现了 CANopen 协议中的对象字典、PDO 过程数据对象、SDO 服务数据对象、NMT 网络管理和心跳报文，达到了主从 RTU 节点间基于 CANopen 通信的功能。

6.2 问题和展望

由于时间和能力所限，论文不可避免存在一些漏洞和缺陷，有待于进一步的改进和完善，总结如下：

（1）本文设计采用预定义连接集的默认分配方案，运行期间标识符是固定的，不能用动态方式分配节点 ID。

（2）本文搭建的 CANopen 监控网络系统是基本的 CANopen 网络结构，网络中只有一个主节点，即一个 NMT 主机。现场应用中处于安全原因，网络中应包含多个 NMT 主机，即“动态主机”（Flying NMT Master）的方式：当活动的 NMT 主机出现故障时，另外一个设备将会自动承担 NMT 主机的工作。

（3）对于 CANopen 协议的紧急报文的研究力度不够，没有使用紧急事件

（Emergency），需要进一步将其应用于系统中，提高系统的错误预警能力。

（4）硬件电路主要为基本协议实现而设计，没有充分考虑电磁兼容性设计，还需在以后的实践设计中进一步考虑。

CANopen 协议的功能十分丰富和强大，CANopen 规范全集除了包括应用层和通信协议之外，还包括各种不同的框架、建议，以及标准的设备规范、接口协议与应用技术规范，在工业现场总线中的优势在实际应用中逐渐被体现出来。而且，RTU 的应用场所不受环境因素的影响，其诸多优点使得其在恶劣的环境下亦能够稳定安全的运行。所以，基于 CANopen 协议的 RTU 不仅能够应用于油田、气田、煤矿等诸多国家经济命脉的能源企业中，而且可以应用于各个工业领域中。

致　谢

衷心感谢曹庆年老师、孟开元老师对作者的悉心指导。在三年的硕士研究生学习期间，不仅从两位老师那里学到了丰富的专业知识和独立进行科研工作的方法，而且老师渊博的知识，严谨的治学态度，求实的工作作风，勇于探索的精神和对科学孜孜不倦的追求都给作者留下了深刻的印象，并深深影响着作者。在此，特向尊敬的曹庆年与孟开元两位老师表示最衷心的感谢。

衷心感谢硕士研究生阶段培养作者的西安石油大学，感谢北京安控科技发展有限公司西安办事处的庄贵林总工程师、汪长海工程师以及相关的工作人员为本文研究和论文写作提供的帮助。

感谢 RTU 联合实验室的赵博师姐、曹蕾师姐、侯贵双师兄、陈晨师姐、杨鹏、张珂和董鑫在学习和生活上给予的帮助和支持。并感谢西安石油大学计算机学院的老师们和作者的同学们，在三年的硕士研究生阶段给作者的帮助和支持。

感谢作者的家人，在作者人生的过程中，家人一直在默默的关心和支持着，同时也感谢 RTU 联合实验室基金对本课题的资助。

最后，向在百忙之中评审本文的各位专家老师们致以深深的谢意！

参考文献

[1] 王黎明，夏立等．CAN 现场总线系统的设计与应用[M]．北京：电子工业出版社，2008．

[2] 阳宪惠．现场总线技术与应用[M]．北京：清华大学出版社，1996．309~320．

[3] 饶运涛，邹继军，王进宏，郑勇芸．现场总线 CAN 原理与应用技术（第二版）[M]. 北京：北京航空航天大学出版社，2007．

[4] 程丽平，赵协广，杨婕．基于 DSP 和 CAN 总线的 RTU 的设计[J]．微型机与应用，2011，30（2）：35~37．

[5] 孙健，陶维青．CAN 应用层协议 CANopen 浅析[J]．仪器仪表标准化与计量，2006，2：22~24．

[6] 陈在平，王峰．基于 CANopen 协议从节点研究[J]．制造业自动化，2010，2：27~30．

[7] 肖红翼，翁惠辉，毛玉蓉．采油场井口 RTU 测控系统[J]．石油仪器，2007，21（6）67~69．

[8] 田泽．嵌入式系统开发与应用教程[M]．北京：北京航空航天大学出版社，2005．6~26．

[9] 韩山，郭云，付海燕．ARM 微处理器应用开发技术详解与实例分析[M]．北京：清华大学出版社，2007．8~9．

[10] 马忠梅，徐英慧，叶勇建等．AT91 系列 ARM 核微控制器结构与开发[M]．北京：北京航空航天大学出版社，2003．1~17．

[11] 吕少中，赵国志，张丽杰．基于 AT91SAM7X256 的智能分站控制系统[J]．微计算机信息，2008，24（4）：91~92．

[12] ATMEL 公司，AT91SAM7X256_datasheet[pdf]．http://www.atmel.com，2006．

[13] ATMEL 公司，AT24C256_datasheet[pdf]．http://www.atmel.com，2008．

[14] 赵博．基于 CAN 总线的 PLC 模块通信协议研究与实现[D]．西安：西安石油大学，2009．37~38．

[15] Philips Semiconductors，TJA1050_datasheet[pdf]．Philips Semiconductors，2000．

[16] CiA DS202．CAN Application Layer for Industrial Applications，CMS Service Specification[S]，1996．18~24．

[17] CiA DS203．CAN Application Layer for Industrial Applications，NMT Service Specification[S]，1996．3~20．

[18] CiA DS204．CAN Application Layer for Industrial Applications，DBT Service Specification[S]，1996．2~13．

[19] CiA DS202．CAN Application Layer for Industrial Applications，LMT Service Specification[S]，1996．2~9．

[20] CiA DS301．CANopen Appplication Layer and Communication Profile，Verison4.02[S]，

2002．

[21]Holger Zeltwanger 著，周立功等译．北京：北京航空航天大学出版社，2011．203~204．

[22]JEAN J.LABROSSE 著，邵贝贝等译．嵌入式实时操作系统μC/OS-Ⅱ（第二版）[M]. 北京：北京航空航天大学出版社，2003．7~9．

[23]黄燕平．μC/OS-Ⅱ移植要点详解[M]．北京：北京航空航天大学出版社，2005．72~103．

[24]饶怡欣，胥布工，匡付华．基于 CANopen 的电动执行机构远程监控主站的实现[J]. 计算机测量与控制，2010，18（2）：373~375．

[25]王芳．基于汽车总线模型的 CANopen 协议的实现[D]．天津：天津工业大学，2008. 48~49．

[26]赵峰．CANopen 协议研究及一体化适配器开发[D]．北京：北京化工大学，2009．52~53．

[27]程俊.基于 CANopen 协议的电动执行机构从站研究[D]．广州：华南理工大学，2010. 52~53．

[28]CiA DSP-401．Device Profile for Generic I_O Modules，Version2.1[S]，2002．

攻读硕士学位期间发表论文

[1] 曹庆年，胡欣欣，孟开元等．基于 HART 协议的工业无线监控系统设计[J]. 西安石油大学学报（自然科学版），2012，27（1）：99～103．