西南交通大学

硕士学位论文

CANopen协议在冗余系统中的应用研究

姓名：周跃峰

申请学位级别：硕士

专业：模式识别与智能系统

指导教师：王玉松

20090501

摘 要

CANopen协议是基于CAN总线的高层协议，它是用于标准的嵌入式网络开发的具有灵活配置能力的开放式协议。目前已成为基于CAN总线的分布式自动化系统的标准协议（EN50325-4），广泛应用于工业控制的各个领域。

最初设计的CAN总线只实现了开放式系统互连参考模型的物理层与数据链路层。一般用户必须直接用驱动程序操作链路层，在较复杂的应用系统中，不能直接满足工业控制网络的组态和产品互连要求。故出现了多种上层协议（如CANopen、DeviceNet和 SDS等）。本文选择开放的、通用的、实时的、流行欧洲的CANopen协议作为研究对象。另外，在实际的工业领域中冗余控制作为一种能够满足连续生产要求、提高系统可靠性的有效手段而被广泛应用。而标准的CANopen协议并没有对冗余控制进行明确规定。因此，本文选CANopen协议及其冗余控制作为研究对象，具有较强的理论意义和实用价值。

本文首先对CANopen协议的基本框架、对象字典、通讯对象以及标识符的分配等进行了深入剖析。接着解决了CANopen协议在实际应用系统中实现的关键技术（冗余和实时）。然后应用模块化、结构化的设计思想，把基于CANopen协议的冗余系统划分为：硬件层、抽象层和协议层，并分别进行实现。特别在协议层的设计中，充分依据“开-闭”原则，对各个模块的消息语法细节、主要数据结构及其程序实现进行了设计。首次提出了一套应用CANopen协议管理冗余网络的管理机制。

最后，基于上述研究成果，实际开发了机务段股道管理自动化系统的CAN通讯模块。搭建了以合肥客机段为模型的调试平台。经检验，该模块的实时性强、可靠性高，且具有良好的可移植性与通用性。

关键词：CAN总线；CANopen协议；冗余

Abstract

CANopen is a high-level protocol which is is based on CAN bus, It is an open-ended protocol with flexible configuration capabilities and is used in embedded network developed .It has become the standard protocol of distributed automation systems which is based on CAN bus (EN50325-4), it is widely used in various industrial control areas

The CAN bus originally only designed the physical layer and data link layer of the OSI model. Common users have to deal with the data link layer with the drive program directly, in more complex applications, CAN bus can not directly fullfill the needs of interconnection configuration and product demands for industrial control network. Thus, there emerged a variety of upper-layer protocols (eg CANopen, DeviceNet and SDS, etc.). In this paper, we choose the CANopen protocol which is open、common、real-time and popular in Europe as the study object.The research has the strong theoretic significience and realistic value.

In this paper, we expound the basic framework、the object dictionary、communication objects and the allocation of COB-ID of CANopen protocol first. Then we solved the key technologies (redundancy and real-time)CANopen protocol used in realistic application system.And then the application adopted the modular、 structured design ideal, it divided the redundant system which is based on CANopen protocol into three parts: the hardware layer, abstract layer and protocol layer, and each part were carried out to achieve. Particularly in the design of protocol layer, sufficient basis for "Open - Closed" principle, each module for the details of the message syntax, the main data structures and procedures designed are achieved. We first bring forward a set of management mechanism for redundant CANopen network management protocol.

Finally, based on the results of the study, the actual development

of the Locomotive Depot Road Management Unit Automation System CAN communication module is designed. A debugging platform based on the model of the Hefei Passenger Locomotive Depot was build. After testing, the module is high in real-time、reliability and is good at portability and versatility.

key words: CAN bus; CANopen protocol; redundancy

西南交通大学

学位论文版权使用授权书

本学位论文作者完全了解学校有关保留、使用学位论文的规定，同意学校保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权西南交通大学可以将本论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复印手段保存和汇编本学位论文。

本学位论文属于

1．保密□，在　　年解密后适用本授权书；

2．不保密☑，使用本授权书。

（请在以上方框内打“√”）

学位论文作者签名：周跃峰　　指导老师签名：

日期：09.6.5　　日期：09.6.5

西南交通大学学位论文创新性声明

本人郑重声明：所呈交的学位论文，是在导师指导下独立进行研究工作所得的成果。除文中已经注明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的研究成果。对本文的研究做出贡献的个人和集体，均已在文中作了明确的说明。本人完全意识到本声明的法律结果由本人承担。

本学位论文的主要创新点如下：

1. 首次提出了一套应用 CANopen 协议管理并联系统的管理机制。
2. 结合机务段股道管理自动化系统对总线应用的需求，应用 CANopen 协议，设计并开发了 CAN 总线通讯模块。

学位论文作者签名：

日期：09.6.15

第1章 绪论

1.1 课题的提出

基于以下四个方面的原因提出本课题:

1、CAN应用的广泛性

CAN(Controller Area Network)即控制器局域网络。由于其高性能、高可靠性、及独特的设计,CAN越来越受到人们的重视。CAN最初是由德国的BOSCH公司为汽车监测、控制系统而设计的。由于得到Motorola，Intel，Philip，Siemens，NEC等公司的支持，使得CAN-bus成为国际上应用最广泛的现场总线之一。最初广泛应用于欧洲的中高档汽车中；近几年来，CAN-bus被广泛地应用于工业自动化、船舶、医疗设备、工业设备等方面。

2、CAN的不足

由于CAN总线是针对相对较少信息的发送而设计优化的一种串行通讯协议，因此传输大数据量的能力较差，一次最多只能传送8字节。

它只定义了物理层和数据链路层，本身并不完整，有些复杂的应用问题需要一个更高层次的协议——应用层协议来实现。

CAN的技术特点允许各厂商在CAN协议的基础上自行开发自己的高层应用协议，给用户提供了一个面向应用的清晰接口。

3、CANopen的优点

- 提出了CAN传输8字节以上数据的一套解决方案。
- 通过CANopen协议及其设备子协议，使得多家制造商的产品能够用于任何 CANopen网络，解决了一致性、互用性及互换性问题。

另外，规定了一套缺省的ID分配表，根据帧的类别分配ID的优先级；规定了一套CAN总线的数据管理方式；规定了一套网络的管理办法；提出了一套系统的同步操作方案。

4、冗余应用的必要性

在现代的军用电子设备和某些特殊工作环境中，对系统的可靠性提出了更高的要求。为了提高系统的可靠性，采用冗余备份技术,使系统在出现故障

时,仍可以保持正常工作。冗余是常用的保护系统可依赖性的技术，也是构建可生存网络系统的重要技术。冗余技术可以减少网络系统中存在的单点故障，而对冗余机制做出变化得到的多样性和不可预测性可以减少系统组件的共模故障和相同的脆弱点。

1.2 课题的国内外研究现状

CANopen协议在国外蓬勃发展，受到了足够的重视。在欧洲，CANopen协议已被广泛的应用于医疗装置中，并进一步扩展应用到保安控制系统中；在美国，CANopen协议已经成为装载机械和公共运输设备的协议标准，同时也应用于嵌入式系统的控制。目前，国外已有许多大公司开发了CANopen软件和硬件产品，比如：Northhampton公司的CANopen开发工具，Downers Grove公司的CANopen控制模块，Elkhart公司的CANopen的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uCAN.Open.er和Philips公司的CANopenIA Developer's Kit。

在我国CAN的应用层协议主要应用DeviceNet，很少用到CANopen。CANopen的实现尚处于探索阶段。[47]

1.3 课题的研究意义

通过本课题的研究，分析了以CANopen高层协议为基础的标准CAN总线应用层协议，使CAN总线的应用层协议的开发摆脱了各家各不相同的局面，实现了标准化和通用性。切实地掌握CAN总线的硬件和软件的开发应用技术。CANopen协议解决了复杂的并一串联系统的管理控制。CANopen应用于机务段股道管理自动化系统中，提高了系统的实时性与可靠性，增强了系统的可移植性与通用性。还对冗余和实时进行了探讨。相信本课题对CANopen和冗余及其相关技术的应用和发展有一定的借鉴和推动作用。

第2章 CAN总线及上层协议CANopen的剖析

2.1 CAN总线协议的分析

CAN(Controller Area Network)既控制器局域网络。由于其高性能、高可靠性及独特的设计，CAN越来越受到人们的重视。CAN最初是由德国的BOSCH公司为汽车监测、控制系统而设计的。由于得到Motorola，Intel，Philip，Siemens，NEC等公司的支持，使得CAN-bus成为国际上应用最广泛的现场总线之一。最初广泛应用于欧洲的中高档汽车中；近几年来，CAN-bus开始进入各个行业的应用。

2.1.1 CAN 总线的技术规范

随着CAN在各个领域的应用和推广，对其通信格式的标准化提出了要求。为此，1991年9月Philip Semiconductors制定并发布了CAN技术规范（Version 2.0）。该技术规范包括A和B两部分。2.0A给出了CAN报文标准格式，而2.0B给出了标准的和扩展的两种格式。此后，1993年11月ISO正式颁布了轨道交通运输工具——数据信息交换——高速通信控制局域网（CAN）国际标准ISO－11898，为控制器局域网的标准化、规范化铺平了道路。

2.1.1.1 CAN 节点的分层结构

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（包括逻辑链路控制子层LLC和媒体访问控制子层MAC）和物理层，而在CAN技术规范2.0A的版本中，数据链路层的LLC和MAC的服务和功能被描述为“目标层”和“传送层”。CAN的分层结构和功能如图2-1所示。

LLC子层的主要功能是：为数据传送和远程数据请求提供服务，确认由LLC子层接收的报文实际已被接收，并为恢复管理和通知超载提供信息。在定义目标处理时，存在许多灵活性。MAC子层的功能主要是传送规则，亦即控制帧结构，执行仲裁、错误检测、出错标定和故障界定。MAC子层也要确定，为开始一次新的发送，总线是否开放或者是否马上开始接收。位定时特性也是MAC

子层的一部分。MAC子层特性不存在修改的灵活性。物理层的功能是有关全部电气特性。自然，在一个网络内物理层的所有节点是相同的。然而，在选择物理层时存在很大的灵活性。

CAN技术规范2.0B定义了数据链路层中的MAC子层和LLC子层的一部分；并描述与CAN有关的外层。物理层定义信号怎样进行发送，因而，涉及位定时、位编码和同步的描述。在这部分规范中，未定义物理层中的驱动器/接收器特性，以便有序根据具体应用，对发送媒体和信号电平进行优化。MAC子层是CAN协议的核心，它描述由LLC子层接收的报文和对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。MAC子层由称为故障界定的一个管理实体监控，它具有识别永久故障或者短暂扰动的自检机制。LLC子层的主要功能是报文滤波、超载通知和恢复管理。

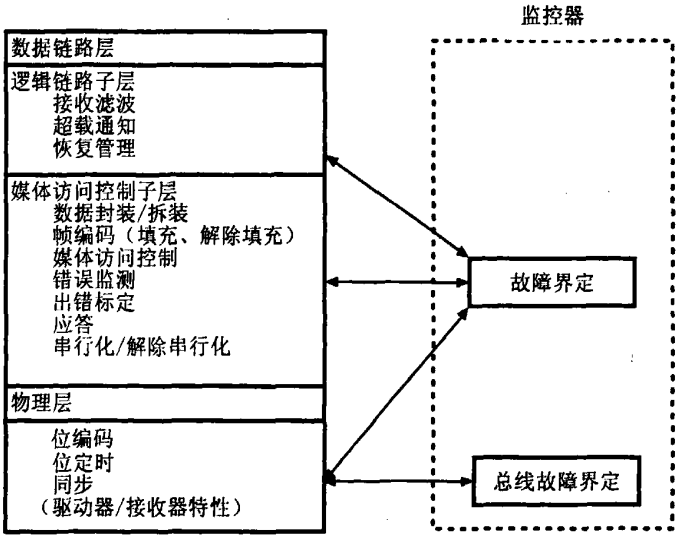


图2-1 CAN的分层结构和功能

2.1.1.2 CAN传送的报文及其帧格式

在进行数据传送时，发出报文的单元称为该报文的发送器。该单元在总线空闲或丢失仲裁前恒为发送器。如果一个单元不是报文发送器，并且总线不处于空闲状态，则该单元为接收器。

对于报文发送器和接收器，报文的实际有效时刻是不同的。对于发送器而言，如果直接到帧结束末尾一直未出错，则对于发送器报文有效。如果报

文受损，将允许按照优先权顺序自动重发送。为了能同其他报文进行总线访问竞争，总线一旦空闲，重发送立即开始。对于接收器而言，如果直到帧结束的最后一位一直未出错，则对于接收器报文有效。

构成一帧的帧起始、仲裁场、控制场、数据场和CRC序列借助位填充规则进行编码。当发送器在发送的位流中检测到5位连续的相同数值时，将自动地在实际发送的位流中插入一个补码位。数据帧和远程帧的其余位场采用固定格式，不进行填充。出错帧和超载帧同样是固定格式，也不进行位填充。

报文传送有4种不同类型的帧表示和控制：数据帧携带数据由发送器至接收器；远程帧通过总线单元发送，以请求发送具有相同标识符的数据帧；出错帧由检测出总线错误的任何单元发送；超载帧用于提供当前的和后续的数据帧的附加延迟。

1、CAN2.0A标准帧

CAN标准帧有11个字节，分为两部分：信息部分（前3个字节）和数据部分（后8个字节）。报文标识符（CAN-ID）为11位，有效数据最大长度为8个字节。具体格式如表2-1所示。

其中，FF(Frame Format)表示帧格式，FF=0为标准帧格式，FF=1为扩展帧格式；RTR（Remote Transmission Request）表示远程发送请求位，RTR=0为数据帧，RTR=1为远程帧。

2、CAN2.0B扩展帧

CAN扩展帧和标准帧格式对比，扩展格式除标识符长度多了两个字节，其它部分均相同。CAN扩展帧有13个字节，前5个字节为信息部分，后8个字节为数据部分。表2-2为扩展帧格式。

表2-1　CAN2.0A标准帧格式

	7	6	5	4	3	2	1	0
Bytes0	FF=0	RTR	×	×	DLC(数据长度)			
Bytes1	(报文标识符) ID.10~ ID.3							
Bytes2	ID.2 ~ ID.0			×	×	×	×	×
Bytes3	数据字节1							
Bytes4	数据字节2							
Bytes5	数据字节3							
Bytes6	数据字节4							
Bytes7	数据字节5							
Bytes8	数据字节6							
Bytes9	数据字节7							
Bytes10	数据字节8							

CAN报文按帧格式分为标准帧和扩展帧。标准帧节省了2个字节，而采用29位标识符的扩展帧在组网的灵活性、方便性和扩充性等方面更优越。

表2-2　CAN2.0B扩展帧格式

	7	6	5	4	3	2	1	0
Bytes0	FF=1	RTR	×	×	DLC(数据长度)			
Bytes1	(报文标识符) ID.28 ~ ID.21							
Bytes2	ID.20 ~ ID.13							
Bytes3	ID.12 ~ ID.5							
Bytes4	ID.4 ~ ID.0					×	×	×
Bytes5	数据字节1							
Bytes6	数据字节2							
Bytes7	数据字节3							
Bytes8	数据字节4							
Bytes9	数据字节5							
Bytes10	数据字节6							
Bytes11	数据字节7							
Bytes12	数据字节8							

2.1.2 CAN 总线的特点

CAN总线的数据通信具有突出的可靠性、实时性和灵活性。其特点可概括为：

（1）CAN为多主方式工作，通信方式灵活，网络上任一节点均可在任意时刻主动地向网络上其他节点发送信息，而不分主从，且无需站地址等节点信息；

（2）在报文标识符上，CAN网络上的节点信息分成不同的优先级，可满足不同的实时要求，传输时延得到保证；

（3）CAN采用非破坏性的基于优先级的总线仲裁技术。当总线空闲呈隐性电平时，任何一个节点都可以向总线发送一个显性电平作为一个帧的开始。如果有两个或两个以上的节点同时发送，就会产生总线冲突。CAN的总线仲裁技术对标识符按位进行。各发送节点在向总线发送电平的同时，也对总线上的电平进行读取，并与自身发送的电平进行比较，如果电平相同则继续发送下一位，不同则停止发送，退出总线竞争，优先级最高的节点获得了总线的使用权；

（4）CAN中的报文滤波技术实现了多种传送和接收数据方式（点对点、一点对多点及全局广播等）的选择和转换，而无需专门调度；

（5）CAN的直接通信距离最远可达10km（速率5kbps以下）；通信速率最高可达1Mbps（此时通信距离最长为40m）；

（6）CAN总线通信格式采用短帧结构，传输时间短，受干扰概率低，从而保证了通讯的实时性；

（7）CAN的每帧信息都有CRC校验及其他检错措施，保证了数据通信的可靠性；

（8）CAN的通信介质可选择双绞线、同轴电缆或光纤，选择十分灵活；

（9）CAN节点在严重错误的情况下，具有自动关闭输出的功能，以使总线上其他节点的操作不受影响，而且发送的信息遭到破坏后，可以自动重发。[2][4]

2.1.3 CAN 总线的不足及其上层协议的介绍

实际中，在执行基于CAN的分布式系统时，除了基本的数据链路层服务之外，还要求或希望有更多的功能，如发送长于8字节的数据块，响应或确定数据传送，标识符分配，网络启动或监控节点。由于这些附加的功能直接支持应用过程，所以它可以被认作“应用层”。

特别在工业自动化应用中，越来越需要一个开放、标准化的较高层协议，这个协议支持不同生产厂家设备的互用性和可交换性。因此，要求有标准设备模型的规范,即基本功能性的“标准设备”和“标准应用”的规范，以作为对标准化应用层的补充 。

工业应用中，主要代表开放式分布系统的标准是CANopen、DeviceNet和SDS。开放式分布系统标准的工业应用包括工业自动化中由工业器件(传感器、执行器、控制器、人机接口）组成的底层网络。这种应用主要要求有：可配置、灵活性和可扩展性。为了保持生产厂商的独立性，必须以“设备子协议（Device Profile）”的形式定义器件的功能性。因此，哪种类型的通讯系统解决方案提供了一个完整的通讯框架和系统服务、设备建模、以及设施，其目的是便于系统配置和设备参数化。

DeviceNet、CANopen和SDS这三种基于CAN的开放式系统已经趋于成熟，其标准已经形成。它们具有相同的功能性，虽然这三个标准采用的方法不尽相同。最明显的不同是信息标识符的使用。信息标识符的使用在SDS里是基于完整的预定义方式，而在CANopen里，信息标识符的使用是向系统设计者或综合者保持开放。DeviceNet 却有些限制。DeviceNet和SDS基于面向连接的观点，CANopen基于面向信息的观点。

DeviceNet和SDS的应用在美国普遍应用，而CANopen却盛行于欧洲。SDS在重型汽车、卡车及相关领域中得到了广泛的应用。幸运的是，所有的这些解决方案都基于相同的物理层和数据链路层(其中DeviceNet对于物理层的要求较高）。这就是说，当应用专有的总线接口时，只需要在软件方面下功夫。值得一提的是CANopen在故障----安全产品方面已经有了成功的应用(如轨道交通方面），CANopen在国内还善在探索阶段，也是本文选择CANopen作为研究对象的主要依据。

2.2 CANopen协议的剖析

2.2.1 CANopen 协议的简介

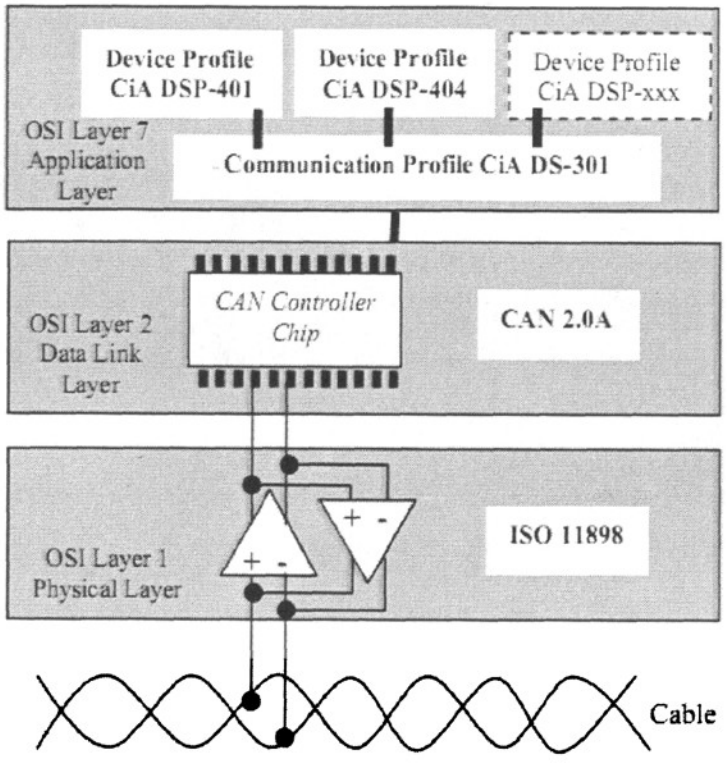


图2-2 CAN、CANopen标准在OSI网络模型中的位置示意图

CANopen协议是CAN-in-Automation(CiA)定义的标准之一，并且在发布后不久就获得了广泛的承认。尤其是在欧洲，CANopen协议被认为是在基于CAN的工业系统中占领导地位的标准。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、操作设备、控制器、可编程控制器或编码器，都在称为“设备描述”的协议中进行描述；“设备描述”定义了不同类型的标准设备及其相应的功能。依靠CANopen协议的支持，可以对不同厂商的设备通过总线进行配置。在OSI模型中，CAN标准、CANopen协议之间的关系如图2-2所示。[36]

2.2.2 CANopen 协议的技术规范

CANopen是在 CAL基础上开发的，使用了CAL通讯和服务协议子集，提供了分布式控制系统的一种实现方案。CANopen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。

CANopen的核心概念是设备对象字典（OD：Object Dictionary），在其它现场总线 （Profibus，Interbus-S）系统中也使用这种设备描述形式。

下面先介绍对象字典（OD：Object Dictionary），然后再介绍CANopen通讯机制。[36]

2.2.2.1 对象字典 OD

对象字典（OD：Object Dictionary）是一个有序的对象组；每个对象采用一个16位的索引值来寻址，为了允许访问数据结构中的单个元素，同时定义了一个8位的子索引，对象字典的结构参照表2-3。一个节点的对象字典的范围在0x1000到0x9FFF之间。

CANopen网络中每个节点都有一个对象字典。对象字典包含了描述这个设备和它的网络行为的所有参数。

一个节点的对象字典是在电子数据文档（EDS：Electronic Data Sheet）中描述或者记录在纸上。不必要也不需要通过CAN- bus“审问”一个节点的对象字典中的所有参数。如果一个节点严格按照在纸上的对象字典进行描述其行为，也是可以的。节点本身只需要能够提供对象字典中必需的对象（而在CANopen规定中必需的项实际上是很少的），以及其它可选择的、构成节点部分可配置功能的对象。

表2-3　CANopen对象字典通用结构

索引	对象
0000	Not used
0001 - 001F	静态数据类型(标准数据类型，如 Boolean, Integer 16)
0020 - 003F	复杂数据类型 (预定义由简单类型合成的结构体如PDOCommPar, SDOParameter)
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	Reserved
1000 - 1FFF	通讯子协议区域 (如设备类型，错误寄存器，支持的PDO数量)
2000 - 5FFF	标准商特定子协议区域
6000 - 9FFF	标准的设备子协议
A000 - FFFF	Reserved

CANopen由一系列称为子协议的文档组成。

通讯子协议（communication profile），描述对象字典的主要形式和对象字典中的通讯子协议区域中的对象，通讯参数。同时描述CANopen通讯对象。这个子协议适用于所有的CANopen设备。

还有各种设备子协议（device profile），为各种不同类型设备定义对象字典中的对象。目前已有5种不同的设备子协议，并有几种正在发展。

设备子协议为对象字典中的每个对象描述了它的功能、名字、索引和子索引、数据类型，以及这个对象是必需的还是可选的，这个对象是只读、只写或者可读写等等。

一个设备的通讯功能、通讯对象、与设备相关的对象以及对象字典的缺省由电子数据文档(EDS:Electronic Data sheet)中提供。

单个设备的对象配置的描述文件称作设备配置文件(DCF: Device Configuration File),它和EDS有相同的结构。二者文件类型都在CANopen规范中定义。

设备子协议定义了对象字典中哪些OD对象是必需的，哪些是可选的；必需的对象应该保持最少数目以减小实现的工作量。

可选项----在通讯部分和与设备相关部分----可以根据需要增加以扩展CANopen设备的功能。如果需要的项超过了设备子协议中可以提供的，在设备子协议中已预留有足够空间提供给厂商的特定功能使用。

对象字典中描述通讯参数部分对所有CANopen设备（例如在OD中的对象是相同的，对象值不必一定相同）都是一样的。对象字典中设备相关部分对于不同类的设备是不同的。

2.2.2.2 CANopen 通讯模式

CANopen通讯模型定义了4种报文（通讯对象）：

1、管理报文

- 层管理，网络管理和ID分配服务：如初始化，配置和网络管理（包括：节点保护）。
- 服务和协议基于主从通讯模式：在CAN网络中，只能有一个主节点以及一个或多个从节点。

2、服务数据对象SDO(Service Data Object)

- 通过使用索引和子索引（在CAN报文的前几个字节），SDO使客户机能够访问设备（服务器）对象字典中的项（对象）。
- SDO通过CAL中多元域的CMS对象来实现，允许传送任何长度的数据（当数据超过4个字节时分拆成几个报文）。
- 协议采用确认服务类型：为每个消息生成一个应答（一个SDO需要两个ID）。SDO请求和应答报文总是包含8个字节（没有意义的数据长度在第一个字节中表示，第一个字节携带协议信息）。SDO通讯有较多的协议规定。

3、过程数据对象PDO（Process Data Object）

- 用来传输实时数据， 数据从一个生产者传到一个或多个消费者。数据传送限制在1到8个字节（例如，一个PDO可以传输最多64个数字I/O值，或者4个16位的AD值）。
- PDO通讯没有协议规定。PDO数据内容只由它的CAN-ID定义，假定生产者和消费者知道这个PDO的数据内容。
- 每个PDO在对象字典中用2个对象描述：
  - PDO通讯参数：包含哪个COB-ID将被PDO使用，传输类型，禁止时

间和定时器周期。

- PDO映射参数：包含一个对象字典中对象的列表，这些对象映射到PDO里，包括它们的数据长度。生产者和消费者必须知道这个映射，以解释PDO内容。

- PDO消息的内容是预定义的（或者在网络启动时配置的）：

映射应用对象到PDO中是在设备对象字典中描述的。如果设备（生产者和消费者）支持可变PDO映射，那么使用SDO报文可以配置PDO映射参数。

- PDO可以有多种传送方式：
  - 同步（通过接收SYNC对象实现同步）
    - 非周期：由远程帧预触发传送，或者由设备子协议中规定的对象特定事件预触发传送。
    - 周期：传送在每1到240个SYNC消息后触发。
  - 异步
    - 由远程帧触发传送。
    - 由设备子协议中规定的对象特定事件触发传送。

表2-4给出来了由传输类型定义的不同PDO传输模式，传输类型为PDO通讯参数对象的一部分，由8位无符号整数定义。

表2-4 PDO传输类型定义

传输类型	触发PDO的条件 (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 - 250	-	-	-	Reserved
252	B	B	-	异步，在RTR之后
253	-	O	-	异步，在RTR之后
254	-	O	O	异步，制造商特定事件
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 - 例如数值改变或者定时器中断。
- 传输类型为：1到240时，该数据代表两个PDO之间的SYNC对象的数目。

- 一个PDO可以指定一个禁止时间，即定义两个连续PDO传输的最小间隔时间，避免由于高优先级信息的数据量太大，始终占据总线，而使其它优先级较低的数据无力竞争总线的问题。禁止时间由16位无符号整数定义，单位100us。
- 一个PDO可以指定一个事件定时周期，当超过定时时间后，一个PDO传输可以被触发（不需要触发位）。事件定时周期由16位无符号整数定义，单位1ms。
- PDO数据传送没有上层协议，而且PDO报文没有确认（一个PDO需要一个CAN-ID）。每个PDO报文传送最多8个字节（64位）数据。

4、预定义报文或者特殊功能对象

- 同步（SYNC）
  - 在网络范围内同步（尤其在驱动应用中）：在整个网络范围内当前输入值同时保存，随后传送（如果需要），根据前一个SYNC后接收到的报文更新输出值。
  - 主从模式：SYNC主节点定时发送SYNC对象，SYNC从节点收到后同步执行任务。

- 在SYNC报文传送后，在给定的时间窗口内传送一个同步PDO。
- CANopen建议用一个最高优先级的COB-ID以保证同步信号正常传送。SYNC报文可以不传送数据以使报文尽可能短。

- 时间标记对象（Time Stamp）
  - 为应用设备提供公共的时间帧参考。
- 紧急事件（Emergency）
  - 设备内部错误触发。
- 节点/寿命保护（Node/Life guarding）
  - 主从通讯模式。
  - NMT主节点监控节点状态：称作节点保护（Node guarding）。
  - 节点也可以(可选择)监控NMT主节点的状态:称作寿命保护(Life guarding）。当NMT从节点接收到NMT主节点发送的第一个Node Guard报文后启动寿命保护。
  - 检测设备的网络接口错误（不是设备自身的错误）：通过应急指示报告。
  - 根据NMT节点保护协议实现： NMT主节点发送远程请求到一个特定节点，节点给出应答，应答报文中包含了这个节点的状态。
- Boot-UP
  - 主从通讯模式。
  - NMT从节点通过发送这个报文，向NMT主节点说明该节点已经由初始化状态进入预操作状态。

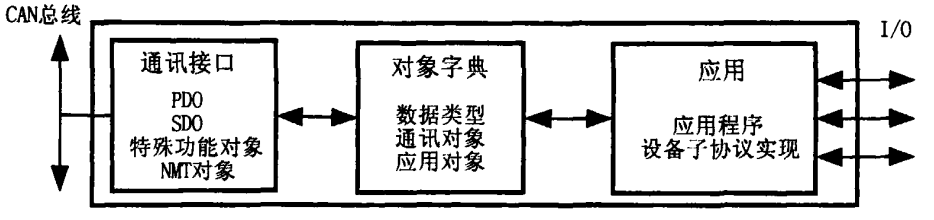


图2-3　CANOpen设备模型

上面提到的通讯对象类型中有两个对象用于数据传输。它们采用二种不同的数据传输机制实现：

- SDO 用来在设备之间传输大的低优先级数据，典型的是用来配置CANopen网络上的设备。
- PDO 用来传输8字节或更少数据，没有其它协议预设定（意味着数据内容已预先定义）。

一个CANopen设备必须支持一定数量的网络管理服务（管理报文，administrative messages），需要至少一个SDO。每个生产或消费过程数据的设备需要至少一个PDO。所有其它的通讯对象是可选的。一个CANopen设备中CAN通讯接口、对象字典和应用程序之间的联系如图2-3所示。

2.2.2.3 CANopen 预定义连接集

为了减小简单网络的组态工作量，CANopen定义了强制性的缺省标识符（CAN-ID）分配表。这些标志符在预操作状态下可用，通过动态分配还可修改它们。CANopen设备必须向它所支持的通讯对象的提供相应的标识符。

缺省ID分配表是基于11位CAN－ID，包含一个4位的功能码部分和一个7位的节点ID(Node-ID)部分。如图2-4所示。

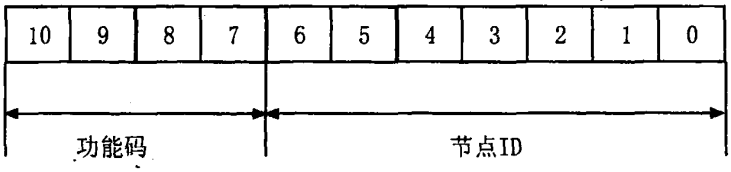


图2-4 预定义连接集ID

Node-ID由系统集成商定义，例如通过设备上的拨码开关设置。Node-ID范围是1--127（0不允许被使用）。

预定义的连接集定义了4个接收PDO（Receive PDO），4个发送PDO（Transmit PDO），1个SDO（占用2个CAN-ID），1个紧急对象和1个节点错误控制(Node Error Control)ID。也支持不需确认的NMT Module Control服务，SYNC和Time Stamp对象的广播。

缺省ID分配表如表表2-5所示。

表2-5 CANopen预定义主/从连接集CAN标识符分配表

CANopen预定义主/从连接集的广播对象			
对象	功能码 (ID-bits 10-7)	COB-ID	通讯参数在OD中的索引
NMT Module Control	0000	000H	-
SYNC	0001	080H	1005H, 1006H, 1007H
TIME STAMP	0010	100H	1012H, 1013H
CANopen主/从连接集的对等对象			
对象	功能码 (ID-bits 10-7)	COB-ID	通讯参数在OD中的索引
紧急	0001	081H-0FFH	1024, 1015H
PDO1(发送)	0011	181H-1FFH	1800H
PDO1(接收)	0100	201H-27FH	1400H
PDO2(发送)	0101	281H-2FFH	1801H
PDO2(接收)	0110	301H-37FH	1401H
PDO3(发送)	0111	381H-3FFH	1802H
PDO3(接收)	1000	401H-47FH	1402H
PDO4(发送)	1001	481H-4FFH	1803H
PDO4(接收)	1010	501H-57FH	1403H
SDO(发送/服务器)	1011	581H-5FFH	1200H
SDO(接收/客户)	1100	601H-67FH	1200H
NMT Error Control	1110	701H-77FH	1016H-1017H

注意：

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

2.2.2.4 CANopen 标识符分配

ID地址分配表与预定义的主从连接集（set）相对应，因为所有的对等ID是不同的，所以实际上只有一个主设备(知道所有连接的节点ID)能和连接的

每个从节点（最多127个）以对等方式通讯。两个连接在一起的从节点不能够通讯，因为它们彼此不知道对方的节点ID。

CANopen网络中CAN 标识符（或COB-ID）分配3种不同方法：

- 使用预定义的主从连接集。ID是缺省的，不需要配置。如果节点支持，PDO数据内容也可以配置。
- 上电后修改PDO的ID（在预操作状态），使用（预定义的）SDO在节点的对象字典中适当位置进行修改。
- 当网络初始化完毕，并且启动后，主节点首先通过“Connect_Remote_Node”报文（NMT报文）和每个连接的从设备建立一个对话。一旦这个对话建立，CAN通讯ID服务分配好，这需要节点支持扩展的boot-up。

2.2.2.5 CANopen boot-up 过程

在网络初始化过程中，CANopen支持扩展的boot-up，也支持最小化boot-up过程。

扩展boot-up是可选的，最小boot-up则必须被每个节点支持。两类节点可以在同一个网络中同时存在。

如果使用CAL的DBT服务进行ID分配，则节点必须支持扩展boot-up过程。

可以用节点状态转换图表示这两种初始化过程，如图2-5所示。扩展boot-up的状态图在预操作和操作状态之间比最小化boot-up多了一些状态。

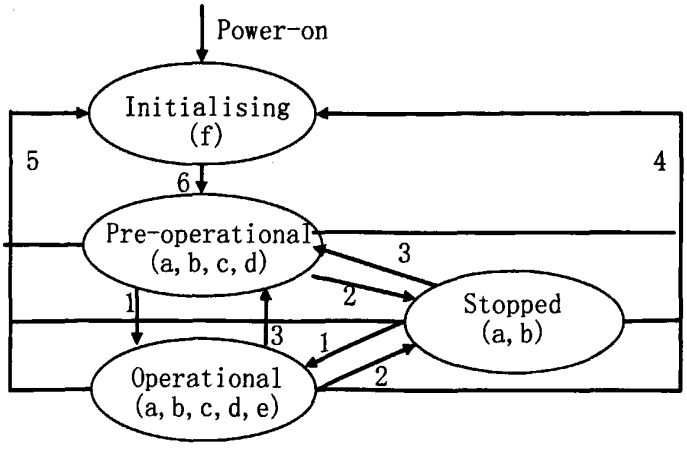


图2-5 CANopen最小化boot-up节点状态转换图

注意：

- 图3-3中括号内的字母表示处于不同状态那些通讯对象可以使用。

a.NMT ，b.Node Guard ，c.SDO ，d.Emergency ，e.PDO ，f.Boot-up

- 状态转移（1－5由NMT服务发起），NMT命令字（在括号中）：

1：Start_Remote_node (0x01)

2：Stop_Remote_Node (0x02)

3：Enter_Pre-Operational_State (0x80)

4：Reset_Node (0x81)

5：Reset_Communication (0x82)

6：设备初始化结束，自动进入Pre_Operational状态，发送Boot-up消息

在任何时候NMT服务都可使所有或者部分节点进入不同的工作状态。NMT服务的CAN报文由CAN头(COB-ID=0)和两字节数据组成：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（此时寻址所有节点）。

仅支持最小化boot-up的设备叫最小能力设备。最小能力设备在设备初始化结束后自动进入预操作1状态。在这个状态，可以通过SDO进行参数配置和进行COB-ID分配。

设备进入准备状态后，除了NMT服务和节点保护服务（如果支持并且激活的话）外，将停止通讯。

2.2.3 CANopen 协议的特点

（1）基于CAN总线的CANopen网络通讯具有以下特点：

- 缺省的ID分配表，由于CAN总线是用ID决定在网络上发送的优先级别，所以是很重要的。
- 使用对象字典（OD：Object Dictionary）对设备功能进行标准化的描述。提供了一套数据的管理方式。采用了主索引与子索引的方式，提高了数据访问的速度。全网络提供统一的错误标识码。
- 通过NMT帧来管理网络的运行，包括主对从的管理、启动、新加入的节点，如何加入系统。为了系统稳定的运行，还提供了心跳帧及其节点保护。
- 数据交换使用PDO、SDO交换数据。其中的SDO提供了一套发送大于8

个字节数据的机制。PDO主要实现了实时数据的发送。

- 在实时的实现方面，采用了同步操作和PDO来实现，还有用了远程帧触发，应急报文的发送等。

（2）基于CAN总线的CANopen网络通讯的实现需完善的内容：

- 缺乏一套针对冗余的实现机制。
- 心跳、节点保护等周期的选取细则需要开发者自行设计。
- 实时时钟的实现。
- 发送调度算法。

第3章 CANopen协议在冗余系统中的应用技术研究

3.1 网络控制系统的设计

3.1.1 网络控制系统设计的一般步骤

网络控制系统设计不仅指建立网络,而且包括补充新节点或扩充网段。网络设计还包括对现存网络进行定期优化。一个大型的网络控制系统的设计过程是一个复杂的分析、模块化及集成的过程。一般而言,一个正常的网络控制系统设计过程应包括以下几点。

3.1.1.1 需求分析

(1) 现状分析:现状、人员现状、资金状况、节点数目、地理分布、业务特点、数据流量和流向,现有软件和使用情况以及通信线路情况等。

(2) 功能需求和性能要求:希望达到的功能、所需存储容量、响应时间、信息传送的误码率、将来的扩充等。

(3) 成本/效益分析:对建立网络系统的人力、财力、物力的投入与可能产生的经济、社会效益进行分析。

(4) 书写需求分析报告。

需求分析任务应该由用户方人员和设计人员共同来承担,二者缺一不可。实际上,用户需求在网络系统设计中占有举足轻重的作用。可以说,一个好的用户需求报告意味着这一系统已经成功了一半。相反,如果需求分析不详尽、不到位、配备再好的设备,选用再好的系统也无济于事。如果把设计网络系统看成是盖一座大楼,用户需求分析就相当于地基,这一点是许多人往往重视不够的。

3.1.1.2 系统初步设计

(1) 确定网络的规模和应用范围:确定网络覆盖范围、定义网络应用的

边界。

（2）统一建网模式:确定网络的总体框架,比如是集中式还是分布式,采用主从方式还是其它方式等。

3.1.1.3 网络系统详细设计

(1) 网络协议体系结构的确定:根据应用需求,确定用户端系统应该采用的网络类型,是否采用中继系统等,并确定整个网络应该采用的协议体系结构。

(2) 节点规模设计:确定网络主要节点设备的大小和应具备的功能。

(3) 选定通信介质:为用户端系统选定传输介质,为中继系统选定传输资源。

(4) 结构化布线设计:对网络传输电缆进行规范的结构化布线。

(5) 确定详细设计方案:确定网络总体及各部分的详细设计方案,并形成正式文档。

3.1.1.4 计算机系统及应用系统设计

(1) 计算机系统设计:对整个系统的服务器、工作站、终端以及打印机等外设进行配置和设计。

(2) 应用系统设计:确定应用系统的框架和对网络系统的要求。

(3) 硬件设备的选型和配置:根据网络系统方案,选择性能价格比最优的硬件设备,并加以有效的组合。

(4) 系统软件选择:为计算机系统选择应该采用的数据库系统、管理系统以及开发平台。

(5) 机房环境设计:确定用户端系统的服务器所在机房以及一般工作站机房环境,包括温度、湿度等要求。

(6) 确定系统集成详细方案:将整个系统设计的各个部分加以集成,并最终形成系统集成的正式文档。[18]

3.1.2 网络控制系统设计应遵循的原则

网络控制系统技术复杂、涉及面广，为了使整个网络系统更合理、更经济、性能良好，在它的设计过程中应该遵循以下设计原则。

3.1.2.1 整个网络应具有良好的性价比

一方面，要保护现有的硬件资源，对某些应用状况良好的应用软件，也要加以保护，并尽可能利用公用和现有的通信资源。另一方面，由于网络设备更新换代的周期比较短，应用需求的变化也比较频繁，因此在网络设计时，应尽可能防止系统迅速淘汰。

3.1.2.2 实用性和先进性

设计网络系统时首先应该以注重实用和成效为原则，紧密结合具体应用的实际需要，在技术上应该具有世界先进水平，选用的设备应该是技术成熟和实用效果好、市场占有率高、通用性好的设备。

3.1.2.3 开放性和可扩充性

为适应需求的发展变化，网络系统应该具备良好的开放性和可扩充性，以保证网络节点的增加、数量的增长、网络延伸距离的扩大及多媒体应用。比如应该采用标准的结构和协议，采用具有良好扩充性的网络设备和网络拓扑，采用结构化布线的方法等。

3.1.2.4 高可靠性和稳定性

在网络设计时，不论是网络节点、通信线路，还是网络拓扑的设计，都应该对可靠性加以考虑。比如选用设备应该充分考虑冗余、容错能力，在万一出现局部故障时应不影响系统其他部分的正常运行，并且故障易于诊断和排除。

3.1.2.5 安全性

所建立的计算机网络系统应能保证各种数据的完整性、安全性的要求，以及对整个网络的安全性要求。比如应该采用具有良好网络安全性的网络设备和网络操作系统，具有较小误码率和较好抗干扰能力的通信线路，采用一定的加密措施等。

3.1.2.6 可维护性

整个网络应具有良好的可维护性，不仅要保证整个网络系统设计的合理性，还应该配置相关的检测设备和网络管理设施。[18]

3.1.3 本系统设计中的注意事项

为与CANopen通讯协议和相应的设备子协议保持一致，以使制造商的产品能够用于任何CANopen网络，以下3种层次的兼容性要求需要满足（对日益增长的设备兼容性的要求）：

- 一致性：

设备连接到CANopen网络后不能影响其他设备的通讯：应用层的一致性。

- 互用性：

设备能够同网络上的其它节点交换数据：通讯协议的一致性。

- 互换性：

设备能够代替另外一个同类设备：设备子协议的一致性。

一个CANopen设备至少应该具有（最小能力设备）：

- 一个节点ID。
- 一个对象字典（内容由设备功能决定）。
- 一个SDO，能够访问对象字典中必需的对象（只读）。
- 支持下列NMT从设备服务：

Reset_Node，Enter_Preoperational_State，Start_Remote_Node，Stop_Remote_Node，Reset_Communication。

- 缺省的标识符的分配。

3.2 系统实现关键技术研究之——冗余

3.2.1 冗余的概述

在实际生产过程中，一些机电设备或者生产线往往要求24小时连续生产运行而不能停顿，在这种条件下即使可靠性在高的控制系统也不能保证故障为零。在要求“绝对”可靠的地方，容错控制技术都将有一席之地，如核电站、战略武器、大型地面电站、24小时连续运转设备等。这些地方的工作系统一旦失效，不仅造成人力、物力损失，有些使之产生灾难性的后果。要提高系统的可靠性所限应尽量提高系统元器件本身的可靠性，这是提高系统整体可靠性的基础，但这有一定的限度。为了更现住地提高系统的可靠性，是系统在产生故障的情况下仍能继续工作，就必须采用冗余控制技术，冗余控制技术主要建立在“冗余”的设计上，即利用资源的冗余来提高系统的可靠性。当系统中的某一套装置发生问题时，通过故障监控予以切除或隔离，然后将完好装置接入系统，使系统在不停顿的情况下继续工作，因此，冗余控制成了一种满足连续生产要求、提高系统可靠性的有效手段。

容错控制的根本特征是当控制系统发生故障时，系统仍然能够维持自身运行在安全状态，并尽可能地满足一定的性能指标要求。从容错控制的对象来看，由于构成一个控制系统的典型环节包括控制器、检测元件或传感器、执行器以及被控对象，所以对不同环节的考虑理论上都可衍生出特定的容错控制问题。本节主要研究被控制对象本身的容错控制，使得被控制对象能够适应它的参数在一定范围内变化，甚至适应结构化变化。

本节首先给出了智能容错控制系统的基本结构及原理，从多模冗余智能容错控制方案及配置等方面分析了之智能容错控制技术。给出了多模冗余控制系统的可靠性模型，并分析了他们的各项可靠性指标。[49]

3.2.2 智能容错控制的基本结构及原理

一般来说，“错”可以分为两类:第一类是先天性的固有错，如元器件生产过程中造成的错或线路与程序在设计过程中产生的错。这一类错只对其拆除更改或改正，是不能容忍的。第二类错是后天性的，是由于系统在运行中

产生的缺陷所导致的故障。故障有永久性的、瞬间性的及间歇性故障的区别。对瞬间性及间歇性的故障，不能采取检测定位等措施（因为可能在检测就不见了，但过一段时间又会出现），但可以考虑随机地消除其作用，使其不影响设备的正常运转。由于瞬间性故障占全部故障的大多数，它成为容错技术的主要对象。对运行中产生的永久故障，也可暂时消除其影响，但根本的办法仍是检测出它的存在。最好诊断出它存在的方位或部件，将有关部件或分系统切换、修理。如任其存在，就会使它与另一故障（永久性或瞬间性）相遇而构成双重故障，造成灾难性故障。因此，一个永久性故障的延续存在，能使系统从单故障中自动恢复的概率大为降低。另外一点要说明的是，容错控制技术所考虑的并加以补救处理的错主要是单独地、孤立地存在的错。

容错控制的目的在于针对不同的故障和故障采集，采取相应的容错处理措施，对故障进行消除、修复或隔离，以保证设备继续安全可靠运行，或以牺牲性能损失为代价，保证设备在规定时间内完成其基本功能。智能容错控制的基本结构如图3-1所示。智能容错控制系统一般由硬件和软件两部分组成。

硬件主要由信号检测传感器、二次仪表、接口板和计算机等组成。传感器检测的信号经放大和多路切换开关送入A/D卡，将模拟信号转换成数字信号。该信号由计算机进行分析和处理，分离出故障信息，并进行显示、报警和记录。对于可控故障信号可有计算机送入容错控制器，再通过相应的执行机构对故障实现容错控制。

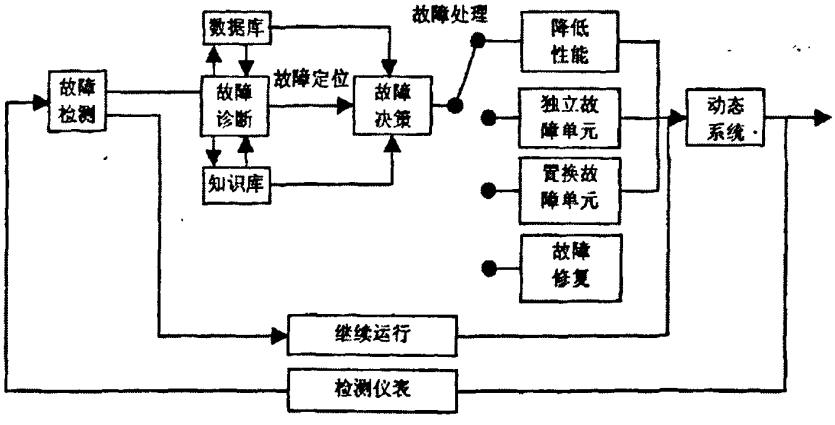


图3-1 智能容错控制的基本结构

系统运行分两种模式。当没有故障发生时，故障检测和容错控制模式之起监视和预测作用，并将设备的运行状态和检测参数以图形和数据形式进行

显示。当出现故障或即将出现故障时，容错控制器才正式投入工作，并调用相应的容错子模块，保证设备在固定时间内完成其规定功能。系统的特点是能有效地提高设备的安全性和可靠性。[49]

3.2.3 基于多模冗余的智能容错控制技术

冗余控制的概念，严格来讲是采用一定或成倍量的设备或元器件的方式组成控制系统来完成控制。当某一设备或元器件发生故障而损坏时，它可以通过硬件、软件或人为方式，相互切换作为后备或元器件，代替因故障而损坏的设备或元器件，保持系统正常工作，使控制设备因意外而导致的停机所示降到最低。

多模冗余结构中通常包含有两个或两个以上结构完成相同的模块，而且其中任意一个模块都能独立完成系统所要求的功能。因此这种结构具有抑制或校正整体性故障的能力，即一个故障模块仅靠自身的调节补偿能力已不能恢复或维持它应有的能力，只有用其它一直无故障的模块去替换它才能使系统正常工作。当然多模冗余结构对抑制局部性故障也是适用的。在实际应用中局部性故障与整体性故障之间在各种不同的系统中有不同的界限，二者之间没有严格的鸿沟。[49]

3.2.3.1 多模冗余系统的分类

冗余控制的方式在工控领域根据不同的产品和客户不同的需求有多种多样，采用的方式也不尽相同。

按照冗余模件数一般分为：①双模冗余，即主-备式冗余方式，该方式由两台相同的系统组成，当主系统运行时，备用系统处于热储备状态，当主系统发生故障时，备用系统立即接入主机，代替主系统进行工作；②三模冗余；③四模冗余。在工业系统冗余容错系统中双模、三模冗余应用较普遍。

按冗余的实现方式来分大致分为：①硬冗余，即采用特殊的硬件模块故障切换的冗余方式；；②软冗余，即采用编程的方式来实现故障切换的冗余方式。

按冗余的切换方式来分大致分为：①热冗余，即硬件冗余方式，当主设备故障时，通过特定硬件判别、备份方式无缝地自动切换到备用设备上，保

持系统正常运行；②暖冗余，即软冗余方式，主要通过编程方式来实现冗余。由于软冗余的实现受多方因素制约，系统切换的时间较硬件冗余稍长，因此部分软冗余可能会使主设备在发生切换时有间隙或需要人为简单干预或预备才能得以完善；③冷冗余，即一套或部分冗余的设备不通电、不工作们准备待命（人为预置好）。当主设备故障时需要人工恢复系统运行。按照现在的严格定义，这种方式，并不算冗余，只作备份理解。这种冗余一般应用与实时性不强、工艺连续性要求不高的场合。[49]

3.2.3.2 多模冗余智能容错控制方案

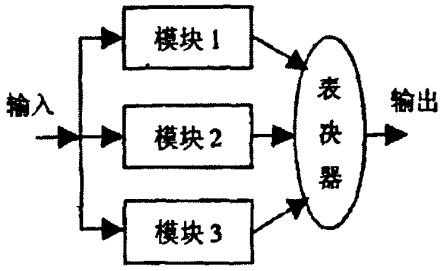
一般地，多模冗余智能容错控制常用方案有三种，即静态冗余、动态冗余和混合冗余。

1、静态冗余容错控制

静态容余容错控制系统的输入同时作用于三个采用同一设计、同一工艺制造出的模块，三个模块的运行结果同时送到表决器，而表决器则通过表决，以多数结果为系统的最终结果，故障模块的运行结果就被正确模块的运行结果所屏蔽。这种容错控制方案的结构如图3-2所示。其主要特点为：(1)由于故障被屏蔽，所以不需要检测和识别故障；(2)容易与无冗余系统进行转换；(3)所有模块都消耗能量。

2、动态冗余容错控制

动态冗余容错控制系统有主模块和N个备用模块，但任何时候都只有一个模块在运行。系统的输入首先要通过转换装置，转换装置通过系统输出前的检查装置来决定系统输入接入主模块还是备用模块。若当前工作模块出现故障，则检测装置将出错信息传送给转换装置，装换装置则负责将备用模块替代故障模块。这种容错控制方案的结构如图3-3所示。该方案特点为：(1)仅有一个模块消耗能量；(2)模块数目可随任务而改变，不会影响系统工作；(3)转换装置和检测装置中任意故障都会导致系统失效。



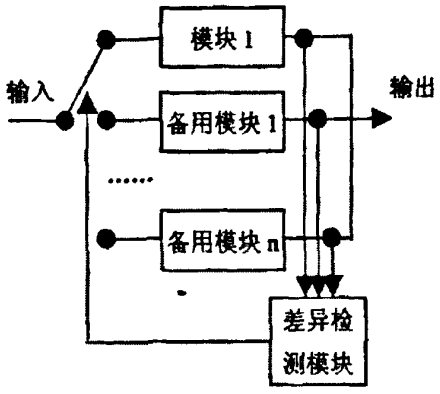


图3-2 静态冗余控制方案（三模冗余）图3-3 动态冗余控制方案

3、混合冗余容错控制

混合冗余容错控制系统总有N个相同模块同时工作，系统输出同时到达这N个模块，这些模块中的K个模块的输出通过表决器表决，以多数结果作为系统的表决结果。系统中另外的N-K个模块作为表决系统中模块的备份。当参与表决的模块总有一个故障模块时，位于表决器之前的转换开关负责切换表决器输入，以一个备份模块的输出作为表决器的一个输入、故障模块的输出。这种容错控制方案的结构如图3-4所示。该方案的特点为：综合了静态冗余与动态冗余的优势。

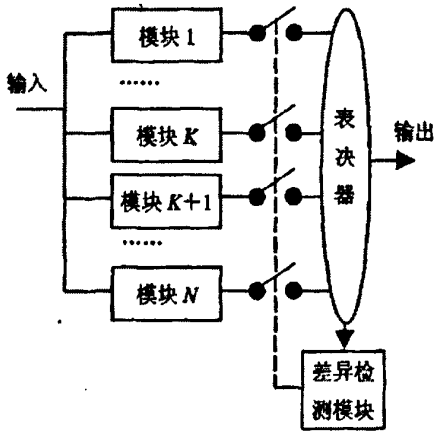


图3-4 混合冗余控制方案

3.2.3.3 冗余度的配置

冗余控制系统设计的目的，除了提高安全可靠性、实现特定的冗余度等级外，同时要兼容基本可靠性。因此，为了满足上述要求而达到较高的故障检测覆盖率及降低技术水平，将是冗余度配置要解决的问题。冗余度配置方案的确定，主要考虑以下三个问题：

1、冗余度的选取

目前无论是部分冗余还是系统冗余，大多数采用双、三和四冗余度方案。选取冗余度的多少，要从系统的可靠性指标要求出发，并兼顾系统体积、重量、费用和冗余管理方式的要求。设计人员的任务是以最小的余度数满足规定的可靠性指标和工作等级。

2、表决、监控面的设置

设置的原则是：首先，应该满足系统指标要求，将余度系统分为若干级，成为分级冗余形式，使生存通道增加，从而提高整个系统的可靠性；但表决、监控面也不宜过多，检测部件的增加，会使可靠性的提高受到限制，因此需要斟酌适当的选取。其次，要考虑满足信号一致性的要求。另外，还要考虑满足控制率重构的要求。为了保证系统出现故障时，能够采取系统控制率重构，必须对传感器进行表决。这对多模态控制的实现也是必不可少的。

3、信号传递方式

冗余控制系统中，各个部件除了前后连接外，通道还需要相互交换信息。通道部的连接方式是与信号的表决、监控设备密切相关。[49]

3.2.4 基于多模冗余智能容错系统的可靠性分析

3.2.4.1 可靠性指标

可靠性指机器、零件或系统，在规定的工作条件下，在规定的时间内具有正常工作性能的能力。衡量可靠性的指标，常用的是可靠度、平均故障时间MTBF、平均寿命MTTF及故障率，先简述如下：

可靠度：指机器设备在规定条件下和规定时间内完成对应功能的概率，即无故障正常工作状态的概率。可靠度用R(t)表示，即R(t)=P（X>t），它是

规定时间t的函数，规定时间越长，R(t)越小。

平均无故障间隔时间MTBF（Mean Time Between Failures）:是可以边修理边使用的机器、零件或系统，相邻故障期间的正常工作时间的平均值。

平均故障前时间（MTTF--Mean Time To Failures）:指不能修理的机器、零件或系统，至发生故障位置的工作时间平均值，即指不可修改产品的平均寿命。

故障率（Failure Rate）:通常指瞬时故障率，又称失效率、风险率。即指工作到某时刻尚未发生故障的产品，在单位时间内发生故障的概率，用$\lambda(t)$表示。

根据统计试验获得可靠度R(t)的方法是:取N个同样的单元，在同样的规定条件下，从t=0时刻开始试验，设到t时刻有$N_0(t)$个产品未发生故障，N_F（$=N-N_0$）个单元失效。当$N\longrightarrow\infty$，有

$$R(t)\approx N_0(t)/N$$

可靠度具有下列性质：

① $R(0)=1$;

② $\lim_{t\longrightarrow\infty} R(t)=0$;

③ $0\leq R(t)\leq 1$;

④ R（t）是时间t的单调递减函数。选取不同的t，可得R(t)随时间变化的曲线。设X是该单元能正常使用的期限，即从开始使用到第一次故障的时间为止的时间间隔。则X是随机变量，设它的分布函数为F(t)，分布密度函数f(t)，则有：

$$R(t)=1-F(t)$$

根据定义，平均故障间隔时间MTBF有：

$$MTBF=\int_0^\infty tdF(t)=\int_0^\infty tf(t)dt=\lim_{t\longrightarrow\infty}tR(t)+\int_0^\infty R(t)dt \qquad (3.1)$$

根据可靠度R(t)性质，则上式的第一项为零，故有

$$MTBF=\int_0^\infty R(t)dt \qquad (3.2)$$

而对故障率则有

$$\lambda(t)=f(t)/R(t) \tag{3.3}$$

$$\because R(t)\approx N_0(t)/N$$

$$\frac{dR(t)}{dt}\approx\frac{1}{N}\frac{dN_f(t)}{dt}$$

$$\therefore \lambda(t)\approx\frac{1}{N_0(t)}\frac{dN_f(t)}{dt} \tag{3.4}$$

上式表明，故障率$\lambda(t)$等于t时刻后的单位时间内失效单元数与时刻仍有效的单元数之比。

系统由特定功能的、相互间具有有机联系的和一定逻辑关系的许多单元构成的一个整体。根据系统中各个单元或子系统间的逻辑关系，可以把系统分为串联系统、并联系统、串——并联混合系统、表决系统、旁路系统等。

在下面多模冗余系统可靠性模型的分析中，假设第i个部件的寿命为x_i，可靠度为R_i=P（x_i>t）（i=1, 2, 3, …, n）假定x_1，x_2，…，x_n为随机变量且相互独立，所有部件的故障度服从指数分布，即

$$R_i(t)=e^{-\lambda_i t} \tag{3.5}$$

式中λ_i为第i个部件的故障率。

3.2.4.2 并联系统系统可靠性模型

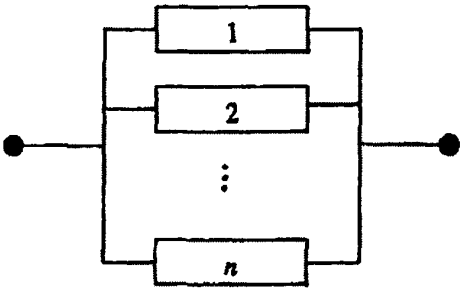


图3-5 并联系统

只有当所有部件都失效时才能把此系统称为并联系统。其功能逻辑关系图如图3-5所示。它表明任何一个部件有效（畅通），系统就有效（路线可通过）。

显然旁路系统的寿命为：

$$X_s = \max\{X_1, X_2, ..., X_n\}$$

系统的可靠度为：

$$R_s = P\{X_s > t\} = P\{\max(x_1, x_2, ..., x_n) > t\} = 1 - P\{\max(x_1, x_2, ..., x_n) \le t\}$$

$$= 1 - P\{x_1 \le t, x_2 \le t, ..., x_n \le t\} = 1 - \prod_{i=1}^{n}[1 - R_i(t)] \qquad (3.6)$$

当 $R_i(t) = e^{-\lambda_i t}$，i=1, 2, …, n, 且 $\lambda_1 = \lambda_2 = \cdots\cdots = \lambda_n = \lambda$ 时

$$R_s(t) = 1 - (1 - e^{-\lambda t})^n \qquad (3.7)$$

上式表明并联系统的可靠度高于任何一个部件的可靠度。

系统的平均寿命为：$\mathrm{MTTF} = \int_0^{\infty} (1 - (1 - e^{-\lambda t})^n) dt = \sum_{i=1}^{n} \frac{1}{i\lambda}$ （3.8）

3.2.4.3 旁路系统系统可靠性模型

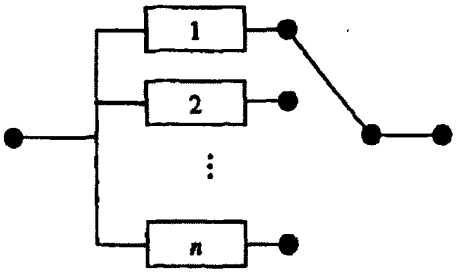


图3-6 旁路系统

为了提高系统的可靠度，除了多安装一些单元外，还可以储备一些单元，以便当工作单元失效时，能立即通过转换开关使储备的单元逐个地去替换，直到所有单元都发生故障时为止，系统才失效，这种系统称为旁路系统，其功能逻辑框图如图3-6所示。

在下面的分析中，假设：①储备单元完全可靠，即备用的单元在储备期

间内不发生失效也不劣化，储备期的长短对以后的使用寿命没有影响；②转换开关完全可靠，是指使用开关时，开关完全可靠，不发生故障。

若系统由n各单元组成，其中一个单元工作，n-1个单元备用，且第i个单元的寿命为x_i，其分布函数为$F_i(t)$，i=1, 2, …, n, 且相互独立；系统的工作寿命为X_s，故有$X_s=\sum_{i=1}^{n}x_i$，系统的可靠度为

$$R_s(t)=P\{X_S>t\}=P\{\sum_{i=1}^{n}x_i>t\}=1-P\{\sum_{i=1}^{n}x_i\le t\}$$

$$=1-\iint_{x_1,x_2,\cdots,x_n}\cdots\int dF_1(t)dF_2(t)\ldots dF_n(t)=1-F_1*F_2*\ldots*F_n \qquad (3.9)$$

式中$F_1*F_2*\ldots*F_n$表示卷积。

系统的平均寿命为：

MTTF=$\sum_{i=1}^{n}T_i$，其中T_i是第i个部件的平均寿命

当$R_i(t)=e^{-\lambda t}$，$i=1,2,\ldots\ldots,n$，且$\lambda_1=\lambda_2=\ldots\ldots=\lambda_n=\lambda$时，系统的可靠度和平均寿命为：

$$R_s(t)=\sum_{i=0}^{n-1}\frac{(\lambda t)^i}{i!}e^{-\lambda t} \qquad (3.10)$$

$$\text{MTTF}=\frac{n}{\lambda} \qquad (3.11)$$

由上述讨论可知：并联系统的寿命为单元中最大的寿命。而转换开关与储备单元完全可靠的旁路系统的寿命为所有单元寿命之和，这说明转换开关、储备单元均完全可靠的旁路系统的可靠性较大。[49]

3.2.4.4 表决系统系统可靠性模型

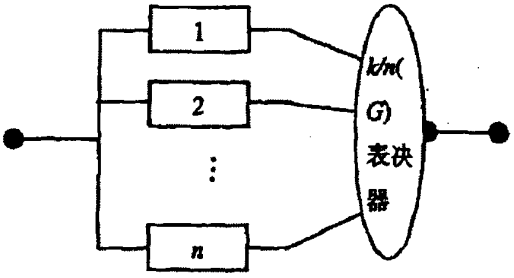


图3-7 表决系统

设系统由n个部件和表决器组成。系统工作时，每个部件的信息输入表决器中与其余信号相比较，系统成功完成任务需要其中至少K个部件是好的，这种系统称为k/n(G)结构，或称n中取k表决系统（1≤k≤n），其中G表示系统完好。表决系统的模型如图3-7所示。[49]

表决系统有以下三种特殊情况：

（1）当k=n，n/n（G）系统等价于n个部件的串联系统；

（2）当k=1，1/n(G)系统等价于n个部件的并联系统；

（3）当k=m+1，m+1/(2m+1)(G)系统称为多数表决系统；

假设各个部件的可靠度均为R(t)，且$R(t)=e^{-\lambda t}$，则表决系统的可靠度为：

$$R_s(t)=\sum_{i=k}^{n}C_n^i R(t)^i(1-R(t))^{n-i}=\sum_{i=k}^{n}C_n^i e^{-i\lambda t}(1-e^{-\lambda t})^{n-i} \qquad (3.12)$$

系统的平均寿命为：

$$MTTF=\sum_{i=k}^{n}\frac{1}{i\lambda} \qquad (3.13)$$

3.2.5 小结

由以上分析可以看出：1.并联系统的故障率低于各单元的故障率；2.而可靠度高于各单元的可靠度；3.并联系统的平均寿命高于个单元的平均寿命。通过并联可以提高系统的可靠度。两部件并联系统平均寿命提高50%，三个并联第3个对提高系统平均寿命的贡献为33.3%，四个并联，第4个的贡献为25%，随着并联部件的增多，对提高系统可靠度的贡献程度下降，所以一般只采用2个或3个并联来提高系统的可靠度。

旁路系统与并联系统的区别在于：并联系统中每个单元一开始就同时处于工作状态，而旁路系统中仅有一个单元工作，其余单元处于待机工作状态。

本论文以两部件并联系统为研究对象。因为其可靠性较大，而且适合于冗余热备。在实际项目中应用广泛，且具有很强的通用性。

3.3 系统实现关键技术研究之——实时

3.3.1 实时的概述

网络控制系统是一种典型的分布式计算机实时应用系统。所谓分布式实时系统是由一组分布的计算设备组成、每个计算设备执行一系列任务，且具有实时能力的系统。实时是指信号的输入、计算和输出都要在极短的时间内完成，并根据生产过程工况的变化及时地进行处理。而实时系统指在事件或数据产生的同时，能够在规定的时间内给予响应，以足够快的速度处理，及时地将处理结果送往目的地的一种处理系统。实时性将系统对输入信息做出响应的时间加以约束，只有系统处理信息的结果正确且在规定时间范围内得到，系统才是实时系统。系统接受到信息后，必须在一定的时间内做出响应，若得到结果正确但超过了时限，就认为系统失败，满足一项任务的实时性是指其响应时间小于规定的时限。

实时与快速并不完全等同，不论网络的传输速度有多快，只要在规定的响应时间内发生响应动作，则称系统具有实时性；而实时网络是指网络中数据传输是具有时限的。

因此，控制系统的实时性即包括系统中控制器的实时性又包括通讯网络的实时性。一般的，控制器所要处理的事务不止一项，且各项事务对实时性要求各不相同。各项事务对实时性的要求可由控制器中的实时多任务管理程序来决定。控制系统具有通信功能的基本单元通过通信网络联系在一起，这些单元称为“站”或“节点”，当某站向通信子网请求通信时，它对响应时间是有要求的，不同的站对实时性要求可能不同，同一站中的不同通信任务对实时性的要求也可能不同。

在网络控制系统中，需要传输的数据可分为3类：周期数据、突发数据和非实时数据。

(1) 周期数据：如各种传感器和控制器的I/O信号和部分系统状态监测数据，周期数据对时间有严格要求，一般不允许有秒级的时延，在某些特殊场合下甚至不允许有毫秒级的时延，并且有截止期限制。此外，对周期数据而言，有些情况下只有最新数据是有意义的，如果在某一时段内，一周期数据由于某种原因未能达到或出错，而此时下一个数据已经产生，则该数据将被丢弃，因此周期数据一般不要求重发。

(2) 突发数据：如报警信号和紧急操作指令。突发数据对实时性要求极高，一般应有比周期数据更高的优先级，并且要求准确无误，但数据长度一般较短，数据量相对较少，对带宽的占用率较低。

(3) 非实时数据：如用户编程数据和组态数据，对时间的要求并不严格，允许有一定的时延，但这类数据的长度较长且不确定，数据量较大，对带宽的占用率较高，一般以小型文件的形式出现。所传送的数据一般都是有意义的，不允许丢失，需要差错控制和重发机制保证数据的完整和准确。

上述3类数据中，周期数据和突发数据属于实时数据，但对实时性的要求不同，前者应满足截止期要求，对时延特性较为敏感，时延的不确定性影响闭环控制的稳定性和控制性能；后者则希望尽可能快地得到响应，时延越小越好，但并不介意时延是否具有确定性，同时对可靠性的要求高于周期数据。

3.3.2 影响控制网络实时性的主要因素

(1) 网络自身的硬件性能：网络的拓扑结构、通信媒体、网络接口的传输速率等。通信媒体的传输速率越高、网络接口的传送速率越快，网络的实时性越高。

(2) 网络的通信协议：媒体访问控制方式、网络通信协议的层次结构、传输的可靠性、有无连接控制等。一般来说，层次结构越简单，系统的实时性越高。而可靠性与实时性相互矛盾，对于无连接、无应答的通信方式要比有连接、有应答的通信方式的实时性要高，但可靠性差。其中媒体访问控制方式是控制网络各站点向(或从)媒体发送信息(或取得信息)的时刻，是影响网络实时性的重要因素。

(3) 网络负载，即传输的信息量，指网络在一定时间内需要传送信息的多少。网络传送信息量越少，实时性越高。

3.3.3 控制网络实时性的实现研究

3.3.4.1 硬件方面的考虑

根据需要，选择速度快的硬件设备，包括网络控制器的选取、传输介质的选择、网络设备的选取（如交换器、中继器、网关等）等等。而在实际设计中还要考虑成本等综合因数。

3.3.4.2 数据实时传输的方法

为完成实时数据的传输，通常有4种实时增强方法：集中轮询、令牌传递、总线仲裁和主从式静态调度。其中，集中轮询类似于TDMA方式，该方法立足于节点间带宽分配的公平性，应用中的主要难题是难于保证每条消息的访问时间和传输延迟，满足节点应用任务间的约束关系。令牌传递方式使用令牌作为网络资源的仲裁器，确定网络上的计算机发送数据的时间，这种方式可以确定网络数据发送的最大延迟。并且，令牌传递方式可以灵活地添加和删除网络结点，从而适应网络结点的动态变化。但是由于其协议的实现对设备资源的要求较高，因此对现场级设备不太适合。总线仲裁的方法加入统计规则进行准入控制，比较适合于软实时的数据传输，而对于硬实时的数据传输要求无法满足。主从静态调度机制采用主从结构的方式进行网络资源的分配，所有的网络资源都受到主站的控制，并由主站确定哪一个从站可以向网络发送数据，因此这种方式较适合于硬实时数据传输，但这种方式对网络固定性要求较高，不适合于动态的网络结点的变化。根据上述分析，每一种网络技术都有它适合的某一个方面，很难找到一种网络技术能够解决所有的网络传输。

3.3.4.3 减少控制网络负载

(1) 尽量缩减通信协议的层次和每层的协议，所以目前的DCS或FCS都采用三层协议，这样既减少了无效数据的传输，也能减少打包拆包的时间，从而提高网络的实时性。

(2) 减少通信过程中应答的次数，即尽可能采用同步传输，因为异步传输中，每传送一个字节，双方就要联络应答一次，因而有效数据传输效率低；而对于同步传输，每传送一帧数据，只联络一次。在一帧内不必联络应答，有效数据传输效率明显提高。另外采用无连接、无响应的数据通信方式，既不需要在通信前建立连接．也不需要在传送数据后作出应答，这就更加减少无效数据的传送：而采用无连接、无应答方式的要求是网络的误码率低，在过程控制网络中，由于传输距离一般说来都比较近，为几百米到几千米，这种情况下，网络的误码率是比较低的，能够达到控制的要求，故可以采用此法。

(3) 例外报告法：对毫无变化或变化甚微的过程数据不再进行重复传送，都以第一次传送的数据为准，等到过程数据有了明显的变化时再进行传输。为提高实时性节省通信资源，例外报告法先要规定一些条件范围，对超出范围的过程数据构成“例外报告”给予传送；没有超出则认为过程数据无变化，不生成报告，不予传送，只把上次数据保留下来作为本次数据。这样可以提高有效数据传送效率，从而提高系统的实时性。为避免避免有些数据变化太慢，长时间形不成例外报告，使得通信子网上长时间没有通信发生，与通信子网发生故障不能通信混淆起来，规定最大无传输时间，到了时限即使仍未生成例外报告，也应进行一次传输，以表明通信子网的正常运行而没有故障。

(4) 在不影响控制系统正常运行的情况下，尽可能选择大的采样周期，这样可减少通信子网的负载，提高网络实时性。

(5) 设置合理的控制系统的测控点、操作站和监控站的数量，即合理设置挂在控制网络上的节点数。在不影响系统正常运行的情况下，应尽可能减少控制网络上的节点数，这样既可减少通信子网的负载，又可以降低网络的复杂性和实现整个控制系统的投资。

(6) 采用分布式数据库。在工业控制网络中，有些数据共享性比较高，如果在网络中的站点间反复交换，将占用通信资源。若采用分布式数据库技术，在系统的各站点内设置局部数据库、在每个局部数据库中皆存放共享数据，并使其具有较高的自治性，这样可以减轻网络负载，提高网络的实时性。

在采用前面这些方法的基础上，工业控制网络的信息量会大大减少。然而它们对网络信息率的减少有一定的限度，并且在有些情况下，有些方式不能够减少网络的信息量，所以在这些情况下，只能采用其它的方式来提高网络的实时性。而且工业控制网络中，通信网络的负载不是固定不变的，而是随着控制系统的运行情况不断变化的。在控制系统正常运行的情况下，通过

采用例外报告法可使网络信息率大为减少，并且比较稳定，网络的负载较轻；在某些因素的作用下，控制系统发生波动，“例外报告法”不起作用，使网络的信息量比网络正常运行时大大增加，网络负载加重。[50]

3.3.4 小结

CANopen协议在制定时就充分考虑了实时性。1.在实时的传输控制上：CAN总线已经实现了根据ID进行总线仲裁，但是不能满目强实时；主从式静态调度不符合尽量减少网络负载的原则；采用集中轮询的方法，实现较繁琐；CANopen协议选用的是集中轮询，时间戳和同步是实现实时的关键，具体详见本文3.4.4中相关部分。2.在减少网络负载方面，有多处体现。数据帧传输方面应用了同步、PDO和远程帧的机制。在网络控制帧的传输上，NMT、心跳和节点保护的数据长度也都只使用了一个字节进行传输。

3.4 CANopen协议的设计与实现

3.4.1 基于 CANopen 协议的 CAN 总线层次结构

从OSI网路参考模型的角度来看，CAN总线网络只需要实现第1层（物理层）、第2层（数据链路层）和第7层（应用层）。CAN总线协议的层结构如图3-8所示。因为CAN总线通常只包括一个网段，因此不需要第3层（网络层）第4层（传输层），也不需要第5层（会话层）和第6层（表示层）的作用。其中第1层、第2层已由CAN控制器的硬件实现，设计者只需要选购独立的CAN控制器或者带CAN模块的主控制器。

许多器件厂商推出各种CAN总线器件，如Philips的82C250、Intel的82527CAN通信控制器等，已逐步形成系列。它们的发送模块和接收模块的实现都有所区别，因此在设计中应添加一层所谓的硬件抽象层(Hardware Abstraction Layer, HAL),即与硬件相关的驱动程序来实现对CAN控制器的驱动。将来移植到不同类型的CAN控制器硬件上时，只需修改硬件抽象层里的程序即可。如此一来，即可让CANopen协议应用的可移植性增强。[48]

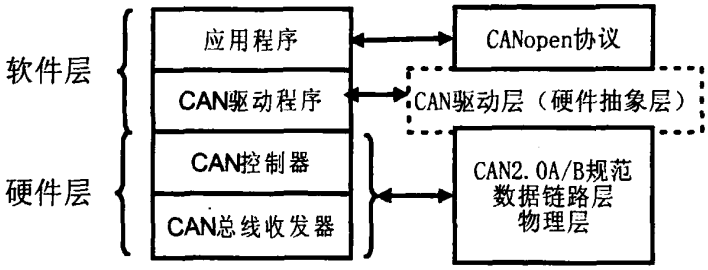


图3-8 基于CANopen协议的CAN总线层次结构

3.4.2 硬件层的设计

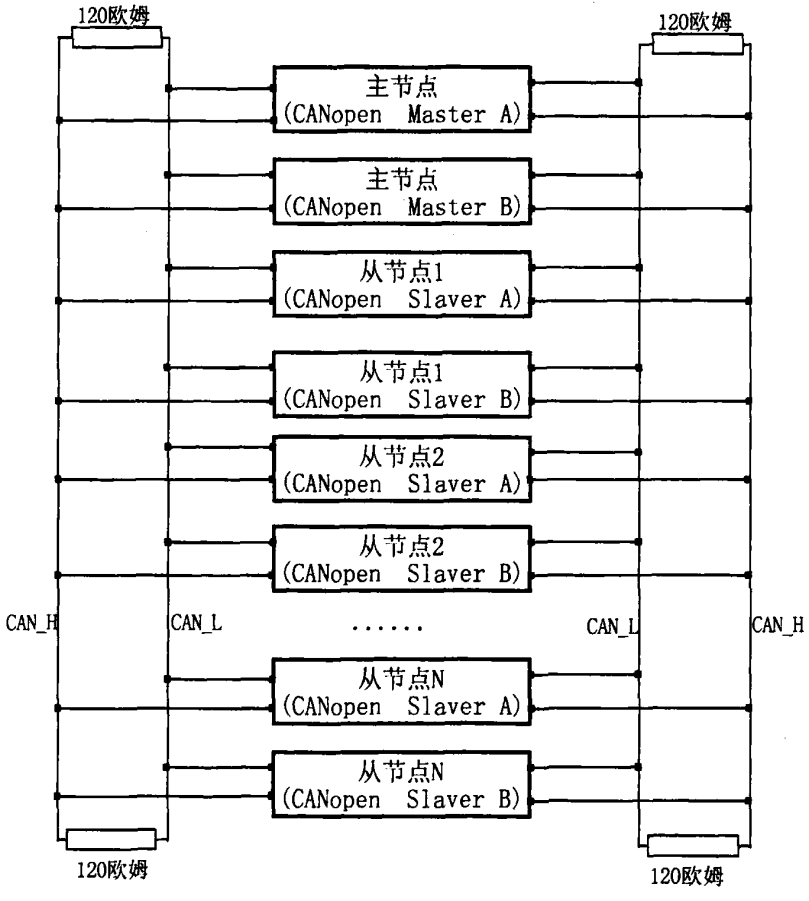


图3-9 网络拓扑示意性结构图

本设计针对并联系统CAN网络进行研究。网络中相同功能的节点都有两个，而且每个节点都带有双CAN口。系统的网络拓扑示意性结构如图3-9所示。

在实际应用中，为了提高系统的稳定性，相同功能的节点常常还配合其它硬件连接进行信息的交互，如串口、其它总线等等。

3.4.3 驱动层的设计

对于主控制器来说，CAN控制器完全是基于事件触发的，即CAN控制器会在本身状态发生变化时，会将状态变化的结果告诉主控制器。所以即可以采用中断控制的方式，也可以采用轮询查看CAN控制器状态的方式来对CAN控制器作出相应的处理。但采用轮询方式可能会降低设备中的主控制器的性能，所以一般CAN驱动层的设计都采用中断控制的方式来完成。CAN驱动层的主要模块有：CAN控制器的初始化模块、中断处理模块、发送报文模块和接收报文模块。对不同的CAN控制器，这些模块的实现有所区别，本文采用SJA1000为例，进行CAN驱动层的设计。

对于驱动层的设计，设计原则是越简单越好。对于多端口的CAN控制器：接收时，直接放入各自的接收队列；发送的时候把编号的数据帧很快的进行发送。对于冗余的体现、接收发送的控制策略、与上层有关的部分在抽象层实现。

3.4.3.1 SJA1000 的介绍

SJA1000 是一个独立的CAN控制器，它在汽车和普通的工业应用上有先进的特征。由于硬件和软件的兼容，它将会替代PCA82C200。它与PCA82C200相比具有更先进的特征，因此特别适合于轿车内的电子模块、传感器、制动器的连接和通用工业应用中，特别是系统优化，系统诊断和系统维护时特别重要。

3.4.3.2 初始化模块

该模块主要用来设置CAN工作时的参数。正常工作模式和复位模式是CAN控制器的两种状态模式。因为要得到配置信息的寄存器仅在复位模式可写，在进入正常工作模式之前，CAN控制器必须先进入复位模式，对CAN控制器的所有内部寄存器进行初始化，再进入工作模式。CAN控制器的初始化程序流程

图如图3-10所示。

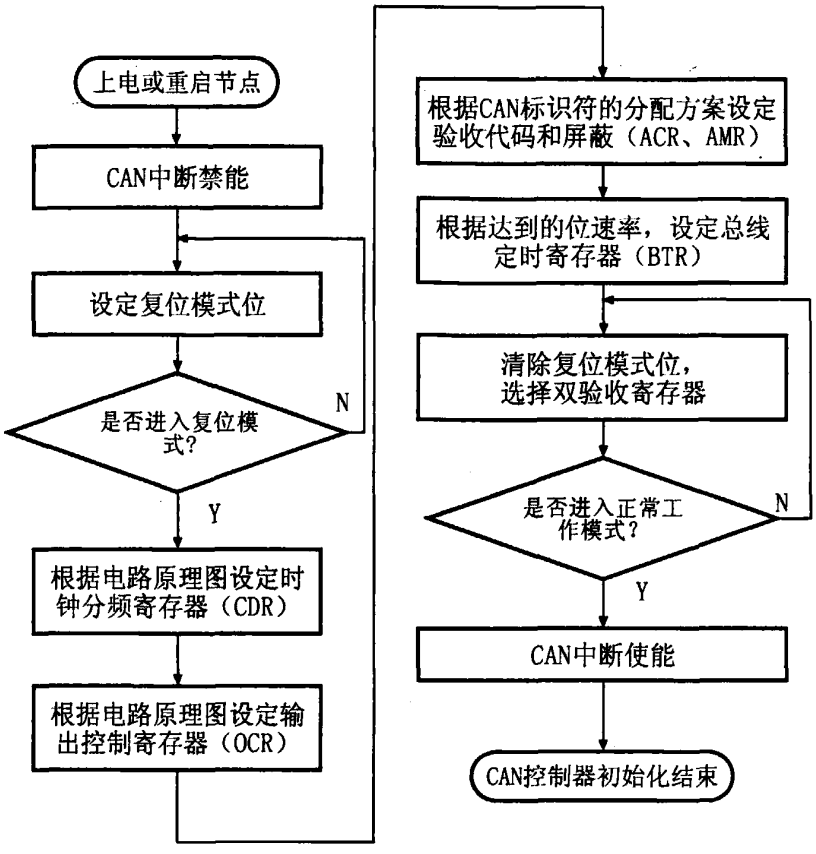


图3-10 初始化程序流程

3.4.3.3 中断处理模块

该函数主要对CAN的状态变化进行监控。当CAN控制器产生中断时，主控制器通过读取CAN控制器的中断寄存器（IR），并对中断采取相应措施。图3-11为中断处理流程图。

这个流程里不同的中断被处理的次序仅是一种可能的解决方法。所有的出错终端可以执行系统的一个通用的错误处理策略程序，这个策略可完成系统调试时的系统优化、运行时的系统自动优化和系统维护时的诊断。

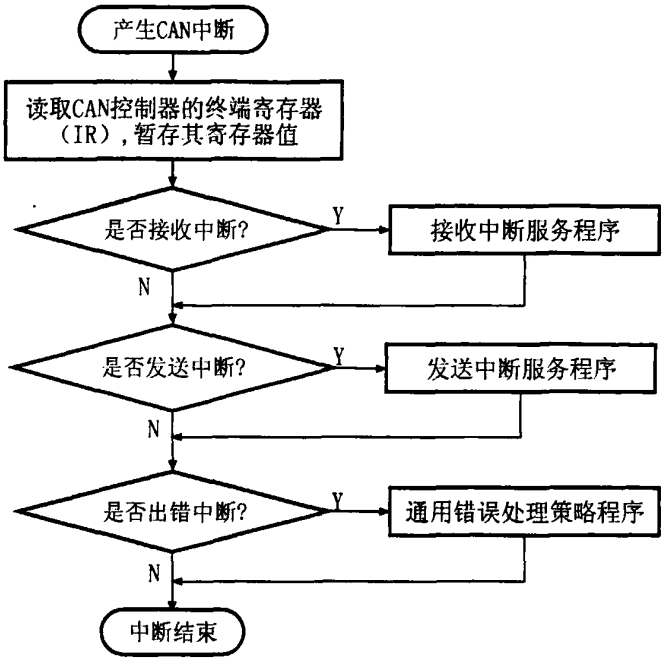


图3-11 中断发送程序流程图

3.4.3.4 接收报文模块

如果接受FIFO里有接收到的报文，置位命令寄存器“释放接收寄存器”位(RRB)将产生一个新的接受中断。由于SJA1000的接收FIFO有64个字节，最多可同时存放4个扩展帧，因此不急着从接收缓存区中读取报文，先处理好接收到的报文，再释放接收缓冲器，如图3-12所示。

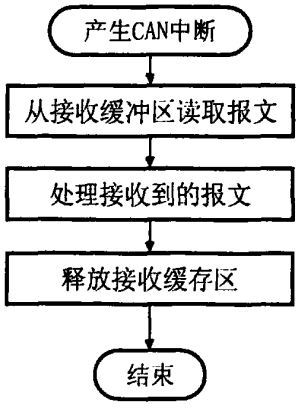


图3-12 接收中断服务程序流程图

3.4.3.5 发送报文模块

将本地数据打包成符合CAN帧格式的报文后，就可调用该模块进行报文的发送。图3-13为发送函数的程序流程图。因为CAN控制器在发送报文时，发送缓存区禁止读操作，这样如果在模块里一直等待数据发送完毕，才发送下一帧报文，会使整个主控制器的性能下降。为了避免上述这种情况，在放置一个新报文到CAN控制器的发送缓存区之前，主控制器要先检查状态寄存器的“发送缓存器状态”位（TBS）；当发送缓存区被锁定时（TBS = 0），CPU将下一个要发送的报文存入临时存储区。在该情况下，下一个发送报文的流程图将在发送中断服务程序中处理，如图3-13所示。置位命令寄存器的“发送请求”位（TR），使CAN控制器启动发送。

为了避免由于不同的原因一个发送报文长时间等待处理，而使发送缓存区一直被锁定的情况，置位命令寄存器的“中止发送”位，则发送缓冲区被释放，同时产生一个发送中断，来将一个高优先级的报文放入发送缓存区进行发送，其流程如图3-13所示。[48]

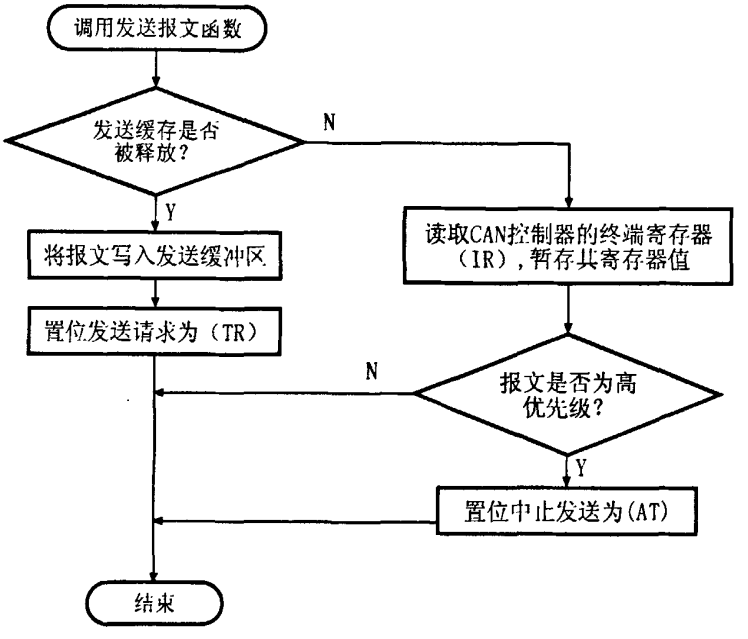


图3-13 发送报文函数的程序流程图

3.4.3.6 硬件抽象层的设计

硬件抽象层的主要功能有：(1)实现底层的控制策略；(2)实现底层与上层数据的无缝连接；(3)实现上层应用与底层的无关性。此处不再详述，具体详见第四章中相关部分的实现。

3.4.4 CANopen 协议栈的设计

3.4.4.1 软件结构

根据上一章CANopen协议的剖析，协议栈的实现软件可分为以下几个模块：PDO模块、SDO模块（包括SDO-Server模块和SDO- Client模块）它们实现数据传输的功能；NMT模块用来实现网络管理，NMT模块分NMT-Server模块和NMT-Client模块,其中网络的主节点中应该有NMT-Server模块，其它的设备中应该有NMT- Client模块；对象字典的软件模块是CANopen协议层的核心部分，以表的形式管理应用对象和通讯对象的参数，提供一个全局的数据结构；特殊功能对象包括同步（Sync）对象、心跳（Heartbeat）、应急（Emergency）对象和时间戳（Time Stamp）对象，此部分在实际的应用中，可根据具体的需要添加；API（Application Programming Interface,应用程序编程接口）提供上层具体应用和CANopen协议软件协议模块的衔接，也是CANopen协议实现开放式业务结构的关键技术之一。软件结构如图3-14所示。

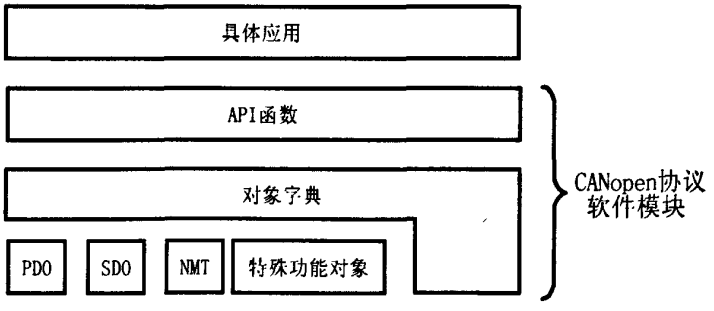


图3-14 CANopen协议的软件结构

3.4.4.2 软件模块的设计

1、模块的设计思想[6]

为了使设计的CANopen协议栈符合“开-闭”原则。“开-闭”原则讲的是：一个软件实体应当对扩展开放，对修改关闭。在设计一个模块的时候，应当使这个模块可以在不被修改的前提下被扩展。换言之，应当可以在不必修改源代码的情况下改变这个模块的行为。

符合“开-闭”原则的一个软件系统具有两个无可比拟的优越性：

(1) 通过扩展已有的软件系统，可以提供新的行为，以满足对软件的新需求，使变换中的软件系统有一定的扩展性(Extensibility)、灵活性(Flexibility)、可插入性(Pluggability)；

(2) 已有的软件模块，特别是最重要的抽象层模块不能再修改，这就使变换中的软件系统有一定的稳定性和延续性。具有这两个优点的软件系统是一个在高层次上实现了复用的系统，也是一个易于维护的系统。

2、模块的具体设计

本系统在驱动层的设计上就充分考虑了迪米特法则，硬件抽象层的设计使得CAN驱动层与上层的CANopen协议栈分离。

由于CAN驱动层数据帧的格式和CANopen协议定义的数据格式是不一样的。应用中采用适配器模式。在接收中把CAN驱动层接收的数据帧转换成CANopen协议定义的数据格式，在发送时把CANopen协议定义的数据转换成CAN驱动层的数据帧格式。

在接收处理时，采用策略模式，根据接收到的数据帧功能码的不同，采用不同的处理策略。

对象字典的设计依据“依赖倒转原则”。传统的过程性系统的设计倾向于使高层次的模块依赖于低层的模块，抽象层次依赖于具体层次。本系统对象字典OD是把这个错误的依赖关系倒转过来，对象字典含有宏观的和主要的商务逻辑，由它决定具体层次的实现和具体算法的改变，对象字典来决定上层的具体应用。对象字典在整个模块中只有一个，所以在实现中宜采用单例模式来实现。

3.4.4.3 节点的工作流程

系统的工作流程采用状态机的管理方式，通过控制系统的工作状态机（bStateMachineState），控制系统的工作状态。状态机的改变有两种情况：一种是由于收到CAN网络的NMT帧而改变，另外一种是程序内部逻辑控制它的变化。

其核心代码如下：

```
void main()
{
    bStateMachineState = INITIALISATION_COMMUNICATION;
    while(1)
    {
        Switch( bStateMachineState)
        {
            case INITIALISATION_COMMUNICATION:  // 通讯初始化
                Init_Can_Phy();                 //初始化CAN物理层
                Init_Canopen();                 //初始化CANopen
                bStateMachineState = INITIALISATION_NODE;
                break;
            case INITIALISATION_NODE:           //初始化节点状态
                Initialisation();
                bStateMachineState = PRE_OPERATION;
                break;
            case PRE_OPERATIONAL:               //进入预操作状态
                PreOperational();
                break;
            case OPERATIONAL:                   //进入操作状态
                Operational();
                break;
            case STOPPED:                       //进入停止状态
                Stopped();
                break;
```

```
            default:                                    //异常，重新初始化节点状态
                bStateMachineState = INITIALISATION_NODE;
                break;
        }
    }
}
```

3.4.4.4 对象字典的设计

对象字典由一组对象组成，每一个对象又有一个或多个子对象，而子对象拥有实际的对象数据。对象字典仪表的实行组织，存储在ROM或RAM中。在ROM中存有一张索引表。子索引表可被存放在ROM或RAM中，它的每一项的内容（字段）有：子索引号、访问权限、数据长度、当前值的地址、默认值的地址和回调函数指针，为了优化访问速度，对象以索引号升序排列，子对象以子索引号升序排列。

以下是为设备中的对象字典主索引与子索引定义的数据结构：

```
typedef struct                  /*Index*/
{
    uint8       Index;          /*object index of the entry to be define*/
    uint16      Count;          /*number of subindex within this index entry*/
    OD_Subindex *psub;          /*address of subindex table*/
}OD_Index;
typedef struct                  /*Subindex*/
{
    uint8       Sunindex;       /*subindex for the subindex entry to be define*/
    uint8       AccType;        /*access rights for the object*/
    uint16      Size;           /*maximum length of the object data int RAM*/
    void        *pData;         /*value for the object data*/
    uint32      (*pFunc)();     /*callback function*/
}OD_Subindex
```

3.4.4.5 NMT 模块控制的消息语法细节

只有NMT-Master节点能够传送NMT模块控制（NMT Module Control）报文。所有从设备必须支持NMT模块控制服务。NMT消息由NMT-Master 发给NMT-Slave(s)，NMT Module Control消息不需要应答，其格式如表3-1。

表3-1 NMT消息帧格式

COB-ID	Byte0	Byte1
0x000	CS	Node-ID

当Node-ID=0，则所有的NMT从设备被寻址。CS是命令字，其含义见表3-2。

表3-2 CS命令字含义

命令字	NMT服务
1	Start Remote Node
2	Stop Remote Node
128	Enter Pre-operation State
129	Reset Node
130	Reset Communication

3.4.4.6 NMT 节点保护的消息语法细节

通过NMT节点保护（NMT Node Guarding）服务，NMT主节点可以检查每个节点的当前状态，当这些节点没有数据传送时这种服务尤其有意义。NMT消息由NMT-Master发给NMT-Slave(s)一个远程帧（无数据），其帧格式如表3-3。NMT-Slave(s)节点收到后发送给NMT-Master一个应答帧，其格式如表3-4。

表3-3 NMT-Master发送的节点保护帧格式

COB-ID
0x700+ Node_ID

表3-4NMT-Slave(s)发送的节点保护帧格式

COB-ID	Byte0
0x700+Node_ID	Bit7: toggle　Bit6-0:状态

数据部分包括一个触发位（bit7），触发位必须在每次节点保护应答中交替置“0”或者“1”。触发位在第一次节点保护请求时置为“0”。位0到位6（bits0～6）表示节点状态，节点状态的含义为表3-5中的值。

表3-5 节点状态的含义

Value	状态
0	Initialising
1	Disconnected
2	Connecting
3	Preparing
4	Stopped
5	Operational
127	Pre-operational

状态0从不在节点保护应答中出现，因为一个节点在这个状态时并不应答节点保护报文。

3.4.4.7 Heartbeat 的消息语法细节

为了实现网络管理中的错误控制（NMT Error Control）功能，获得网络节点的当前状态而引入的。Heartbeat协议采用Producer/Consumer模式，Heartbeat-Consumer通常是NMT-Master节点，则Heartbeat-Producer是NMT-Slave节点。Heartbeat报文周期性的由NMT-Slave发出，表明该节点目前仍然在工作，并向Heartbeat-Consumer报告它的状态，其报文的格式如表3-6，节点状态的含义为表3-7中的值。

表3-6 心跳帧的格式

COB-ID	Byte0
0x700+ Node-ID	状态

表3-7 节点状态的含义

Value	状态
0	Boot-up
4	Stopped
5	Operational
127	Pre-operational

如果Heartbeat-Consumer在规定的时间间隔内没有接收到Heartbeat报文，就发送一个NMT报文来重启通讯或重启节点，相当于一个网络“看门狗”。实际上，当一个Heartbeat-Producer节点重启后在转入预操作状态所发送的

Boot-up报文是其第一个Heartbeat报文，其帧格式为表3-8中所示。

表3-8 NMT - Slave发送的Boot-up帧格式

COB-ID	Byte0
0x700+ Node-ID	0

Heartbeat-Producer发送Heartbeat报文的时间间隔（Producer heartbeat time）将在对象字典中索引0x1017处被定义：Heartbeat Consumer接收heartbeat报文的最小时间间隔（Consumer heartbeat time）将在对象字典中索引0x1016处被定义，它的对象类型是数组，管理者每一个Heartbeat Producer的节点编号和时间间隔。两者的时间单位为毫秒（ms），如果两者时间间隔定义为0，则Heartbeat协议不起作用。Heartbeat Consumer定义的时间间隔应比Heartbeat Producer定义的大。

一个节点不能够同时支持Node Guarding和Heartbeat协议，本系统的实现中采用Node Guarding实现，而没有采用Heartbeat实现。

3.4.4.8 双模冗余网络管理的实现

本设计中主要研究基于CANopen协议的双模冗余网络管理的实现，对于多模冗余网络的管理具有一定的借鉴作用。而且也是实现CANopen在冗余系统中应用的关键技术。网络的实现上属于并联冗余系统，较常用的串联冗余系统具有更高的可靠性。系统的切换方式上，为了表述的方便采用了冷冗余。因为对于热冗余的系统而言，为了提高系统的稳定性，切换的策略一般不只是靠本网络的信息做出决定，而是要综合串口等其它硬件冗余，多方面的信息做出网络切换的策略。

本设计基于双模冗余网络管理的实现主要通过NMT控制模块和节点保护实现。（1）对于系统中的任意一个从节点而言，是不区分网络中有多主节点或者单个主节点的。它只需根据NMT Module Control的控制，做好自己内部的管理即可，使得实现很简单。采用节点保护实现，而没有采用心跳实现，是因为节点保护比心跳多发送一种从主节点到从节点的保护帧。对于双CAN接收端口而言，从节点据此可判断本节点的底层是否正常，可以实现单个端口故障的自诊断、自恢复。单个端口故障的自诊断、自恢复的时间应该是节点保护帧发送间隔的3倍左右较为合适。（2）冗余的主节点需要周期性的发送节点保护帧。强主节点通过心跳帧，判断从节点的工作状态。弱主节点处

于热备状态，随时准备接管强主节点管理网络的正常运行。弱主节点也要发送节点保护帧。帧可以考虑是否完全相同。建议系统中相同功能的节点模块，在ID中设计一个表示位，识别具有相同功能的节点是哪个设备发送的数据，便于网络的诊断、扩展、维护。

1、主节点的启动过程

主节点启动比较的简单，启动示意性流程见图3-15。启动成功后默认进入的是预操作状态。是否进入操作状态需要上层应用提供切换，具体的实现上有手动切换和软件上的智能切换。预操作状态只消费心跳和发送节点保护。在操作状态不紧紧消费心跳和发送节点保护，还要对网络进行配置（只配置一次）、管理。对于主节点对从节点的配置，是在从节点发送启动帧后通过SDO进行，包括心跳时间等网络运行的各种参数。

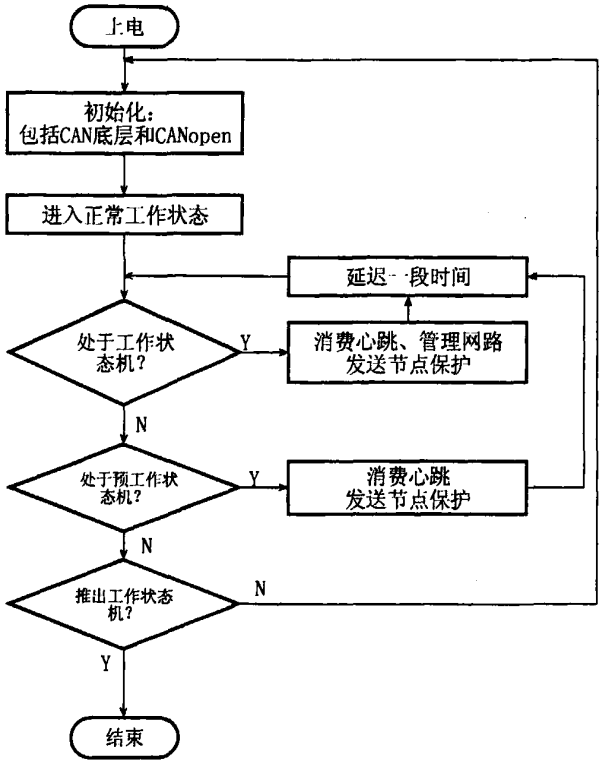


图3-15 主节点启动示意性流程

2、从节点的启动过程

系统的设计中，对于从节点加入系统，不区别是正常启动还是故障恢复后的加入网络。启动成功后，进入正常工作状态，开始发送心跳帧，默认进入的是预操作状态。从节点的启动流程如图3-16所示。

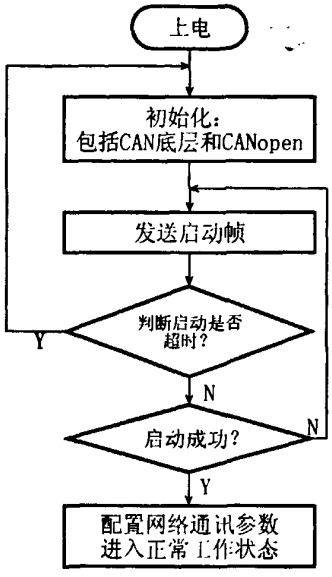


图3-16 从节点的启动流程

3、主节点对于网络的管理

从节点启动成功后根据主节点配置的心跳参数，周期性的发送心跳帧。对于心跳周期的选取（不同节点不同心跳时间），变速心跳机制的实现（相同节点变化心跳时间），心跳周期和系统高性能之间的关系，以及心跳周期与采集周期的选取，可参看相关文献[21][43][44][52]，本论文中不再赘述。

对于Heartbeat-Consumer也就是主节点，要有一个变量，定义消费心跳的时间。而且这个时间要略大于产生心跳的周期。还需要一个表来登记和管理总线上的NMT-Client节点。消费心跳的时间和心跳管理表的数据结构如下：

对于Heartbeat-Consumer也就是主节点，需要一个表来登记和管理总线上个NMT-Client节点，该表的数据结构如下：

```
typedef struct
{
    uint8   DeviceType;          /*设备类型*/
    uint8   NowRunFunID;         /*当前运行的设备ID*/
    uint8   InsertFlag_1;        /*第一个设备加入表中标志*/
    uint8   DeviceFunID_1;       /*第一个具有相同功能设备的编号*/
    uint8   NmtSlaveNodeID_1;    /*第一个从节点编号*/
    uint8   NmtSlaveState_1;     /*第一个从节点状态编码*/
    uint8   RecvNum_1;           /*第一个从节点处理前接收的次数*/
    uint8   InsertFlag_2;        /*第二个设备加入表中标志*/
```

```
    uint8   DeviceFunID_2;          /*第二个具有相同功能设备的编号*/
    uint8   NmtSlaveNodeID_2;       /*第二个从节点编号*/
    uint8   NmtSlaveState_2;        /*第二个从节点状态编码*/
    uint8   RecvNum_2;              /*第二个从节点处理前接收的次数*/
    uint16  ProduceHeartbeatTime;/*生产心跳的时间（单位：ms）*/
    uint16  ConsumerHeartbeatTime;/*消费心跳的时间（单位：ms）*/
    uint64  LastConsumerTime;       /*最近一次消费心跳的时刻*/
    uint32  Reserve;                /*保留*/
} NmtSlaveNode;                     /*心跳管理表*/
uint64  NowTime;                    /*当前时刻，从1984年1月1号到现在的毫秒数*/
```

NmtSlaveNode NMTSlaveTab[]={……}; /* 从节点的节点登记表*/

NMT- Server节点需要有API函数能发送NMT命令来改变NMT-Slave节点的状态，同时也处理接收到的Heartbeat报文：

- Send_NMT();　　/*发送NMT报文*/
- Process_Heartbeat();/*处理Heartbeat 报文*/

主节点收到从节点发送的启动帧后，对从节点进行配置，配置完成后加入节点登记表NMTSlaveTab。对于固定的网络，可以提前初始化好节点登记表。具有相同功能的从节点，先被配置好的从节点作为当前运行的设备。

心跳生产者周期性的发送心跳帧。主节点收到心跳帧就给NMTSlaveTab中对应的接收心跳次数加1。心跳消费者定时遍历从节点登记表对网络内的所有从节点进行管理。其流程见图3-17。

图3-17中的“进入对应的处理”为：首先判断现在的时刻和上次的处理时刻是否达到处理心跳的时间要求。如果时间未到，则退出，负责执行以下的分类操作。（1）对于都没有加入系统的，直接退出。（2）对于都加入系统的，又分四种情况处理。A、如果当前运行的设备没有心跳，而预操作机有心跳，则发送复位当前设备命令，启动预操作机进入操作状态。给当前工作的设备的加入系统标志InsertFlag 置0。B、如果当前的设备有心跳，而预操作机没有心跳，则发送复位预操作机命令。给预操作机加入系统标志InsertFlag 置0。C、如果预操作机与运行机都没有心跳，则两个都发送复位命令且加入系统标志InsertFlag 置0，当前运行的设备ID置0。D、对于全部正常的，不进行操作。最后给心跳计数清零，记录此次的处理时间。（3）对于只有一个加入系统的情况，如果他没有心跳就复位此设备，给InsertFlag置0且当前运行的设备ID置0。如果有心跳，心跳计数清零，记录此次的处理

时间。

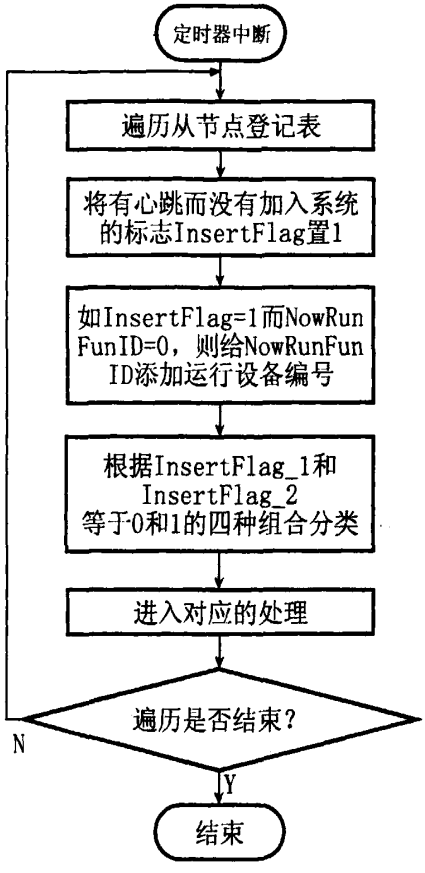


图3-17 主节点通过心跳管理系统的流程图

4、上层应用的接口

- 提供切换主节点工作状态的功能。
- 提供查询从节点的工作状态功能。
- 主节点，要给上层提供网络状态，为系统的冷冗余管理提供信息。
- 主节点要根据心跳的信息，对于没有心跳的设备信息进行及时更新，此点对于系统的安全及其主要。

3.4.4.9 SDO 模块的实现

1、SDO 的消息语法细节

对象字典作为应用层与通讯层的数据交换的媒介，一个遵循CANopen协议的设备中的所有数据项都是由对象字典管理，而其中的每一项可用索引号和

字索引号寻址。如果总线上的某一设备需要访问另一个设备的对象字典，可通过SDO的通讯机制来实现，如图3-18所示。

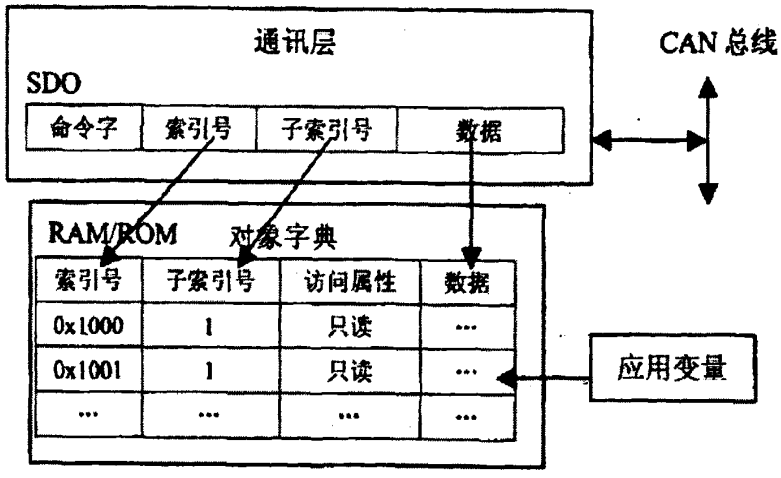


图3-18 SDO访问对象字典

由于SDO可用来访问对象字典中的所有数据，所以上述的映射到PDO的应用对象，在实时性要求不高的情况下，也可采用SDO的通讯机制传送。

SDO可以用来传输任意成都的应用对象。SDO采用Client/Server模式传送数据，所以一个SDO-Client的请求一定有来自SDO-Server的应答。对象字典的访问者被称为SDO-Client,被访问且提供请求服务的CANopen设备被称为SDO-Server。所有请求服务有两类：SDO下载（Download,对对象字典的写操作）和SDO上传（Upload,对对象字典的读操作）。一个下载/上传是由SDO-Client发起，SDO-Client先发送一个启动域请求报文。对于不多于4个字节的数据可以只使用启动域报文来发送（Expedited Transfer）,如果要传送多于4个字节的数据时，则需要分段传送（Segmented transfer）,如图3-19所示。

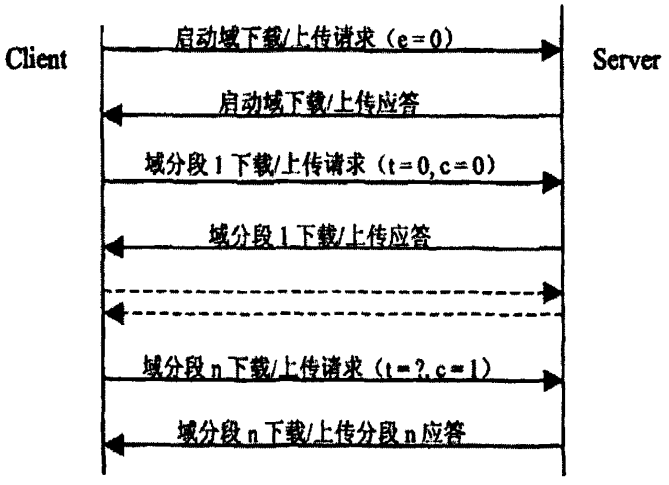


图3-19 SDO分段传送

SDO报文有规定的格式：CAN报文的数据部分的第一个字节包含必须的命令字（数据流的控制信息）；对于剩余的7个字节，除了启动域的请求/应答报文内的第2～4字节包含访问对象字典所需要的索引号和子索引号，最后的4个字节可用于存放不多于4个字节的数据（对于Expedited Transfer）或4个字节的字节计数器（对于Segmented Transfer）。SDO启动域的请求/应答报文的基本结构和域分段报文的基本结构分别见表3-9和表3-10。

表3-9 SDO启动域的请求/应答报文的基本结构

Byte0	Byte1-2	Byte3	Byte4～7
SDO命令	对象索引号	对象子索引号	数据（或字节计数器）

表3-10 域分段报文的基本结构

Byte0	Byte1～7
SDO命令字	数据

在SDO传送过程中，在SDO-Client和SDO-Server任何一方都可通过发送一个中止传送的报文来通知对方中止SDO传送。在域传送中止报文中，数据部分的0～2字节的内容是对象索引号和子索引号，4～7字节包含32位的中止代码，描述中止报文传送原因，如表3-11所示。

表3-11域传送中止SDO：16进制中止代码表（部分）

中止代码	代码功能描述
0x 0503 0000	触发位没有交替改变
0x 0504 0000	SDO协议超时

0x 0504 0001	非法或未知的Client/Server
0x 0601 0000	对象不支持访问
0x 0601 0041	对象不能够映射到PDO
0x 0602 0000	对象字典中对象不存在
0x 0606 0010	数据类型不匹配，服务参数成都不匹配
0x 0609 0011	子索引不存在
0x 0800 0000	一般性错误
……	……

2、SDO的实现

SDO的数据结构主要由SDO参数和暂存上传或下载数据的缓存组成：

```
typedef struct
{
    uint32  ReqCobID;   /*COB-ID Client->Server(请求)*/
    uint32  AckCobID;   /*COB-ID Server->Client(应答)*/
    uint8   NodeID;     /*通讯对方节点的编号*/
    uint16 Timeout;     /*SDO超时，单位：ms*/
    uint8   Status;     /*SDO当前状态，0：未被创建，1：启动域阶段，*/
                        /* 2：分段传输阶段*/
    uint16  Index;      /*索引号*/
    uint8   Subindex;   /*子索引号*/
    uint16  Size;       /*缓存区大小*/
    uint8   Buf;        /*缓存区地址*/
    uint16 Offset;      /*缓存区读写地址偏移量*/
    uint8   Toggle;     /*触发位标志*/
}SDOParam;
    SDOPara SdosTab[] = {…};    /*SDO-Server表*/
    SDOPara SdocTab[] = {…};    /*SDO-Client表*/
```

SDO-Server根据请求报文中的第一个数据字节的5-7bit的命令字(详见参考文献[36])，调用以下4个不同的处理函数：

- Initiate_Download():启动域下载，其程序流程图如图3-20所示。

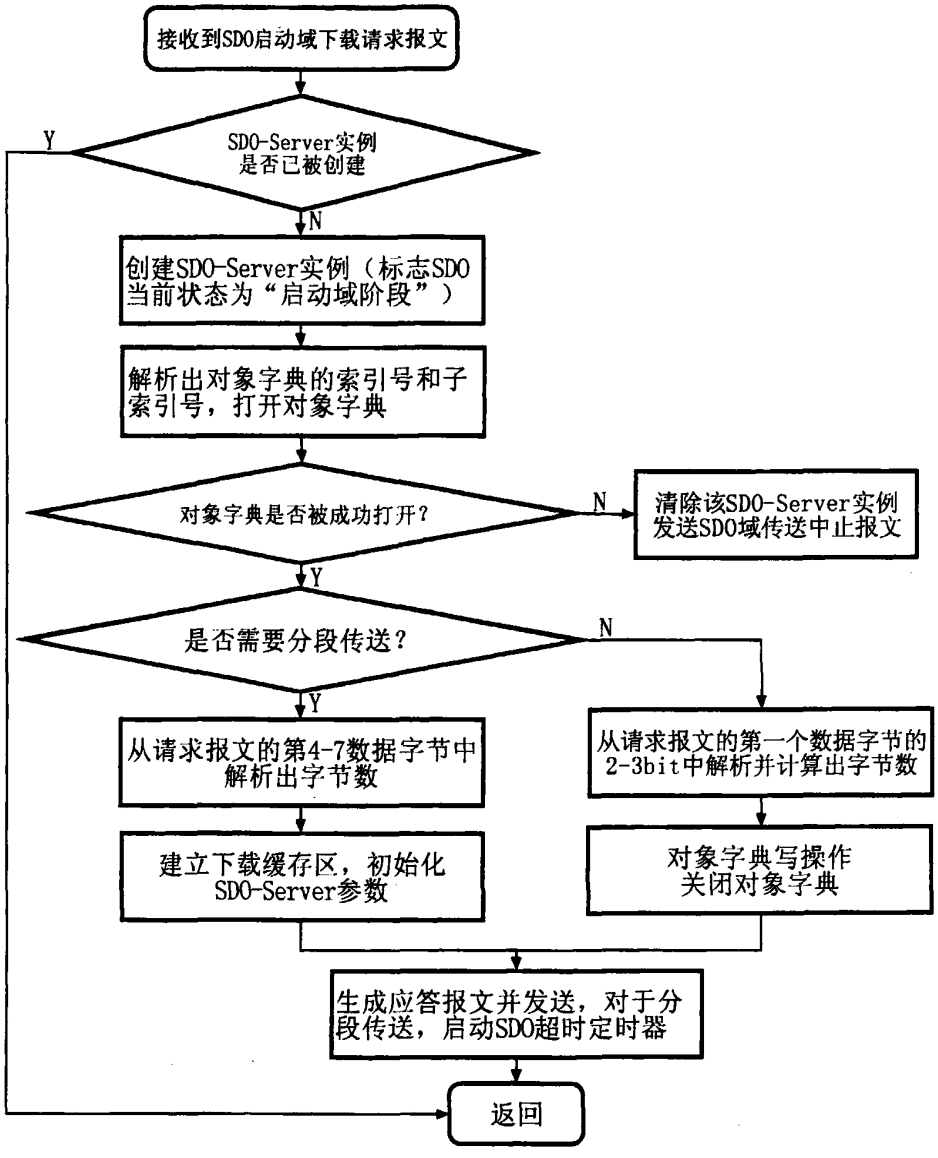


图3-20 域启动下载的程序流程图

- Initiate_Upload():启动域上传，其程序流程图如图3-21所示。

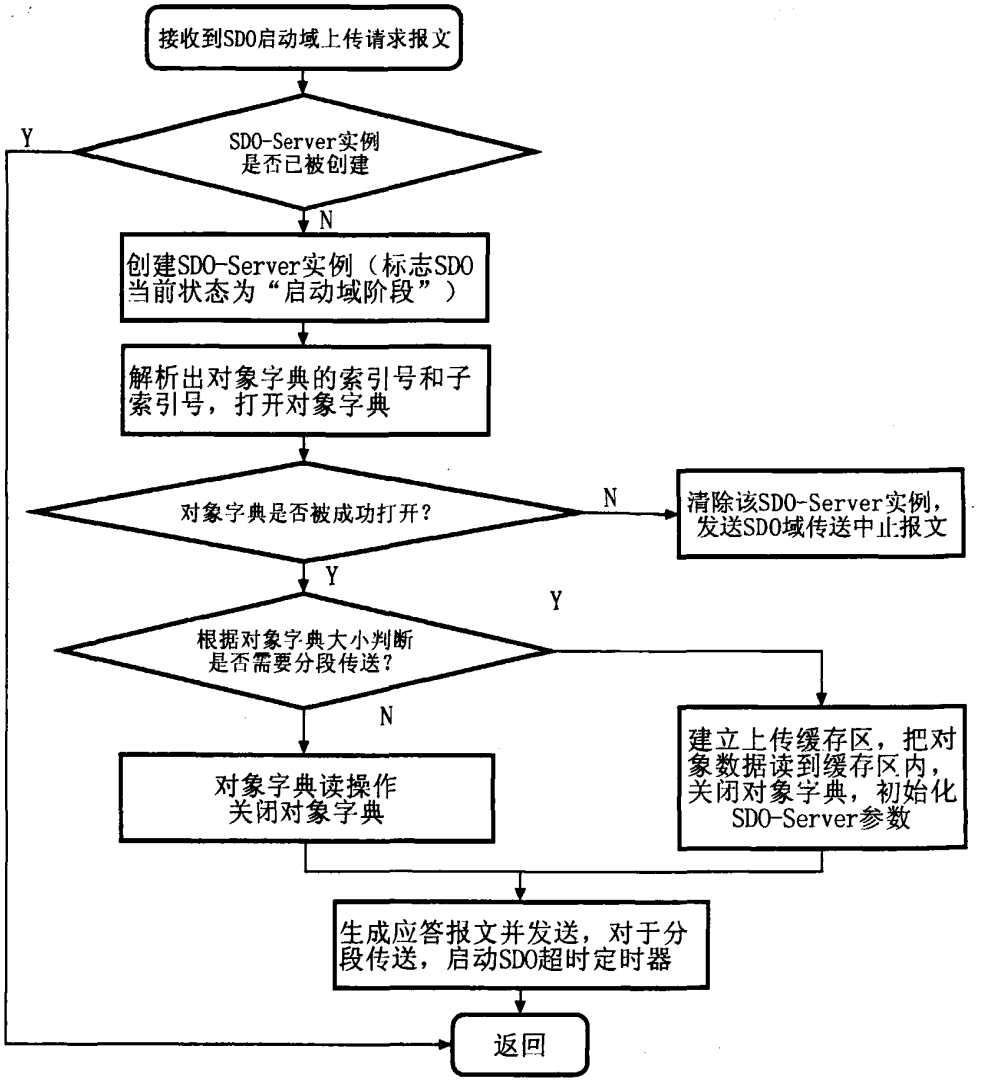


图3-21 域启动上传的程序流程图

需要说明的是对于多于4个字节的数据传送不需要开辟缓存区。而对于需要分段传送的数据应该先在启动域阶段用malloc函数开辟数据缓存区，把需要下载的对象数据先都写入缓存区内，再一下子复制到对象字典中：或者把需要上传的对象数据先从对象字典中一下子复制到缓存区，再分段传送。这是因为分段传送的过程可能较长，在其过程中对象数据有可能被其它应用程序直接修改，而导致前后传送的数据不一致，或者在分段传送的过程中由于某些原因发生SDO域传送中止，而导致传送的数据发生错误，对象字典被破坏。

对于分段传送，SDO-Server在发出应答报文后，启动SDO超时定时器，等待下一个请求报文，如果通讯链路出现问题或是SDO-Client节点发生故障而

使通讯无法进行下去，就会导致定时器超时，于是SDO-Server中止SDI传送并发送域传送中止报文（中止代码为0x05040000,SDO协议超时），这样就不会使SDO-Server一直处于等待状态而无法进行后面的工作。同理，在SDO-Client方也应有一个同样的SDO超时定时器，用以进行超时控制。

- Download_Fragment():域分段下载，其程序流程图如图3-22所示。

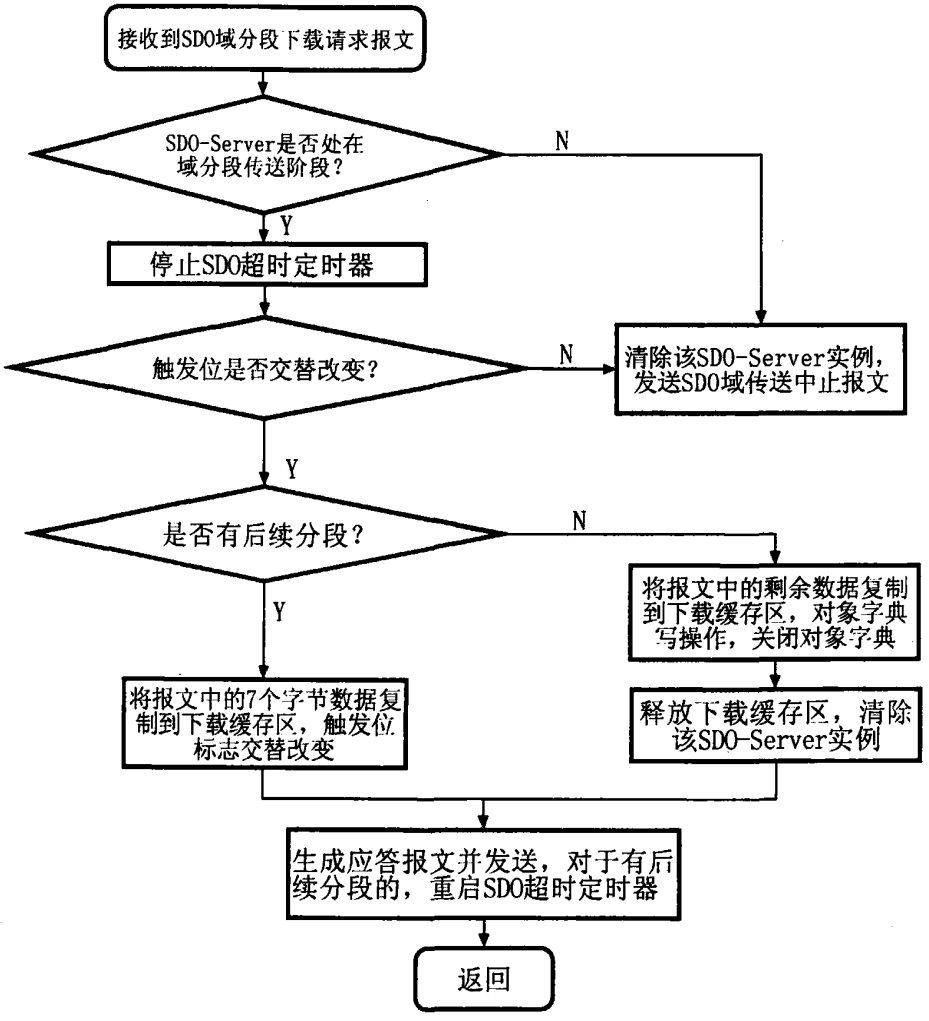


图3-22 域分段下载的程序流程图

- Upload_Fragment():域分段上传，其程序流程图如图3-23所示。

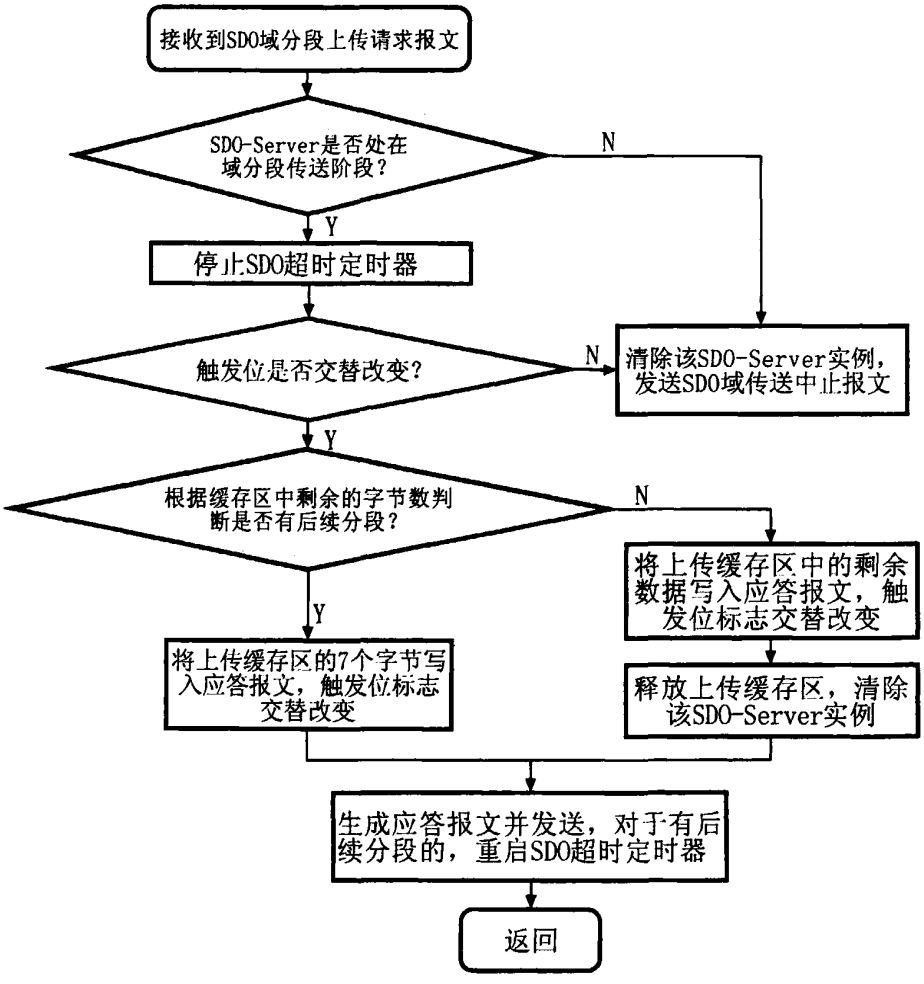


图3-23 域分段上传的程序流程图

SDO-Client也有启动域下载、启动域上传、域分段下载和域分段上传这4部分，也应有相应的发送请求报文和处理应答报文的函数及其程序流程，遵循SDO传送协议和SDO命令字（SDO报文的第一个数据字节）的语法和细节，有关具体的程序流程省略。[48]

3.4.4.10 Sync 的实现

SYNC的通讯模式属于生产者/消费者通讯方式。网络中有且只有一个SYNC生产者，一般有多个消费者。基于冗余CANopen的控制网络SYNC生产者为网络的主节点发送。SYNC的生产者可以在网络的启动过程中动态配置，也可在系

统设计中静态配置。对象字典中索引1005h的bits30存储着节点是否是SYNC的生产者。SYNC生产者时钟周期触发，触发周期存储在对象字典中的索引为1006h。SYNC生产者通过广播的方式发送SYNC数据帧。数据帧的网络标识符为80h，基本上除了NMT优先级最高；数据帧的程度为0。对象字典的索引1007h中存储着同步时间窗的宽度。由于程序流程较简单，故不做详细介绍。

SYNC是CANopen管理各节点同步数据收发的一种方法，相当于网络节拍，基于同步的PDO按照这个网络节拍来执行实时数据的收发。因此是CANopen实现实时的关键环节。

3.4.4.11 PDO模块的实现

PDO的数据结构主要由PDO得通讯参数和映射参数组成：

```
typedef struct
{
    /*PDO通讯参数*/
    uint32  CobID;          /*COB-ID used by PDO*/
    uint8   TxType;         /*传送类型*/
    uint16  InhibitTime;    /*禁止时间*/
    uint16  EventTime;      /*事件定时周期*/
    /*PDO映射参数*/
    uint8   NumOfEntries;   /*被映射的项的个数*/
    uint32  MappedParam[8]; /*被映射的项（索引号+子索引号+位数）*/
    /*标志*/
    uint8   SYNCflag;       /*同步标志*/
    uint8   RTRflag;        /*远程请求标志*/
    uint8   EVENTflag;      /*内部事件标志*/
}PDOPara;
    PDOPara    TPdoTab[]={……}; /*发送PDO表*/
    PDOPara    RPdoTab[]={……}; /*接收PDO表*/
```

其中，同步PDO时根据映射参数NumOfEntries、MappedParam[8]和通讯参数CobID来封装报文，在不短于禁止时间InhibitTime和不长于事件定时周期EventTime内部PDO报文发送出去，相应的API函数为：

- Send_PDO():发送PDO报文。

在接收到远程帧或SYNC报文，会触发某些传送类型的PDO，所以要有远程帧和SYNC报文的处理函数：

Process_RTR():根据PDO传送类型，处理远程帧；

Process_SYNC():根据PDO传送类型，处理SYNC报文。

接收PDO的步骤是：（1）根据PDO报文的COB-ID,查找接收PDO表RPdoTab，找到相应的参数；（2）根据通讯参数NumOfEntries和MappedParam[8]，解析出相应的对象数据后写入对象字典，相应的API函数为：

- Receive_PDO():接收PDO报文。[48]

3.4.4.12 时间戳的实现

时间戳对象表示从1984年1月1日后的天数和午夜过后的毫秒数，一共是6个字节。

特别在一些大型网络里，网络的传输速度比较慢，那些对时间要求苛刻的设备就要求非常准的时间同步，有可能需要把各自的时间同步在毫秒级。这就可以用这种高精度的同步协议来达到这种要求，用特殊的时间戳来调整不可避免的时间漂移。这种高精度的时间戳是一个无符号32位数，精度为1毫秒，这意味着时间计数器每72分钟重新计数一回。

对于一个节点是不是时间戳的生产者，在对象字典的1012h的bits30中进行了描述。这个高精度时间戳在对象字典的1013h，发送时映射到一个PDO里。时间戳的传输过程如图3-24所示。

图3-24 时间戳

时间戳对象传输时，发送设备作为生产者，接收设备作为消费者。另外，对于时间戳对象，标识符为256。

对于冗余网络时钟同步的实现，建议采用集中式策略，网络的强主节点

定期与UTC进行同步，并向内部提供基准时钟。在系统中弱主节点作为备份，实现主节点的冗余。时钟同步的方式可以采用推模式(push)和拉模式(pop)。

push模式实现相对简单，服务器定期将时钟信号发送给各客户端，在主节点上保存一张客户列表，记录从节点反馈的信息，主节点可根据该信息自动调整下一次的发送时间。在该模式下，几乎所有的节点都在中心节点上进行，使得中心节点负载过重，另外，无法确定网络延迟对更新的影响。

pop模式，时钟同步由从节点发起。客户端根据时钟精度的要求，由自身时钟偏移速度和历次同步时间的统计，计算出下一次的同步周期，从而实现自适应的时钟同步。

在基于冗余网络的CANopen实现中，时间戳和SYNC的实现应综合进行考虑设计、两者的结合对于减少网络的负荷，提高时钟的准备度，提高系统的实时性有很重要的意义。

3.4.4.13 API 的设计

- 提供关于系统运行的运行指示灯。CANopen协议中已经有了明确的规定。
- CANopen应该提供给上层应用关于CAN网络的工作状态，特别是主节点的设计中。
- 状态机的设计既提供了网络内部的管理（包括自复位功能），也便于上层应用的控制。
- 上层通过读写对象字典提供与系统进行数据的交互。对象字典中设有回调函数，用于CANopen与应用之间的交互，很好的实现了接口隔离。

3.4.4.14 设计中的一些经验总结

- 对于一个轻量级的系统，没有必要一定实现SYNC、PDO发送禁止时间等关于实时的实现，CAN已经提供了根据优先级竞争总线的机制。
- 对于降低开发的难度，可以去掉SDO的配置网络。
- 注意对象字典的设计，特别是在嵌入式中，内存有限，不必要的内容尽量精简，但也要考虑移植时候的一致性、可维护性。

- 系统的设计中，使用统一的错误返回值。

第4章 CANopen协议在股道管理自动化系统中的实际应用

4.1 应用背景

4.1.1 股道管理自动化系统的层次结构

股道管理自动化系统是实现以进路控制为主要内容的微机联锁功能控制系统。随着铁路运输发展的需要和科学技术的进路，股道管理自动化系统的功能、体系结构、操作方式等各个方面都在不断地完善。股道管理自动化系统是以色灯信号机、动力转辙机和轨道电路作为室外三大基础设备，以电气设备或电子设备实现联锁功能对轨道区段状态、信号状态和道岔状态进行检测并对信号机和道岔实施控制的系统。根据系统各主要部分的功能和设置地点的不同，系统的层次结构如图4-1所示。

操作表示层：操作人员通过操作向联锁机构输入操作信息和接收联锁机构输出的反映设备工作状态和行车作业情况的表示信息。人机会话层的设备设于车站值班室。

联锁运算层：联锁机构是联锁控制系统的核心，联锁机构除了接收来自人机会话层的操作信息外，还接收来自监控层的反映信号机、动力转辙机和轨道电路状态的信息，根据对输入的操作信息和状态信息，以及联锁机构的当前内部信息进行处理，产生相应的输出信息，即信号控制命令和道岔控制命令，并交付监控层的控制电路予以执行。联锁机构所处理的信息都是二值逻辑信息，因此，联锁机构又是逻辑处理机构。联锁层设备设在车站信号楼的机械室外。

输入/输出测控层：接收来自联锁层的控制命令，经过信号机控制电路，改变信号显示；接收来自联锁层的道岔控制命令，经过道岔控制电路，驱动道岔转换；向联锁机构传输信号状态信息、道岔状态信息，以及道岔电路状态信息。[11]

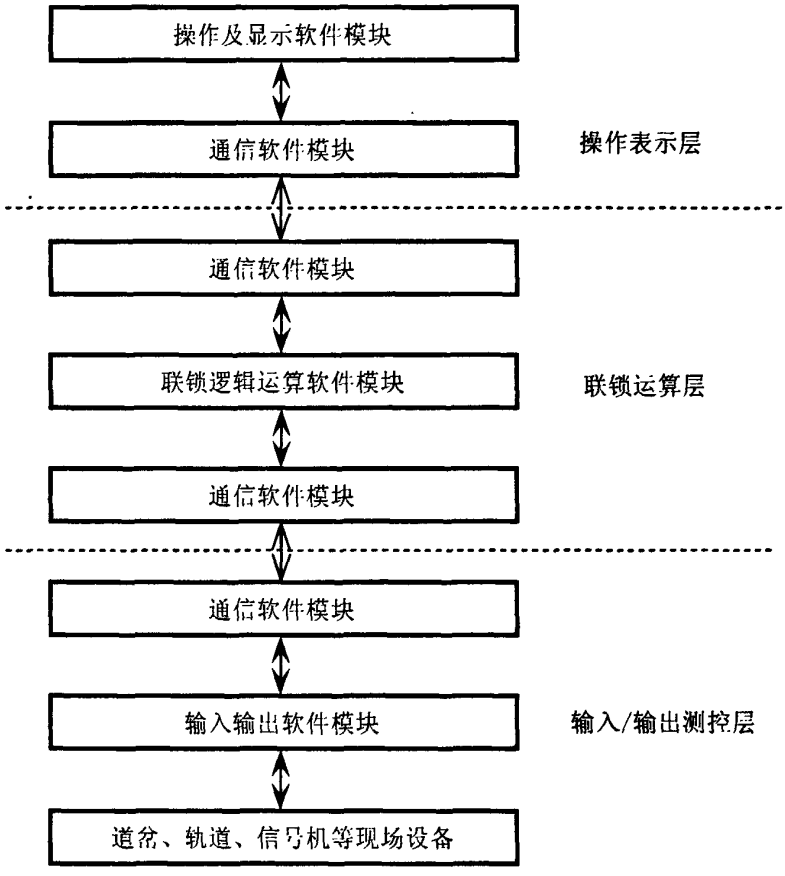


图4-1 股道管理自动化系统软件的总体结构

4.1.2 股道管理自动化系统对现场总线的基本要求

1、要有简单可靠的信息通道

这不仅关系到信息能否畅通，也关系到工程实用性，特别对铁路信号和工务系统来说，可靠的通信系统是现场设备和系统工作的保证。

2、要有灵活方便的设备和系统操作能力

铁路系统是一个庞大的系统，因此要求应用现场总线技术所提供的设备和系统的操作必须灵活方便。

3、要有可靠的信息安全技术保证

现场总线的基本特征就是以信息为基础，因此对于工业控制系统来说信息安全十分重要。

4、要有系统故障安全特性

铁路系统运行对各种设备设施的基本要求之一就是设备和设施具有故障安全特性，这是保证列车安全运行的基本要求。

5、要有在恶劣环境下稳定工作的能力

这不仅包括自然因素还包括诸如电气化铁路的电磁环境列车运行引起的振动等。[57]

4.2 基于CAN总线的股道管理自动化系统的特点

实现分散的铁路信号现场设备智能化控制，网络化连接，使联锁功能集中控制，执行设备分散自治，设备故障进一步分散、透明，便于站场改建和重新组态。具有以下主要优点：

1、技术先进。车站设备实现了网络化连接条件下的智能控制，易于与其它铁路指挥、管理、控制系统互联互接，实现信息共享，提高运输效率和行车指挥管理水平。

2、节约设备。控制设备站用技术房屋大幅减少，室外控制电缆使用减少，取消了大量的安全型继电器，不仅降低了安装成本，而且减少了维修工作量，便于故障分析和处理，有效减少了故障停时。

3、方便站场改建。站场的改变和设备的增加只需要对有关数据和组态文件进行修改，新增设备挂在现场总线上，接通电源即可完成，成本低，速度快，非常方便。

4、维修方便。现场智能设备即可以远程控制和故障诊断维修，也可以在本地通过便携维修仪进行故障诊断和处理。

5、组态方便。股道管理自动化系统的通信系统是一个独立的，它不随联锁关系及站场的形态大小而改变，当通信系统组态完成后，采用的设备只于传输介质相关，其嵌入的通信软件是标准和通用的，便于系统的维修和组态。

6、通用性。现场设备控制模块的硬件结构是通用和可互换的，但对于不同的现场设备，接口模块是不同的应用软件也是不同的。

股道管理自动化系统是当今铁路信号车站控制系统智能化的基础设备，是原有继电联锁设备的更新换代产品，是今后智能交通运输的发展方向。新型以现场总线为纽带组成的网络化车站信号控制系统是以智能设备为基础，结合计算机技术、网络技术、现代通信技术及电力电子技术的结晶，总体的

设计思想是相关安全的联锁功能集中在工业控制层，现场信号机电设备是现场智能终端实现高度自治控制，智能终端对现场设备进行控制和监督，对收集到的相关信息通过控制器局域网反馈到联锁层，联锁机在对人机对话层下传的信息进行处理时，结合智能终端的表示信息形成相关控制命令，并交由网络传输层下传到相关智能终端。[57]

4.3 系统设计

4.3.1 系统拓扑结构的设计

系统网络拓扑结构如图4-2所示，每台联锁机上配置有两块PCI总线接口的CAN适配卡，每块适配卡为四个通道，共8个通道的CAN总线物理链路。由于CAN总线的总线驱动能力所限，一个网络最多可容纳110个节点，而系统的节点数较大，故分为四种类型的双冗余CAN总线网络：道岔组合架总线网络、信号组合架网络、轨道电路组合架网络和零散组合架网络。每个组合架为双CAN冗余总线结构。其中，组合架上的智能I/O板模块示意图如图4-3所示。每一个智能I/O板有两个CAN端口，分别连接联锁机的两块CAN适配卡。智能I/O板之间通过IIC读取铁电中的数据获得本板的CAN节点ID。MCUA和MCUB通过两个端口的高低电平判断自己是A还是B。智能I/O板之间还可通过串口和IIC进行信息的交互。

在运行过程中，只要联锁机有一个CAN适配卡是好的，仍可正常工作。MCUA、MCUB只要有一个是好的，仍可对现场设备进行采集和控制。

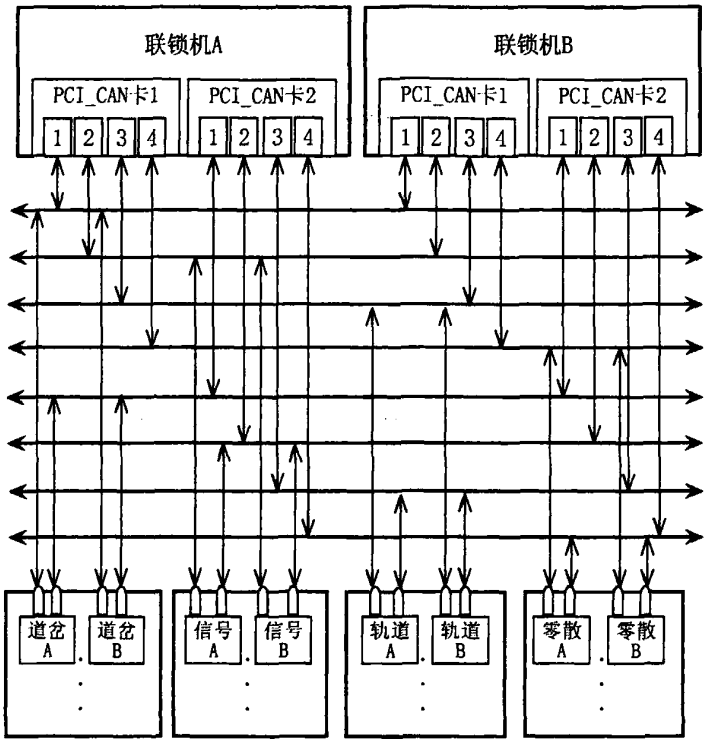


图4-2 CAN网络拓扑结构

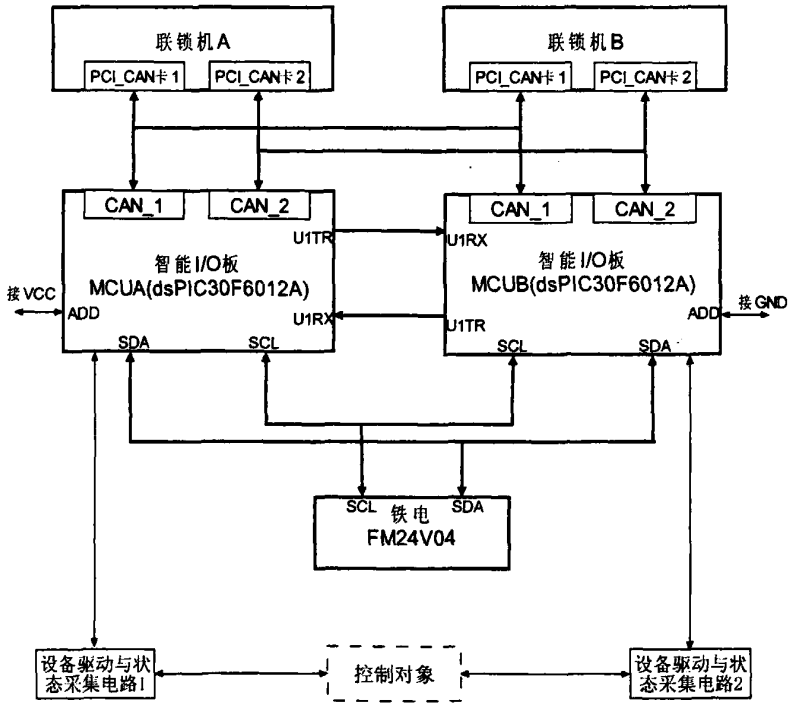


图4-3 智能I/O板模块示意图

4.3.2 系统的软件设计

4.3.2.1 系统软件的层次结构

系统软件的层次结构分为四层。CAN驱动层、CAN实现层、抽象层和CANopen应用层。上层的应用层与底层通过抽象层分离。CAN驱动层、CAN实现层的实现与上层处理采用哪种协议实现是无关的。系统软件的层次结构见图4-4所示。下文的表述主要侧重于联锁机的实现，智能I/O从节点的的实现较简单，与联锁机大同小异，不再赘述。

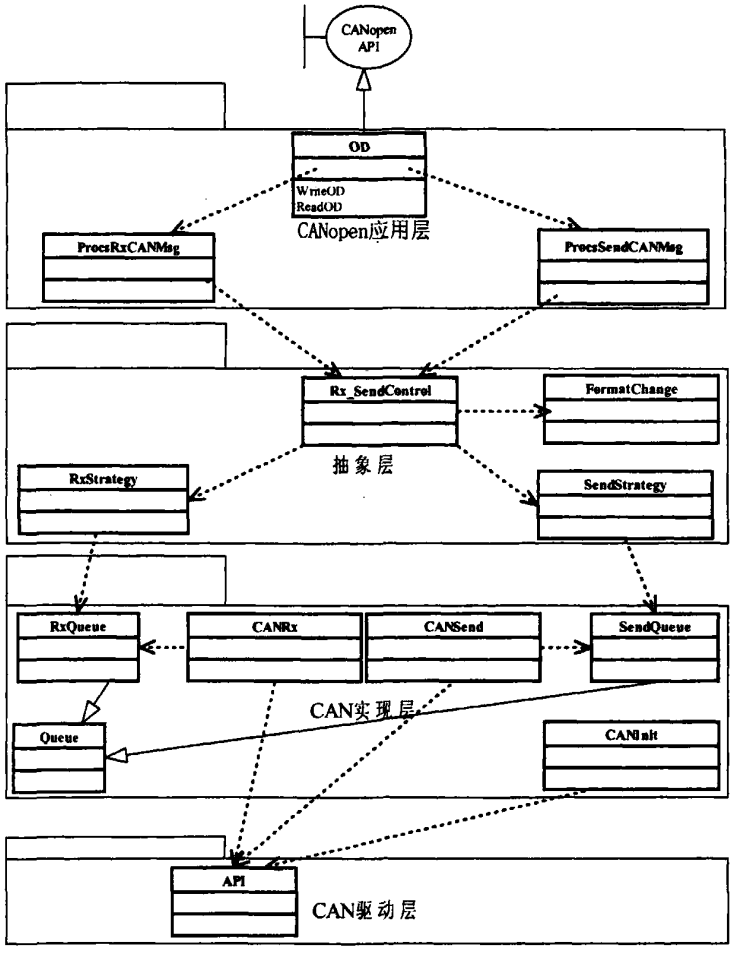


图4-4 系统软件的层次结构

4.3.2.2 CAN 报文的帧格式

本系统中的数据帧采用CAN2.0B扩展帧格式。CAN标识符的分配为：把29位的标识符分成2个部分，即功能段和目的节点ID，如表4-1所示。节点ID0—ID19根据组合架上智能I/O组合板的位置所定。功能码的定义如表4-2。功能码的大小根据信息的优先级别分配，其优先级别越高，功能码越小。数据部分根据需要分配，如智能I/O组合架向联锁机发送的状态帧格式为:索引+子索引+数据+CRC校验信息。

表4-1 标识符分配方案

功能段	节点ID（Node ID）				
ID28-ID20	ID19-ID16	ID15-ID12	ID11-ID8	ID7-ID4	ID3-ID0
功能码	组合柜行号	组合柜序号	层序号	控制板序号	控制端口序号

表4-2 功能段定义

功能码	信息类别	功能码	信息类别	功能码	信息类别
000000100	故障信息	000001000	心跳	000010010	命令
000010000	命令应答	000100000	状态变化	001000010	状态查询
001000000	节点保护	001000000	状态应答		

4.3.2.3 系统中定义的一些主要的数据类型

- 索引及数据：

```
typedef struct
{
    USDWORD     index;      // 主索引
    USBYTE      count;      // 子索引数量
    Odsubindex  *psub;      // 子索引表首地址
}OD_INDEX;
```

- 子索引及数据：

```
typedef struct
{
    USBYTE      subindex;
```

```
    USBYTE      acctype;    //访问类型
    USBYTE      size;
    union
    {
        USBYTE  bdata[4];
        USSHORT sdata[2];
        USDWORD idata[1];
    }data;
    USSHORT  (*pfunc)();    //回调函数
}OD_SUBINDEX;
```

- 扩展帧帧头格式：

```
typedef  struct
{
    USDWORD unuse    :3;    //未用
    USDWORD deviceid :20;   //设备地址
    USDWORD funid    :9;    //功能码
} CANID;
```

- CAN数据帧格式：

```
typedef  struct  _VCI_CAN_OBJ{
    USDWORD CANID;
    USDWORD TimeStamp;
    BYTE    TimeFlag;
    BYTE    RemoteFlag;     //是否是远程帧
    BYTE    ExternFlag;     //是否是扩展帧
    BYTE    DataLen;        //数据长度
    BYTE    Data[8];
    BYTE    Reserved[3];
} CANOPEN_FRAME,*PCANOPEN_FRAME;
```

- 缓存区格式:循环队列

```
typedef struct
{
    INT         Head;
    INT         End;
```

```
    CANOPEN_FRAME    Buf[MAX_NUM];
}BufData;
```

4.3.2.4 CAN 实现层

“CAN实现层”的功能实现较简单。初始化模块，就是实现底层收发的一些参数的设定，如速率、屏蔽码、校验码等；接收模块就是把各个端口收到的数据放入对应的接收队列；发送模块提供函数接口，把指定的数据发送到指定的网络。

对于接收，要提供一个静态的变量，每次收到数据，就给此变量加一。另外的一个函数定时清零此变量。通过此方法可以判断此接收端口是否接收正常，如果长时间没有收到数据，调用初始化对其进行修复。这样，也就实现了底层硬件的自制功能。

关于发送，网络控制系统的发送调度方法主要包括基于优先级别的调度方法、基于节点发送时间间隔的调度方法以及基于死区的调度方法等。基于优先级别的调度方法主要有静态调度(Static Scheduling)和动态调度(Dynamic Scheduling)。在静态调度（又称离线调度）中，各节点任务是预先确定的，如时限、计算时间、优先权关系和任务的释放时间等。如RM(Rate Monotonic)及其衍生算法属于静态调度法。静态调度法是缺省的调度算法，在任务执行前就已经确定了各任务的优先级，不会随时间变化。利用静态调度算法可以设置某些节点在其它节点之前多次访问网络。动态调度法（又称在线调度）是在网络控制系统运行中，网络资源的分配，根据网络资源的分配，根据任务的某些特性随时间的变化，动态地调整任务之间数据的发送顺序。如EDF (Earliest Deadline First)是最优的动态调度算法。TOD技术与优先级控制技术相结合，可以实现动态的调度策略。[58]

4.3.2.5 抽象层

“抽象层”是“实现层”与“应用层” 分离的关键，是系统实现的纽带，其作用不可忽视。其主要的功能有：进行上层与下层传输数据的格式转换；对于应用层发送的数据，分发到不同的网络；对于多端口接收的数据进行合并，包括相同的帧和功能相同的帧，然后交由上层处理。

4.3.2.6 应用层

“应用层”主要有实现系统“依赖倒转”的对象字典。上层应用通过读写对象字典与系统进行信息的交互。对于命令的发送与接收、实时查询的实现以及运行中的变化信息的及时上报等通过回调函数实现。还有对于接收的数据进行处理和需要发送的数据进行编帧。其心跳的处理是实现对于网络管理的关键。其中处理接收到的数据帧的核心代码如下：

```
void PrcsRxCANMsg(CONOPENFRAME cof)
{
    BYTE FUNID;
    FUNID = cof.CANID.funid;
    switch(FUNID)
    {
        case  lifeguardfunid:
            PrcsRxLifeguard(cof);    //处理节点保护
            break;
        case  errfunid:
            PrcsRxErr(cof);          //处理故障
            break;
        case  heartbeatfunid:
            PrcsRxHeartbeat(cof);    //处理心跳
            break;
        case  cmdfunid:
            PrcsRxCmd(cof);          //处理命令
            break;
        case  cmdanswerfunid:
            PrcsRxCmdAnswer(cof);    //处理命令应答
            break;
        case  statequeryfunid:
            PrcsRxStatequery(cof);   //处理状态查询
            break;
        case  stateanswerfunid:
```

```
            PrcsRxstateAnswer(cof); //处理状态应答
            break;
        case  statechangefunid:
            PrcsRxstateChange(cof); //处理状态变化
            break;
        default:
            break;
    }
```

4.3.2.7 上层应用接口

最后，把整个关于CAN网络相关的模块做成DLL。DLL提供的接口包括：如何初始化系统；如何切换系统工作的模式，包括进入主工作模式还是备工作模式；如何创建系统运行的线程，包括底层的收发和上层接收的处理与发送；如何获得网络的运行状况，包括各个CAN端口是否收发正常、如何控制现场的设备以及查询现场设备的工作状态。

4.3.2.8 软件抗干扰设计

- 总线传输的信息进行软件 CRC 校验。
- 采用不对称编码表示涉及安全的信息。
- 关键数据异地多份存储。如智能 I/O 节点 ID 存储于铁电中的不同区域，读取时，多个进行比较获得。
- 软件陷阱的设计，包括地址陷阱、指针陷阱、时钟陷阱和内存陷阱。
- 实时运行一个检验软件，实时检查数据的正确性。
- 在读写对象字典的时候要进行互斥的操作，防止脏数据的读取。
- 看门狗的设计。当系统由于干扰导致运行紊乱时，由外部看门狗电路使系统自动恢复。

4.4 系统测试与验证

为了测试系统，搭建了以合肥客机段股道管理自动化系统为模型的调试平台。测试界面如图4-5所示。测试软件基于.NET平台开发，可以控制PCI卡的连接、启动、关闭，可以设置系统通讯的参数,可以接收并显示收到的数据，可以向各个网络发送数据，可以通过接收的数据显示出设备的实时状态。

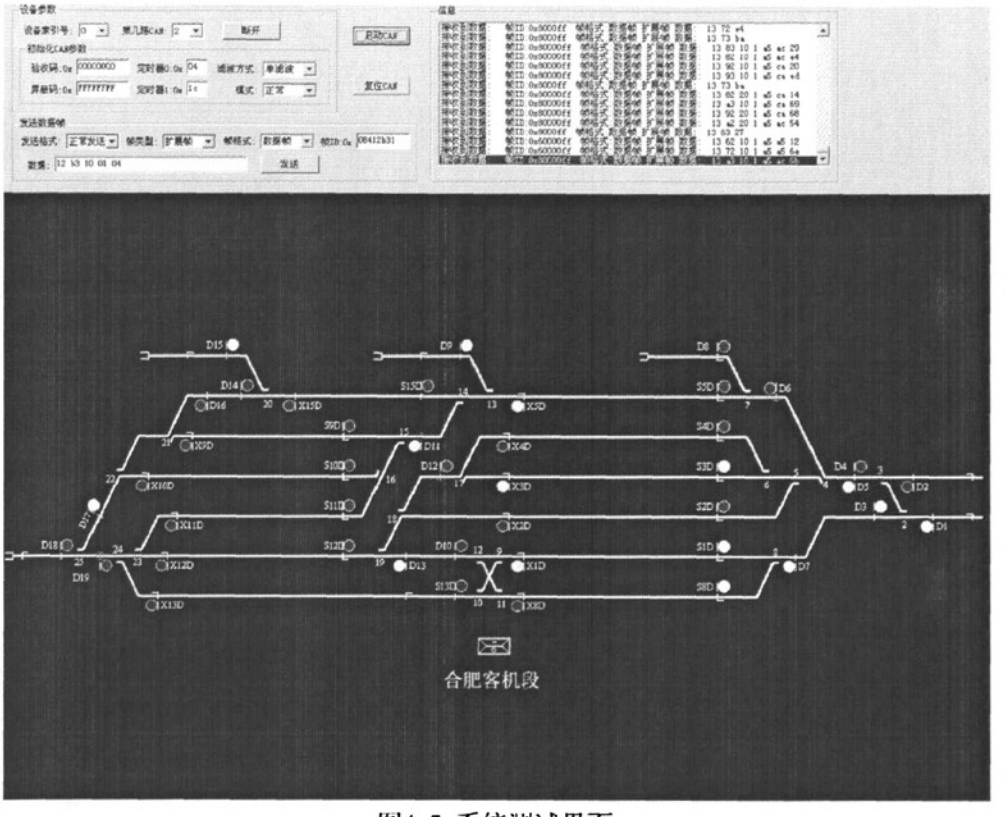


图4-5 系统测试界面

结　论

本文以CANopen协议和冗余为研究对象，主要的研究工作包括以下两个方面：

1. 本文解决了CANopen协议在实际应用系统中实现的关键技术（冗余和实时）。然后应用模块化、结构化的设计思想，把基于CANopen协议的冗余系统划分为：硬件层、抽象层和协议层,并分别进行实现。特别在协议层的设计中，充分依据“开-闭”原则，对各个模块的消息语法细节、主要数据结构及其程序实现进行了设计。首次提出了一套应用CANopen协议管理冗余网络的管理机制。
2. 最后，基于上述研究成果，实际开发了机务段股道管理自动化系统的CAN通讯模块。搭建了以合肥客机段为模型的调试平台。经检验，该模块的实时性强、可靠性高，且具有良好的可移植性与通用性。

未来工作的展望：

1. CAN总线通讯系统中不同设备间时钟同步的研究。
2. CAN总线通讯系统中不同设备间心跳周期的选取，心跳与PDO发送比例的关系。
3. 主节点的实时调度。

由于本人水平有限，对课题的研究开发还有待深入，缺点和不足之处在所难免，恳请各位老师提出宝贵的指导意见。

致 谢

首先，由衷的感谢我的导师王玉松老师！在这三年多的时间里，王老师深厚全面的学术素养，渊博的学识、严谨的治学态度使我受益非浅；敏锐的洞察力、高屋建瓴的全局控制能力、谦和而不厌其烦的处事风范，使我高山仰止，无尽感怀。在这期间，王老师在生活上的关心与照顾，为人处事上的谆谆教导更将让我终身难忘。在我做论文期间，从论文的选题开始，直到论文的定稿，王老师所付出的心血，让我切身体会到“师者，父母心”这句话的真正含义。在此论文完成之际，特向王老师表示深深的谢意！

同时，感谢何鸿云老师、李家武老师、朱金陵老师和龚南平老师给予我实践的机会。感谢周美玉老师、许志淳老师、陈安邦老师对我在学习、生活中的指导，营造了中心良好的学习氛围。感谢中心所有兄弟姐妹们三年来给予我的关心和帮助，特别感谢中心05届的师兄、师姐：宋红霞、周强、况长虹、赵丽娟、王利峰、危华进和白亚玲。

感谢钟剑老师、吴勇老师、李玲、吴鹏、陈星、徐雁、刘杰、刘成林等，在运达创新科技有限公司实习的这段时间，他们给予了我很大的帮助和指导。感谢项目组成员的大力协助，与他们共同的学习和进步是我一生中难忘的经历。

最后，感谢父母、女友、亲朋们对我一如既往的关爱，他们的关爱，是我一生的动力。

参考文献

[1] 谢希仁.计算机网络. 第4版，电子工业出版社，2006
[2] 阳宪惠.现场总线技术及其应用.清华大学出版社，1998
[3] 阳宪惠.工业数据通信与控制网络.清华大学出版社，2003
[4] 邬宽明.CAN总线原理和应用系统设计.北京航空航天大学出版社,1996
[5] 史久根,张培仁,陈真勇.CAN总线系统设计技术.国防工业出版社,2004
[6] 阎宏.JAVA与模式.电子工业出版社，2002
[7] 严蔚敏，吴伟民.数据结构（C语言版）.清华大学出版社，2007
[8] 华兴.排队论与随机服务系统.上海翻译出版公司，1987
[9] 晨风.嵌入式实时多任务软件开发基.清华大学出版社，2004
[10] 吴炜煜.面向对象分析设计与编程.清华大学出版社，2007
[11] 徐洪泽，岳强等.车站信号计算机联锁控制系统——原理及应用.中国铁道出版社，2005
[12] 邵贝贝.嵌入式实时操作系统uC/OS-II.第二版.北京航天航空大学出版社，2003
[13] Jane W.S.Liu.REAL-TIME SYSTEMS.高等教育出版社，2005
[14] Joseph Schmuller.UML基础、案例与应用.李虎，赵龙刚.人民邮电出版社，2007
[15] Ian Sommerville.软件工程.程一剑，陈霞.机械工业出版社，中信出版社，2003
[16] Martin Fowler.UML精髓.徐家福.第三版.清华大学出版社，2005
[17] Andrew S.Tanenbaum，Vrije Universiteit，Amsterdam，The Netherlands.计算机网络.潘爱民.第四版.清华大学出版.2004
[18] 门维江.网络系统设计的一般步骤和应遵循的原则.甘肃教育学院学报.2000第14卷第4期
[19] 赵一心，李鉴宝.双网通信系统的设计与实现.计算机与数字工程.1997，第25卷第3期
[20] 史久根等.CAN数据传送的实时性研究及其应用.信息与控制.2004（3）
[21] 谢斌,高扬. Linux高可用集群心跳机制研究.计算机工程与应用. 2004
[22] 邓遵义，宁祎.基于CANopen协议的主节点通讯实现.微计算机信

息.2008，第24卷第8-2期

[23] 程坷飞等.基于CANopen协议的车载平台数据通讯系统.嵌入式与SOC.2005

[24] 孙健，陶维青.CAN应用层协议CANopen浅析.仪器仪表标准化与计量.2006.2

[25] 吴爱国,刘莉.CAN总线控制系统的应用层协议CANopen剖析.微计算机信息.2003

[26] 杜陶钧，黄鸿.模块化设计中模块划分的分级、层次特性的探讨.机电产品开发与创新.2003年第2期

[27] 郭远东，黄荣瑛，陈友东，王田苗.基于模块化设计的嵌入式软件测试方法.单片机与嵌入式系统应用.2005年，第1期

[28] Olaf Pfeiffer, Andrew Ayre, Christian Keydel. Embedded Networking with CAN and CANopen. RTC Books. 2003

[29] M. Farsi, K. Ratcliff, Manuel Barbosa. An introduction to CANopen. Computer & Engineering Journal. 1999:161～168

[30] K. Etschberger, R. Hofmann, A. Neuner, U. Weissenrieder, B. Wiulsroed .A Failure-Tolerant CANopen System for Marine Automation Systems. IXXAT. 2001

[31] BOSCH. CAN Specification 2.0 PartA. 1991

[32] BOSCH. CAN Specification 2.0 PartB. 1991

[33] SUP. CANopen 8 channel Multifunction Digital Input Node. 2006

[34] SUP. CANopen 8 channel Current/Voltage Input Node. 2006

[35] PHILIPS Semiconductors. CAN Specification V2.0. 1991

[36] H. Boterenbrood. CANopen high-level protocol for CAN-bus v3.0. 2000

[37] CANopen Application Layer and Communication Profile. CiA Draft Standard 301. V4.02, 2002

[38] CANopen Framework for CANopen Managers and Programmable CANopen Devices. CiA Draft Standard Proposal 302. V3.2.1. 2003

[39] CANopen Layer Setting Services and Protocol. CiA Draft Standard 305. V1.1.1. 2002

[40] CANopen Electronic data sheet specification for CANopen. CiA Draft Standard 306. V1.3. 2005

[41] CANopen Framework for Maritime Electronics. CiA Draft Standard 307. V1. 0. 1. 2002

[42] CANopen Device Profile for Generic I/O Modules. CiA Draft Standard 401. V2. 1. 2002

[43] 姜晓东,许正华等. 基于网络和主机负载的变速心跳机制的实现方法. 专利. 2004

[44] 陆立,梁柏青等. 一种网络实体分别独立控制协议心跳的实现方法. 专利. 2006

[45] Thomas Nolte, Hansson, Christer Norstrom. Minimizing CAN response-time jitter by message manipulation. Real-Time and Embedded Technology and Appliations Symposium. 2003

[46] CANopen high-level protocol for CAN-bus. NIKHEF internal documentation. 2000

[47] 宋晓强. CAN-bus高层协议CANopen的研究以及在模块化CAN控制器上的实现. 天津大学硕士学位论文. 2004

[48] 傅海滨. 公交车车载智能系统的解决方案. 华东师范大学硕士学位论文. 2006

[49] 熊瑞平. 面向网络化制造的智能监控技术研究. 四川大学博士学位论文. 2006

[50] 方来华. 网络控制系统分析、建模及控制研究. 天津大学博士学位论文. 2005

[51] 窦连旺. 网络控制系统建模、稳定性分析及其调度的研究. 天津大学硕士学位论文. 2003

[52] 胡庆平. 新型心跳监测技术的研究与实现. 华中科技大学硕士学位论文. 2004

[53] 刘海花. 基于CANopen的智能监控下位系统研究. 河北工业大学硕士学位论文. 2007

[54] 申江. 电力机车DCU软件的模块化研究与设计. 西南交通大学硕士学位论文. 2007

[55] 李敏. 列车DC600V三相逆变电源研究. 西南交通大学硕士学位论文. 2007

[56] 周强. 双机热备计算机联锁系统软件的设计与实现. 西南交通大学硕士学位论文. 2008

[57] 张太卿.基于CAN总线的全电子化微机联锁系统设计. 西南交通大学硕士学位论文.2005

[58] 任旭东.网络控制可并行化方法及其调度管理.浙江大学硕士学位论文.2004

攻读硕士期间发表的论文

[1] 周跃峰，王玉松等. 一种基于 CANopen 协议的微机联锁总线控制解决方案. 工业控制计算机. 已录用