华中科技大学

硕士学位论文

CANopen协议在伺服系统中的软件实现与植入研究

姓名：刘思捷

申请学位级别：硕士

专业：控制理论与控制工程

指导教师：李叶松

2011-02-23

摘 要

近年来随着微电子技术，计算机技术的发展，伺服系统在提高控制精度的同时，也在向网络化控制的方向发展。分布式伺服系统作为常见的运动控制系统，在实际的工业生产线，数控机床系统以及机器人产品中都被广泛应用。

本文首先介绍了分布式交流伺服系统的发展概况，对交流伺服系统的网络化控制实现方法进行了分析。然后以 CAN 总线应用层协议 CANopen 作为网络协议，采用软件植入的实现方案，研究了在实时控制系统软件框架中加入 CANopen 协议模块的方法。针对软件植入方案的要求，对 CANopen 协议进行了深入的分析，分析了实现该方案需要解决的主要问题，根据实际的软硬件平台、控制软件的参数结构以及程序流程，设计了协议植入软件模块。在原有硬件平台上，首先利用 TMS320F2812 型 DSP 的 CAN 总线模块，使用该部分的邮箱管理功能实现了 CAN 报文的分类处理，利用同步和异步收发的功能，完成了 CANopen 通信协议栈的植入；然后结合原有驱动器的参数结构，采用了高效的对象字典数据存储结构和索引算法，在避免参数二义性的前提下实现了 CANopen 对象字典；最后利用原有驱动器的控制程序，结合 CANopen 通信协议和对象字典，实现了 CANopen 运动控制子协议 DS 402。同时以 PC 机和 PCI 扩展卡为硬件平台，在 Windows 下编程实现了简易的 CANopen 主站功能，对从站的进行了实验。实验结果证明了从站正确的实现了 CANopen 协议。

关键词：分布式伺服系统　　CANopen 协议　　软件植入

Abstract

In recent years, with the development of electronic and computer technology, servo system is improving its control accuracy, and developing to the direction of network control at the same time. Distributed servo system is a kind of common motion control system, which is widely used in assemble lines, numeric controlled machine tool, robots and so on.

This thesis firstly introduces the overview of servo system development, and analyzes the realization of network-controlled distributed servo system. Then, CANopen protocol is adopted as network protocol, and software-implant way is taken to realize CANopen protocol onto an existing servo system. According to the program, the protocol is analyzed thoroughly, the main problems are figured out, and the module design is worked out based on the software and hardware platform. With the original platform, this thesis firstly uses eCAN module and CAN mailbox of TMS320F2812 DSP to realize CAN message sorting and handling. Both synchronous and asynchronous function are used to realize CANopen communication protocol; Secondly, with original system parameters, efficient data structure and algorithm are designed for dictionary, and standard CANopen Object Dictionary is built up under the premise of avoiding dictionary ambiguity; Finally, based on existing motion control module and CANopen Object Dictionary, CANopen DS 402 slaver protocol is realized onto the original servo. On the other hand, this thesis also realizes CANopen master protocol on PC machine with PCI-CAN card, which is programmed in Visual C++. At the end, this thesis takes an experiment, the result of which confirms the correctness of the CANopen device.

Key words: Distributed servo system, CANopen protocol, software implant

独 创 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，除文中已经标明引用的内容外，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，均已在文中以明确方式标明。本人完全意识到，本声明的法律结果由本人承担。

学位论文作者签名：

日期：2011 年　月　日

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，即：学校有权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。本人授权华中科技大学可以将本学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文。

本论文属于

保　密□，在____年解密后适用本授权书。

不保密□。

（请在以上方框内打“√”）

学位论文作者签名：　　指导教师签名：

日期：2011 年　月　日　　日期：2011 年　月　日

1. 绪论

1.1. 现代智能化交流伺服系统的应用背景

伺服系统，又称位置随动系统，是现代工业生产中广泛应用的典型的运动控制系统。其主要功能为对位置指令进行跟随响应，广泛应用于数控机床，工业自动化生产线，机器人等多个领域[1-4]。随着现代微电子技术的发展，现代微处理器(MCU)以及数字信号处理器(DSP)的运算性能和运行频率有了飞跃性的提升，而电力电子技术的发展，使得当今的电力电子器件有了更高的开关频率，作为交流伺服系统的执行器单元，能够提供更好的控制性能。上述的所有条件，使得电机控制理论中早就提出的交流永磁同步电机的矢量控制算法有了具有实践意义的实现方案。因此交流永磁同步电机在伺服系统中已经逐步取代直流电机，成为伺服系统的主要解决方案。

与此同时，随着现代通信技术，计算机技术的发展，伺服系统的功能也随之不断扩展。在以交流永磁同步电机为主要控制对象的现代交流伺服系统中，实现友好的用户交互功能，实现智能化控制，已经具备了可能性和必然性。是现代交流伺服系统发展的另外一个潮流。下图是典型的现代数控机床系统控制台的界面：

图 1-1 典型控制台界面

可以看到，现代数控机床控制系统可通过键盘接受用户输入设定，通过液晶屏显示当前多轴控制系统的各个轴的状态。这样的智能化交互式控制台有以下优点：1、易操作易上手，能减少使用者的适应时间和使用难度。2、友好而强大的用户界面，用户能直接通过显示屏来监控各个轴向的状态，便于用户掌握整个系统的运行情况，并且做出相应的指令。保证这些功能正确实现的基础，是现代交流伺服系统的合理系统构架以及完善的功能模块。下面将对现代交流伺服系统的系统构架和功能模块进行简单介绍。

1.2. 现代交流伺服系统的系统构架

现代交流伺服系统的主要控制构架与传统交流伺服系统相同，都是采用了三闭环的控制结构，内环电流环和速度环可以采用 PID 控制，而外环位置换则采用比例

控制，再加上前馈单元以减小位置环余差。其典型系统结构框图如下所示：

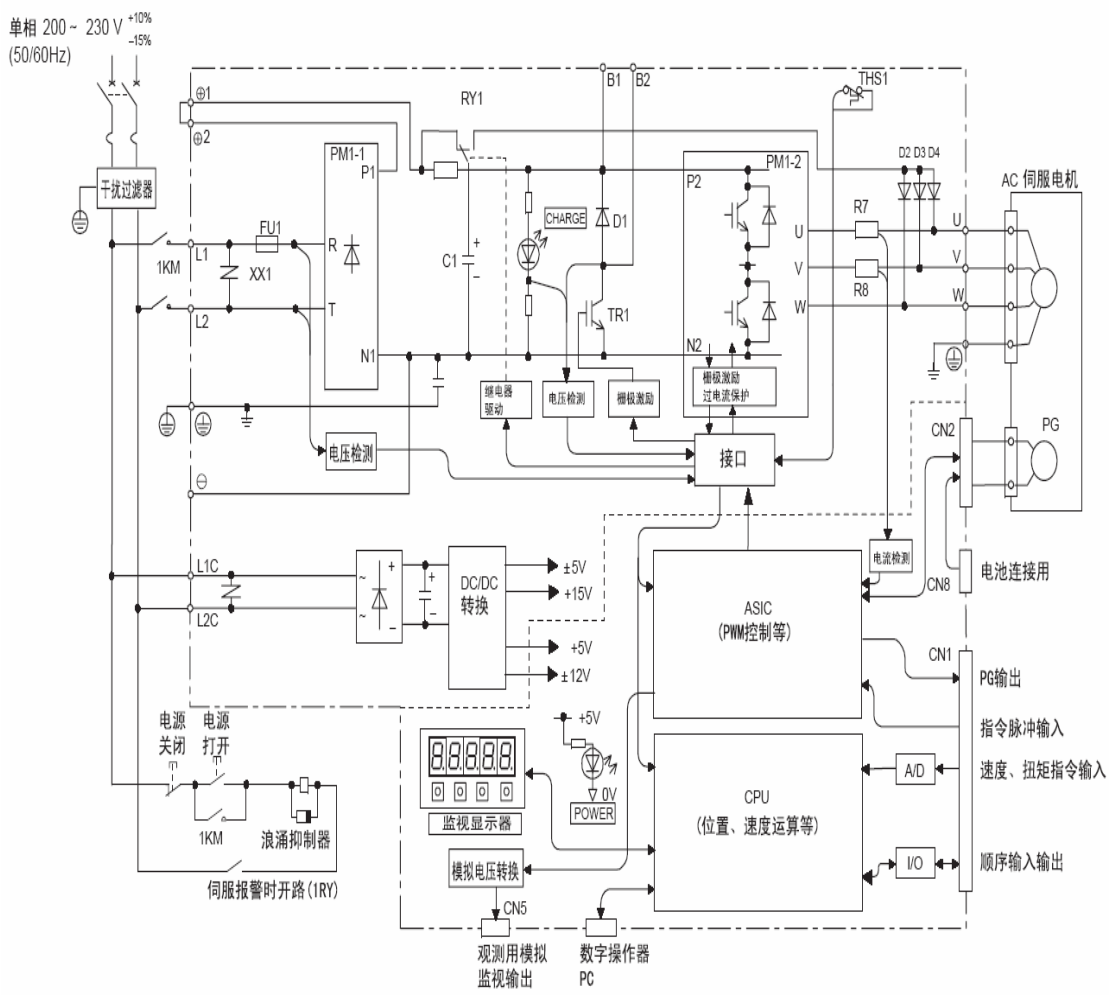

图 1-2 典型伺服系统结构框图

现代交流伺服驱动器在完成位置随动控制的同时，还要求加入用户可观测可操作的功能，常采用的实现方式是利用各种外设模块，例如键盘，LED，液晶屏等，来接受用户输入的指令，如位置指令，速度指令等，以及显示系统的参数和状态量。而一般情况下，出于成本的考虑，这样的外设模块通常都比较简单，例如上图所示系统，系统采用的参数显示模块是简单的八段式 LED。因此通常情况下外设能实现的用户交互功能非常有限。并且操作比较复杂，需要编写专门的操作手册供使用者进行参考。基于上述原因，一般的交流伺服驱动器所能完成的用户交互功能有限，而在实际应用中，大型多轴伺服系统往往包含大量的运动控制参数，需要进行统一管理，因此大多数交流伺服系统均采用了单主控台——多驱动器的系统结构。驱动

器主要负责位置随动系统的运动控制功能，而主控台的则集中负责参数管理，实现用户参数和命令的输入以及系统参数的显示。组成典型的分布式交流伺服系统系统。

1.3. 分布式交流伺服系统

1.3.1. 传统式分布式交流伺服系统

在传统的分布式交流伺服系统中，控制台和各驱动器的数据传输方式主要采用I/O 方式直接传输，伺服驱动器采用直接的方式将驱动器参数发送到控制台的 I/O 上。这种方式的好处是传输方式简单直观，而存在的问题主要有以下几个方面：

1) 设备间必须预先约定参数传输的格式。
2) 当需要更换其他型号的设备时时，将因为参数格式不匹配而无法使用。
3) 在一主多从的运动控制系统中，需要解决主设备 I/O 争用的问题。
4) 无法实现对高级的控制功能，例如对多个从设备的优先级控制等。

针对这些问题，采用现场总线解决 I/O 信息通道的管理问题成为一种有效的办法。

1.3.2. 基于现场总线的分布式交流伺服系统

现代分布式交流伺服系统多采用这种控制方式。分布式交流伺服系统一般采用单主控台——多驱动器的控制结构，而现场总线能够很好的实现这种控制结构下的数据交互。基于现场总线的分布式交流伺服系统的优点主要有如下几个方面：

1) 串行数字信号传输数据，抗干扰能力强，数据传输效率高。
2) 采用统一的物理层接口和信号电平，支持统一标准设备；数据链路层具有校验功能，确保传输的可靠性。
3) 采用标准的应用层协议，对网络报文进行统一处理，对设备参数的格式和结构进行统一定义，所有支持标准协议的设备之间可以兼容。
4) 可实现多从设备之间的同步操作，优先级操作等高级控制功能。

1.3.3. 现场总线简介

和传统的分布式交流伺服系统相比，基于现场总线的分布式交流伺服系统在数据传输，设备兼容以及控制功能几方面都具有明显的优势，因此逐渐成为现代分布式交流伺服系统的设备间通信的主要方式[5-10]。现在市面上主流的现场总线有以下几种：

1) RS485 串行总线

RS485 串行总线是美国电子工业协会 EIA（Electronic Industry Association）制定的一种串行物理接口标准，作为最常用的现场总线，和 RS232 串口的单工通信方式不同，RS485 串行总线采用了全双工的通信方式，可以支持多节点的总线式通信结构。RS485 串行总线标准成熟，且应用广泛，几乎所有的控制设备，如 PC 机，伺服系统控制台/驱动器，PLC 等都支持 RS485 串口通信。其主要缺点是数据链路层协议过于简单，且传输速度较慢[11-13]。

2) CAN 总线

CAN 总线是德国 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[14]。

CAN 总线主要定义于 ISO/OSI 的 7 层模型的物理层[15]，数据链路层，而一般的基于 CAN 总线的通信协议应用，都是直接利用 CAN 总线标准定义的物理层和数据链路层协议，然后自定义应用层协议，实现 3 层的通信模型。

与 RS232/485 总线相比，CAN 总线具有如下优势[16]

(1) 物理层信道传输波特率更高，信号传输距离更长。一般情况下 CAN 总线在信号在传输距离为 40m 以内时，支持 1M bps 的波特率。而 CAN 总线的最大信号传输距离更是达到了 10km。

(2) 数据链路层功能更加强大。在数据链路层，CAN 总线协议规定了 CAN 报文的数据帧格式，并且进行 CRC 校验以保证报文传输的完整性和准确性，CAN 总线报文的帧格式如下图所示：

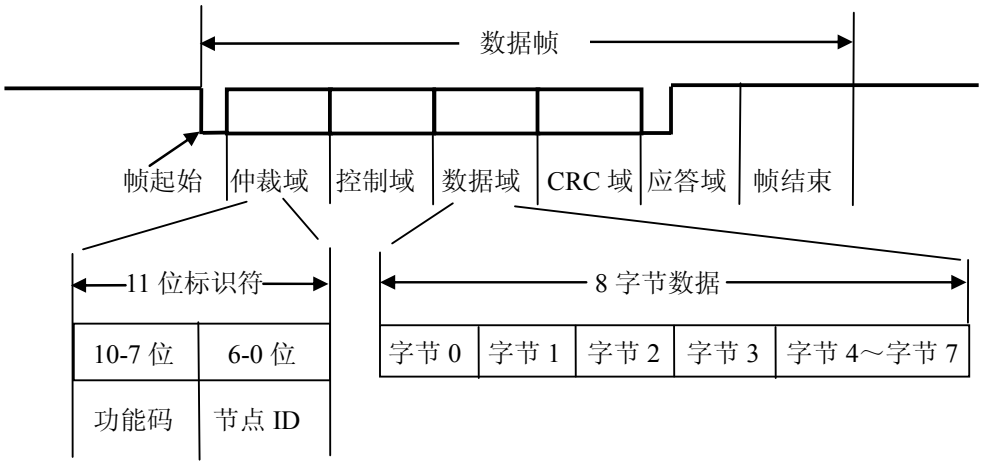

图 1-3 CAN 报文帧格式

相比之下，RS323/485 串行总线只定义了每个字节的传输格式，而对于数据帧格式的定义和数据帧起止的甄别则要由应用层协议实现；对于数据帧的校验也由应用层协议实现。CAN 总线则将数据帧的相关操作的实现集成在总线控制芯片中，与 RS232/485 串行总线相比，减少了应用层软件的开销。

(3) 针对不同设备，可支持多种标准的应用层协议。其中 CANopen 协议作为广泛采用的 CAN 总线应用层协议，充分利用了 CAN 总线的优点，并且针对不同的设备定义了多种应用层子协议。其中 CANopen DS 402 协议作为运动控制设备专用协议，具有简单易用，功能强大的优点，已经被许多运动控制设备厂商采用为标准的 CAN 总线应用层协议[17-22]。

3) 工业以太网 EtherCAT[23]

EtherCAT 是开放的实时以太网络通讯协议，最初由德国倍福自动化有限公司 (Beckhoff Automation GmbH) 研发。和 CAN 总线相比，EtherCAT 具有更高的传输速率和更好的通信实时性，适用于对实时性有较高要求的运动控制系统，例如多轴联动系统。但是和 CAN 总线相比，EtherCAT 成本更高，因此在实时性要求不高的系统中，常采用 CAN 总线通信。EtherCAT 中常用应用层协议 COE 与 CANopen 协议类似。

本文所研究的分布式交流伺服系统的网络接口采用 CAN 总线，而应用层协议则采用 CANopen 协议。

1.4. 论文主要研究内容介绍

根据 CANopen 协议的结构，后文将对 CANopen 协议的内容进行详细分析，并且在已有伺服驱动器中实现 CANopen 协议。本文主要从以下方面来进行论述：

1) 详细分析 CANopen 协议的通信模型和主要内容，提出协议实现方案，分析实现该方案的主要工作内容。
2) 确定从节点伺服驱动器的软硬件平台，针对方案分析中所提出的要求，采用软件植入方法进行从节点通信协议的编程实现。
3) 开发主节点程序，使用主从式通信的方式，对 CANopen 协议规定的标准运行模式进行测试，验证系统设计方案的可用性。
4) 总结工作，对下一步工作进行展望，提出可以改进和优化的内容。

2. CANopen 协议及其实现方案分析

2.1. CAN总线应用层协议简介

CAN 总线标准协议只定义了物理层协议和数据链路层协议，没有定义应用层协议。在实际应用中，需要用一个应用层协议来定义 CAN 报文中的 11/29 位标识符和 8 字节数据的使用[24]。

常用的 CAN 总线应用层协议主要是 Devicenet 协议和 CANopen 协议。

Devicenet 协议由美国罗克韦尔公司提出，是基于 CAN 总线的应用层协议。协议定了 CAN 报文中 11/29 位标识符和 8 字节数据的使用方式，最多支持 64 个网络节点，支持 125kbps，250kbps，500kbps 等多种传输波特率，是一种早期提出的节约成本的 CAN 总线应用层协议。

CANopen 协议是由 CiA(CAN in Automation)提出的基于 CAN 总线的应用层协议，跟 Devicenet 协议相比，CANopen 协议具有以下几个方面的优势[25-28]：

1) 支持更多数量的节点，CANopen 协议最多可支持 127 个节点[29]。

2) 支持更高的波特率，CANopen 协议最高可支持 1Mbps 波特率通信

3) 根据不同类型的设备定义了多种专属应用层子协议，其中包含运动控制设备专用子协议 CANopen DS 402 协议。

基于 CANopen 协议的设备通信模型如下所示，从模型图可以看出，OSI 模型中的 Layer1 和 Layer2 已经分别由电缆和 CAN 控制芯片实现，而 CANopen 协议根据不同类型的设备定义了不同的应用层子协议，因此实现 CANopen 协议的主要工作就是实现这些应用层协议[30, 31]。

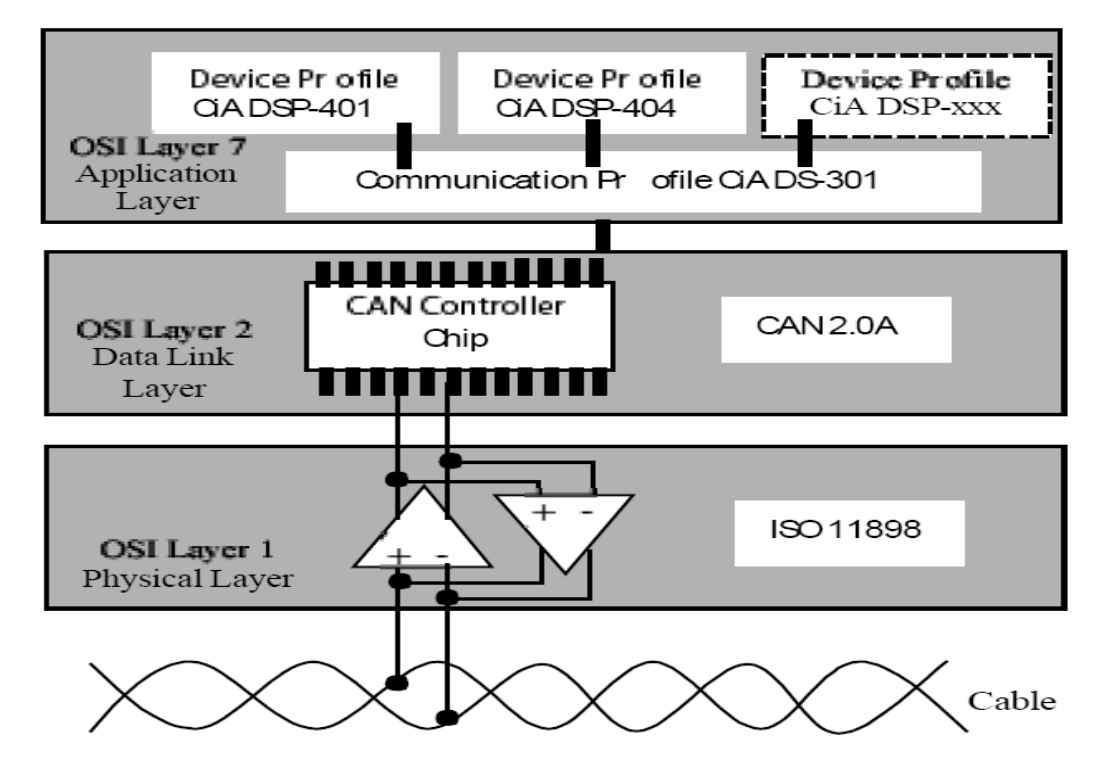

图 2-1 CANopen 通信协议模型

CANopen 协议与 Devicenet 相比有许多优势，因此 CANopen 协议是当前主流的 CAN 总线应用层协议。本文也将采用 CANopen 协议作为 CAN 总线应用层协议。下面将对 CANopen 协议进行分析。

2.2. CANopen协议分析

在提出实现 CANopen 协议的具体方案，并进行具体的设计工作之前，有必要先对 CANopen 协议进行详细的分析，进而结合原有系统的特点，确定应该采用何种方案实现 CANopen 协议，以及其中设计的细节工作和难点。下面首先从四个方面对 CANopen 协议的通信模型进行详细的分析。

2.2.1. CANopen协议报文处理

CANopen 协议作为 CAN 总线的应用层协议，主要对 CAN 报文中可用于报文控制的 11/29 位 CAN-ID 和 8 字节数据进行定义，实现应用层功能[10]。

CANopen 协议使用 11/29 位 CAN-ID 对报文进行分类，协议支持的报文有以下几类：

1) SDO（服务数据对象）报文[25, 27]。SDO 报文的主要用途是主从节点的数据交互。使用 SDO 报文进行通信时，接收报文的节点需要回送报文进行应答，因此 SDO 报文实现的是一种面向连接的可靠性通信。关于 SDO 报文通信的详细定义和处理流程将在后面讨论。

2) PDO（过程数据对象）报文[26, 27]。PDO 报文的主要用途与 SDO 报文类似，也是用于主从节点数据交互。与 SDO 报文的主要区别在于，使用 PDO 报文通信时，接收报文的节点不需要回送应答；另外 PDO 报文对数据的访问方式与 SDO 报文不同，PDO 报文的数据访问效率比 SDO 报文更高，因此 PDO 报文常用于需要频繁操作的数据，以及需要高效率传输的数据。PDO 报文可采用同步方式或者异步方式进行传输。当需要同步传输 PDO 报文时，设备的对象字典需要支持同步周期字典项(索引值 0x1006)。

3) NMT（网络管理）报文[25]。该报文的主要功能是进行网络管理，CANopen 主节点通过发送 NMT 报文控制从节点的启动，停止，运行等状态切换，从节点通过 NMT 报文(具体来说是心跳报文)来通知主节点自己上线或者下线。

4) 特殊功能报文[25]，主要包括同步报文，紧急事件报文，时间邮戳报文等。

CANopen 协议支持最多 127 个节点以及每个节点的四类报文，协议将 CAN 报文的 11/29 位 CAN-ID 分为两个域，称为 COB-ID(CAN Object ID)，其中 7 位为地址域，而 7 位地址域为 0 表示广播地址，因此最多支持 2^7-1=127 个节点。4 位为功能码，不同的功能码代表不同类型的报文。具体的 COB-ID 分配如下表所示[28]：

表 2-1 CANopen 协议预定义主/从连接集 COB-ID 分配表

对象	功能码 (ID-bits 10-7)	COB-ID
NMT 报文	0000	000H
同步报文	0001	080H
时间邮戳	0010	100H
紧急报文	0001	081H-0FFH

PDO1(发送)	0011	181H-1FFH
PDO1(接收)	0100	201H-27FH
PDO2(发送)	0101	281H-2FFH
PDO2(接收)	0110	301H-37FH
PDO3(发送)	0111	381H-3FFH
PDO3(接收)	1000	401H-47FH
PDO4(发送)	1001	481H-4FFH
PDO4(接收)	1010	501H-57FH
SDO(发送)	1011	581H-5FFH
SDO(接收)	1100	601H-67FH

针对不同类型的 CAN 报文，CANopen 协议定义了 8 个字节数据域代表不同的含义[27]。支持 CANopen 协议的设备的通信模型图如下所示：

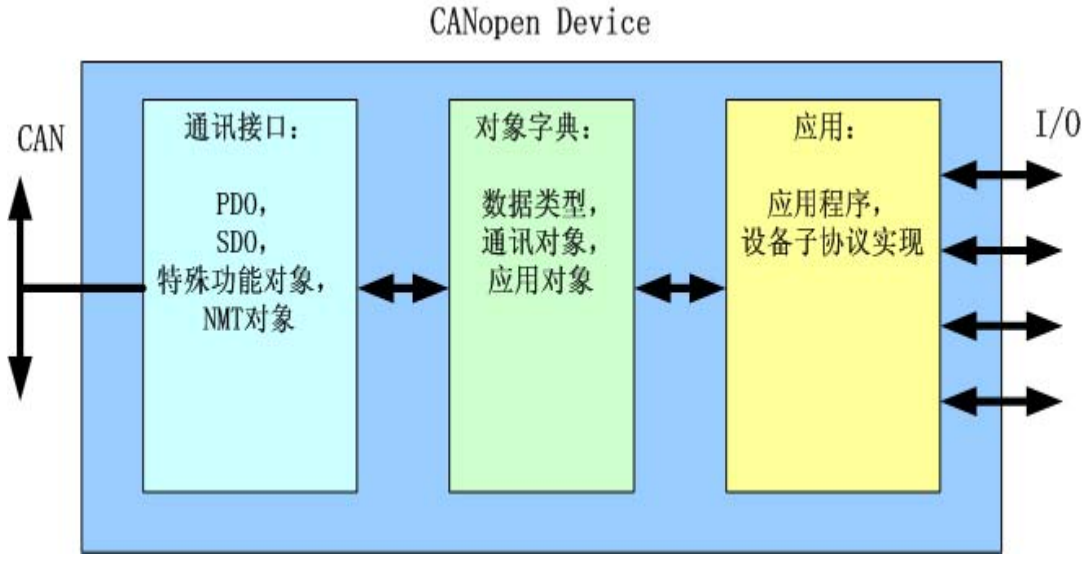

图 2-2 CANopen 设备通信模型

可以看到所有的 CAN 报文与 CANopen 设备之间的数据传输都需要通过中间数据区进行，CANopen 协议将中间数据区命名为对象字典。CANopen 协议规定的对象字典项的通用数据结构如下表所示：

表 2-2 CANopen 对象字典通用数据结构

Byte 0-1	Byte 2	Byte 3-6
16 位字典索引	8 位字典子索引	最多 4 字节的对象值

对 CANopen 对象字典进行操作需要按照索引表来索引目标对象，CANopen 协议规定了两种 CAN 报文访问对象字典的方式[27]：

1) 字典索引方式：该方式是 SDO 报文访问对象字典的方式。在该方式下 SDO 报文的 8 字节数据域被分为四个子域，如下表所示

表 2-3 直接索引方式 SDO 报文数据域定义

Byte 0	Byte 1-2	Byte 3	Byte 4-7
操作码	对象索引	对象子索引	对象数据

其中索引域，子索引域以及对象数据与 CANopen 协议对象字典的定义完全相同，而首字节的操作码则规定了许多复杂的字典操作。实际应用中主要使用了主节点读写从节点字典的操作码。具体来说：当主节点读取从节点字典项时，主从节点的 8 位操作码的位定义如下表所示：

表 2-4 主节点读从节点字典操作码位域定义

Bit0	bit1	bit2	bit3	bit4-5	bit6	bit7
0	1	0	0	n	1	1

其中两位二进制数 n 表示 SDO 报文后四字节中无效的字节数。

而当主节点写从节点字典项时，主节点和从节点的操作码位域所示如下：

表 2-5 主节点写从节点字典时主节点操作码位域定义

Bit0	bit1	bit2	bit3	bit4-5	bit6	bit7
0	0	1	0	n	1	1

表 2-6 主节点写从节点字典时从节点操作码位域定义

Bit0	bit1	bit2	bit3	bit4-5	bit6	bit7
0	1	1	0	n	1	1

其中 n 表示 SDO 报文后四字节中无效的字节数。

在使用 SDO 报文进行字典操作时，只需按照协议规定格式填充 8 字节数据域，并且根据具体的字典操作和对象数据长度得到相应操作码并填充即可。

2) 直接映射方式：该方式是 PDO 报文访问对象字典所特有的方式。采用直接映射方式时，需要对报文映射对象进行预定义，CANopen 协议对象字典中也有专门的对象用于存放 PDO 映射的预定义。下面举例说明具体的 PDO 映射过程。例如在对象字典中定义了如下的 RPDO1 的 8 字节数据映射表时：

表 2-7 PDO 映射表

PDO	占用的字节	映射的对象意义	索引值
RPD01 0x200+ID	2	控制字（Controlword）	0x6040
	2	运行模式	0x6060
	4	目标位置（Target position）	0x607A

若节点设备收到以下的 PDO 报文(假设设备的地址为 1)：

表 2-8 PDO 报文示例

COB-ID	Byte0-1	Byte2-3	Byte4-7
0x201	0x001F	0x0001	0x000A0000

设备节点将根据 RPDO1 的映射预定义，跳过对字典的索引过程，直接进行数据映射，将数据 0x001F，0x0001，和 0x000A0000 分别写入索引值为 0x6040，0x6060 和 0x607A 的对象中。

使用直接映射方式的优点是 CAN 报文数据利用率高，缺点是要预先定义 PDO 映射表，因此一般只对极少的对象采用直接映射的访问方式。在 CANopen 协议中每个设备支持 4 个 PDO 对象报文，能进行映射的字典对象有限。通常只对设备中读写操作最频繁的参数进行映射，这样做能最大优化通信数据传输的效率。

2.2.2. CANopen协议对象字典分配

如前文所述，CANopen 协议对象字典采用 16 位索引值+8 位子索引值+4 字节数据的结构，因此最多可有 65536 个字典项，每个字典项可包含一个或多个子对象。根据协议规定，每个包含多个子对象的字典项，其子索引为 0 的子对象的值就是该字典项包含的子对象个数，因此每个字典项最多可包含 127 个子对象。CANopen 协议对字典对象的索引值分配如下表[27]：

表 2-9 CANopen 对象字典分配表

索引值(HEX)	对象说明
0000	Reserved
0001-001F	标准数据类型
0020-003F	自定义数据结构
0040-005F	设备商定义的数据类型
0060-007F	标准设备规范(例如 CANopen DS 402)定义的静态数据类型
0080-009F	标准设备规范(例如 CANopen DS 402)定义的复杂数据结构
00A0-0FFF	Reserved
1000-1FFF	通信子协议(例如 CANopen DS 301)参数区
2000-5FFF	设备制造商参数区
6000-9FFF	标准设备规范(例如 CANopen DS 402)参数区
A000-BFFF	接口规范说明区域
C000-FFFF	Reserved

其中主要可供使用的是索引值为 0x1000-0x1FFF 的通信子协议区，索引值为 0x2000-0x5FFF 的设备制造商参数区，以及索引值为 0x6000-0x9FFF 的标准设备规范区。CANopen 协议支持一系列的子协议文档[27]，这些文档用于定义索引值为 0x1000-0x1FFF 的通信子协议区和索引值为 0x6000-0x9FFF 的标准设备规范区。而索引值为 0x2000-0x5FFF 的设备制造商参数区则主要由用户使用和定义。

2.2.3. CANopen通信子协议——DS 301 协议

CANopen DS 301 协议定义了 CANopen 协议对象字典的通信子协议参数区(字典索引值 0x1000-0x1FFF)，主要定义了 CANopen 协议设备的通信参数，包括节点设备号，主节点同步时间，从节点心跳报文时间，存储字典操作码等。以从节点心跳报文周期为例来说明：

表 2-10 字典对象——子节点心跳报文周期

索引 (hex)	子索引	数据类型	读写权限	取值范围	PDO 映射	默认值
0x1017	0	Uint16	R/W	-	否	10

该对象定义了从节点向网络发送心跳报文的时间，以毫秒为单位，数据类型为 16 位无符号数，而读写权限为 R/W，表示该字典项的访问权限为可读写；该字典项不被 PDO 映射，因此主节点可通过 SDO 报文，以字典索引的方式访问该字典项，而无法通过 PDO 映射的方式修改字典项。默认值为 10，在从节点初始化后，默认每 10ms 发送一次心跳报文，向主节点发送上线通知，直到主节点修改该字典项为止。

2.2.4. CANopen运动控制子协议——DS 402 协议

CANopen DS 402 协议定义了运动控制设备使用 CANopen 协议通信时，CANopen 对象字典的标准设备规范参数区(索引值 0x6000-0x9FFF)的内容。相对于通信子协议 CANopen DS 301，CANopen DS 402 协议对字典的定义更加复杂。

CANopen DS 402 协议规定了若干种标准运行模式，根据这些标准运行模式对对象字典进行分区和定义。这些标准运行模式包括[26]：

1) 轨迹位置模式。该模式的对象字典区定义了设备的位置轨迹规划参数，包括轨迹模式(梯形轨迹规划和 S 型轨迹规划选择)，目标位置(驱动器位置指令)，轨迹速度(梯形轨迹和 S 型轨迹中最大的速度值)，轨迹加速度(梯形轨迹和 S 型轨迹中最大加速度值)。

2) 轨迹速度模式。该模式的对象字典区定义了设备的速度轨迹规划参数，主要

包括目标速度，轨迹加速度，实际速度，速度环控制参数给定等。

3) 回零模式。该模式的对象字典区定义了设备的回零运行模式参数，包括回零模式，最大回零速度，最小回零速度等。

而以上标准运行模式之间的切换，以及设备运行时的运行状态切换，则需要通过设备的控制字/状态字来完成。这两个对象是 CANopen DS 402 协议所规定的标准对象，也是所有标准运行模式公用对象。不同的运行模式下，控制字/状态字有不同的位定义，在本文后面的章节将有详尽的叙述。这里首先列出它们在 CANopen 对象字典中的定义：

CANopen 协议控制字对象定义：

表 2-11 CANopen DS 402 字典对象——控制字

索引 (hex)	子索引	数据类型	读写权限	取值范围	PDO 映射	默认值
0x6040	0	Uint16	R/W	-	是	-

CANopen 协议状态字对象定义：

表 2-12 CANopen DS 402 字典对象——状态字

索引 (hex)	子索引	数据类型	读写权限	取值范围	PDO 映射	默认值
0x6041	0	Uint16	RO	-	是	-

CANopen DS 402 协议还定义了另一组重要的公用对象——指令运行模式(索引值 0x6060)以及实际运行模式(索引值 0x6061)。这个字典对象实际上采用了双缓冲的模式来存储设备的运行模式。其中指令运行模式(索引值 0x6060)用于系统的运行模式切换。当系统完成当前运行模式的最后一个用户命令，且控制字被修改，设备状态机允许运行模式切换时，设备将切换到指令运行模式。而在此之前，虽然主节点可以通过字典操作修改指令运行模式，但是可能由于设备当前运行模式尚未结束，不具备模式切换的条件，因此模式切换并未实际发生；此时若主节点查询设备当前运行模式，需查询实际运行模式对象(索引值为 0x6061)，该对象将返回当前系统的运行模式。该对象为只读对象，只有当系统完成模式切换，进入新的运行模式时，该对象才会被从节点设备自动更新，主节点无法更新该字典对象。

在对 CANopen 协议进行了详细的介绍后，下面将对实现 CANopen 协议的具体对象——交流伺服驱动器进行介绍，以便进一步分析实现 CANopen 协议的方案。

2.3. 交流伺服驱动器软硬件结构介绍

交流伺服驱动器的处理器一般都采用专用的 MCU 或者 DSP，这些 MCU 和 DSP 往往都是半导体厂商专门为运动控制系统定制推出，外设和处理单元都专门针对运动控制系统的功能要求进行优化。与其他的嵌入式系统相比，运动控制系统的处理器专用性强，而通用性比较差，并且需要同时考虑到成本和实时性的问题。因此一般情况下，交流伺服驱动器没有嵌入式实时操作系统的软件环境，这是运动控制系统和其他嵌入式系统的重要区别。交流伺服驱动器作为常用的运动控制系统，其内部软件需要在没有操作系统的环境下进行任务调度，保证控制的实时性。常见的交流伺服系统的顶层软件流程图如下所示：

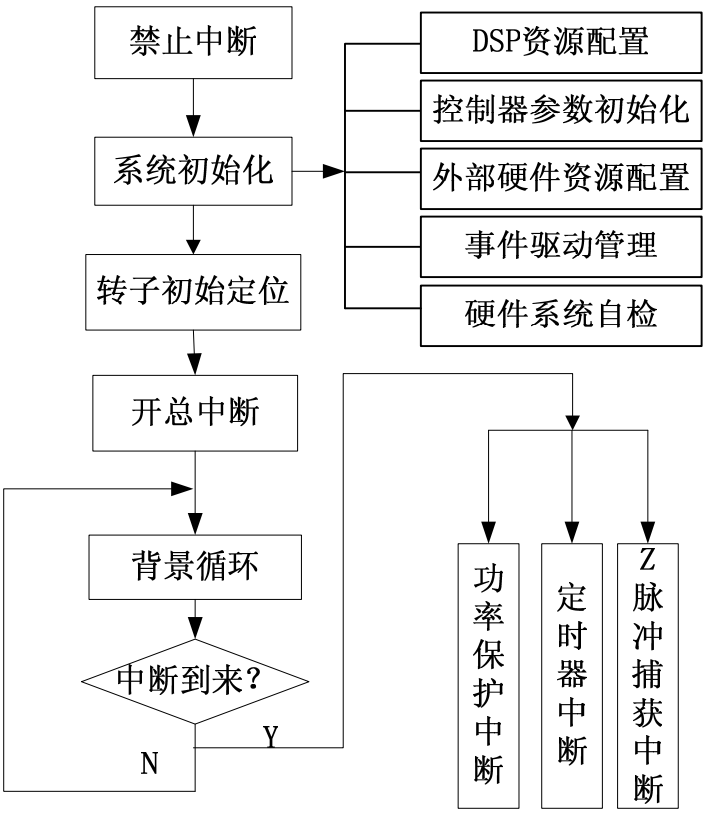

图 2-3 常见交流伺服系统软件流程

从上图可以看出，交流伺服驱动器常用的软件结构类似于分时操作系统，利用MCU/DSP 的定时中断，以中断周期作为系统最小的时间片并进行计数处理，实现多种系统周期分配，然后在主程序中采用背景循环的方式，将系统各种任务(例如 I/O任务，电流环任务，通信任务等)分配到不同的循环周期中，实现周期性调度。以定时器中断周期为 100μs 为例，驱动器软件调度采用周期式实时任务调度，可将不同任务按周期分为 500μs，1ms，10ms，100ms 几类；在定时器中断服务程序中对系统全局时间标识变量进行增计数，并在背景循环程序中对相关标识量进行判断，即可实现简单的周期式实时任务调度。常见的主循环程序流程图如下：

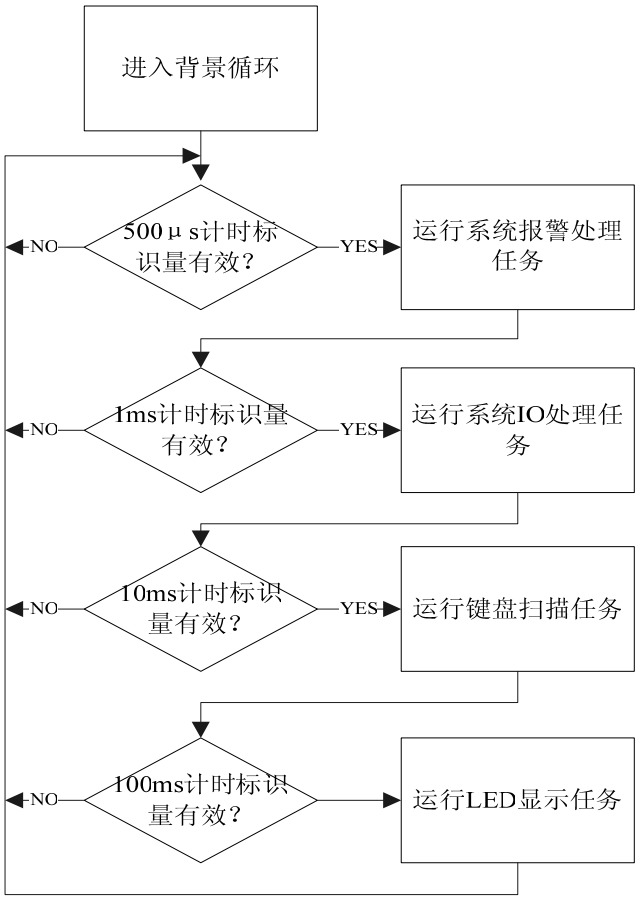

图 2-4 交流伺服系统背景循环流程

在介绍了 CANopen 协议的主要内容，以及交流伺服驱动器软硬件结构特点以后，下一节将对交流伺服驱动器的 CANopen 协议的实现方案进行分析。

2.4. CANopen协议的软件植入式实现方案

2.4.1. CANopen协议实现方案选择

实际设计中，需要在已有的伺服驱动器上实现 CANopen 应用层协议，具体的实现方案主要有两种：

1) 在驱动器中增加专用的处理器和硬件电路板，实现 CANopen 通信协议。该方案的优点是不需要对原有驱动器的软件进行大规模修改，缺点是需要额外的硬件开销。

2) 在不增加专用的通信协议处理器的前提下，利用已有的驱动器硬件自带的 CAN 总线模块，对原有驱动器的软件进行修改，在其中加入 CANopen 模块来实现协议。该方案的优点是不需要额外的硬件开销，缺点是需要大量的软件方面的工作，同时需要对原驱动器的程序进行修改，另外还需要考虑在原驱动器的程序中加入 CANopen 模块时软件模块兼容的问题，以及额外的软件空间开销和运行时间开销所带来的问题。

根据实际条件，现有交流伺服驱动器采用了 TI 公司的 TMS320F2812 型 DSP 作为主处理器，该 DSP 最高运行频率可达到 150MHz，处理能力强大；同时有丰富的外设，片上自带两套 CAN 总线模块可以使用，并且支持最新的 CAN 2.0b 协议标准，支持 29 位的 CAN 报文 ID。由于该 DSP 强大的处理能力，加入 CANopen 协议软件模块所带来的额外软件开销并不会影响到原有的运动控制功能，而该 DSP 对 CAN 总线支持良好，为软件实现 CANopen 协议提供了良好的基础。而解决加入 CANopen 模块时的软件兼容问题，则是实现该方案的主要工作之一。综合考虑，最终决定在实际的项目中采用软件植入 CANopen 协议的方案。

2.4.2. CANopen软件植入方案详细分析

根据前面对 CANopen 协议的分析，该协议的通信模型特点可以由下面几点概括：

1) 以 COB-ID 对 CAN 报文进行分类；不同类型的 CAN 报文在应用中需要不同

的处理流程，实现不同的功能。

2) 以对象字典为核心进行数据交互，所有系统参数都采用系统对象字典的方式进行统一管理，所有 CAN 报文通信进行数据交互都是通过对象字典实现。

3) 根据实际应用，定制标准的子协议；子协议中定义标准的系统运行模式，根据标准的运行模式将对象字典分区定义，并定义标准的状态机切换，实现多种控制功能。

因此软件植入协议的主要工作内容可以围绕这三个方面进行分析。

2.4.2.1. CAN报文的分类和处理

CANopen 协议的四大类报文处理流程的复杂程度差别较大，在实际中主要体现为报文的处理时间和系统硬件存储资源上的差别。处理器对于 CAN 报文的处理方式主要是查询方式和中断方式两种。对于流程复杂，处理时间较长的报文，如果采用中断方式处理，则中断服务程序的运行时间过长，可能影响到电流环这样需要实时快速反应的系统模块；而对于处理流程简单的报文，虽然查询法和中断法都是可行的，但是当报文需要系统进行快速应答时，如果采用查询方式，将报文处理程序放到系统背景循环的某个周期任务（例如 1ms 周期任务）中，会导致系统无法及时做出回应。因此正确的选择每种报文的处理方式，是 CANopen 报文分类和处理的关键。根据报文处理过程的复杂度，从两个方面来分析不同种类 CAN 报文的处理方式。

1) 网络管理报文 NMT 和特殊报文（主要是同步报文），这两大类报文的处理流程较为简单。普通 NMT 报文通常只需要设备在收到报文后进行一些开关动作（通常是修改系统状态位），进行运行状态切换即可，同步报文的处理流程也类似，只需修改同步报文的标志位即可。但是实际应用中，网络管理报文对系统的反应时间通常要求不高，而同步报文在多轴联动时作为设备的同步信号，需要进行快速反应，接收或回送相关的同步 PDO，因此网络管理报文应采用查询方式处理，而同步报文应采用中断方式进行处理。

2) 数据传输相关报文 SDO 报文和 PDO 报文。这两类报文的处理流程均比较复杂，SDO 报文在处理过程中需要对字典进行查找，并且修改相关字典项；PDO 报文

虽然采用直接映射方式访问字典对象，无需字典查找，但 PDO 报文的数据映射往往和运动控制参数相关，需对系统三闭环结构中的指令量和状态量进行操作，同时完成一些较复杂的状态机切换，因此这两类报文应采用查询方式进行处理。

2.4.2.2. 对象字典与系统参数处理

对象字典是 CANopen 设备的核心数据结构[32]。以软件植入方式实现 CANopen 协议时，需要解决的问题主要有以下几点：

1) 字典项在实际程序中应该采取什么样的数据结构。根据 CANopen 协议，每个字典项至少应包括对象索引，子索引，对象值这三个数据区，而协议文档中对每个字典项的读写权限也有相关定义，因此实际应用中每个字典项的数据结构应至少包含四个数据区：对象索引，子索引，对象值，读写权限。

2) 对象字典表在实际程序中应采取什么样的存储结构。考虑到系统的字典操作主要是查找，读取/修改字典项数据，不支持动态字典，即不支持删除字典项，插入字典项等操作。因此采用线性表的存储结构就能很好的满足需求。进行字典项的二分查找和读取/修改字典项的操作时程序效率也较高。

3) 如何解决 CANopen 协议的对象字典与原驱动器参数结构的兼容问题。由于原有的伺服驱动器具有自身的参数结构，因此可能并不包含 CANopen 对象字典所定义的标准对象，这就需要在原有的参数结构中加入 CANopen 协议标准的字典项；另一方面，原有的系统参数结构可能并非按照 CANopen 协议所规定的“索引-子索引”的数据结构进行定义，因此必须对系统原有的参数结构进行适当的改造。

2.4.2.3. CANopen运动控制子协议DS 402 实现方法分析

CANopen 运动控制子协议 DS 402 定义了若干标准运行模式，设备可以选择性支持。其中常用的标准运行模式有轨迹位置模式，轨迹速度模式，插补模式，回零模式，转矩模式等。在植入 CANopen 协议软件模块之前，驱动器可能并不支持这些标准运行模式；另一方面，驱动器可能已经实现了一套自定义的运动控制模式，并且

定义了相关的三闭环控制程序结构和运动控制参数结构。要加入 CANopen DS 402 协议所规定的标准运行模式，首先要对原驱动器的程序结构和参数结构进行分析，确定加入该模式的可行性[26]。

对于常用的 CANopen 标准运行模式，如轨迹位置模式，轨迹速度模式等，可以在驱动器中直接增加相应的轨迹规划函数模块，该模块通过对象字典与驱动器的指令量和状态量接口，然后在原程序位置环或速度环中增加分支语句，当系统选择轨迹位置模式或轨迹速度模式运行时，程序便进入轨迹规划函数分支，根据用户指定的轨迹模式进行增量计算。而对于较特殊的插补模式，由于需要多轴同步，并且计算过程比较复杂，因此该模式下的增量计算通常不是由驱动器程序完成，而是由主控台的程序完成，主控台在完成增量计算后，直接将位置增量通过同步 PDO 报文的方式发送到从节点，从节点只进行三闭环位置伺服动作。

CANopen 子协议 DS 402 另外一个主要内容是定义了标准的设备运行状态机。实际中，原有的驱动器程序也定义了系统运行状态机，但是两者无法兼容，原有的驱动器定义的状态机比较简单，只包含启动，弱电合闸，强电合闸少数几个状态，而 CANopen 子协议 DS 402 的定义的状态机比较复杂，导致原驱动器并不支持其中的某些状态，例如预运行状态等。要使得两者能够兼容，一个可行的办法是将 CANopen 状态机当作原系统的子状态机，可以在原系统中定义一个标识位来标识“CANopen 协议运行状态”，当驱动器在 CAN 总线网络中发送心跳报文，通知主站自己上线以后，驱动器进入 CANopen 协议运行状态。这时系统参数，指令量，状态量都可以通过 CAN 总线通信进行操作。在此基础上，单独编程实现 CANopen 状态机，让其作为“CANopen 协议运行状态”下的系统状态机，进行 CANopen 协议定义的标准状态切换。

2.5. 本章总结

本章主要介绍了课题的研究内容——CANopen 协议，课题的实现对象——交流伺服驱动器，课题的主要工作——在原有的交流伺服驱动器中实现 CANopen 协议。

提出了两种可行的方案：硬件实现和软件实现，分析了两种方案的优缺点，最终采用了软件实现 CANopen 协议的方案。然后根据 CANopen 协议的内容和原有交流伺服驱动器的特点，分析了软件实现 CANopen 协议需要解决的三个方面的问题。下一章将对这三方面的问题分别提出具体的解决方案，并加以实现，从而在原有的交流伺服驱动器中，以软件的方式实现 CANopen 协议从节点功能。

3. CANopen 协议的软件实现

上一章对 CANopen 协议的特点以及常见的交流伺服驱动器的软件结构进行了详细分析，抽象出了实现 CANopen 协议时所需解决的问题，而并未讨论到实现的具体细节，如软硬件平台，编程语言，程序流程等。本章的主要内容是介绍实际的设计工作。首先介绍课题中所使用的实际驱动器设备的系统结构以及软硬件平台，然后介绍在该设备上采用软件方式实现 CANopen 协议的详细设计工作。

3.1. 实际伺服驱动器平台介绍

实际使用的交流伺服驱动器的硬件模块图如下：

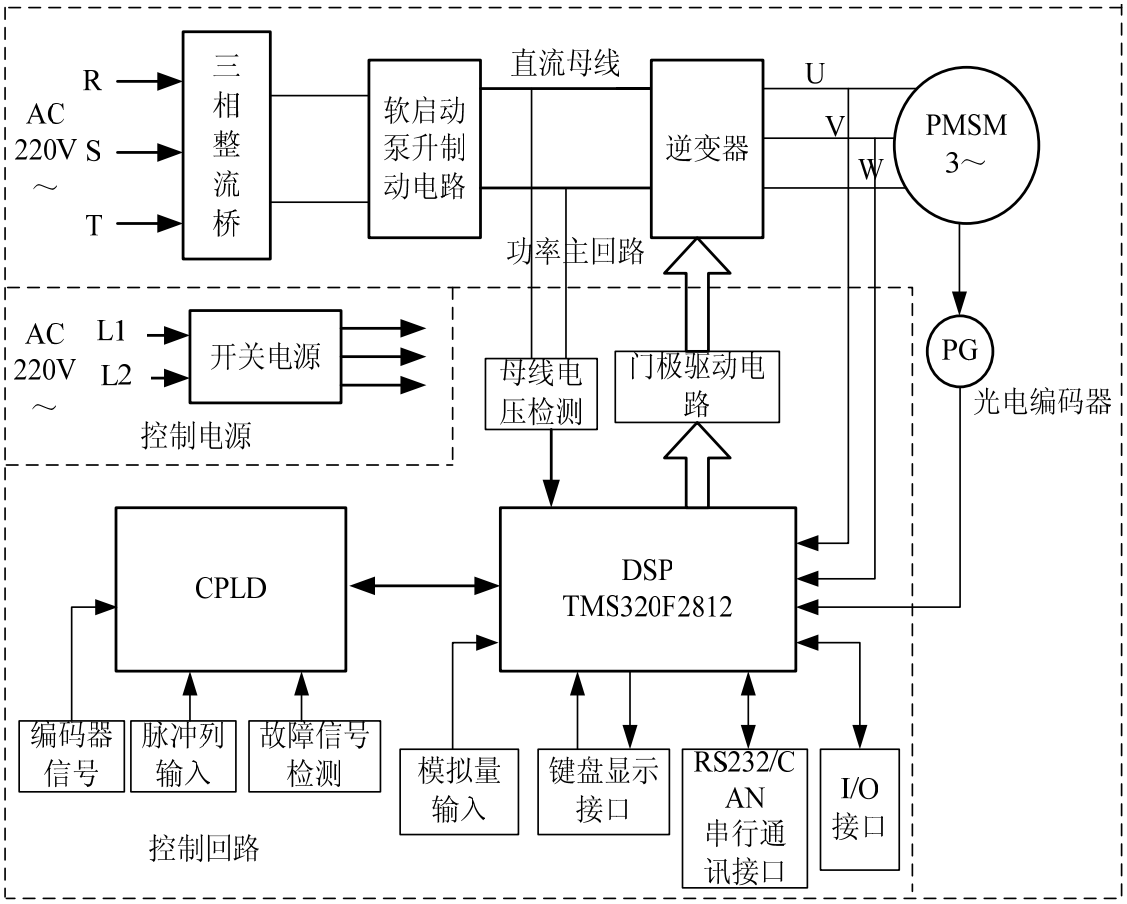

图 3-1 伺服驱动器硬件总体结构图

本课题主要使用TI公司生产的TMS320F2812型DSP作为主处理器。利用该DSP的eCAN模块来实现CANopen协议。下面对TMS320F2812型DSP及其eCAN模块进行简单介绍：

TMS320F2812型DSP为32位DSP，其内存中数据单元以双字节为最小单位。内建有两组CAN总线模块，均支持CAN2.0B协议标准，支持29位的CAN-ID，同时能兼容11位CAN-ID模式[33-37]。

3.1.1. TMS320F2812型DSP的CAN报文邮箱管理方式

DSP对CAN总线报文的管理采用了邮箱方式。由DSP硬件直接分配内存定义CAN邮箱收发报文，内部支持最多32个CAN邮箱，每个邮箱都可用于发送报文或者接收报文。邮箱寄存器的起始地址位于DSP地址空间中的0x6100处，邮箱寄存器在地址空间中的总长度为256个字长，因此每个邮箱分配到的存储空间为8个字，也就是16个字节。每个邮箱寄存器主要由以下几个部分组成[36]：

1) 29位的CAN-ID地址域，占用每个邮箱的前两个字，其中CAN-ID的低字在前，高字在后。以起始地址为0x6100的第一个邮箱为例，其29位CAN-ID的低16位被放在0x6100地址处，高13位被放在0x6101的地址处。这样还剩下最高的3位可以用来进行设置，实现一些邮箱控制功能。其中第29位为自动应答位，可以实现报文自动应答；第30位为接收屏蔽使能位，若设置成1，则该邮箱只能接收CAN-ID和邮箱寄存器完全相同的报文，若设置成0，则该邮箱可接收任意CAN-ID的报文；第31位为11/29位CAN-ID标志位。由于eCAN模块在支持CAN 2.0B的29位CAN-ID的同时必须兼容11位CAN-ID，因此设置了一个标志位进行CAN-ID格式选择。

2) CAN报文控制寄存器，占用每个邮箱寄存器的第3，4个字，同样低字在前高字在后，是CAN邮箱的主要控制寄存器。32位寄存器位定义如下表所示：

表 3-1 CAN邮箱控制器位定义

bit 0-3	bit 4	bit 5-7	bit 8-12	bit 13-31
报文数据长度	远程帧标识	保留	传输优先级	保留

每个 CAN 邮箱控制寄存器在使用过程中主要起作用的是以下三个位段：数据长度，远程帧标识，传输优先级，下面分别对这三个位段进行说明。

报文数据长度：4 位二进制数，取值范围为 0-15，由于 CAN 报文的数据域可以为 0-8 个字节，因此该位段的二进制数值只能取 0-8 以内。邮箱在发送和接收 CAN 报文时根据该 4 位二进制数的值对 CAN 报文的数据进行处理。

远程帧标识：1 位二进制数，当该位为 1 时，相应的邮箱被设置为远程帧接收/发送邮箱。

传输优先级：5 位二进制数，取值范围为 0-31。传输优先级主要用于多个邮箱同时发送数据的发送顺序排列，数值越大的邮箱优先级越高。当多个邮箱优先级相同时，DSP 将按照邮箱编号的大小顺序从编号最大的邮箱开始依次发送。

3) 8 字节的报文数据区，该数据区位于邮箱的第 4-7 个字，实际 DSP 的 CAN 邮箱数据寄存器都是 32 位寄存器，故该数据区共包含两个邮箱寄存器：CANMDL 寄存器和 CANMDH 寄存器。报文数据仍采用低字节在前的存储方式。这样，DSP 每个 CAN 报文数据寄存器实际上是 4 字节长度，因此接收或发送 CAN 报文时，首先要对报文长度和 4 进行比较，如果长度超过 4 字节则需要使用 CANMDH 寄存器，如果报文长度小于 4 字节则不需要使用 CANMDH 寄存器。

3.1.2. TMS320F2812 型DSP的通用CAN总线寄存器

除了使用 32 个邮箱来管理 CAN 总线报文以外，该 DSP 还提供了一组通用的寄存器来实现一些全局的 CAN 总线管理功能，例如 CAN 总线波特率设置，eCAN 模块中断的使能/禁止，每个邮箱的使能/禁止，邮箱报文发送错误记录等。DSP 主要的通用 CAN 寄存器有以下几个[36]：

1) 邮箱使能寄存器，长度 32 位，对应 32 个邮箱，当使能位设置为 1 时对应邮箱被使能，设置为 0 时对应邮箱被禁止使用。

2) CAN 总线主控制寄存器 CANMC，长度 32 位，该寄存器位定义非常复杂，这里只列出主要使用的控制位的定义：

表 3-2 CANMC 控制寄存器主要位定义

位域	说明
bit 0-4	5 位二进制数值表示当前所使用的 CAN 邮箱的数量。
bit 6	CAN 模块自检模式使能位，当该位为 1 时，CAN 模块运行于自检模式，为 0 时运行于正常模式。
bit 12	更改设置请求位，当更改 CAN 模块的控制寄存器时，需要先将该位置位。

3) 波特率设置寄存器 CANBTC，该寄存器的主要作用是设置 CAN 总线传输波特率，其中经常使用到的是 bit16-bit23 的 8 位 BRP，bit6-bit3 的 4 位 TSEG1 以及 bit2-bit0 的 3 位 TSEG2，这三个二进制数值直接决定了 CAN 总线的波特率，其计算公式如下：波特率 $=\frac{SYSCLK}{BRP\times Bit\_time}$，其中 BRP 为 CANBTC 寄存器 bit16-bit23 的 8 位二进制数值 BRP_{REG}+1，Bit_time=(TSEG1+1)+(TSEG2+1)+1。而 SYSCLK 位 DSP 的时钟频率，常用为 150MHz 和 120MHz。常用的 CAN 总线波特率为 1Mbps，500Kbps，250Kbps 和 125Kbps，本文的采用系统时钟为 120MHz，TSEG1=10，TSEG2=2。通过调节 BRP 在常用的波特率中进行切换。BRP 和波特率的关系如下表所示：

表 3-3 CANBTC 寄存器中 BRP 域与 CAN 总线波特率关系表

BRP	波特率
9	1Mbps
19	500Kbps
39	250Kbps
79	125Kbps

4) CAN 邮箱中断控制寄存器：CANMIM 寄存器和 CANMIL 寄存器。这两个寄

存器分别为 CAN 邮箱中断屏蔽寄存器和 CAN 邮箱中断优先级寄存器。两个寄存器均为 32 位并分别对应 32 个 CAN 邮箱，CANMIM 寄存器对应位置 1 时使能相应邮箱中断，置 0 时中断被屏蔽。CANMIL 寄存器相应位置 1 时，对应邮箱执行 CAN 模块 1 号中断服务程序，置 0 时执行 0 号中断服务程序。

在使用以上 CAN 总线模块寄存器时，应特别注意，所有的 32 位寄存器均不支持直接的位运算，如果对上述 32 位寄存器直接进行位运算将导致未知错误。写入寄存器的值时需要一次性写入全部 32 位。因此实际应用中必须采用缓冲方式，当需要修改寄存器的值时，先在通用内存中申请临时内存变量，将寄存器的当前值赋给该临时变量，进行位运算后再一次性写回对应寄存器中。例如要对 CANMIM 寄存器的最低位进行或操作，不能直接对寄存器进行位操作 CANMIM|=1；而需要申请临时的内存变量 CANMIMShadow，然后进行如下操作：

```
CANMIMShadow.all=CANMIM.all;     //对映射变量赋初值
CANMIMShadow|=1;                 //对映射变量进行位操作
CANMIM.all= CANMIMShadow.all;    //将操作后的映射变量送回寄存器
```

从上面的介绍可以看出，TMS320F2812 型 DSP 的 CAN 总线模块功能强大，以此为基础能够编程实现各种常见的 CAN 总线应用层协议。下面将阐述如何利用该 DSP 的 eCAN 模块，解决 CANopen 协议软件实现方案的三个问题，并在驱动器中实现 CANopen 协议。

3.2. CAN报文的分类处理及其软件实现

TMS320F2812 型 DSP 的 eCAN 模块采用邮箱方式管理 CAN 报文，因此在实现 CANopen 协议的报文分类处理时，可以直接利用 DSP 的邮箱管理的特点，通过 29 位的 COB-ID 将报文类型和邮箱的绑定，实现报文分类。再将每类报文的处理函数设计成独立的子模块，实现分类报文的处理。实际设计中，CANopen 协议定义的每类报文的处理函数都被封装成独立子函数，而顶层的 CAN 报文主处理函数通过轮流调用各个分类报文处理函数实现，CAN 报文主处理函数的流程图如下：

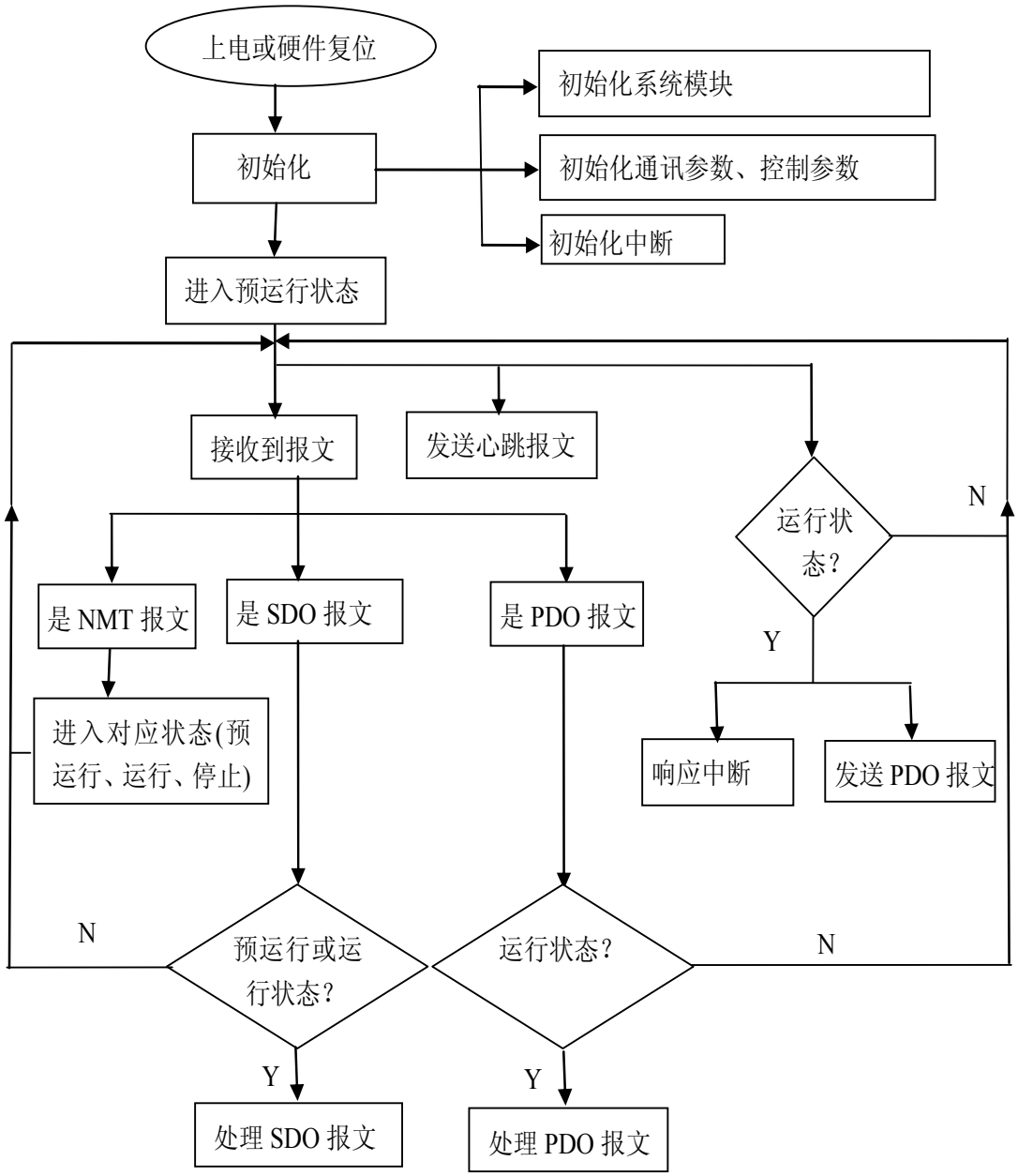

图 3-2 CAN 报文处理流程图

对于网络同步报文和需要同步处理的 PDO 报文，都采用 CAN 邮箱中断方式进行处理。由于交流伺服驱动器是 CANopen 从节点，只需接收主节点发送的网络同步报文，因此采用 CAN 邮箱的接收中断即可实现。

所有报文的收发和处理过程，都需要把相应的 CAN 报文与指定的邮箱绑定才能实现。要进行邮箱绑定，只需将特定 CAN 邮箱的 CAN-ID 寄存器设置为协议定义的

值，并使能该邮箱的 CAN-ID 屏蔽功能即可。这样该邮箱只接受/发送特定 ID 号的报文。实际设计中邮箱与报文对象的绑定如下表所示：

表 3-4 实际设计中的邮箱——CAN 报文绑定

邮箱编号	邮箱 CAN-ID 寄存器	报文对象
0 号邮箱	0x461C0000	接收远程帧 TPDO
6 号邮箱	0x1C1C0000	NMT 错误控制
7 号邮箱	0x181C0000	SDO（接收）
8 号邮箱	0x141C0000	PDO4（接收）
9 号邮箱	0x101C0000	PDO3（接收）
10 号邮箱	0x0C1C0000	PDO2（接收）
11 号邮箱	0x081C0000	PDO1（接收）
12 号邮箱	0x021C0000	紧急报文（接收）
14 号邮箱	0x02000000	同步报文（接收）
15 号邮箱	0x00000000	NMT 报文（接收）
23 号邮箱	0x161C0000	SDO（发送）
24 号邮箱	0x121C0000	PDO4（发送）
25 号邮箱	0x0E1C0000	PDO3（发送）
26 号邮箱	0x0A1C0000	PDO2（发送）
27 号邮箱	0x061C0000	PDO1（发送）
28 号邮箱	0x021C0000	紧急报文（发送）
30 号邮箱	0x02000000	同步报文（发送）

完成了 CAN 报文处理程序的设计后，考虑其植入原驱动器的方法。CAN 报文主处理函数是独立于原驱动器的任务，可以直接放到背景循环中执行。在综合考虑 CAN 报文处理函数的运行时间和 CANopen 从节点的反应时间后，将该任务作为驱动器的 1ms 定时任务进行处理。对于 CAN 总线中断服务函数，可视为直接在原驱动器的中加入一个中断服务，响应 CANopen 主节点的同步请求。在植入以上 CAN 报文处理过程后，驱动器顶层流程图如下图所示：

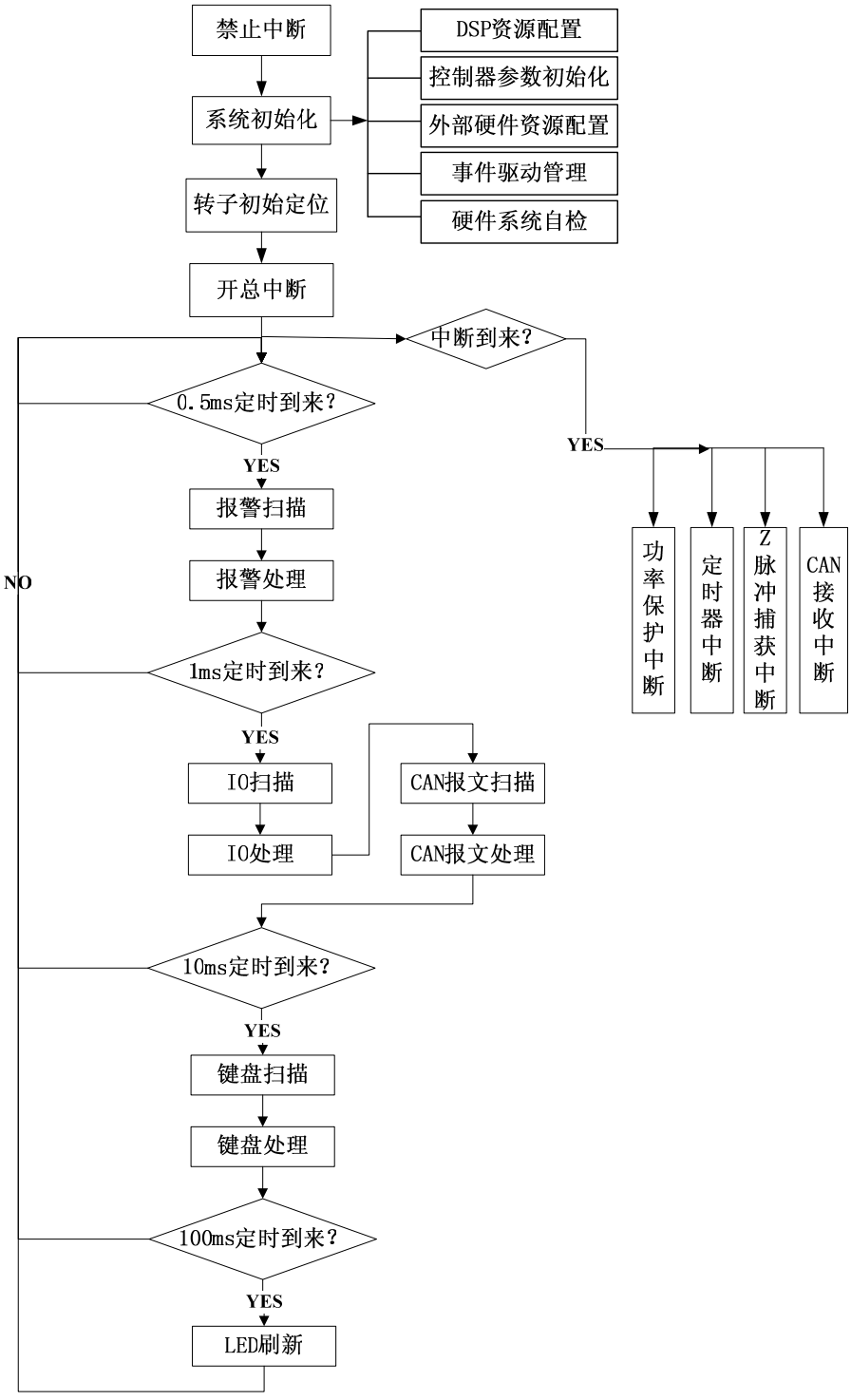

图 3-3 加入 CANopen 模块后的驱动器顶层流程图

从以上的流程图可以看出，CANopen 模块以独立的形式植入原驱动器程序，兼容性良好，可视为驱动器程序的即插即拔模块。

按照自顶向下的顺序，下面分别论述 CAN 报文主处理函数的各个子模块的实现方式。

3.2.1. SDO报文处理流程以及对象字典操作

SDO 报文的主要功能是读写对象字典。鉴于 SDO 报文对字典的操作方式是直接索引方式，且通信模型为请求——应答的确定性通信模型，因此 SDO 报文适合用来操作字典中的非实时对象。就具体的交流伺服驱动器来说，非实时的数据主要是一些控制器参数（例如 PID 参数），通信参数（例如同步时间）等，这些参数的特点是一般只在驱动器进入运行状态前进行读写。SDO 报文处理流程图如下：

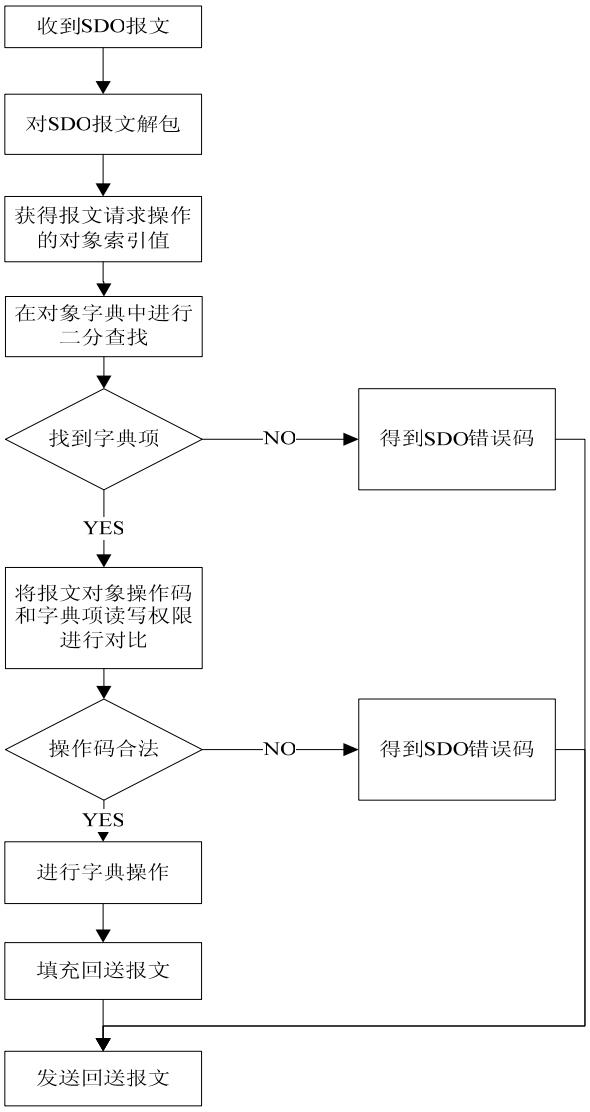

图 3-4 SDO 报文处理流程图

SDO 处理中的关键步骤是在对象字典中进行二分查找，该步骤能够极大优化字典索引效率，减少 SDO 报文处理函数运行时间，确保 SDO 报文能在规定的系统任务时间 1ms 内完成。对于具体的二分查找步骤，由于与对象字典采用的存储结构相关，将在后文介绍对象字典的具体存储结构时一并进行介绍。

3.2.2. PDO报文处理流程以及对象字典映射

PDO 报文是 CANopen 协议中非常重要的一类报文对象，尤其是对于伺服系统来说，在执行三闭环定位或者速度伺服时都需要驱动器与主控台进行大量数据交互，而在多轴联动的情况下，更是需要多个驱动器和主控台进行实时的同步数据传输。PDO 报文采取直接映射的方式访问 CANopen 对象字典，绕过了对象字典中索引——子索引的二份查找过程，且每个 PDO 报文的 8 字节数据全部用于传输数据，不包含任何读/写操作码，索引——子索引等信息，数据密度大，同时不需要回送应答。这些特点决定了 PDO 报文在数据交互方面的高效率。对于 PDO 报文的处理，首先需要根据协议定义的 PDO 传输控制功能（例如禁止时间，事件周期等），确定 PDO 报文对象的数据结构；然后选择 PDO 映射的方法：固定映射或者动态映射；最后根据不同 PDO 报文的传输方式：同步方式或者异步方式，分别进行处理。

3.2.2.1. 实际系统中PDO对象的数据结构

按照发送/接收的功能，PDO 报文主要分为两大类：TPDO（传输）和 RPDO（接收）。一般来说 RPDO 对象的数据结构只要包含 CAN-ID 和 8 字节数据即可，数据结构比较简单。而 TPDO 对象的数据结构应包含 PDO 传输类型，事件时间，禁止时间，同步对象计数等数据域，其定义如下：

```
typedef struct
{
    Uint16   event_time;              //事件定时周期(ms)
    Uint16   EvTimCnt;                //定时周期计数器（ms）
    Uint16   inhibit_time;            //禁止时间(单位 100us)
```

```
    Uint16   InhTimCnt;              //禁止时间计数器
    Uint8    SyncObjCnt;             //接收同步对象(SYNC)计数器
    ……                               //以下数据域省略
}   TPDO_CONFIG;
```

CANopen 协议对于 TPDO 传输规定了事件时间（event time）和禁止时间（inhibit time），其中事件时间以毫秒为单位，主要用于产生 TPDO 的传输事件（event）。例如驱动器如果需要周期性（例如 10ms）回送某个 PDO，将自己的某些状态量传输到控制台，则可以设定事件时间为 event_time =10ms。事件时间的计时可以直接利用驱动器的定时中断实现。由于驱动器原有程序中已经包含了 1ms 定时的分支处理，这样在原驱动器程序基础上实现事件时间非常简单，只需在原驱动器程序的 1ms 定时分支中对 EvTimCnt 进行自加操作，当 EvTimCnt=event_time 时传输 TPDO 报文并清零 EvTimCnt 即可[25]。

禁止时间以 0.1 毫秒为单位，主要用来解决 CAN 总线的争用问题。当总线上同时有两个节点请求发送报文时，DSP 的 CAN 模块将检测到总线冲突，TPDO 发送失败，这时将设备应该重新发送 TPDO 报文，如果没有设定禁止时间，两个节点都直接重发，将导致无穷尽的总线冲突，信道将因为冲突而完全不可用。为避免这种情况的发生，CANopen 协议定义了 TPDO 的禁止时间，当 TPDO 发送失败时，可以根据禁止时间定义相应的重发算法，避免总线再次冲突。最简单的重发算法就是直接等待一个禁止时间，而 CANopen 协议最多支持 127 个节点，每个节点支持 4 个 TPDO，如果采用直接等待一个禁止时间的重发算法，则所有 TPDO 的禁止时间应该各不相等，且任意两个 TPDO 的禁止时间间隔应该大于当前波特率下总线上一帧数据的发送时间。

由于禁止时间的单位为 0.1 毫秒，正好等于原有驱动器的定时中断周期，因此禁止时间可以直接在驱动器的定时中断服务程序中实现，例如设定某个 TPDO 报文的重发禁止时间为 1ms 则可以设定禁止时间 inhibit_time=10，每个定时中断中对 InhTimCnt 进行自加，当 InhTimCnt=inhibit_time 时传输 TPDO 报文并清零 InhTimCnt 即可。TPDO 报文根据禁止计时进行重发。

3.2.2.2. 接收RPDO处理：固定的PDO映射

PDO 报文对字典的映射方式有两种，一种是固定映射方式，不支持用户通过通信方式动态修改 PDO 与字典的映射关系。另外一种是动态映射方式，用户可以通过 SDO 报文修改对象字典中 PDO 映射管理项，进而实现 PDO 与字典的动态映射。

在运动控制系统中，需要进行 PDO 映射的数据主要是三环控制程序中的指令量和状态量，且需要映射的数据较少，4 组 TPDO 和 RPDO 已经能满足系统需求，不存在需要映射的数据过多而导致同一个 PDO 在不同情况要进行不同映射的情况。固定映射方法操作简单，软件开销小，适合实际应用系统。采用固定 PDO 映射后，对于 RPDO 报文的处理则变得很简单，在背景循环中查询 PDO 邮箱时，在收到某个 RPDO 报文后，根据固定映射表，直接将该 RPDO 报文的 8 字节内容写入对应的字典项中。

3.2.2.3. 发送TPDO处理：同步与异步

TPDO 的发送方式分两种，同步方式和异步方式[38]。

同步方式和异步方式对应不同的 TPDO 触发事件，CANopen 定义了 8 位的传输类型值，每种类型对应不同的触发事件，如下表所示[25]：

表 3-5 CANopen 传输类型表

传输类型	触发PDO条件 B：两者必须同时成立， 0：只要有一个成立即可触发			PDO传输
	收到 SYNC 消息	收到 RTR 消息	事件	
0	B	—	B	同步，非周期
1-240	O	—	—	同步，周期
254	—	O	O	异步，制造商指定的事件
255	—	O	O	异步，设备子协议指定的事件

传输类型为 0 的 TPDO 需要同步报文和 TPDO 事件同时发生才能触发，在实际

设计中并未使用该方式。

传输类型 1-240 的 TPDO 触发条件为 x 个同步报文时间，其中 x 为传输类型值，实际设计中同步 TPDO 的发送主要采用的是这种触发方式，而对同步报文的计数可以直接放在同步报文所绑定的邮箱的接收中断服务程序中进行，实际设计的中断服务程序流程如下：

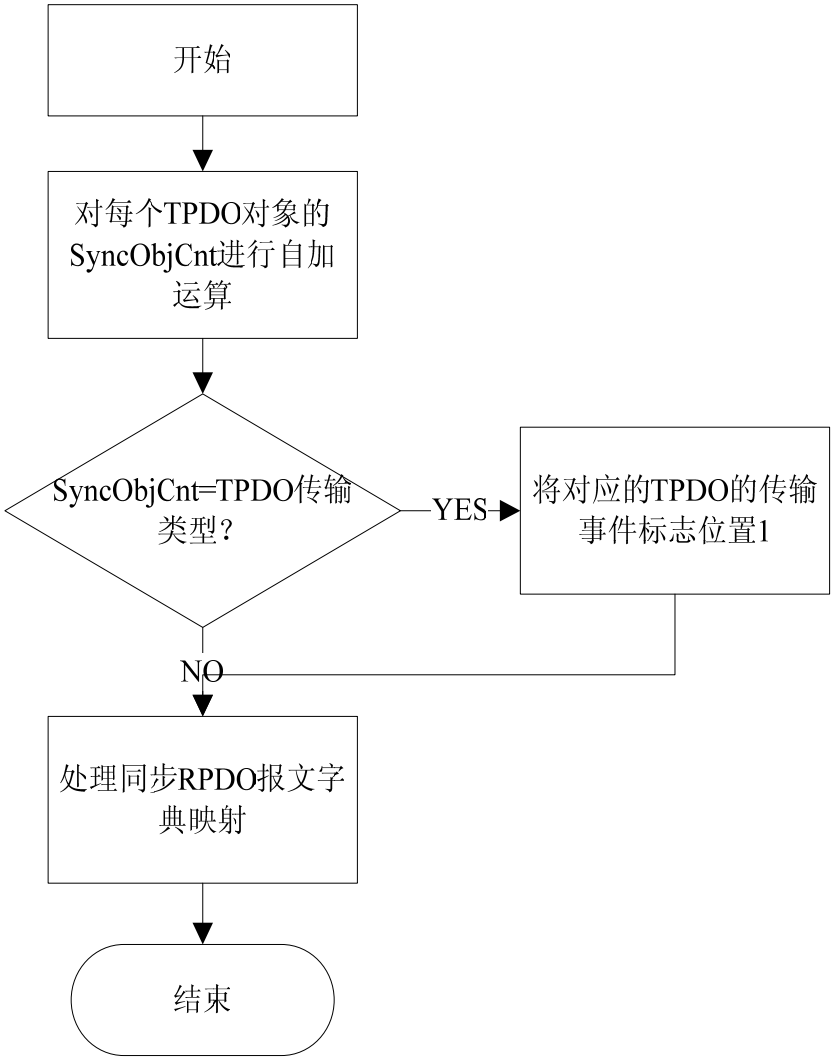

图 3-5 同步中断服务程序流程图

传输类型为 254 和 255 的 TPDO 报文为异步传输方式，其触发事件由厂商或者 CANopen 子协议定义。例如可以定义“驱动器对象字典中特定的对象值改变”为触发事件，或者直接定义一个事件时间（event_time），在驱动器的定时中断服务程序中通过对“事件时间”的方法来产生 TPDO 发送事件。

背景循环中对所有的 TPDO 进行发送条件轮询，实现各种 TPDO 报文的传输，每个 TPDO 发送函数流程如下：

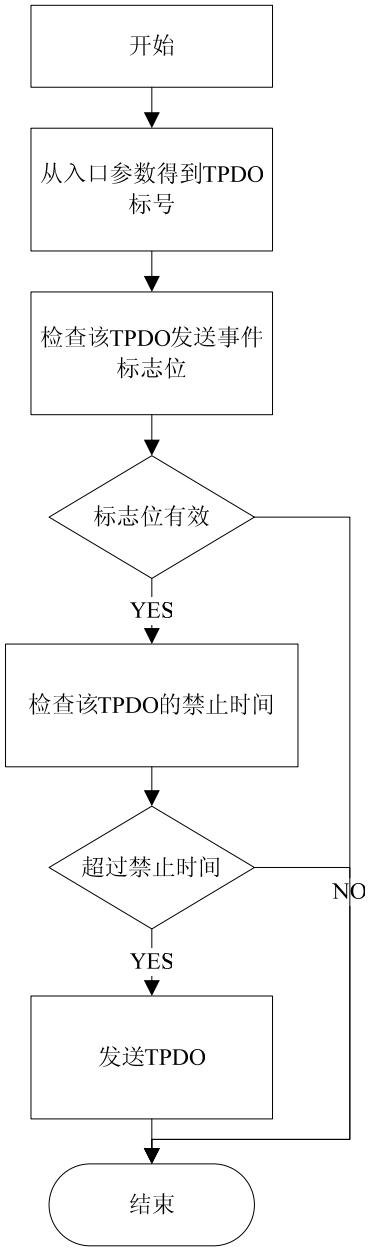

图 3-6 TPDO 传输函数流程图

实际设计中，为了简单起见，系统的异步 TPDO 的传输类型均设置为 254。对于实际应用中的异步 TPDO 来说，其触发条件一般为映射的字典对象改变，因此实际设计中所有的异步 TPDO 统一采用该事件作为 TPDO 触发事件，在背景循环中的 CANopen 协议主函数对每个异步 TPDO 所映射的字典项进行周期性扫描，另外每个

TPDO 都开辟一片缓存区缓存上次发送时的映射内容，将这两者进行对比，若不相等，则触发该 TPDO 的发送事件，并将该 TPDO 的发送标志位置位，调用 TPDO 发送函数时 TPDO 被发送；而对于实际应用中的同步 TPDO，考虑到运动控制系统中很少需要对同步周期计数，一般都是收到同步报文就直接回送同步 TPDO，因此所有的同步 TPDO 的传输类型都被设置为 1，即收到同步报文就马上触发同步 TPDO 的传输事件。

3.2.3. NMT报文处理：系统网络状态机

CANopen 协议中 NMT 报文主要用于主节点对从节点进行网络状态管理[39]。对于每个 CANopen 设备来说，其在 CAN 总线网络上有启动，停止，预运行，运行这几种状态。CANopen DS 301 协议定义的标准状态机如下图所示：

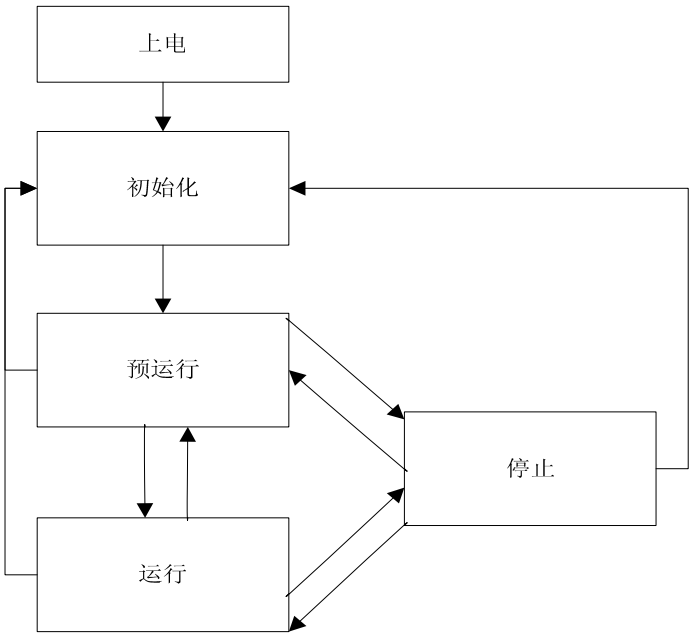

图 3-7 CANopen 标准状态机

从节点收到主节点 NMT 报文时按照标准状态机进行状态切换。标准的 NMT 报文只包含 1 个字节的命令字，CANopen 协议定义的命令字如下表所示：

表 3-6 CANopen 协议 NMT 报文命令字

命令字	含义
1	启动远程节点命令
2	停止远程节点命令
128	进入预运行状态
129	重置节点
130	重置通信

在背景循环中，当 CANopen 从站对接收邮箱进行轮询时，若 NMT 报文绑定的邮箱接收到 CAN 消息，驱动器程序将按照标准状态机对驱动器的相应状态位进行更新。

NMT 报文除了用于主节点对从节点发送命令以外，还可以作为从节点的心跳报文使用。从节点根据对象字典中的心跳周期，定时向网络发送心跳报文，通知主节点自己当前的运行状态，CANopen 协议定义的心跳报文状态字含义如下：

表 3-7 CANopen 协议心跳报文状态字

心跳状态字	节点状态
0	初始化
4	停止状态
5	运行状态
127	预运行状态

主节点则可以通过对心跳报文进行计时来确定网络中节点是否在线，实现网络节点管理功能，具体做法如下：主节点维护一个数组记录当前在线的从节点，当主节点收到某个从节点的心跳报文时，则数组中添加该节点编号，同时对该节点的心跳报文进行计时，若该节点的心跳报文超时，则认为该节点心跳停止，主节点将该节点从当前在线节点数组中删除。

3.3. 驱动器中CANopen对象字典的实现

上文论述了软件实现 CANopen 协议报文分类处理的细节，而所有类型的 CAN 报文经过分类处理后，都需要进行相应的对象字典操作。下面将结合第 2 章对于 CANopen 对象字典的相关介绍，分析在设备已经具有自身参数结构情况下，如何实现 CANopen 对象字典，并且以实际课题中的交流伺服驱动器作为 CANopen 设备，在设备中实现标准 CANopen 对象字典。

3.3.1. 字典项的数据结构

根据第 2 章的介绍，字典项的数据结构应该包含对象索引值，子索引值，读写权限，对象数据这几项。CANopen 协议规定的索引值为双字节，而子索引值和读写权限均为单字节，对象数据则根据协议定义可能为单字节，双字节或者双字长度。于是得到初步的数据结构定义：

```
typedef struct{
    unsigned int idex;              //索引值
    unsigned char subindex;         //子索引值
    ……                              //以下定义省略
} OD;
```

但是在实际设计中，驱动器使用的是 32 位的 TMS320F2812 型 DSP，在使用 C 语言编程时，DSP 会自动将 8 位的无符号 char 型变量对齐到 16 位字的低 8 位，高 8 位为无效数据。如果采用以上定义，按照 unsigned int 来存储 8 位的无符号字节变量，则每个字典项将浪费与自身空间大小相等的存储空间。对于片上存储资源有限的交流伺服驱动器来说，这种空间利用效率低下的数据结构定义应该要尽量避免。

为了减少直接定义字典数据结构所带来的空间浪费，实际中采用位域的方法来定义对象字典数据结构，以双字节的字典项为例，考虑如下定义：

```
typedef struct{
    unsigned int index;   //索引
```

```
    unsigned char subindex:8; //子索引
    unsigned char access:8;   //读写权限
    OD_DATA data;          //字典数据，具体定义也采用位域方式，这里省略
}OD;
```

以上定义中对象数据变量类型 OD_DATA 也采用划分位域的方式定义，这样定义对象字典项可以提高内存利用效率，同时对字典项的访问也非常方便。

3.3.2. 对象字典的存储结构以及字典项的二分查找

CANopen 协议中并未定义对象字典的存在形态，对象字典既可以以静态形式存在，只支持一般的静态字典操作，例如索引字典项，读/写字典项等；也可以以动态形式存在，除了支持一般的索引和读写操作外，还支持新增字典项，插入字典项，删除字典项等动态字典功能。

在交流伺服驱动器中，参数结构是固定的，新增字典项等动态功能意味着用户将会改变驱动器的参数结构。对于普通用户而言，开放字典动态操作操作具有很大的危险性，有潜在的损坏系统甚至造成安全事故的可能；另一方面，实际用户一般也没有动态字典的需求。因此本课题的实际设计中采用了静态字典，只支持索引字典项，读/写字典项的操作。其中索引字典项是最复杂的字典操作。如何设计对象字典的存储结构，并设计高效率的索引程序，是实现字典操作的关键。

对于静态字典，由于不需要插入/删除操作，最有效率的存储结构就是线性表，即以字典项数组的方式存储。所有字典项按照索引值升序的方式在对象字典中初始化，而对于同一索引值的各个子索引对象，也按照子索引升序的方式直接放入数组中。对于排好序的线性数组来说，进行二分查找的效率很高，且程序流程简单。但是对象字典数组的二分查找和一般的流程有所不同，需要考虑同一个索引值的子项问题，根据 CANopen 协议，同一个字典项的子索引值必然是从 0 开始的连续正整数，且最多不超过 127 个子索引，而这些子项在对象字典初始化时已经都按照子索引升序的方式放入数组，因此在二分查找到匹配的索引值后，只需求出查找到的字典项的子索引值和需要查找的子索引值的差，再对数组下标进行一次加减法运算，即可

查到正确的子索引项。字典索引程序流程如下图所示：

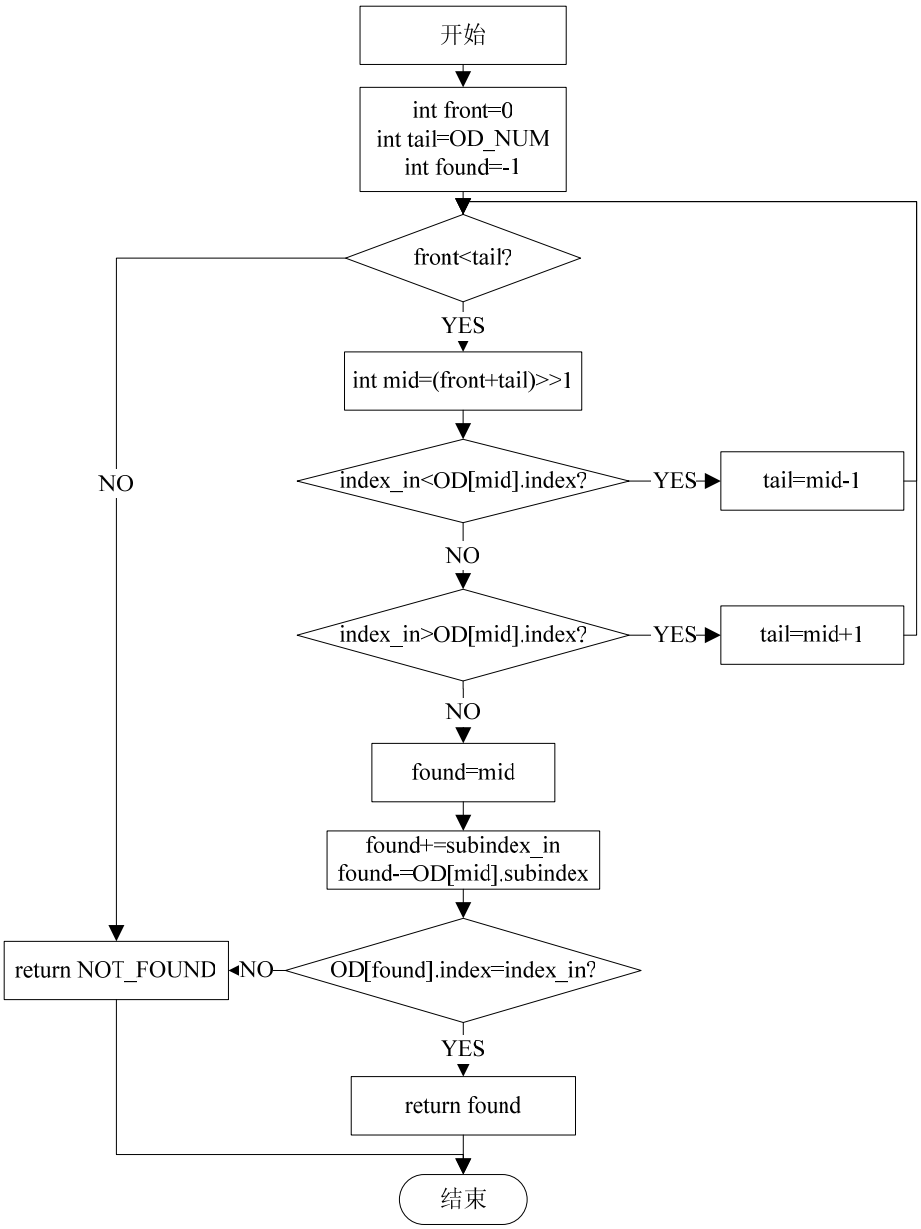

图 3-8 二分法查找字典流程图

3.3.3. 对象字典与原驱动器参数结构的接口

原有驱动器中已经定义了自己的参数结构，在设计对象字典与原驱动器参数的接口时，首先要考虑参数结构的统一接口问题，其次要避免参数的二义性。实际伺服驱动器中的一些控制参数，比如三环控制的 PID 控制器参数、滤波器参数等，在

原有的驱动器程序已经定义相应的参数表，这些参数在三闭环控制程序中都多次被引用，如何避免对原驱动器的程序进行大幅度修改，同时引入这些参数的对象字典管理模式，是需要解决的问题。

依照 CANopen 协议的定义，在实际设计中，将伺服驱动器原有的控制参数定义到字典的厂商自定义区(索引号 0x2000-0x5FFF)。为了防止全局控制参数产生二义性，对象字典中并未加入这些参数的实体字典项，只是加入了参数索引号；定义索引值为 0x2002 的字典项为原驱动器的控制参数表，而该字典项的所有子项索引则直接采用伺服驱动器原有的参数编号。通过通信方式对字典中原有控制参数的访问和操作，则以程序区分的方式与一般的字典操作区别开，流程图如下：

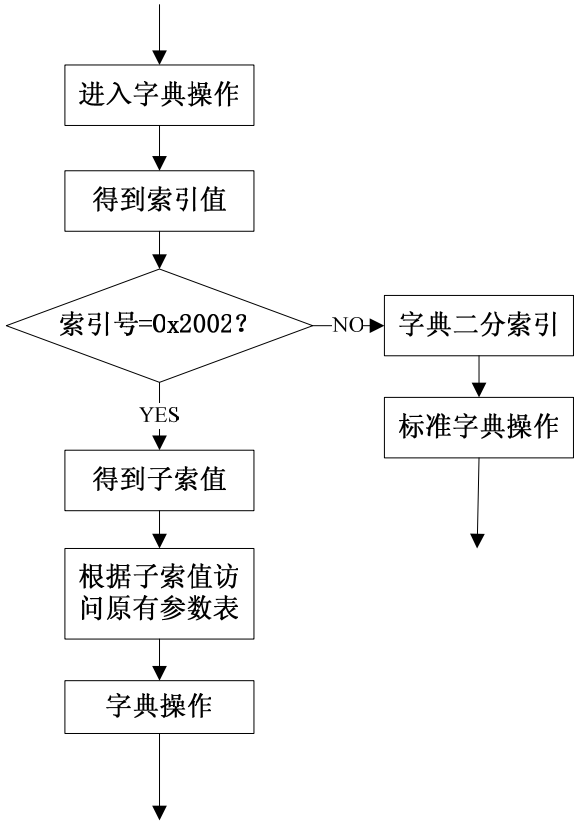

图 3-9 字典索引全局控制参数流程图

如上图所示，实际中对象字典索引全局控制参数的处理方法是：在字典操作函数中增加分支处理，当判定需要操作的索引值指向原有驱动器控制参数时，直接对参数数组进行操作。这样处理既统一了参数通信接口，又避免了二义性。

伺服驱动器运行过程中的指令和状态量，如电机的转速指令和转速反馈，电流指令和电流反馈等，均包含在 CANopen 子协议 DS 402 定义的对象字典中。原有运动控制程序将这些变量定义在特定的数据结构中，通过数据结构实现运动控制模块间的接口。这种接口方式简单，且不会造成模块之间的强耦合，因此在 CANopen 模块中也借鉴了这种做法，根据不同的运行模式，对伺服驱动器运行过程中的指令和状态变量，定义相应的中间数据结构进行接口，对象字典的操作（通常是 PDO 映射方式）只访问中间数据结构，而运动控制程序也通过访问中间数据结构来接收和发送相关的指令量和状态量，以缓冲的方式进行通信模块和运动控制模块的数据隔离和接口，保证了 CANopen 模块的独立性。下面以框图的形式来举例说明具体的接口方法。

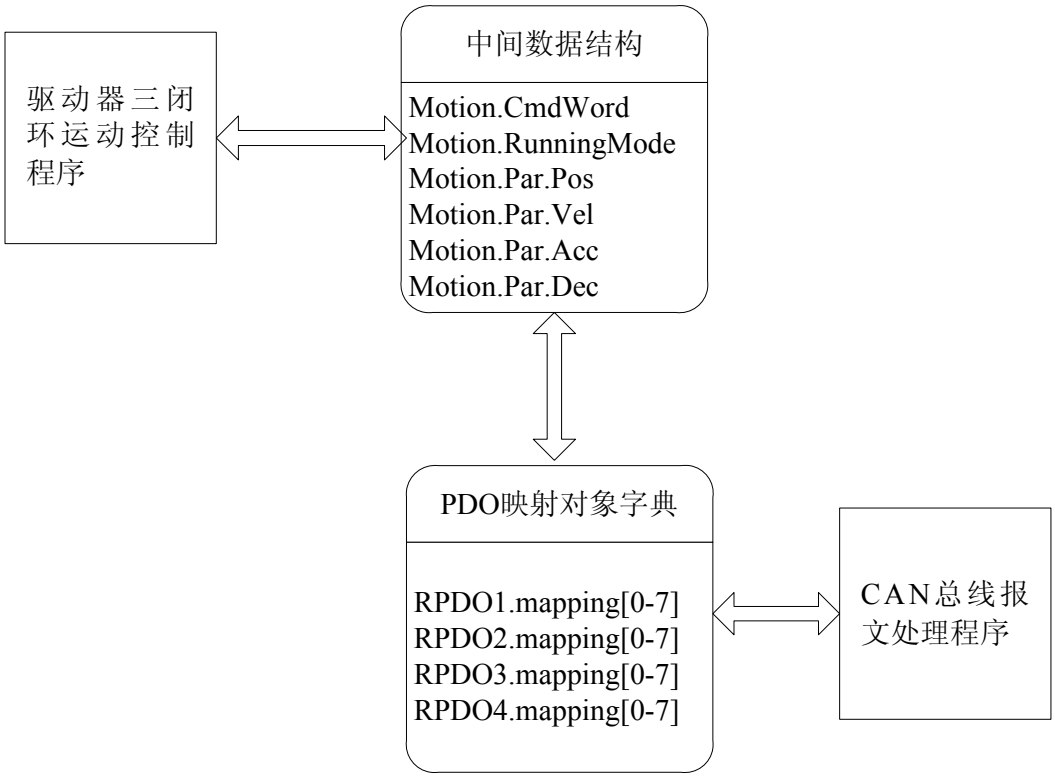

图 3-10 CAN 通信程序与运动控制数据接口图

在采用了中间数据结构实现 CAN 通信程序与运动控制程序的数据接口后，需定义 CANopen DS 402 协议字典项的 PDO 映射，最终实现通信程序到运动控制程序的数据接口。具体的 PDO 映射表将在后面的章节给出。

3.4. CANopen运动控制子协议DS 402 与PDO映射

CANopen 运动控制子协议 DS 402 是实际应用中的核心协议。协议定义了设备的标准状态机，同时定义了标准的运行模式以及每种模式相关的对象字典，这些字典项均位于索引值为 0x6000-0x9FFF 的 CANopen 标准子协议区。下面将论述在现有驱动器中，以软件方式实现 CANopen DS 402 协议的具体方法。

3.4.1. CANopen DS 402 协议及其状态机实现

CANopen DS 402 的标准状态机如下图所示[26]：

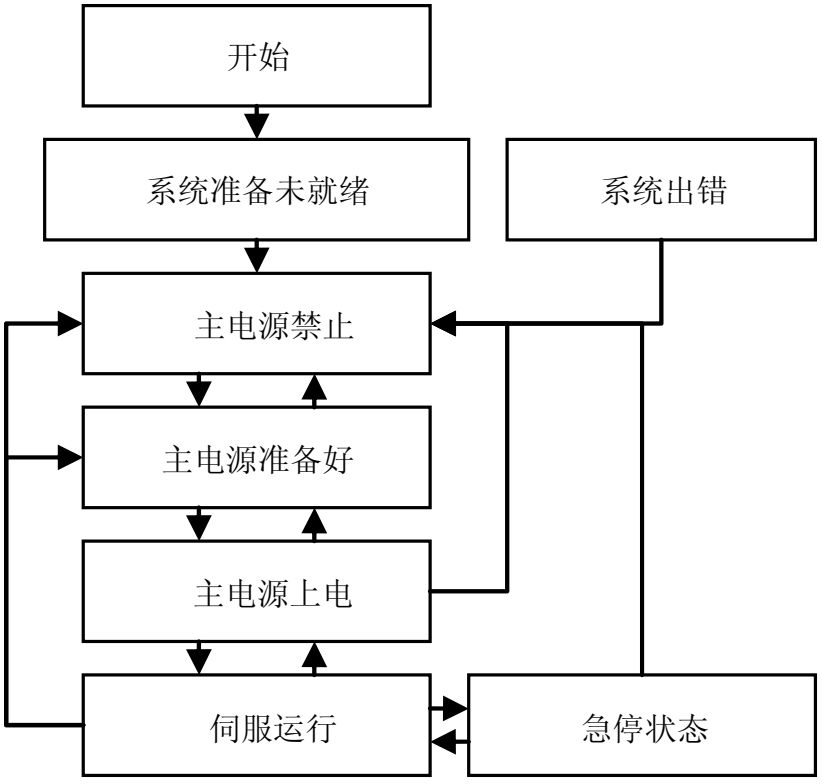

图 3-11 CANopen DS 402 协议状态机

标准状态机的状态切换是通过操作对象字典中的控制字来完成的，作为运动控制子协议 DS 402 字典区中最重要，操作最频繁的字典对象，这里有必要对这设备控制字进行详细的说明。

3.4.1.1. CANopen DS 402 协议的设备控制字

CANopen DS 402 协议的设备控制字是一个 16 位无符号整数，在对象字典中的索引值为 0x6040，该字典项不含子索引。设备控制字的 16 位定义如下表所示：

表 3-8 CANopen DS 402 设备控制字定义

类型	读写权限	单位	取值范围	PDO映射	Memory
Unsigned 16	R/W	-	定义如下	YES	R

位编号	描述
0	Switch On. 表明驱动器已经上控制电，可以完成基本系统和参数初始化
1	Enable Voltage. 表明驱动器已经可以接受高压强电
2	Quick Stop. 低电平有效，执行快停操作
3	Enable Operation. 如果无故障可以使能驱动器
4	指令数据有效：表示驱动器可以刷新当前控制模式和运动参数到缓冲区，1：可以 0：忽略
5	是否立即刷新模式和参数PDO，0：当前运动完成后刷新，1：立即刷新
6	绝对定位模式或增量定位模式
7	Reset Fault. 上跳沿清除驱动器故障
8	Halt. 高电平驱动器暂停接受主机命令
9-10	运行模式定义：0：位置轨迹定位模式；1：速度轨迹模式；2：自动插补模式； 3：回零模式
11-15	保留

设备控制字中 bit0-bit3 与状态机切换相关，bit0 完成从主电源未上电到主电源上电的状态切换，bit1 完成从主电源禁止到主电源准备完毕的状态切换，bit2 完成从伺服运行状态到急停状态的切换，bit3 完成从主电源上电到伺服运行状态的切换。

3.4.1.2. 在原驱动器程序中实现标准状态机的方式

根据控制字定义，可以按照标准状态机切换流程实现独立的状态机程序。对于状态机程序如何植入原驱动器程序的问题，则可以采用和 CAN 报文处理函数类似的方法。状态机程序模块独立于原驱动器程序模块，两者之间不存在耦合关系，因此可以直接放在原驱动器的背景循环中执行。实际设计中状态机切换作为驱动器程序 1ms 的周期任务运行，根据对象字典中的控制字刷新驱动器状态。

3.4.2. 标准运行模式的实现以及PDO映射

CANopen 运动控制子协议 DS 402 定义了多种标准运行模式，包括轨迹位置模式，轨迹速度模式，插补模式，回零模式等，每种模式都定义了相关的对象字典。如何利用伺服驱动器已有的三闭环控制结构，实现标准运行模式，是实现应用层协议的主要问题。

3.4.2.1. 标准运行模式的PDO映射

实现标准运行模式的前提是实现数据交互。DS 402 协议定义了标准运行模式下的指令和状态量，这些数据均属于驱动器的实时数据，适合采用 PDO 映射的方式进行交互。实际设计中，驱动器设备主要支持如下几种标准运行模式：轨迹位置模式，轨迹速度模式，回零模式。对于多轴联动控制中重要的插补模式，由于通信实时性和同步精度要求较高，更合适的解决方案是在工业以太网 Ethercat 中通过 COE 协议实现。

实际设计中标准运行模式的 PDO 映射表如下所示：

表 3-9 标准运行模式中的 PDO 映射

PDO	占用的字节	映射的对象意义	对象索引
RPDO1	2	控制字（Controlword）	0x6040
	2	运行模式	0x6060
	4	目标位置（Target position）	轨迹位置模式：0x607A 回零模式：零位偏移量 0x607C
RPDO2	2	控制字（Controlword）	0x6040
	2	当前轨迹完成后停留时间	轨迹位置模式：0x60F3
	4	目标速度（Target velocity）	轨迹速度模式：0x60FF 轨迹位置模式：0x6081 回零模式：0x6099
RPDO3	4	目标加速度（Target accerleraty）	0x6083
	4	目标减速度（Target accerleraty）	0x6084
RPDO4	8	模式运行控制字	插补模式：0x2012 回零模式：0x6098
TPDO1	2	状态字（Statusword）	0x6041
	2	显示运行模式	0x6061
	4	实际位置（Target position）	轨迹位置模式：0x6064
TPDO2	2	状态字（Statusword）	0x6041
	4	速度实际值（Velocity actual value）	轨迹位置模式：0x6069 速度模式：0x606C
TPDO3	4	速度环指令速度	0x606B
TPDO4	8	模式运行状态字	插补模式：0x2010

以上 PDO 映射在驱动器主背景循环 CANopen 通信协议函数中完成，而紧接着驱动器将调用 CANopen DS 402 协议的状态机程序，如果切换到运行状态，则调用 CANopen DS 402 协议运行模式选择函数，根据用户指定的运行模式（索引值 0x6060 的字典项），将 PDO 映射的数据导入中间数据结构中。这时通信协议流程完成。只需要实现标准运行模式，就可以完全实现 CANopen 从站的功能。

3.4.2.2. CANopen DS 402 协议标准运行模式的实现

CANopen DS 402 协议的实现主要有两方面工作，其中协议状态机的实现已经在前面进行了论述，而标准运行模式的实现则需要借助驱动器原有的运动控制程序。实现标准运行模式，首先要实现通信模块和运动控制模块的数据接口，前文已经讨论了中间数据结构的数据接口方式，下面讨论数据接口的过程。实际中 CANopen DS 402 模块对每种标准运行模式都定义一个中间数据结构，存放该模式下的指令和状态量，CANopen DS 402 函数作为系统的周期任务，根据对象字典中的运行模式项定时刷新相应的数据结构。CANopen DS 402 函数流程图如下：

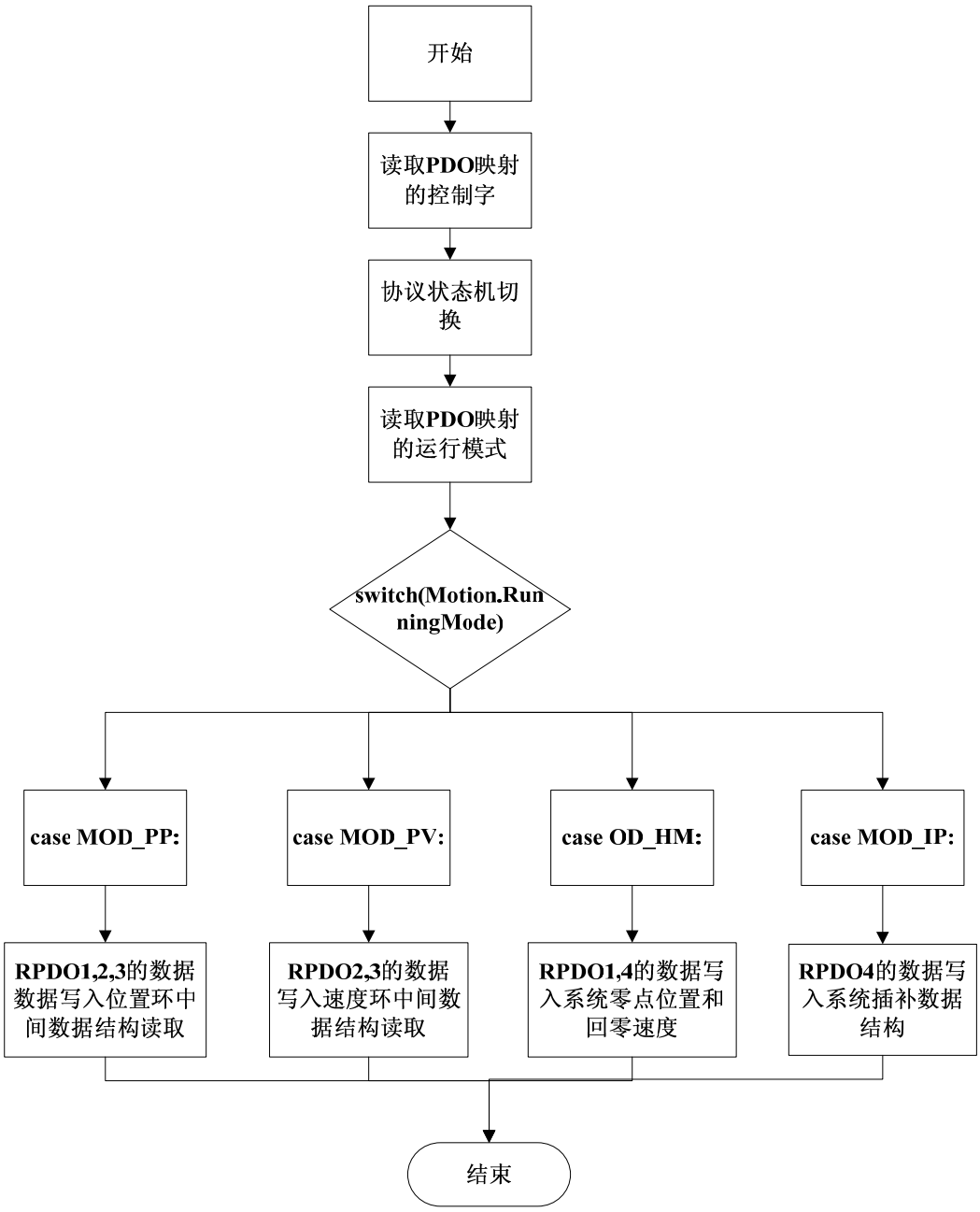

图 3-12 CANopen DS 402 协议函数流程图

实际中 CANopen DS 402 函数作为系统 1ms 周期任务，进行通信模块和运动控制模块的数据接口。在此基础上要实现标准运行模式，只需在伺服驱动器的三闭环控制程序中添加运行模式选择，对 CANopen 标准运行模式进行分支处理，即可在原有系统中实现标准运行模式的软件植入。以轨迹位置模式为例，将其植入后驱动器位置环流程图如下：

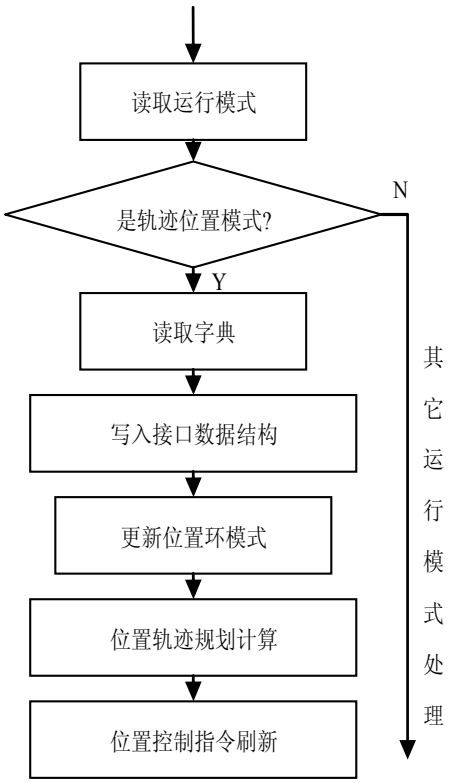

图 3-13 植入轨迹位置模式后的驱动器位置环流程图

在最终实现了 CANopen 标准运行模式后，交流伺服驱动器作为 CANopen 协议的从站，其协议定义的功能已经以软件方式实现并且植入原有程序。下面的工作主要是设计 CANopen 主站，并验证该实现方案的正确性。

3.5. 本章总结

本章围绕着软件方式实现 CANopen DS 402 协议的方案，论述了该方案的关键问题的解决方法。以实际的交流伺服驱动器为研究对象，在驱动器中以软件方式实现了 CANopen 协议的植入，进而在驱动器中实现了运动控制子协议 CANopen DS 402 协议。而交流伺服驱动器作为 CANopen 协议从站，需要 CANopen 主站对其进行网络管理、运行调度以及参数设置。下一章将介绍用 PC 机和 CAN 总线扩展卡实现简易的 CANopen 主站的软硬件解决方案，在此基础上进行详细设计，在 PC 机上实现简易的 CANopen 主站。然后以主从式通信的方式进行相关实验，验证从节点协议实现的正确性。

4. CANopen 主站设计以及协议验证

CANopen 协议的通信模型是非对等的主从通信模型，网络中需要一个主站，由主站进行网络管理，从节点对象字典的维护和读写。在运动控制子协议 DS 402 中，也是由主站接收用户请求，对从节点设备发送标准命令。主站汇总了从站的所有信息，是系统与用户交互的主要平台。因此 CANopen 主站的主要设计任务是实现子站管理和用户交互。

在常见的分布式交流伺服系统中，作为 CANopen 协议主站的控制台一般都是工控机。本文的课题主要是 CANopen 协议及其子协议 DS 402 在伺服驱动器中的实现，主站的作用主要是验证从站协议的正确性，因此直接使用普通 PC 机作为硬件平台，在 Windows 下编程实现简易的 CANopen 协议主站。

4.1. PC机CANopen主站设计

主站硬件平台介绍：一般的 PC 机都不支持 CAN 总线接口，因此需要 PCI 扩展卡实现从 CAN 总线到 PC 机标准 IO 总线的接口转换。实际设计中采用 PCI-7841 扩展卡来完成总线接口转换。该扩展卡采用 SJA1000 作为 CAN 总线控制芯片，支持 CAN 2.0B 总线协议。扩展卡自带两组 CAN 接口，可以方便的扩展网络。设备驱动提供了标准 C 语言函数库，库函数功能丰富，除了读写 CAN 总线数据功能外，还支持多种 CAN 总线控制功能，使主站的编程实现变得非常方便。

主站软件平台介绍：操作系统采用 Windows XP，编程语言为 Visual C++。Windows XP 提供了强大的系统调用功能，通过系统控件可以方便的实现用户交互。虽然 Windows 应用程序无法实现精确的定时，但是在分布式交流伺服系统并不像多轴联动系统那样要求精确的通信同步，因此采用 Visual C++编程实现简易 CANopen 主站是可行的方案。

主站程序需要对网络中所有从节点传输的报文进行监听，接收相关报文，显示在相关控件上，然后对报文做出相应处理；同时还要根据用户需求向指定从站发送

报文，读写从站对象字典或者命令从站执行标准的运行模式。这些工作具有并发的特点，因此主站程序采用了多线程编程，以方便并发性任务处理[40, 41]。主站程序界面如下所示：

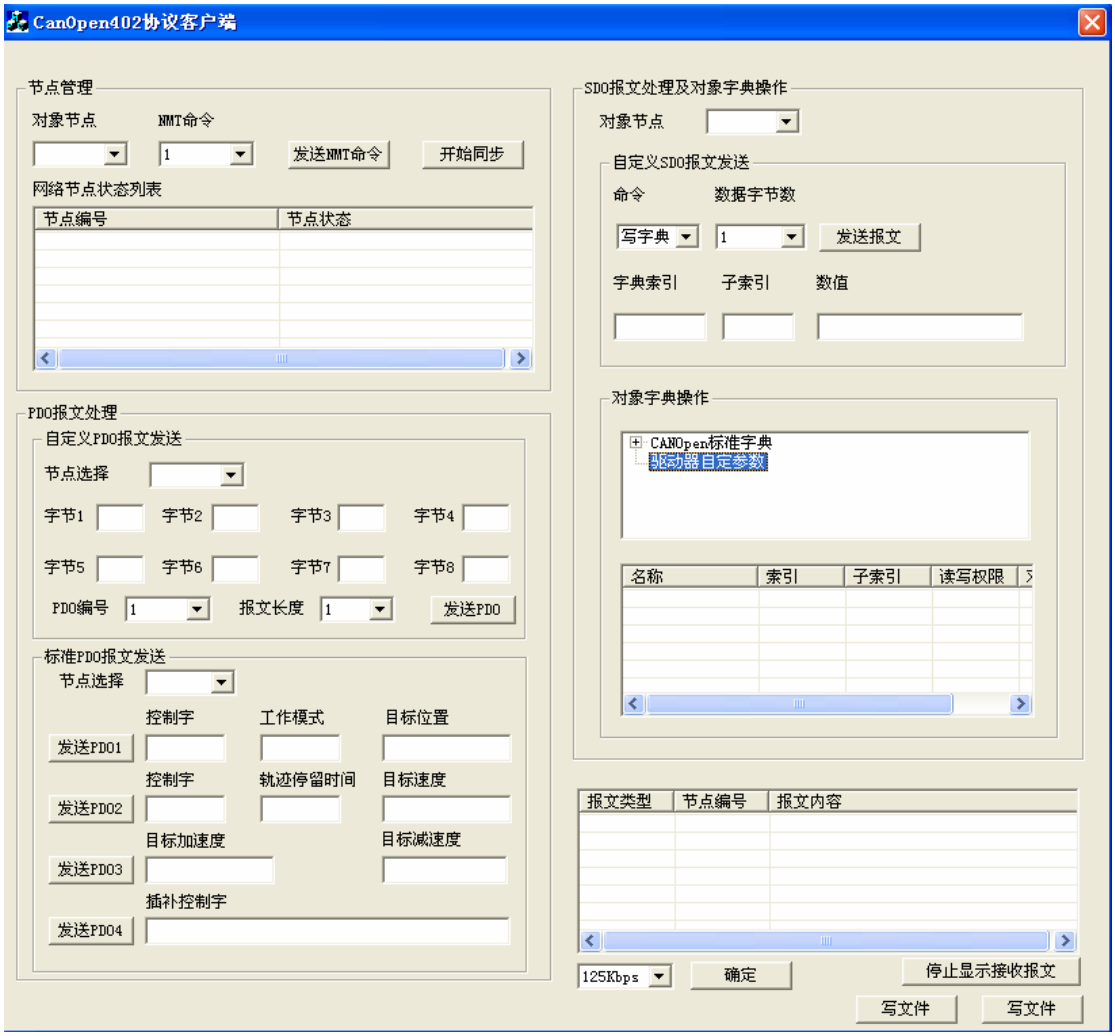

图 4-1 PC 机 CANopen 主站程序界面

主站程序包含 4 个功能模块，网络节点管理，SDO 报文处理及对象字典操作，PDO 报文处理，通信参数设置及接收报文显示等。以下分别说明。

1) 网络节点管理。程序以列表形式列出当前 CAN 网络中的所有在线节点编号以及节点状态。这个部分的功能通过 NMT——心跳报文完成。在主站程序中创建单独的接受线程，用于接收从站心跳报文。同时在主站程序中维护了一个在线节点数组，存放当前在线节点号。对每个在线节点的心跳报文计时，如果超过 10 个心跳周

期未收到该节点的心跳报文，则认为节点下线，并将其从在线节点数组中删除。另外通过 NMT 进行从节点网络状态管理，用户可直接在主站程序中向从节点发送运行/停止/重置命令。

2) SDO 报文以及对象字典操作。用户对从站字典的操作可以通过两种方式进行：在相应控件填充自定义 SDO 报文的索引——子索引以及字典操作码，发送自定 SDO 报文操作从节点对象字典；或者直接在用户界面中的对象字典操作区进行操作，操作从节点的对象字典。

3) PDO 报文处理。主站程序同样提供两种方式。用户可选择自定义 PDO 报文，使用界面控件填充 8 字节的 PDO 报文数据，再选择相应子站号进行发送。也可以按照从节点的标准 PDO 映射进行填充发送。

4) 网络通信管理。主要的功能是设定网络波特率，查看主站收到的所有 CAN 报文，以及将从站特定 PDO 的映射数据写入指定文件。这部分在调试应用层协议时有重要作用，可以通过控件直接查看从站发送到主站的所有 CAN 报文。而在从站进行大规模上传时，可以使用读写文件命令将从站上传的数据写入指定的磁盘文件中。

4.2. CANopen通信协议实验：梯形位置轨迹规划

下面以 PC 机为主站，伺服驱动器为从站，以 CANopen DS 402 协议标准的轨迹位置模式为运行模式进行试验。试验中，从站伺服驱动器按照主站 PC 机发送的命令来进行梯形位置轨迹规划，以验证软件植入方法实现 CANopen 协议的正确性。

CANopen DS 402 定义的轨迹位置模式支持多种轨迹规划算法，其中梯形轨迹规划是最常用的轨迹规划算法。下面先介绍驱动器中梯形轨迹规划的编程实现。

4.2.1. 梯形位置轨迹规划算法的C语言实现

交流伺服系统的三闭环结构中，速度环和加速度环（即电流环）均采用 PID 控制。伺服电机起动以最大加速度启动，直到达到给定速度附近[42]。其常见的响应曲线如下图：

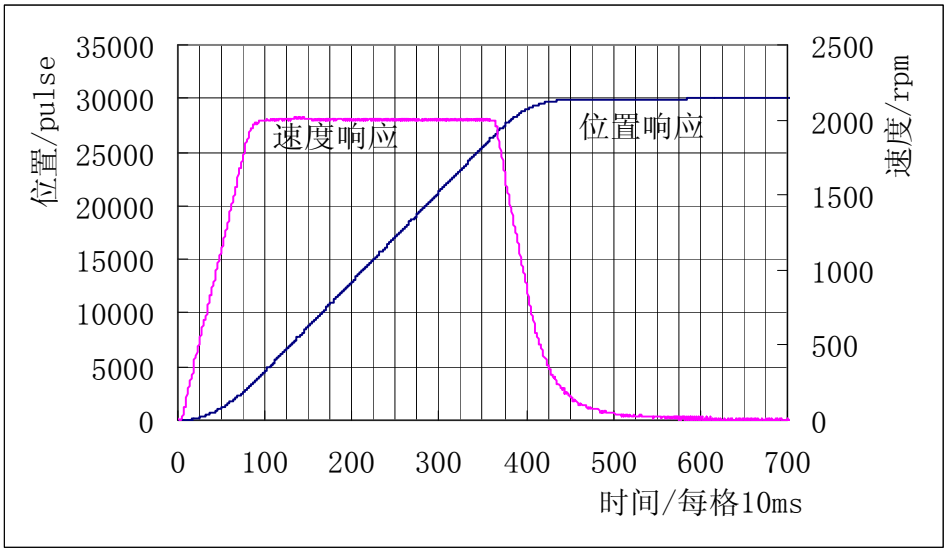

图 4-2 典型交流伺服系统速度响应曲线

而伺服电机直接与传动系统连接，在很多应用场合，传动系统可能无法承受电机最大加速度启动所带来的冲击，而梯形轨迹规划算法能解决这一问题[42]。通过设定轨迹加速度，可以控制速度响应曲线呈近似的梯形曲线，如下图所示：

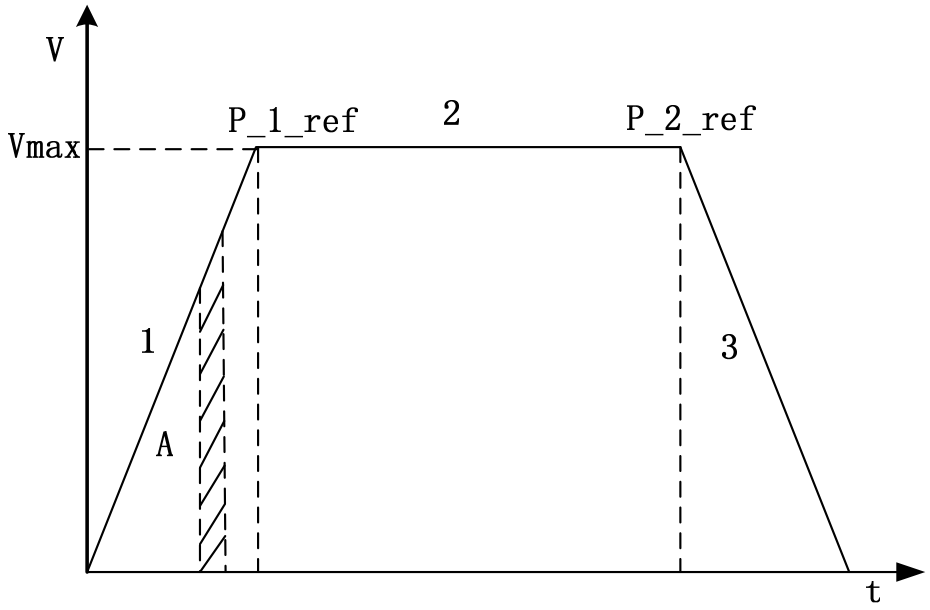

图 4-3 典型梯形轨迹规划速度响应曲线

梯形轨迹规划函数的 C 语言实现主要有两个难点：轨迹拐点的判定和量化余差的处理。对于拐点的判定可直接使用采样速度求累加面积的方法实现，而量化余差的处理则可以将余差与每个采样周期的位置增量比较，在与其最接近的两个周期中

插入量化余差的脉冲量，即可实现无差轨迹规划[42]。C 语言实现无差梯形轨迹规划的流程图如下：

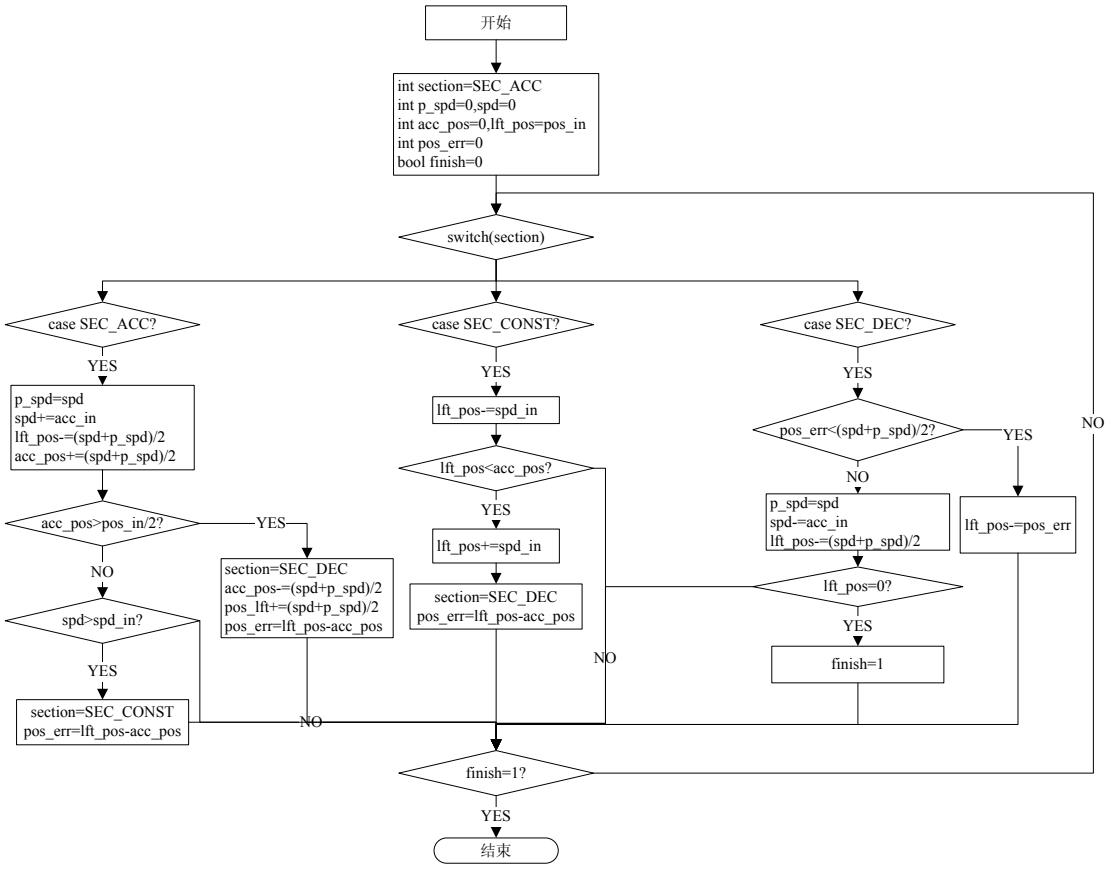

图 4-4 梯形轨迹规划 C 语言流程图

4.2.2. CANopen协议梯形轨迹规划实验结果

在梯形轨迹规划中，轨迹函数入口参数有三个：位置给定、最大速度以及加速度[43]。CANopen DS 402 协议对象字典对上述变量均有定义，它们在对象字典中的索引分别为：位置给定：索引号 0x607A；轨迹速度(即梯形轨迹最大速度)：索引号 0x6081；轨迹加速度：索引号 0x6083。按照本文第三章中 CANopen 运动控制子协议 DS 402 的 PDO 映射表，结合 CANopen 主站程序，可以设计相关实验，主站发送标准的 PDO，命令从站进行梯形轨迹规划。

实验过程如下：利用主站软件对从站进行初始化配置后，让从站进入运行状态，并选择轨迹位置运行模式；然后依次对从站发送 PDO3，PDO2 和 PDO1，其中 PDO3

映射的是轨迹加速度，PDO2 映射的是轨迹速度，PDO1 映射的是目标位置。同时 PDO1 映射从站控制字，将控制字中的运行位置位，从站伺服驱动器马上响应主站的轨迹规划命令，进行梯形轨迹规划的相关计算。实际实验的数据如下：交流伺服驱动器光电编码盘分辨率为 10000pulse/r，系统的位置环周期为 T=300μs。位置指令为 P_{ref}=0x1000 pulse，格式为 32 位整数；匀速段的速度指令给定为 V_{ref}=0xA0000 pulse/T，其中低 16 位表示小数部分，则实际速度给定折算后为 V_{ref}=200r/min。加速度给定为 a=0x1000 pulse/T^2，低 16 位表示小数部分。对轨迹规划过程中驱动器速度指令和速度反馈采样，通过 CAN 总线回送采样数据到 PC 机，实验结果如图 5 所示：

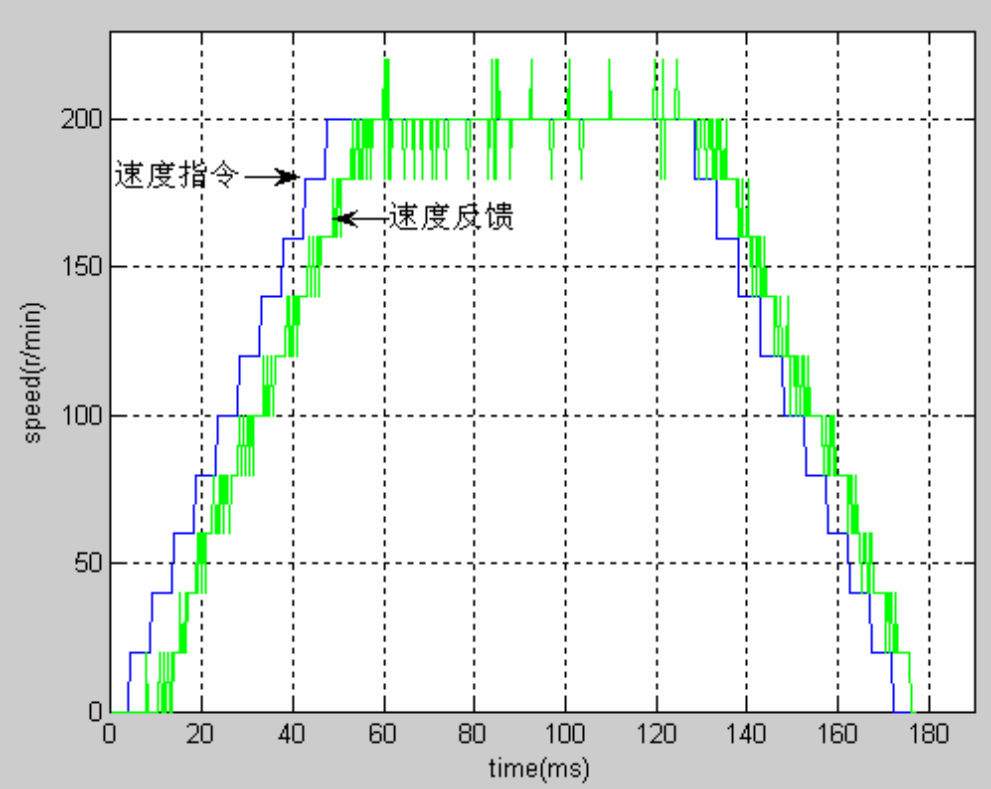

图 4-5　梯形轨迹规划实验结果曲线

从上图可以看到，交流伺服驱动器作为 CANopen 协议从站，能够正确的接收主站发送的指令量（包括位置指令，速度指令，加速度指令等），并能根据主站指定的运行模式，正确响应主站的命令，完成主站指定的梯形轨迹规划任务。实验结果证明了软件实现 CANopen 协议的方案可行性和具体设计的正确性。

4.3. 本章小结

本章以 PC 机为 CANopen 协议主站，通过主从通信的方式，对从站梯形轨迹规划进行了具体实验。其结果证明从站 CANopen 协议已得到了正确的实现，进一步证明了本文中所提出的软件实现 CANopen 协议的方案的可行性，以及前面章节所叙述的软件实现方法的正确性。

5. 全文总结及展望

本文主要针对在交流伺服驱动器中实现 CANopen 协议这一课题，对 CAN 总线和 CANopen 协议进行解析，提出了软件方式实现 CANopen 协议植入的具体方案，并且进行了相关的研究和设计，主要包括如下几方面的工作：

1) CANopen 协议通信栈的具体设计以及编程实现

2) CANopen 运动控制子协议 DS 402 的具体设计以及编程实现

3) CANopen 协议栈植入原驱动器程序的实现

本文在实现标准的 CANopen 协议栈的同时，考虑了在交流伺服驱动器系统资源有限的条件下，如何能经济的实现 CANopen 协议。因此对于 CANopen 协议定义的一些可选功能，例如动态 PDO 映射等，并未加以支持。纵观整个课题，有以下几个方面可以进行进一步研究。

1) 动态的对象字典实现，即实现字典项的新建，插入，删除等功能。如何设计动态字典的数据结构和存储结构，有效率的实现这些动态字典操作，是第一个值得研究的问题。

2) 真正面向对象的协议栈设计。CANopen 协议从本质来说是一个面向对象的应用层协议，如何在不影响效率的情况下，以面向对象的思想实现 CANopen 协议，是第二个值得研究的问题

3) 可裁剪的动态 CANopen 协议栈。如何利用前两个方面的工作，并加入相关的控制模块，实现可剪裁的动态 CANopen 协议栈，以根据不同应用场合的需求，不同档次的产品，来提供不同的协议栈支持，这是第三个值得研究的问题。

致 谢

在华中科技大学里度过的近 7 年时光转眼即将过去，7 年的校园生活令人难忘，而两年半的研究生生涯更是让我收获颇丰。

本文是在李叶松教授的指导下完成的。李老师高深的理论知识，丰富的实践经验和严谨的治学态度是我学习的楷模。而在两年半的研究生学习期间，李老师对我各个方面都予以了极大的关心和帮助，从平时生活到工作学习再到科研实践，这些关心和帮助使我感激不已。而李老师平时的谆谆教诲以及以身作则的风格更是我终生学习的楷模。在这里我要向李老师致以衷心的感激和深深的敬意。

在平时的学习和科研过程中，实验室的各位同学也提供了各种建议和帮助。感谢赵云博士，李文虎硕士，杨运海硕士在学习和科研过程中给予我的帮助，实验室平时的讨论就是最好的科研经验积累方式，最丰富的创意源泉。同时也要感谢实验室的所有研究生同学，平时学习生活中的相互照顾总是能令我深深的感到集体的温暖。

同时我也要感谢我的父母，我的家人，感谢他们为我提供了进入大学学习以及进行研究生深造的条件，并且感谢他们一直以来在背后对我学业和科研的默默支持，在今后走向社会的工作过程中，我将努力奋斗，回报家人对我的支持。

最后感谢所有曾经关心过我的人，我将以我今后的努力工作来回报你们的关怀。

参考文献

[1] 李国厚, 杨青杰. 智能伺服技术及其应用. 微计算机信息, 2005, (9): 128～130

[2] 王健. 现代交流伺服系统技术和市场发展综述. 伺服控制, 2007, (1): 16～21

[3] 秦忆. 现代交流伺服系统: 湖北武汉: 华中理工大学出版社, 1995

[4] 郭庆鼎. 交流伺服系统: 王成元. 北京: 机械工业出版社, 1998

[5] 饶运涛. 现场总线CAN原理与应用技术: 2. 北京: 北京航空航天大学出版社, 2007

[6] 苏奎峰. TMS320F2812原理与开发: 北京: 电子工业出版社, 2005

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

[8] 史久根. CAN现场总线系统设计技术: 北京: 国防工业出版社, 2004

[9] 李博, 李晓汀, 郇极. CANopen运动控制协议驱动程序设计. 组合机床与自动化加工技术, 2007, (4): 52～55

[10] 张厚林. CANopen通讯协议设计与实现: [硕士学位论文]. 华中科技大学, 2009

[11] Modicon Inc. Modicon Modbus Protocol Reference Guide: 1996

[12] Modicon Inc. MODBUS Application Protocol Specification:

[13] Modicon Inc. MODBUS Application Protocol Specification:

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

[15] 杨夫星, 谈世哲. 智能总线协议适配器的设计与实现. 自动化技术与应用, 2009, (12): 86～89

[16] 赵苍荣, 周孟然. 基于ARM的CAN总线井下瓦斯监控系统. 工矿自动化, 2008, (6): 13～16

[17] CAN in Automation. CAN Specification 2.0 Part A:

[18] CAN in Automation. CAN Specification 2.0 Part B:

[19] CAN in Automation. CANopen Electronic Data Sheet Specification for CANopen: 1999

[20] CAN in Automation. CAN Physical Layer for Industrial Applications. CiA Draft Standard 102 Version 2.0: 1994

[21] CAN in Automation. Device Profile for Generic I/O Modules: 2002

[22] Robert Bosch GmbH. CAN Specification Version 2.0[EB]: 1991

[23] 郇极. 工业以太网现场总线EtherCAT驱动程序设计及应用: 北京: 北京航空航天大学出版社, 2010

[24] 周书平, 孙晓民. 基于CAN的OSEK COM规范研究与实现. 计算机工程与设计, 2006, (8): 1314～1316

[25] CAN in Automation. CANopen Application Layer and Communication Profile．CiA Draft Standard 301 ,Version 4.02: 2002

[26] CAN in Automation. CANopen Device Profile Drives and Motion Control CiA Draft Standard Proposal 402 ,Version 4.02: 2002

[27] 广州周立功单片机发展有限公司. CANopen 协议介绍: 2002

[28] Copley Controls Corp. CANopen Programmer's Manual，Version 3: 2006

[29] 陈骥. 基于CANOpen高级协议和ED调度算法的电动汽车网络协议研究: [硕士学位论文]. 天津大学, 2004

[30] SYS TEC electronic GmbH[S]. CANopen Object Dictionary(Software Manual): 2003

[31] ABB Automation Corp. CANopen_适配器模块_NCAN-02_安装和启动指南: 2008

[32] 邓遵义, 宁祎. CANopen协议剖析及其在伺服电机控制中的实现. 机电工程, 2007, (8): 39～41

[33] Texas Instrument Corp. Programming Examples for the TMS320F281x eCAN: 2003

[34] Texas Instrument Corp. TMS320x281x DSP System Control and Interrupts Reference Guide: 2006

[35] Texas Instrument Corp. TMS320x28xx, 28xxx DSP Peripheral Reference Guide: 2006

[36] Texas Instrument Corp. TMS320x28xx,28xxx DSP Enhanced Controller Area

Network(eCAN) Reference Guide: 2006

[37] Texas Instrument Corp. Running an Application from Internal Flash Memory: 2009

[38] 邓遵义, 宁祎. CANopen协议剖析及其在伺服电机控制中的实现. 机电工程, 2007, (8): 39～41

[39] 关静. MicroCANopen协议栈的实现及应用研究: [硕士学位论文]. 天津大学, 2007

[40] 侯捷. 深入浅出MFC: 武汉: 华中科技大学出版社, 2001

[41] 孙鑫. VC++深入详解: 北京: 电子工业出版社, 2006

[42] 杨凯峰. 单轴运动控制器的设计: [硕士学位论文]. 华中科技大学, 2008

[43] 丛爽, 尚伟伟. 运动控制中点到点控制曲线的性能研究. 机械与电子, 2005, (7): 16～19

附　录　攻读学位期间发表的论文

[1] 刘思捷，李叶松. 伺服系统 CANOpen 协议软件植入方法分析. 电气自动化（已录用）

CANopen协议在伺服系统中的软件实现与植入研究

作者： 刘思捷
学位授予单位： 华中科技大学