华中科技大学

硕士学位论文

伺服驱动器工业以太网接口设计——基于EtherCAT与CANopen技术

姓名：李文虎

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

随着工业自动化水平的不断提高，工业控制网络所需负担的工作也日趋繁重，整个网络中传递信息的规模和复杂度也在不断增长，这给控制系统提出了更高的要求。伺服系统作为一种对控制精度、动态响应等性能指标要求很高的控制系统，也必须面对这些问题。

本论文研究了将工业以太网技术应用于伺服系统的方法。通过将 EtherCAT 工业以太网协议与 CANopen 规范相结合，以 TMS320F2812 系列 DSP 为平台，设计并实现了伺服驱动器的工业以太网通信接口，组建了网络化的运动控制系统。

通过分析 EtherCAT 与 CANopen 相关技术细节，阐述了将 CANopen 与 EtherCAT 相结合的关键点，给出了多种运动控制模式的设计方式，分析了软件设计和实现的具体方法和要点。

本文按照分层和模块化的方式给出了通信接口的设计过程，按层次分为三个大的模块：EtherCAT 通信模块、CoE 通信模块与 CANopen 运动控制模块。对各个模块又根据功能分为多个子模块，其中 EtherCAT 通信模块主要包括：EtherCAT 状态机服务、邮箱服务和过程数据服务；CoE 通信模块包括：服务数据对象（SDO）服务、过程数据对象（PDO）服务、对象字典服务；运动控制模块包括设备状态机服务和多种运动控制模式的实现模块。对每个模块本文都给出了具体的设计与实现过程。

本文实现了四种运动控制模式下的实际控制结果，包括周期同步的位置与速度模式以及位置与速度轨迹规划模式。实验结果表明，系统能够满足高速度、高精度、高可靠性和同步协调的控制要求。

最后对所做工作进行了总结与展望。

关键词：伺服驱动系统　　工业以太网接口　　EtherCAT　　CANopen

Abstract

With the development of the Industrial Automation, the control network had to be charged for more communication tasks. The scale and complexity of the information needed to be transported in the network had been grown hugely, which asked for higher requirements of the control system. As a control network which had special requirement in control precision and dynamic response, servo driver system also had to face the same challenge.

This thesis researched on the methods of applying Industrial Ethernet technology in Servo System. Through combining the EtherCAT Industrial Ethernet Protocol with the CANopen Profiles and using TMS320F2812 serial DSP as the hardware platform, Industrial Ethernet communication interface for the servo driver was designed and implemented and finally a networked motion control system was set up.

Based on deep analysis of the EtherCAT and CANopen protocol, the key points of forming the CANopen over EtherCAT structure was elaborated and several motion control modes designs were illustrated, especially on the methods and cruces of the software implementation.

According to the layer and module of the entire software, this thesis gave the design procedure of the communication interface. It was divided into three main modules: EtherCAT and CoE Communication module and the motion control module. Subsequently, it divided each module into sub-modules according to their function. The EtherCAT communication module included mailbox, process data and communication state machine services. The CoE profile module comprised of the Service Data Object, Process Data Object, Object Dictionary services. The motion control module was formed by the device state machine and kinds of motion control modes. The design and implementation of each module was given in detail.

The results of the Cyclic Synchronous Position and Velocity mode and Profile Position and Velocity mode were given in this thesis showed that this system could satisfy the high speed, high precision, high reliability and synchronous unisonous control requirements. Finally, conclusions and exceptions were given.

Keywords: Servo drive System, Industrial Ethernet Interface, EtherCAT, CANopen

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：　　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。

保密□，在_____年解密后适用本授权书。

本论文属于

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：　　年　　月　　日　　日期：　　年　　月　　日

1. 绪论

1.1. 工业以太网技术的发展

工业自动化离不开通信，网络化也是控制系统的发展趋势所在。从 20 世纪 80 年代开始形成和发展起来的现场总线（Filed Bus）技术极大的推动了工业控制技术的发展，解决了传统集散控制系统的诸多问题，经过多年的发展，形成了如 Profibus、CAN、Modbus 等多种总线标准。现场总线控制系统与传统控制系统相比所体现出的可靠性、实时性、可维护性等方面的优势使其在众多领域都得到了广泛的应用。正是由于现场总线技术的普及，才使得基于 PC 的控制系统得以广泛应用。[1, 2]

然而随着现代技术的发展，特别是通信技术和计算机技术的飞速发展，传统的现场总线技术如今逐渐体现出其局限性[3, 4]：

- 控制系统传输信息量的不断增长使得现场总线的网络带宽日趋不足；
- 计算机能力的飞速发展使得现场总线的通信速率逐渐成为了系统的瓶颈，计算机性能得不到最大的发挥；
- 从系统上层信息层到底层设备层一般需要多层的总线结构，增加了系统的复杂度。

另一方面，以太网技术却得到了长足的发展。从出现至今，经过约 30 年的发展时间，其运行速度提高了三个数量级，从 10Mbps 到了 10Gbps（如今已有了 100Gbps 以太网的应用）。而且凭借其低廉的端口价格和优越的性能，以太网占据了整个局域网市场绝大部分的份额。

与传统现场总线技术相比，以太网有着诸多的优势[5]：

- 开放性好，有众多的应用协议支持；
- 通信速率高，100Mbps 的通信速率已经得到了非常成熟和广泛的应用；
- 技术成熟，开发成本低廉，有良好的兼容性。

于是人们开始思考如何将这种优秀的商业技术应用于工业领域，期望凭借其通信速率高、成本低、标准统一等优势逐步取代现有工控领域中纷繁的总线标准，这也就是工业以太网技术的研究和发展的开始。随着可靠性、确定性、实时性等作为以太网无法进入工业领域的障碍被一一扫除，如今工业以太网技术受到越来越多的重视和欢迎，得到了蓬勃的发展。众多厂商（包括传统现场总线厂商）通过如调整 TCP/IP 协议、引入实时调度层或使用专用的硬件设备等方式对以太网技术进行改造，

纷纷推出自己的工业以太网解决方案。目前工业以太网技术已经日趋成熟完善，在诸多领域都有了应用。

工业以太网一般来讲是指技术上与商用以太网（IEEE802.3 标准）相兼容，但在产品设计时，在材质的选用、产品的强度、适用性以及实时性、可互操作性、可靠性、抗干扰性甚至本质安全等方面能符合工业现场需求的以太网技术[4]。

工业以太网与传统现场总线相比有着自己的优势，主要包括[6-9]：

- 更宽的带宽和更大的数据包，能够满足自动化设备间大量、复杂的通信需求；
- 能够应用于同步、实时性能要求更高的领域，如运动控制；
- 灵活的网络管理和控制，利用现有应用协议可以方便的与信息管理层集成；
- 使得基于 PC 的控制系统能够充分发挥其潜能，可以将更多任务交由 PC 完成，实现新的控制模式。

可以预见，工业以太网将进一步促进工业控制系统向着的分散化、一体化、数字化和智能化的方向发展，为系统的实时性、可靠性和可操作性等方面带来新的可能。

1.2. 伺服驱动系统的发展

伴随着电力电子技术、网络技术、控制技术和计算机技术的发展，伺服驱动系统在性能、结构等方面都发生了巨大的变化，基本经历了从开环到闭环、从直流到交流、从模拟到数字的阶段，其总体趋势是全数字化、模块化、智能化和网络化。[10, 11]

在控制结构上伺服驱动系统大致经历了组合式模拟仪表控制系统、集中式数字控制系统(DDC)、集散控制系统(DCS)到现场总线控制系统(FCS)多个阶段，网络化的伺服驱动系统从 20 世纪 90 年代开始就已经有了广泛的研究和应用，是伺服系统发展的趋势所在。[12]

随着计算机能力的不断提高，基于 PC 的开放式数控伺服系统得到了快速的发展，特别是纯软件的 CNC 系统，将参数管理、轨迹规划、位置控制等交给 PC 处理，使得系统的开放性、灵活性、可扩展和可操作性都有了更进一步的提高。而工业以太网技术则适应了这一发展的需求，如今随着工业以太网技术的发展和成熟，更是可以构建从信息层到设备层一网到底的网络化运动控制系统，凭借其优良的性能，使得基于 PC 的伺服驱动系统得以充分发挥其优势，将会是未来的发展方向之一。[5, 13]

1.3. 本文的研究工作

本文致力于研究将工业以太网技术应用于伺服驱动系统中的方法。通过阅读相关文献资料发现在国内外已经将 EtherCAT 技术应用于机器人、驱动系统、数据采集等诸多领域[14-33]，同时有对其实时性能、同步性能、与其它总线协议相比的优势等诸多方面的研究[34-41]，这些信息对本文研究工作的开展起到了很大的帮助作用。本文以工业以太网协议 EtherCAT 为基础，引入 CANopen 通信规范（DS301）为应用层，并结合 CANopen 驱动与运动控制规范（DSP402）为具体应用设计指南，以 TI 公司 TMS320F2812 系列 DSP 为硬件平台，设计实现伺服驱动器的工业以太网通信接口。以普通 PC 机位作为系统主站，以 windows 环境下倍福（Beckhoff）公司 TwinCAT 为主站控制软件，最终组建网络化的运动控制系统。伺服驱动器用于实现对电机高精度的定位控制，其自身的内部软硬件结构相当复杂，鉴于本文的定位是通信接口的设计，因此会将伺服驱动器的内部实现作为一个黑箱处理，重点关注输入输出（指令和反馈）。全文内容的安排如下：

第一章主要总结了当前工业以太网技术的发展概况，展望了伺服驱动系统的发展趋势，给出了本文的主要研究工作。

第二章对本文的研究方案从硬件和软件两个方面给出了一个整体的描述。

第三章详细分析了工业以太网协议 EtherCAT 以及 CANopen 通信和驱动规范，展示了其工作原理。

第四章具体描述了伺服驱动器工业以太网通信接口的软件设计与实现过程。

第五章给出了实际的实验结果。

第六章对全文所做工作进行了总结与展望。

2. 基于工业以太网接口的系统整体方案

EtherCAT 网络采用主从结构的通信模式。在主站方面，EtherCAT 没有任何特殊的硬件要求，本文以带网络接口控制器（NIC）的普通 PC 机作为主站，并直接采用倍福公司在 Windows 环境下的 TwinCAT 作为上位机控制软件。从站方面则采用自主设计的带 EtherCAT 通信接口的伺服驱动器。本文的目标是构建如图 2.1 所示的网络控制系统，整个系统由一个主站和一到多个从站组成，所有设备都处于同一网段中，不存在交换机。每个伺服驱动器均需提供两个 EtherCAT 接口以方便组建网络，主站通过网络接口发送 EtherCAT 报文，来完成对整个网络中设备的参数管理、状态维护以及指令发送等功能，伺服驱动器负责解析报文，执行相关指令并反馈实际状态，最终协同完成对位置、速度等的精确控制。

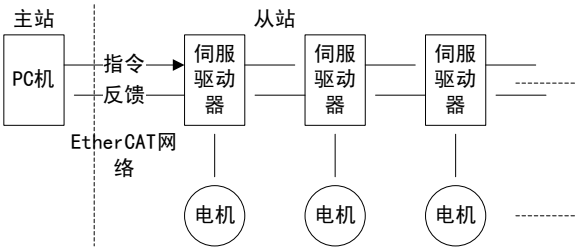

图 2.1 控制网络整体结构

Fig. 2.1 Structure of the Control Network

本文的主要设计与实现工作集中在伺服驱动器 EtherCAT 通信接口上，其整体分层结构如图 2.2 所示，其中物理层与数据链路层的功能完全由硬件实现，而应用层和应用的功能则由软件设计实现。

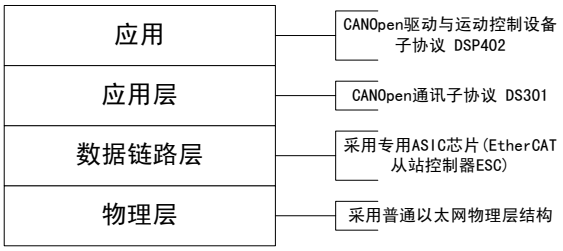

图 2.2 从站分层模型

Fig. 2.2 Layer of the Slave

2.1. 硬件结构

本文设计的工业以太网通信接口的硬件结构框架如图 2.3 所示。EtherCAT 通信协议与普通以太网协议完全兼容，因此物理层采用普通以太网物理层结构，由 RJ45

接口、网络变压器和网络收发器构成。

从站数据链路层采用倍福公司的 ASIC 芯片 ET1100（EtherCAT 从站控制器）实现。ET1100 通过介质独立接口（MII）与物理层通信，同时需要一块外部的 EEPROM 用于存储其初始配置信息和其它从站应用相关信息，此 EEPROM 通过 I^2C 接口与 ET1100 相连。

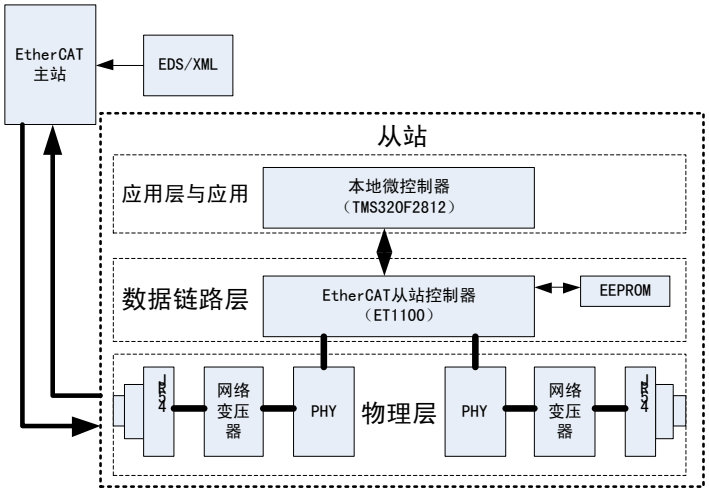

图 2.3 从站硬件结构框图

Fig. 2.3 Hardware Structure of the Slave

本文选用 TI 公司的 TMS320F2812 系列 DSP 作为从站应用层和应用软件的开发平台。DSP 通过外部接口（XINTF）同 ET1100 提供的外部设备接口（PDI）相连，接口结构如图 2.4 所示。ET1100 的 PDI 接口类型以及接口中信号的驱动方式可以通过 EEPROM 灵活的进行配置，以和 DSP 相匹配。

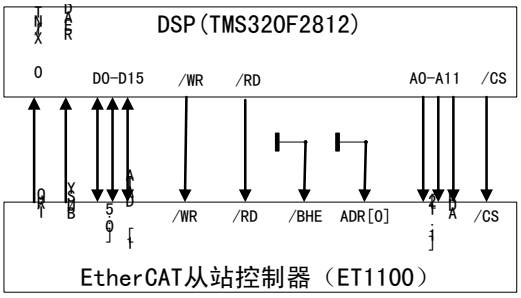

图 2.4 ET1100 与 DSP 的通信接口

Fig. 2.4 Communication Interface between DSP and ET1100

2.2. 软件框架

EtherCAT 通信接口中的应用层和应用的功能将由软件设计实现，此部分将是本文的研究重点。为了使整个软件结构清晰，思路明确，本文将在整体上采用分层的结构进行设计，底层设计为上层应用提供服务接口。对每一层，将根据功能划分为独立的模块。

应用层将主要完成网络通信相关的功能，其功能模块结构如图 2.5 所示，主要需要负责通讯状态的控制以及参数和实时数据的交换。EtherCAT 并没有设计自己完整的应用层协议，而只是为现有应用层协议提供了一个开放的接口，希望能够适应不同的应用需求，带来灵活的设计方案。本文将采用 CANopen 通信规范（DS301）作为应用层通信协议，在整体上构成 CANopen over EtherCAT（CoE）的结构。

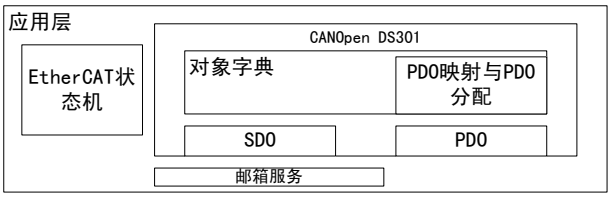

图 2.5 应用层软件框架

Fig. 2.5 Software Structure of the Application Layer

为了实现伺服驱动器运动控制相关的功能，同时提供一个标准的接口，本文将选择 CANopen 驱动与运动控制行规（DSP402）作为应用协议，构成如图 2.6 所示的软件结构框图，此部分将主要负责设备状态以及位置、速度控制等功能。

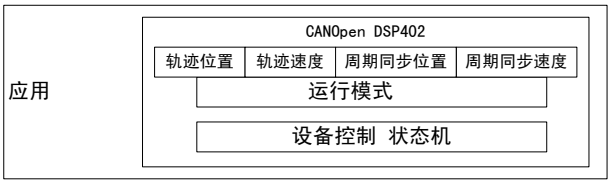

图 2.6 应用软件框架

Fig. 2.6 Software Structure of the Application

3. 网络接口协议分析

3.1. EtherCAT协议

3.1.1. EtherCAT简介

EtherCAT 全称为“Ethernet for Control Automation Technology”，即“用于控制和自动化技术的以太网”，最初是由德国倍福公司推出的开放式实时以太网解决方案，如今已被纳入 IEC61158 标准。现在 EtherCAT 技术由 EtherCAT 技术组（ETG）来独立维护，到目前为止，该组织已经有超过 1500 个成员[42]。

在从站设备中 EtherCAT 不再通过存储转发的机制来处理以太网报文，而是以字节为单位在传输过程中动态处理报文，这就使得报文的处理延时非常低。[43]

EtherCAT 采用集总帧结构，一帧以太网报文可以包含多个 EtherCAT 数据包，分别服务于不同的设备或控制任务，同时引入了逻辑地址映射的功能，使得主站能够灵活高效的组织报文数据，充分利用网络带宽。结合其分布时钟功能，使得 EtherCAT 特别适合于需要同步、实时、小数据量传输的应用领域。[44, 45]

EtherCAT 完全符合以太网协议标准，同时提供了灵活的拓扑结构，多种错误检测机制，对主站无硬件的特殊要求，使得其开发过程能够相对简单灵活。

凭借其在实时性、可靠性等方面的优势，目前 EtherCAT 技术已经得到了广泛的应用，特别是在对实时性要求较高的传感器、I/O 系统和伺服驱动等领域。

3.1.2. EtherCAT运行原理

EtherCAT 的介质访问控制采用主/从模式，以太网报文总是由主节点发送，从节点则从报文中读取指令数据或/并向报文中插入反馈数据，其过程如图 3.1 所示。

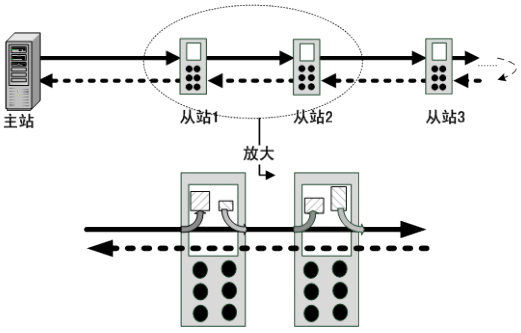

图 3.1 报文处理过程

Fig. 3.1 Frame Processing Procedure

不论物理上 EtherCAT 网段的拓扑结构如何，它在逻辑上都会是一个一端开口的

全双工环路。主节点发出的以太网报文经过从节点进行处理后，由网段中的最后一个从站以设备链相反的方向将经过处理的报文回送，经由第一个从站返回到主站作为回复报文。此过程中两个通信方向独立工作，充分利用了以太网的全双工通信能力。

从站设备根据它们在逻辑环路中的位置以字节为单位在以太网报文传输过程中动态的处理接收到的报文数据。每一个从站依据接收到的数据辨识相应的指令并执行，同时报文向环路中下一个设备传输，此过程典型的延时在 1 微秒以下。在报文经过从站设备时数据链路层负责接收和插入数据，因此和从站设备自身的处理能力没有任何关联。

EtherCAT 协议按照 ISO/OSI 参考模型进行规划，其整体框架如图 3.2 所示[46]。为了保证数据能够高效实时的传输，EtherCAT 在数据链路层引入了实时调度层，需要专门的控制芯片 EtherCAT 从站控制器（ESC）来实现。

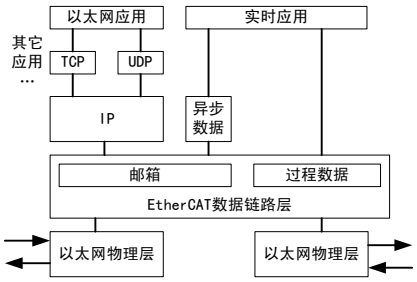

图 3.2 EtherCAT 从站设备模型

Fig. 3.2 EtherCAT Slave Device Model

在工业通信中，系统需要根据数据的传输特性来满足其特殊的要求。参数变量数据一般都会以非周期形式大量传输，实时性要求相对较低，而且传输一般由控制系统发起。诊断变量数据也是非周期和事件驱动的，但实时性要求相对要高一些，而且一般由外围设备发起传输请求。过程变量数据则一般需要以不同的频率周期性的发送，实时性要求最高。EtherCAT 在数据链路层引入了同步管理器和现场总线存储管理单元以及分布时钟技术，使得上层协议能够非常可靠、高效的完成这些不同要求的数据传输过程。

EtherCAT 将网络中的数据分为两个大类：邮箱数据和过程数据。邮箱数据用于传输非实时、非周期性的数据，通信模型采用客户端/服务器模型，采用点对点的方式进行通信。传输过程由客户端发起请求，服务器接收到指示并执行相关动作后给

出回复，最后客户端得到此次服务的确认消息，整个过程如图 3.3 所示。此通信模型采用握手机制，能够保证发送的数据一定能够被读取。过程数据则为实时数据，采用生产者/消费者模型，此模型中存在一个生产者和一个到多个消费者，结构如图 3.4 所示。此类通信过程没有直接的确认服务，但是 EtherCAT 会使用其它机制（如看门狗、工作计数器等）来监测数据的传输过程。此模型能够保证通信双方总是能够得到当前最新的数据信息。

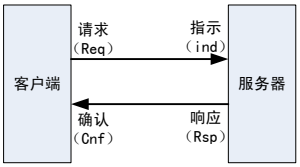

图 3.3 客户端/服务器通信模型

Fig. 3.3 Client/Server Communication Model

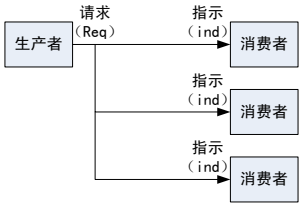

图 3.4 生产者/消费者通信模型

Fig. 3.4 Producer/Consumer Communication Model

在 EtherCAT 网络中，因为从站完全没有主动发送以太网报文的能力，因此上述通信模型的实现有其特殊性。对客户端/服务器模型，主站的确认服务需要其主动发送以太网报文来获取，即至少需要两个通信周期。而生产者/消费者模型中，借助 EtherCAT 从站特殊的报文处理方式，一个周期即可完成所有数据交换工作。

3.1.3. EtherCAT物理层

EtherCAT 使用标准快速以太网物理层结构，传输速率规定为 100Mbps，并需要采用全双工通信模式。物理层一方面为数据链路层提供一个发送数据的接口，对数据链路层的协议数据单元进行编码并添加相关报文信息，按照合理的时序透明的发送比特流，同时需要对发送请求进行确认，指示成功或失败；另一方面指示数据链路层端口有数据到达。物理层和数据链路层之间通过 MII 或 RMII 接口通信。

EtherCAT 对以太网物理层 PHY 芯片会有一些性能上的要求，同时为了能够高效快速的传输以太网报文，本文采用的数据链路层控制芯片 ET1100 对 MII/RMII 接口

进行了一些优化，如为了得到更低的处理/转发延时省略了发送 FIFO，同时为了更安全的通信引入了增强的链接检测和错误处理功能，另一方面为了组建 EtherCAT 通信网络每个从站至少需要两个物理层通信接口，这些都会对物理层结构提出一些要求。总的来说，对本文中物理层的实现会有如下的要求[47]：

- 推荐使用 MII 接口，RMII 接口会引入发送 FIFO，增加了转发时间
- PHYs 符合 IEEE 802.3 100BaseTX 标准，支持 100Mbit/s 全双工通信
- 使用自动协商机制（autonegotiation）
- 支持 MII 管理接口，用于错误处理或链接检测
- 支持 MDI/MDI-X 自动转换
- 链接丢失响应时间低于 15 微妙，减少出错时的数据丢失
- 不能改变报文前同步码的长度，因为 EtherCAT 报文在传输过程中动态进行处理，无法进行补偿
- PHYs 地址与 ET1100 端口的逻辑序列（0/1/2/3）相同或相差 16
- EtherCAT 从站控制器以及与其相连的所有的 PHY 需要有共同的时钟源

3.1.4. EtherCAT数据链路层

EtherCAT 数据链路层（DL）为自动化场合中互连设备之间的数据通信提供基本的实时性保障，以一个抽象的层次来看，数据链路层从以下几个方面为外部提供可见的服务[48]：

1) 服务的原子（primitive）动作和事件；
2) 与每个原子的动作和事件相关的参数以及它们所展现的形式；
3) 这些动作与事件之间的关系以及它们合法的执行秩序。

数据链路层需要计算、比较和产生帧校验（FCS），提供服务用于从以太网报文中提取数据或者向其中插入数据，这些数据通过物理存储空间提供给 DL 用户[48]。它是主站和从站应用之间交换数据的接口，提供了基本的数据访问服务，并提供了保障主从之间进行协调一致交互的相关基础设施。

3.1.4.1. 报文结构

EtherCAT 采用标准 IEEE 802.3 以太网报文结构，并使用保留的以太网类型 0x88A4 来和其它以太网报文相区分，因此，EtherCAT 能够和其它以太网协议并行运行。EtherCAT 也支持 IP/UDP 协议，但是对如本文研究的伺服驱动系统等很多工业领域的控制系统来说，在控制和设备层 IP 协议并没有必要实现。EtherCAT 报文基本结构如

图 3.5 所示[47, 49]。

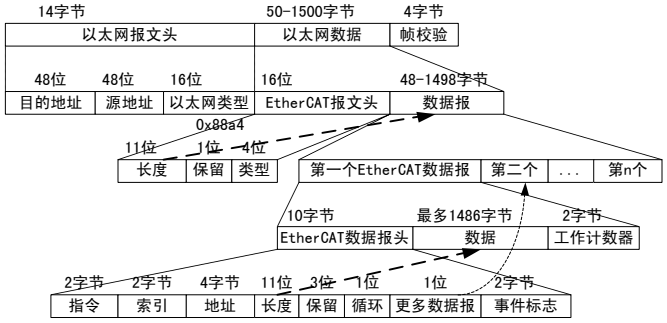

图 3.5 EtherCAT 报文结构

Fig. 3.5 EtherCAT Frame Structure

EtherCAT 将自己的数据报封装于标准以太网报文中，每一帧报文能够封装一到多个 EtherCAT 数据包，只有数据包类型为 1 的 EtherCAT 报文才会被从站处理器处理，即数据链路层将以字节为单位依次获取目的地址、源地址和以太网类型等信息，按照图 3.5 所示以太网报文、EtherCAT 报文和 EtherCAT 数据报三个层次逐步解析，更高层的应用协议将封装于数据报的数据部分并由具体应用进行解析。

本文所建立的控制系统中各个从站并不存在 MAC 地址，同时本文所使用的 EtherCAT 从站控制器也不会关心报文中目的地址和源地址的内容，只有源 MAC 地址会按照默认的处理方式进行处理（将 MAC 地址最高字节的第 2 位变为 1，使其成为本地管理地址(Locally Administered Address)），使得主站能够区分发送和接收的以太网报文。

3.1.4.2. 指令和寻址

EtherCAT 在数据链路层提供了灵活多样的访问从站地址空间的服务，具体对应于报文中数据帧头的指令和地址部分。这些指令和寻址方式依据系统所处的状态和传输数据的类型有着不同的用途。EtherCAT 的指令与寻址方式密切相关，下面先介绍 EtherCAT 支持的寻址模式。

总体上 EtherCAT 支持两种寻址方式：设备寻址和逻辑寻址。设备寻址按照寻址机制又可以分为三种：位置寻址、节点寻址和广播寻址。而节点寻址对应有两种地址匹配方式：配置站地址和配置站地址别名，结构如图 3.6 所示[47]。

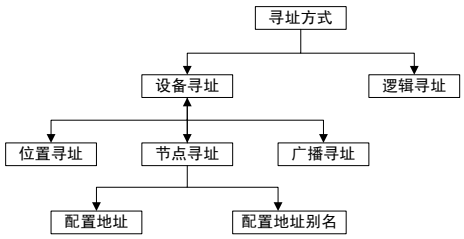

图 3.6 设备寻址模式

Fig. 3.6 Addressing Mode

这里的地址有三种不同的含义：从站节点的地址、从站存储空间的地址以及逻辑地址。为了访问从站存储空间，需要从站节点的设备地址以及节点存储空间地址。逻辑地址是 EtherCAT 引入的用于高效传输过程数据的寻址方式，会最终由 EtherCAT 从站控制器中的现场总线存储管理单元映射为节点物理地址。

对应每种寻址方式的读写操作就构成了 EtherCAT 的指令部分，指令决定了从站对报文中地址的解释方式以及处理方式。

- 位置寻址

对应 EtherCAT 自动增量类型的指令，包括自动增量 R、W、RW 和 RMW。从站控制器在接收到此类指令时会将地址解释为 16 位位置和 16 位偏移量。对位置部分每个从站都会将其加 1，如果发现其值为 0 则表示被寻址，于是按指令访问由偏移量和长度所指示的存储空间。此类指令主要用于在系统初始化阶段获取从站寄存器信息和配置从站站地址等，在运行阶段会很少使用。

- 节点寻址

对应配置设备地址类型的指令，包括配置地址 R、W、RW 和 RMW。从站控制器会将地址解释为 16 位的设备地址和 16 位偏移量。如果设备地址与从站配置地址或配置地址别名（需要主站使能）相匹配，就表明被寻址。此类指令主要用于在初始阶段配置从站寄存器，在运行阶段传输非实时的参数数据。

- 广播寻址

所有设备都将被寻址，对应指令包括广播 R、W 和 RW。地址部分解释为 16 位的位置和 16 位偏移量。位置值会被每个从站加 1，但是并不用于寻址。此类指令一般用于对所有设备都需要执行的操作如请求所有设备进入初始化状态。

- 逻辑寻址

需要从站控制器的 FMMU 功能部件的支持，对应指令包括逻辑 R、W 和 RW。

报文中地址被解释为 32 位逻辑地址，如果从站发现有自己的 FMMU 配置的逻辑地址空间与报文中逻辑地址区域相匹配，会由 FMMU 将逻辑地址转换为物理地址并执行相关指令。此类指令主要用于高效实时的传输周期性的过程数据。

上文中 R 对应读操作，W 对应写操作，RW 对应读写（交换）操作，RMW 表示被寻址的设备执行读操作，其它设备则执行写操作。

具体来说，假定主站在初始化阶段决定将网络中的第二个从站的设备地址配置为 0x1002，此时只能使用位置寻找方式，结合图 3.5 所示 EtherCAT 报文结构，主站发送的指令报文将具有如表 3.1 所示内容。

EtherCAT 数据链路层以字节为单位处理所接收到的报文，且数据格式需要按图 3.5 所示结构分层逐步解析获得。首先从站按照普通以太网报文结构解析数据，如果发现以太网类型为 0x88A4，则按照 EtherCAT 报文结构解析接下来的数据，否则将不再处理报文。如果之后 EtherCAT 报文头指示类型为 1，则继续按照 EtherCAT 数据报结构处理随后的数据，否则不再处理报文。依照 EtherCAT 子报文的结构，获取指令为 2，即自动增量写操作，于是从站将之后 4 字节地址解析为 2 字节位置和 2 字节偏移量，按前述位置寻址方式寻址——如表 3.1 所示，初始位置值为-1，第一个（逻辑位置）从站发现其不为 0，不执行指令，并将此值加 1。第二个从站接收到报文时位置值为 0，于是按照偏移和长度所指示的区域，即从站 RAM 空间地址 0x0010（从站配置地址偏移量）开始 2 个字节的区域，写入数据 0x1002，更新工作计数器，最终完成地址配置工作。同样第二个从站也会将位置值加 1，于是之后的从站也将不会被寻址。

表 3.1 配置节点地址报文示例

Tab. 3.1 Sample of the frame used for slave address configuration

目的地址	xxxxxx	偏移	0x0010
源地址	xxxxxx	长度	2
以太网类型	0x88A4	循环	0
长度	xxxx	更多报文	0
保留	0	事件标志	0
类型	1	数据	0x1002
指令	2	工作计数	0
索引	xxxx	填充	xx…xx
位置	0xFFFF	帧校验	xxxx

EtherCAT 数据报以一个 16 位的工作计数器结束，如果数据报中指令被成功执行

那么工作计数器就会被更新。对每一个数据报主站都会有一个期望的计数值，通过将实际的工作计数器值与期望值进行比较就能够检测数据报是否被正确的处理了。

3.1.4.3. 从站控制器

本文将采用倍福公司的 EtherCAT 从站控制器（ESC）ET1100 来实现 EtherCAT 数据链路层的功能。除了 MAC 层的基本功能，ESC 还提供了丰富的基础设施来保证主站和从站应用之间一致、高效的数据交换，其基本结构如图 3.7 所示[47]。

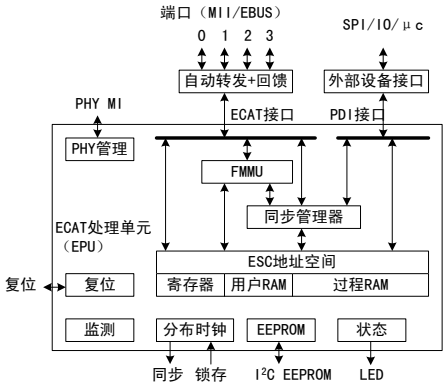

图 3.7 EtherCAT 从站控制器结构

Fig. 3.7 EtherCAT Slave Controller Structure

ESC 内部提供了一块存储空间，作为主从之间数据通信的中转站。主站（通过 MII 接口）和从站应用（通过 PDI 接口）都能够毫无限制的访问整个空间的数据，但此访问方式下如何保证数据传输的一致性，如何通知通信双方数据的变化，如何根据数据特性来满足其特定传输需求都会成为问题。为了解决这些问题，ESC 引入了两个非常重要的功能部件：同步管理器（SynMan）和现场总线存储管理单元（FMMU），与其它一些基础设施相结合，由硬件为主从之间提供协调、一致、高效的数据传输通道。

1) 同步管理器

同步管理器负责管理 ESC 中的存储空间，保证主站和从站之间一致安全的数据交换，是 EtherCAT 协议通信调度的关键所在。其通信方向和通信模式等参数都可以由主站自由配置，对所管理存储空间的读或写操作将可以产生相应事件，这些事件可以映射到事件标志寄存器中用于产生中断信号或供查询，以便指示主站或从站应用数据的变化。同步管理器的存在使得 ESC 区别于一个双口 RAM，因为对此存储区域的访问将会依赖于同步管理器的当前状态。为满足数据传输过程的不同需求，同

步管理器支持两种通信模式：邮箱模式和缓冲模式。

- 邮箱模式

邮箱模式实现了数据交换的握手机制，它只允许对存储空间交替的读写操作，这样就能保证不会有数据被丢弃。邮箱模式下只需要一块缓冲区，激活之后，此缓冲区能够被写入数据。在写操作完成后，只有数据被另一方完全读出，缓冲区才能够被再次写入数据。缓冲区必须由起始地址开始访问，否则访问无效，在结束地址被访问后会引起缓冲区状态的改变并引发相关事件。ESC 通过在内部为每个同步管理器维持一个状态机来实现此机制。对应接收和发送两个方向上的邮箱模式下同步管理器状态机如图 3.8 与图 3.9 所示。邮箱模式一般用于传输非周期的参数数据，实现多种应用层协议（包括本文中应用的 CANopen 协议）。[49]

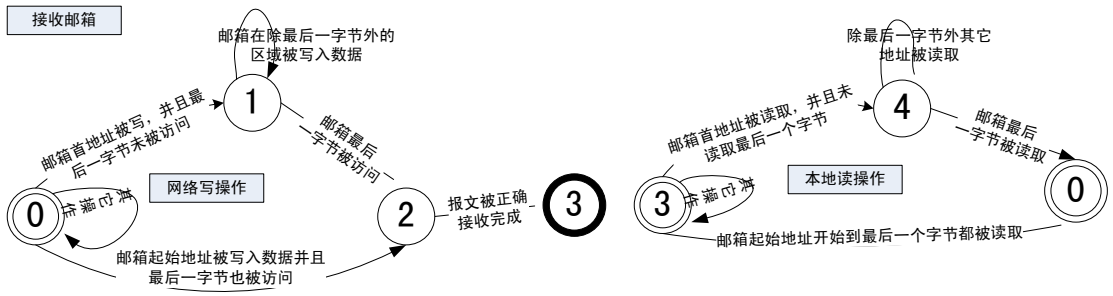

图 3.8 接收邮箱同步管理器内部状态机

Fig. 3.8 SyncManager State Machine for Receive Mailbox

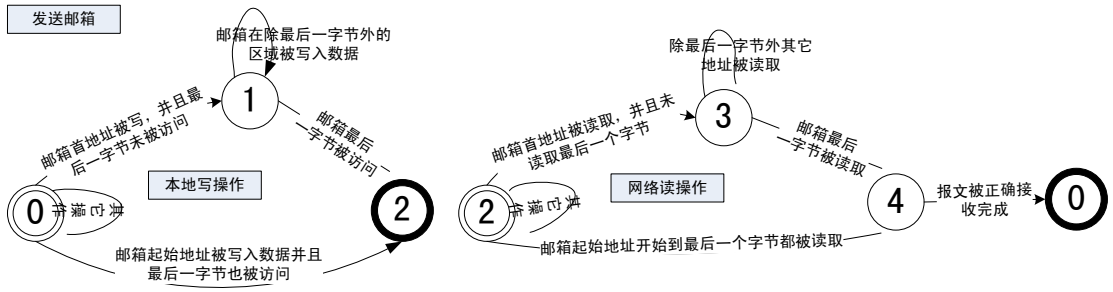

图 3.9 发送邮箱同步管理器状态机

Fig. 3.9 SyncManager State Machine for Send Mailbox

以接收邮箱为例，如图 3.8 所示，只有在状态 3 下本地应用才能开始有效的读操作，这就意味着只有主站成功完成对此缓冲区的写操作后本地读操作才能有效执行。同时，读操作要从其起始地址开始访问，这之后（状态 4）就可以访问其空间的任意地址，包括再次访问起始地址。当邮箱最后一个字节被访问之后，状态机进

入结束状态（状态 0），即意味着此方向上的访问被禁止并开启了主站写服务，同时也会产生相应的同步管理器事件。对于主站的访问，只有在报文被正确完整的接收到之后才会产生相关事件（即正确接收 FCS 之后）。

- 缓冲模式

缓冲模式使得 EtherCAT 主站和本地应用在任何时候都能够访问缓冲区。消费者总是能够得到由生产者发布的当前最新的数据，生产者也总是能够更新数据。如果对缓冲区的写操作快过读操作，那么旧的数据将被丢弃。缓冲模式一般用于传输周期性的过程数据。

缓冲模式使得读写操作能够同时互不干扰的进行，为了实现此机制，物理上缓冲模式需要 3 块连续的缓存空间，分别用于生产者访问、消费者访问和中间缓存。主站和从站对此区域的访问总是通过第一个缓冲区地址和大小来进行，也就是同步管理器所配置的区域。ESC 在内部会根据同步管理器当前的状态对访问进行重定向，实现机制如图 3.10 与图 3.11 所示。[49]

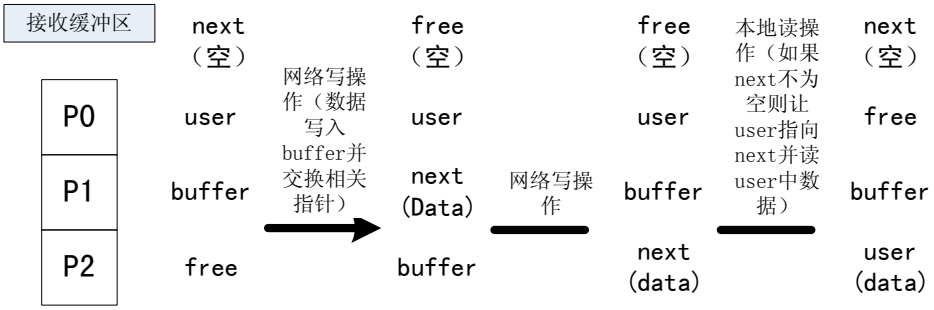

图 3.10 同步管理器接收缓冲区内部变换

Fig. 3.10 SyncManager Interconnection Used as Receive Buffer

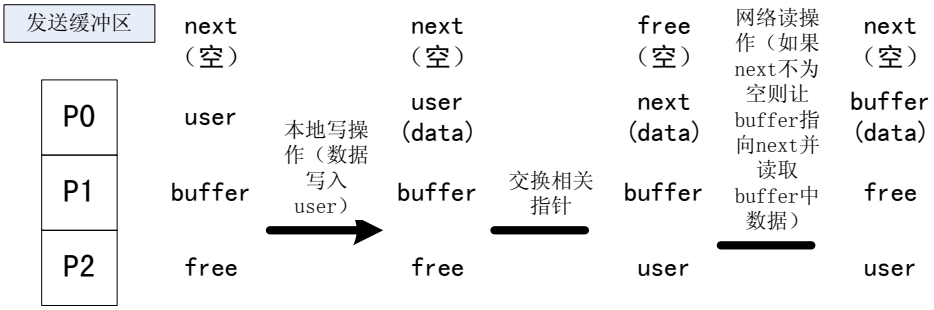

图 3.11 同步管理器发送缓冲区内部变换

Fig. 3.11 SyncManager Interconnection Used as Send Buffer

以接收缓冲区为例，内部分配的三块缓冲区为 P0、P1 和 P2，user（用于本地访

问）、buffer（用于主站访问）和 free（缓存）指针分别指向它们，next 为空。当有主站请求写缓冲区时，数据被写入由 buffer 指向的存储区域，此时如果 next 为空则让 next 指向 buffer，buffer 指向 free，free 置为空，同时产生同步管理器事件。此时若再次有主站写操作（在本地读取缓冲区内容之前），数据依然写入 buffer，然后 buffer 和 next 交换指针（因为 next 不为空），并再次产生同步管理器事件。若此时本地应用发现有同步管理器事件并开始读缓冲区内容，ESC 发现 next 不为空，则让 free 指向 user，user 指向 next，next 置为空，于是本地得到了 user 指向的当前主站所写的最新的数据。若紧接着又有一次本地读操作，因 next 为空，无任何指针交换操作，本地应用依然得到之前所读数据。

表 3.2 同步管理器配置寄存器

Tab. 3.2 Registers for SynchManager Configuration

地址（2 字节）	此同步管理器所管理空间的起始地址
长度（2 字节）	此同步管理器所管理空间的长度
控制字（1 字节）	控制同步管理器工作模式、事件和方向等信息
状态字（1 字节）	同步管理器读写状态和相关事件标志
激活（1 字节）	用于主站使能同步管理器
PDI 控制（1 字节）	用于从站控制同步管理器

同步管理器的配置寄存器如表 3.2 所示。为了展示同步管理器的工作原理与优势，本文假定主站和从站之间需要可靠的传输大量参数，为此，可以选择同步管理器 0 和 1 来负责此项工作。因为同步管理器只能够单方向工作，即对通讯的一方来说只能够执行读或者写一种操作，因此一般都需要两个同步管理器协同工作。因为用于传递参数，故配置为邮箱工作模式，具体来说可以将同步管理器 0 配置为起始地址 0x1000，长度 0x100，方向为主站写从站读，而同步管理器 1 配置为起始地址 0x1100，长度 0x100，方向为主站读从站写。假定主站需要请求获取参数信息，那么可以将请求信息通过配置地址写指令发送到对应从站，其中偏移地址为 0x1000，在主站访问地址 0x10FF 后同步管理器 0 将发生状态变化同时产生事件通知从站。在从站完整获取主站请求信息之前主站将无法再次发送其它请求，即请求信息不会丢失。同样的道理，从站将主站所请求的参数信息写入同步管理器 1 所管理器的存储空间中，主站查看对应事件标志，通过配置地址读指令（偏移量 0x1100）就可以高效可靠的获取参数信息，从而完成一次参数的传递过程。

2) 现场总线存储管理单元（FMMU）

为了更高效的传输过程数据，适应工业应用中小数据量传输的特性，EtherCAT 引入了逻辑地址的概念。EtherCAT 主站能够维护最多达 4GB 的逻辑地址空间，每个从站都能够分配逻辑空间的一部分用于通信，由主站进行计算并在初始化阶段进行配置，这些配置信息就存储于从站的 FMMU 中。FMMU 的唯一职责就是负责将一片连续的逻辑地址空间映射到 ESC 中一片连续的物理存储空间。[47]

FMMU 的存在使得主站能够高效自由的对过程数据进行组织，在实际通信中，一般只需要一帧报文就可以完成所有过程数据的接收发送任务。FMMU 使得主站能够实现类似于以太网中的组播功能，能够独立于从站的物理位置，依据从站的通信周期有选择的进行过程数据的更新。因为映射工作由每个从站硬件实现，因此主站无需对接收到的数据进行额外处理即可使用。在实际应用中，一般都会将同步管理器与 FMMU 相配合，即由 FMMU 将逻辑地址映射到同步管理器（缓冲模式）所管理的缓冲区，以实现高效一致的过程数据服务。

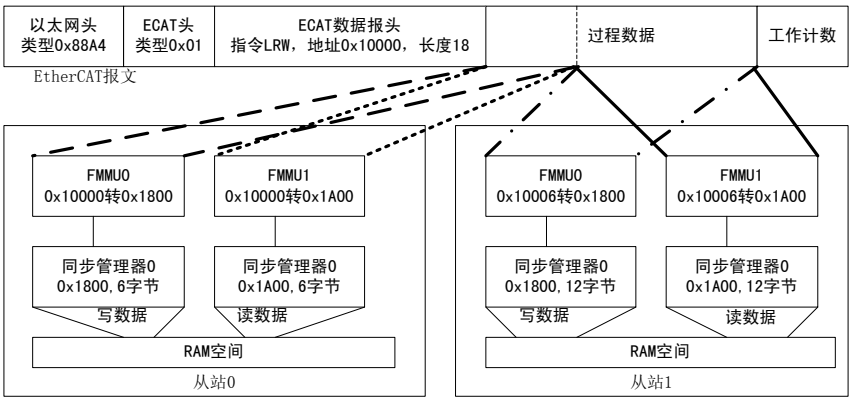

图 3.12 通过 FMMU 和同步管理器处理数据

Fig. 3.12 Data Processing through FMMU and SynchrManager

可以通过一个例子来体验 FMMU 所带来的灵活、高效的数据传输优势。假定 EtherCAT 网络中有一个主站和两个从站，主站和从站需要周期性的传输过程数据，其中从站 0 需要分别 6 个字节输入输出，而从站 1 则需要 12 个字节输入输出。此时对每个从站，可以选择两个 FMMU 配合两个同步管理器就可以高效可靠的完成这项工作。具体来说，从站 0 将同步管理器 2 配置为缓冲工作模式，起始地址为 0x1800，长度为 6 字节，方向为主站写从站读。同步管理器 3 配置为起始地址 0x1A00，方向为主站读从站写，其它与同步管理器 2 相同。从站 1 的同步管理器配置只需将长度设置为 12 字节即可。将从站 0 的 FMMU0 配置为逻辑起始地址 0x10000，长度 6 字

节，物理起始地址为 0x1800（映射到同步管理器 2 所管理区域），方向为主站写从站读，而从站 1 的 FMMU0 则可配置为逻辑起始地址 0x10006，长度 12 个字节，物理起始地址也是 0x1800。同时两从站的 FMMU1 配置为从站写主站读方向，物理起始地址配置为 0x1A00（映射到同步管理器 3 所管理区域），其它与 FMMU0 完全相同。如此一来，主站只需要使用逻辑读写指令，周期性的发送一帧报文，逻辑起始地址 0x10000，长度 18 个字节，就可以完成所有的过程数据输入输出更新工作，如图 3.12 所示。

3) 分布时钟（DC）

ESC 提供的分布时钟单元使得网络中所有设备都能够获得一个彼此相差极小（小于 1 微秒）的绝对系统时间，分布时钟为 EtherCAT 从站提供了如下的特性：

- 从站之间（以及与主站之间）的时钟同步
- 产生同步输出信号（SyncSignals）
- 输入事件的精确时间戳（LatchSignals）
- 产生同步的中断
- 同步的数字信号量输出更新与输入采样

所有支持分布时钟的从站都会有一个自己的本地时钟，此时钟在上电之后开始独立运行并有其独立的时钟源。为了同步各个从站的时钟，需要主站在初始化和运行阶段进行时钟的调整工作。

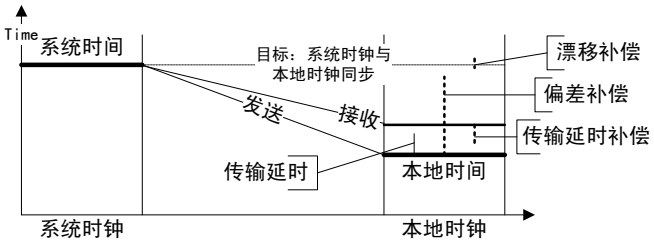

图 3.13 分布时钟同步调整

Fig. 3.13 DC Synchronization Process

EtherCAT 会将网络中逻辑上第一个支持 DC 的从站本地时钟选作系统时钟，作为整个网络的参考时钟，其它从站以及主站时钟都需要以此时钟为基准进行调整，如图 3.13 所示，需要调整的参数包括[47]：

- 传输延时

为各个从站设备提供参考时钟信息时，报文会产生线路上的延时以及处理延时，这些在进行时钟调整时都需要进行考虑。主站首先发送一帧时间戳记录报文，每一个支持 DC 的 ESC 都会记录接收到此报文（各个端口独立记录）的时间（本地时间）。然后利用 EtherCAT 网络逻辑环路的特点以及网络拓扑结构，主站根据所收集的记录时间能够很方便的计算出从站之间的传输延时并最终得到本地时钟与系统时钟间的传输延时。

- 偏差

本地时钟与系统时钟之间因为上电时间的不同或主站对系统时钟的更新等因素会存在初始的偏差。EtherCAT 不会直接调整本地时钟，而是通过记录此偏差并将其与本地时钟相加从而得到一个系统时钟副本来进行调整。在进行传输延时补偿后此偏差可以简单的由主站计算获得并对从站进行配置。

- 漂移

因为各个本地时钟都有着各自独立的时钟源（晶振），它们的时钟周期肯定不会完全一致，也就是存在着快慢，从而导致本地时钟之间产生漂移。为此 ESC 提供了一个闭环控制系统，通过系统时钟与本地副本间的差值来调整本地时钟计数值（每 10ns 计数 9 或 11）以进行补偿。

3.1.4.4. 数据处理机制

在 EtherCAT 网络中，从站控制器是数据解析、管理和交换的中心。同时主站和从站也需要充分利用 EtherCAT 从站管理器提供的基础设施来高效的交换数据。EtherCAT 网络中的数据交换过程有一些基本的特点：

- 访问寄存器采用影子缓冲机制，只有在报文检验正确之后操作才会生效；
- 参数一般通过同步管理器（邮箱模式）点对点的进行传输，采用客户端/服务器模式，因为 EtherCAT 从站没有发送报文的能力，所以回复和确认过程实际上需要主站发送额外的报文主动获取；
- 过程数据一般通过 FMMU 和同步管理器（缓冲模式）使用逻辑寻址传输，对同步管理器所管理的缓冲区，在主站报文校验正确之后才会发生状态变化并产生相关事件标志。

1) 报文处理

主站发送的以太网报文由 ESC 进行处理，并将解析的数据存入本地存储空间。为了能方便的组建通信网络，ESC 一般支持 2-4 个通信端口，用于和主站或其它从

站相连。ESC 内部处理报文过程如图 3.14 所示。[47]

报文处理过程和 ESC 当前端口状态相关，ESC 的每个端口都可以配置为开启、关闭、自动等多种状态。以本文设计中只使用两个端口（端口 0 和 1）为例，主站发送报文从端口 0 进入 ESC，通过自动转发和回馈模块进入 EPU，处理后到达端口 3，此端口未配置，相当于关闭状态，由回馈模块将报文送达端口 1，如果端口 1 处于开启状态，报文将由此端口送往其它从站，当报文经过其它从站处理后回到端口 1 由自动转发器发送到端口 2，如果端口处于关闭状态（对网络中逻辑上的最后一个从站），报文将由回馈模块送往端口 2，与端口 3 类似，报文将由回馈模块送达端口 0，最终回到主站。

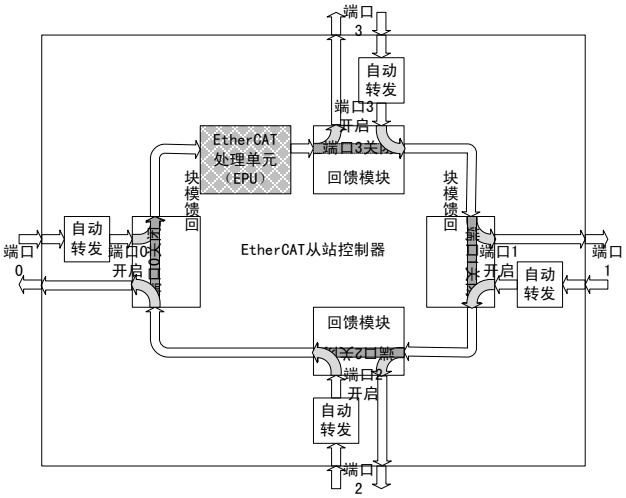

图 3.14 ESC 报文处理

Fig. 3.14 Frame Processing in ESC

2) 数据交换模型

主站和从站通过 ESC 的存储空间交换信息，此过程可以分为多种情况，如图 3.15 所示[48]。

主站写寄存器区域操作（1）可能会引发一个事件（如状态转换请求）通知 DL 用户，从而引发 DL 用户发出一个本地读请求以得到所写数据值（2）。否则，主站写服务将仅仅完成对寄存器区域写操作而不会通知 DL 用户（3）。DL 用户在任何时候都能够发起本地读请求。

DL 用户可以通过本地写操作来设置和更新寄存器（4）。主站读服务将仅仅获取寄存器的值而不会通知 DL 用户（5）。

对 DL 用户 RAM 空间的访问由同步管理器来协调。当然，也可以像访问寄存器一样直接访问此存储区域，但是此访问方式缺乏一致性保障，同时也无法产生相关事件来指示读写操作引起的存储区变化。

主站对 DL 用户 RAM 空间的写服务（6）会引发一个指示事件，DL 用户获取此指示（通过查询或中断）并发出本地读请求（7）以得到主站所写数据。

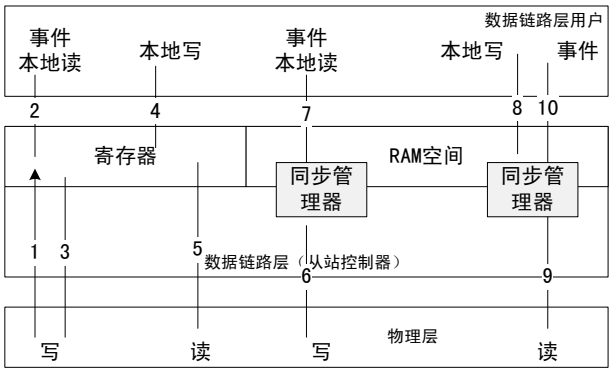

图 3.15 数据交换模型

Fig. 3.15 Data Exchange Model

DL 用户通过发出本地写请求（8）来写 DL 用户存储区域。主站通过发起读服务（9）来获取此区域所写数据，同时会引发一个事件指示（10）以通知 DL 用户此区域数据已经被读取，可以被再次写入数据。

3) 事件

为了方便的协调从站同主站间通信，ESC 提供了应用层事件请求和屏蔽寄存器，使得本地应用能够灵活的配置和获取自己所需要的事件，其基本结构如图 3.16 所示[47]。根据实时性要求的等级，本地应用可以选择通过查询标志位来获取事件信息，也可以将事件组合为中断信号。

本文关心的事件包括 EtherCAT 状态转换请求（用于通信状态机的维护）、同步管理器激活变化（用于邮箱重发）、同步管理器读/写（用于邮箱服务）以及分布时钟的同步信号 0 事件（用于过程数据服务）。为了保证每个通信周期中实时任务时间分配的确定性，本文只会通过事件屏蔽寄存器开放一个事件用于产生中断（根据同步模式选择分布时钟信号或用于过程数据的同步管理器读写事件），其它非实时事件采用轮询的方式进行处理。

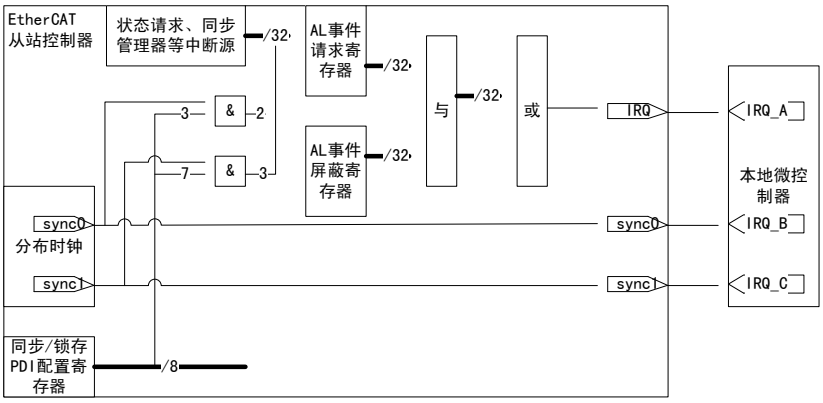

图 3.16 事件请求

Fig. 3.16 Event Request

至此，可以对图 3.7 所示的从站设备的结构有一个更为清楚和深入的认识，特别是主站和从站访问 ESC 的存储空间的各种方式以及其存在的意义有进一步的了解。

- 主站和从站可以直接的访问存储空间，此方式一般用于在初始化节点获取或配置从站寄存器，如通过位置寻址方式配置从站地址；
- 主站可以只通过 FMMU 访问存储空间，此方式一般用于主站批量的获取多个从站的状态信息，如获取用于邮箱通讯的同步管理器的当前状态，从站是不会通过 FMMU 访问存储空间的；
- 主站和从站可以通过只同步管理器访问存储空间，此方式下同步管理器一般工作于邮箱模式，用于可靠的传递参数；
- 主站可以通过将 FMMU 与同步管理器相结合来访问存储空间，此时用于高效的传输过程数据。

3.1.5. EtherCAT应用层

EtherCAT 应用层主要定义了三部分的服务：通信状态机、邮箱和过程数据。这三部分并没有构成一套完整的应用层协议，其主要目的是为其它应用层协议的提供一个方便的接口。本文中对 ESC 的 RAM 空间的访问将默认通过同步管理器实现。

3.1.5.1. 邮箱

EtherCAT 在应用层定义了邮箱协议，通过此协议 EtherCAT 可以支持丰富多样的应用层协议，如 SERCOS、HTTP、CANopen 等，此部分服务主要用于非实时的数据通信。邮箱协议数据将按照图 3.17 的格式封装于 EtherCAT 数据报的数据中进行传

输。

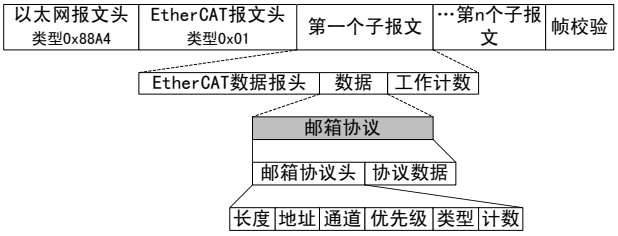

图 3.17 邮箱服务报文

Fig. 3.17 Frame Structure for Mailbox Service

其中图 3.17 所示邮箱协议部分的具体协议结构以及各部分所代表的意义如表 3.3 所示。[50]

表 3.3 邮箱协议结构

Tab. 3.3 Mailbox Protocol Structure

数据域	数据长度	值/描述
长度	2 字节	邮箱服务数据长度
地址	2 字节	如果主站是客户端，则代表源站地址，如果从站是客户端或者目的地址属于另一个网段则代表目的站地址
通道	6 位	0x00 (保留)
优先级	2 位	0x00: 最低优先级… 0x03: 最高优先级（本文未支持）
类型	4 位	0x00：出错 0x01：保留 0x02：EoE 0x03：CoE 0x04：FoE 0x05：SoE 0x06 -0x0e：保留 0x0f：用户自定义
计数	3 位	邮箱服务计数 (0 为初值，7 之后为 1。用于重复的邮箱请求以及检测丢失的邮箱服务。如果为 0 表示不需要检测。
保留	1 位	0x00
服务数据	“长度”字节	邮箱服务数据

那么从站是如何知道接收到的数据为邮箱服务数据呢？这就需要和同步管理器联系起来。首先无论过程数据还是邮箱数据都是通过同步管理器来交换的，如果知道邮箱数据是由哪个同步管理器来管理的，那么通过查询对应同步管理器事件标志，读取同步管理器所管理缓冲区数据，就可以将此数据作为邮箱服务数据来解析。那么如何知道那个同步管理器用于邮箱通信呢？同步管理器是在初始化阶段由主站进行配置的，但是从站在这方面并没有必要给予主站过多的灵活度，从站可以将固定的同步管理器用于邮箱服务（本文将采用此方式），并由从站配置文件记录此信息。主站需要通过配置文件获取同步管理器的配置信息，这样主站和从站就达成一致，可以开始邮箱通信服务了。一般会使用同步管理器 0 和 1 用于邮箱服务，并配置为

邮箱模式，分别用于主站的邮箱请求和从站的邮箱回复，这就意味着同步管理器 0 与 1 所管理的存储空间中的数据将总是具有表 3.3 所示的结构，从站应用根据其中的类型信息决定以何种方式解释之后的服务数据。本文将以此协议为基础实现 CANopen 协议，即服务类型为 0x03。

3.1.5.2. 过程数据

EtherCAT 协议为完成邮箱数据的交换定义了对应的邮箱协议，但对过程数据服务 EtherCAT 在应用层完全没有定义相关协议，因为在数据链路层已经提供了完整的设施以支持过程数据的通信：同步管理器、事件标志和中断、分布时钟、FMMU 以及丰富的读写服务。实际运行中，通常将同步管理器 2 和 3 配置为缓冲模式，用于过程数据服务，同时结合 FMMU 以及分布时钟的同步信号，通过逻辑读写指令，就可以高效一致的完成过程数据的通信。问题的关键在于如何针对具体应用定义过程数据的含义并让主站和从站达成一致，这给予了实现很大的灵活度，用户甚至可以很方便的实现自定义的过程数据服务。为了提供一个统一的接口规范，本文中将引入 CANopen 的应用层协议来协助实现过程数据服务。

3.1.5.3. 通信状态机

每一个 EtherCAT 从站都必须支持 EtherCAT 状态机（ESM），从站所处的状态限制了其当前所能够执行的操作。ESM 是主站和从站之间协调工作的基础，状态的切换一般是由主站发起转换请求，从站获取此事件并执行相关操作后给出状态转换结果。当检测到内部错误时从站也可以主动发起状态转换请求。状态的请求和回复通过 ESC 的 AL 控制寄存器以及 AL 状态和状态代码寄存器来实现。EtherCAT 总共定义了五个状态：初始化（Init）、预运行（Pre-operational）、安全运行（Safe-operational）、运行（Operational）和引导（Bootstrap），其中引导状态是可选的，主要用于实现 FoE（通过 EtherCAT 传输文件），来完成如固件更新等一些需要可靠传输大批量数据的功能，本文将不以支持。

EtherCAT 状态机的转换过程如图 3.18 所示[50]，从初始化到运行状态每次只能前进一步并且必须由主站请求。

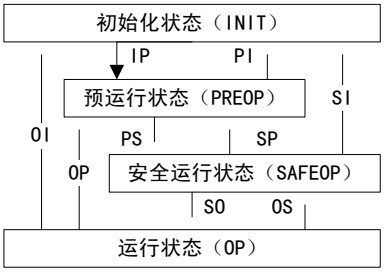

图 3.18 EtherCAT 通信状态机

Fig. 3.18 EtherCAT Communication State Machine

每一个状态都定义了相关的服务，主站在请求状态切换前应先提供相关服务，同样从站在确认状态转换前应执行相关检测工作并启动或停止相关本地服务，相关内容如表 3.4 所示[47, 50]。

表 3.4 EtherCAT 通信状态机服务

Tab. 3.4 EtherCAT Communication State Machine Services

状态/状态转换	服务
初始化	• 应用层无通信 • 主站能够访问从站 DL 寄存器
初始化-预运行	• 主站配置相关寄存器，至少包括： -DL 地址寄存器 -用于邮箱通信的同步管理器通道 • 主站初始化分布时钟同步 • 主站请求“预运行”状态 -主站设置 AL 控制寄存器 • 主站等待 AL 状态寄存器的响应
预运行	• 应用层能够进行邮箱通信 • 无过程数据通信
预运行-安全运行	• 主站通过邮箱配置参数： -如过程数据映射、PDO 分配等 • 主站配置相关 DL 寄存器如 -用于过程数据通信的同步管理器 - FMMU 通道 • 主站请求“安全运行”状态 • 等待 AL 状态寄存器的响应
安全运行	• 应用层能够进行邮箱通信 • 开启过程数据通信，但只处理输入数据，输出应处于“安全”状态
安全运行-运行	• 主站发送合法的输出 • 主站请求“运行”状态 • 等待 AL 状态寄存器的响应
运行	• 邮箱以及输入和输出过程数据都能够进行处理

3.1.5.4. 同步机制

对网络化的控制系统，实时数据的交换一般都会需要进行同步，如输入的采样和指令的执行，这样各个从站才能够协调完成有意义的任务。EtherCAT 支持三种运行模式：[47, 51]

- 自由运行

主站和从站应用以各自的时钟周期独立运行，没有同步关系，一般只会用于开发过程的测试。

- 同步管理器同步

各个从站应用通过同步管理器事件进行同步，也就是同步于报文，一般使用维护输出过程数据的同步管理器事件。受报文传输、处理等因素的影响，同步一般会有微妙级的抖动，可用于毫秒级的同步。本文采用的同步管理器同步结构如图 3.19 所示，在产生同步管理器事件后进行输出过程数据更新和输入过程数据采用。

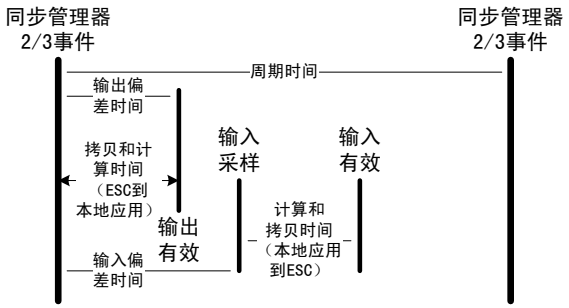

图 3.19 同步管理器同步模式

Fig. 3.19 Synchronous with SyncManager Event

- 分布时钟同步

各从站应用通过分布时钟提供的同步信号进行同步，可以提供更高等级的同步精度，抖动可达到纳秒级，可用于 1 毫秒以内的同步。本文采用的分布时钟同步模式如图 3.20 所示，采用 DC 的同步事件 0 进行同步，在 sync0 事件到达之前主站包含过程数据的报文必须已经由 ESC 处理完成，否则将视为同步出错。同时为了提高可靠性，降低系统的复杂度，在分布时钟同步模式下本文在过程数据到达后（即同步管理器事件发生）不会立即更新，会等到分布时钟同步事件后再做处理。

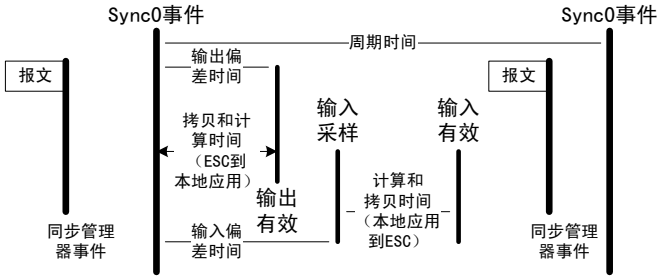

图 3.20 分布时钟同步模式（SYNC0）

Fig. 3.20 Synchronous with DC SYNC0 Event

本文中从站应用的运行平台为 DSP，它不但负责网络通信的应用，也要完成电机控制相关的功能，而这两部分至少需要两个中断处理程序：一个时钟中断，用于完成电流、位置等控制任务，一个则是由同步事件产生的外部中断，用于更新指令和发送反馈等任务。在同步模式下，主站应用和从站数据链路层（即 ESC）之间保证了相对稳定的相位关系，即同步事件与主站应用之间同步了，但是同步事件与 DSP 的时钟中断之间并没有任何关系，而且因为 DSP 与 ESC 使用不同的晶振独立运行，在伺服驱动器的运行状态中，为了在同步事件引发的外部中断与 DSP 内部时钟中断之间建立一个相对稳定的相位关系，需要两部分的工作：

- 在第一次外部中断到来时建立外部中断与 DSP 内部时钟中断之间合理的相位关系；
- 在这之后每次外部中断中，使用一个闭环的控制系统，根据当前实际相位差与期望值之间的差值，对 DSP 时钟周期进行微调。

3.2. CANopen协议

3.2.1. CANopen简介

CANopen 最初是由 CiA（CAN in Automation）开发的基于 CAN（控制局域网）总线的高层协议标准，它实际上是多种协议规范的集合，在发布后不久就获得了广泛的认可，如今在医疗设备、驱动系统、汽车制造等领域都得到了应用。[52]

CANopen 的目的之一就是希望为同种类型的设备提供一个标准的设备描述规范，使得不同厂商的这些设备能够满足可互换性，可以将其看作是一个独立于设备和制造商的描述语言。CANopen 协议提供了非常高效的数据组织和访问方式，在保证标准的同时为设备制造商保留了充足的自由度。CANopen 协议总体可以上分为两类：应用层通信规范以及各种设备子协议，其中通信规范是必须支持的部分，设备

规范实际是基于通讯规范的高层应用协议。

3.2.2. CANopen通信规范

CANopen 应用层和通信子协议（DS301）定义了 CANopen 的通信规范，此协议规定了设备描述的标准方式，定义了设备间的数据交换服务以及网络管理服务，规范了实时数据的配置和通讯以及设备间的同步机制。CANopen 将应用层提供给应用的服务逻辑上划分为不同的服务对象，每一个对象实现一个特定的功能并提供相关的服务。CANopen 将通过图 3.21 所示的模型来构造设备。[53]

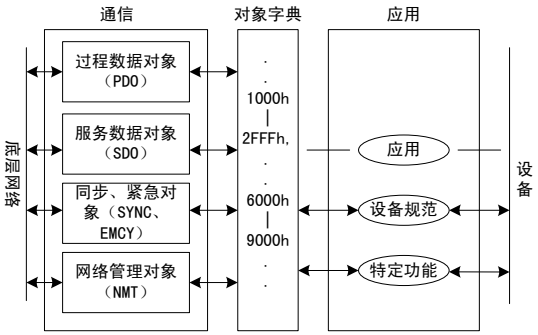

图 3.21 CANopen 设备模型

Fig. 3.21 CANopen Device Model

- 通信——提供了通讯对象以及通过底层网络传输数据的功能
- 对象字典——收集了对应用对象、通讯对象和状态机行为有影响的所有相关数据元素
- 应用——组成了与实际过程环境交互相关的设备功能。

对象字典扮演着通信和应用之间接口的角色。CANopen 通信规范共定义了四类通信对象：服务数据对象（SDO）、过程数据对象（PDO）、网络管理对象（NMT）以及特定功能对象。应用对象以及其对应对象字典中的信息则由相关设备子协议(规范）来描述。

1) 对象字典（OD）

对象字典是设备规范中最重要的部分，它实际上就是一组有序的对象集合，包含有设备参数和过程数据等信息。每一个对象都对应有一个 16 位的索引值，对复杂对象如数组和结构还会对应有一个 8 位的子索引项。对象字典的整体结构如表 3.5 所示。[53]

CANopen 通过对象字典 1000-1FFF 区域定义了所有标准通信对象相关的参数，

此部分内容适用于所有的设备类型，所有 CANopen 设备必须予以支持。

位于 6000h 至 9FFFh 之间的标准设备规范区域包含了对某一类设备通用的数据对象，CANopen 的各种设备子协议（包括本文采用的 DSP402）就是使用此区域对象来规范特定的设备参数和设备功能。

表 3.5 对象字典结构

Tab. 3.5 Object Dictionary Structure

索引(hex)	对象	索引(hex)	对象
0000	未使用	00A0-0FFF	保留供将来使用
0001-001F	静态数据类型	1000-1FFF	通信规范区域
0020-003F	复杂数据类型	2000-5FFF	制造商特定规范区域
0040-005F	制造商特定复杂数据类型	6000-9FFF	标准设备规范区域
0060-007F	设备规范特定静态数据类型	A000-BFFF	标准接口规范区域
0080-009F	设备规范特定复杂数据类型	C000-FFFF	保留供将来使用

对象字典的概念使得制造商能够以一种标准的方式自由的进行功能的裁剪，以及自定义特定的设备规范，但是如果决定提供这些功能那么就需要按照预定义的方式来实现。

2) 服务数据对象（SDO）

SDO 提供了用于访问设备对象字典的服务，采用客户端/服务器模型进行通信。SDO 为两个设备之间建立起一个点对点的通讯通道，用于客户端和服务器（对象字典拥有者）通过对象的索引和子索引来传递非实时的对象信息。

- SDO 上传服务

SDO 上传服务用于客户端向服务器请求对象字典中对应对象的内容（即读对象字典）。此服务需要至少一个启动 SDO 上传服务以及可选的 SDO 分段上传服务（如果请求数据长度大于四个字节）。

启动 SDO 上传服务用于客户端通知服务器准备上传数据，如果请求数据长度不超过四个字节那么服务器将直接回复所请求数据（快速上传服务），否则客户端将得到数据的长度信息（正常上传服务），并需要开始分段上传服务请求，分段上传每次可以最多传输七个字节的数据。

- SDO 下载服务

SDO 下载服务用于客户端下载数据到服务器（即写对象字典）。与上传服务类似，此服务需要一个启动 SDO 下载服务以及可选的 SDO 分段下载服务（如果下载数据超

过四个字节）。

启动 SDO 下载服务通知服务器准备开始接收下载数据，如果数据长度不超过四个字节，那么客户端将直接通过此服务发送数据（快速 SDO 下载服务），接收到服务器确认信息后即结束此次服务，否则此次服务将通知服务器数据的长度信息（正常 SDO 下载服务），并在之后开始 SDO 分段下载服务。

- 终止 SDO 服务

此服务用于在 SDO 服务出现错误时终止当前的服务过程并提供出错代码，客户端和服务器都可以发出此请求且无需确认。

3) 过程数据对象（PDO）

PDO 定义了实时数据的标准传输方式，提供了访问应用对象的接口，用于一次传输不超过八个字节的过程数据。

CANopen 定义了两种 PDO 对象，分别是用于输出数据的接收 PDO（RPDO，位于 1600h-17FFh）和输入数据的发送 PDO（TPDO，位于 1A00h-1BFFh）（输入和输出是以从站设备的角度来看的）。PDO 服务采用生产者/消费者模型，拥有 TPDO 的设备称为生产者，拥有 RPDO 的设备则称为消费者。[53]

为了高效的传输过程数据，PDO 不需要协议头即可以提供过程数据传输服务。为了让通信双方对 PDO 的通信数据达成一致，同时规范 PDO 的传输方式，需要对象字典中的 PDO 映射以及 PDO 参数管理两个对象的支持。PDO 映射用于将应用对象映射到 PDO 对象，其中包含了对象的索引、子索引以及长度信息，其实现机制如图 3.22 所示。PDO 参数管理则定义了对应 PDO 对象的身份标识（COB - ID）、同步方式、禁止时间等信息。

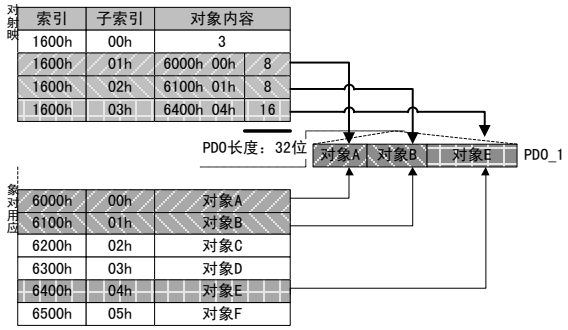

图 3.22 PDO 映射

Fig. 3.22 PDO Mapping

在 CANopen 网络中从站通过过程数据报文 COB-ID 得知此报文中 PDO 数据对应

的 PDO 映射对象，然后通过 PDO 映射找到对应的应用对象及其长度，最终将报文数据映射到具体的应用对象中去。

为了满足不同的应用需求，CANopen 定义同步和异步两种 PDO 传输方式，具体由 PDO 参数对象来指定。其中同步方式需要同步对象（SYNC）的支持。

4) 网络管理对象（NMT）

网络管理对象采用主从模式，主站利用 NMT 提供的服务来完成对从站的初始化、启动、监测、复位或停止等功能。NMT 主要提供了以下服务：

- 模块控制服务

主站利用此服务来控制从站的状态。CANopen 定义的从站通信状态机如图 3.23 所示。[53]

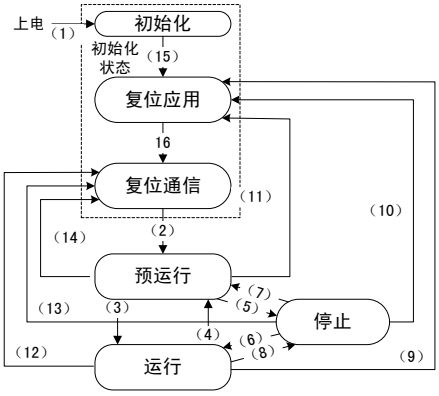

图 3.23 CANopen 通信状态机

Fig. 3.23 CANopen Communication State Machine

图中数字对应的 NMT 服务如表 3.6 所示。从站必须提供启动远程节点服务，其它服务都可以通过本地应用实现。进入预运行状态后从站开始支持 SDO 服务，在进入运行状态后开始启动 PDO 服务。

表 3.6 CANopen 网络管理服务

Tab. 3.6 CANopen NMT Service

(1)	上电之后自动进入初始化状态	(5),(8)	停止远程节点指令（进入停止状态）
(2)	初始化完成，自动进入预运行状态	(9),(10),(11)	复位节点指令（进入复位应用子状态）
(3),(6)	启动远程节点指令（进入运行状态）	(12),(13),(14)	复位通讯指令（进入复位通信子状态）
(4),(7)	进入预运行指令（进入预运行状态）	(15)	初始化结束，自动进入复位应用状态
(16)	复位应用结束，自动进入复位通信状态		

- 错误控制服务

CANopen 定义了两种错误控制方式：监护和心跳，用于监测网络中设备的工作状态，作用类似于看门狗（WatchDog）。

5) 特定功能对象

- 同步对象（SYNC）

同步对象服务采用生产者/消费者模式实现，一般由主站周期性的广播发送，为整个网络提供一个统一的时钟。在 CANopen 网络中，同步就意味着消息的产生和 SYNC 有一个固定的相位关系， PDO 的同步传输方式即以 SYNC 为基础来实现，用于保证网络中设备能以一个固定的时间基更新指令数据和实际数据，原理图 3.24 所示。[53]

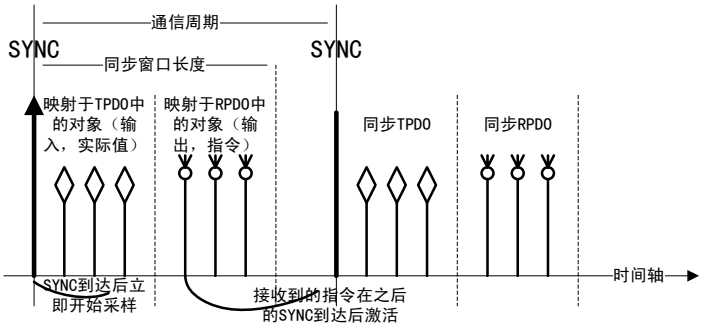

图 3.24 CANopen 同步模型

Fig. 3.24 CANopen Synchronization Model

SYNC 报文的接收时间控制了同步 PDO 与过程环境交互的时间。通常在 SYNC 到达前接收到的带指令数据的同步 RPDO 会在接下来的 SYNC 报文到达时进行驱动。SYNC 的到达也会触发设备进行周期性的采样反馈数据的工作，并将实际数据通过同步 TPDO 尽快发送出去。根据其实际能力，设备可能会配置同步窗口长度，保证在此时间窗口内指令数据一定会到达。

- 紧急对象（EMCY）

紧急对象用于设备报告严重的内部错误。CANopen 定义了丰富的错误代码，设备通过错误寄存器和错误代码来指示当前的错误，每一个错误对应有且只有一个紧急对象的发送。紧急消息结构如表 3.7 所示。

表 3.7 紧急报文结构

Tab. 3.7 Emergency Message Structure

字节	0	1	2	3	4	5	6	7
内容	错误代码		错误寄存器（1001h）	制造商特定错误域				

3.2.3. CoE通信规范

CANopen 是为定义 CAN 总线的高层协议而开发的，而 EtherCAT 的底层实现机制，特别是数据链路层的实现与 CAN 总线的数据链路层有着巨大的差异。将 CANopen 作为 EtherCAT 的应用层，在保证兼容性的同时，为了与 EtherCAT 数据链路层接口，同时充分发挥 EtherCAT 的网络优势，需要对 CANopen 协议进行一番改造和规范，这就形成了所谓的 CoE（CANopen over EtherCAT），其框架如图 3.25 所示。[51, 54]本文将主要分析 CoE 与 CANopen 协议的差异。

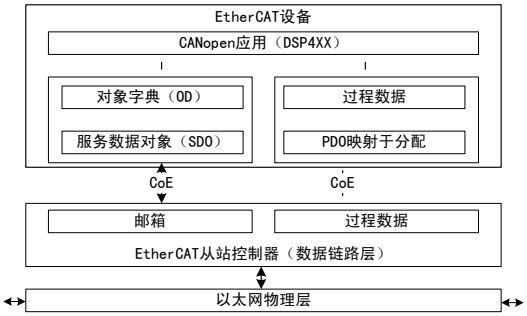

图 3.25 CoE 设备模型

Fig. 3.25 CoE Device Model

1) 通信

在 CANopen 协议中，通信对象标识符（COB-ID）是非常重要的概念，它同时提供了寻址、优先级、指令等多种功能。但这种基于 CAN 总线的实现机制在 EtherCAT 网络中已经完全失去意义了，EtherCAT 有着自己对应的寻址、优先级和指令规范，因此 CoE 中完全没有了与 COB-ID 相关的信息。

2) 状态机

分析比较 EtherCAT 通信状态机（图 3.18）与 CANopen 通信状态机（图 3.23）可以发现它们极为相似，根据对应状态所能提供的服务（表 3.4 和表 3.6），可以基本建立如表 3.8 所示的对应关系。

表 3.8 CANopen 与 EtherCAT 通信状态机映射关系

Tab. 3.8 Mapping of Communication State Machine between EtherCAT and CANopen

EtherCAT 状态	**CANopen 状态**
上电	上电（初始化）
初始化（INIT）	停止
预运行（支持邮箱服务）	预运行（支持 SDO）
安全运行（支持邮箱和过程数据输入服务）	
运行（主从邮箱和过程数据输入输出服务）	运行（支持 SDO 与 PDO）

可以看到它们的主要差别在于：

- EtherCAT 状态机不会在上电后自动进入预运行状态
- CANopen 不支持安全运行状态
- EtherCAT 不支持通过复位指令回到 INIT 状态

经过以上分析可以发现 EtherCAT 状态机可以取代 CANopen 状态机而不会丢失关键的网络管理机能，因此在 CoE 中没有了 CANopen 通信状态机的显式存在。

3) 服务数据对象（SDO）

在 CoE 中以邮箱服务为基础提供的 SDO 服务与 CANopen 中的 SDO 服务完全兼容，但是为了突破原始 SDO 服务 8 字节长度的限制，充分利用以太网报文的资源，CoE 对 SDO 协议进行了增强，使得 SDO 服务能够更加高效方便的传输非实时数据。以 CoE 中 SDO 下载协议为例，其结构如图 3.26 所示。[51]

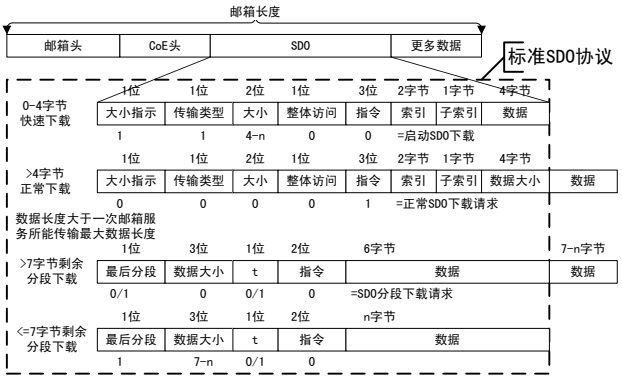

图 3.26 CoE 中 SDO 协议

Fig. 3.26 SDO Protocol in CoE

可以看到在 CoE 中，SDO 协议所能传输数据长度受到以太网报文以及从站同步管理器所管理缓冲区大小的限制。此时 SDO 正常下载协议可以直接包含下载数据，因为 8 字节长度的突破，只要合理配置邮箱缓冲区的大小，使得在实际运行中几乎

肯定不会用到分段传输。同时 CoE 利用标准 SDO 协议中的保留字段增加了一次访问对象字典中复杂对象所有子元素的功能，因此 CoE 实际上可以有效地简化 SDO 服务的实现。

为了更清楚的展示在 EtherCAT 网络中通过 SDO 传输参数的机制，假定主站在 EtherCAT 通信状态机的预运行状态想要获取第二个从站对象字典中设备类型（索引 0x1000）信息，假设这之前主站已经将此从站的设备地址配置为 0x1002，并且用于邮箱通讯的同步管理器 0 和 1 也已经配置成功，起始地址分别为 0x1000 和 0x1100。整个通信流程如下，对其中的 SDO 下载服务，整个过程如图 3.27 所示。

- 主站发送请求报文，其中指令为配置地址写，地址为 0x1002，偏移量为 0x1000
- 从站数据链路层接收并处理报文（如图 3.5），依次判断以太网类型，EtherCAT 报文类型，获取主站指令、地址和偏移量。数据链路层发现主站期望向同步管理器 0 所管理缓冲区写入数据，于是让同步管理器负责处理之后的数据
- 如果同步管理器 0 当前缓冲区状态容许主站写入数据，那么数据将依次写入起始地址为 0x1000 的缓冲区中
- 如果主站所写入数据总长度小于同步管理器 0 的缓冲区长度，主站再次发送指令为配置地址写，地址为 0x1002，偏移量为 0x10FF，长度 1 个字节的报文
- 同步管理器 0 最后一个字节被访问，状态发生变化，产生同步管理器 0 事件，映射到事件标志寄存器中
- 从站应用轮询事件标志寄存器发现有同步管理器 0 事件，于是通过 PDI 接口获取同步管理器 0 所管理缓冲区数据
- 同步管理器 0 用于邮箱通信，于是从站将所获取的数据按照邮箱协议格式（如图 3.17）进行解析，通过邮箱报文头中的类型信息发现其值为 0x04，于是将服务数据按 CoE 协议解析
- 从站应用通过判断 CoE 报文头中的 CoE 服务类型信息，发现其为 0x03，即 SDO 请求，于是将之后服务数据按照 SDO 协议格式进行解析
- 从站应用按照图 3.26 所示格式获取 SDO 服务类型（此处为上传服务）、索引（0x1000）、子索引（0）等信息，并按照此信息访问对象字典
- 从站访问对象字典后发现存在此索引项，并且长度为 4 字节，可以通过 SDO 快速上传服务完成工作，于是按照对应 SDO 回复协议格式封装数据，交 CoE 服务

处理

- CoE 服务按 CoE 协议封装数据，交邮箱服务处理
- 邮箱服务按邮箱协议格式封装数据，并将数据通过 PDI 接口写入同步管理器 1 所管理缓冲区（起始地址 0x1100），引发同步管理器 1 状态变化并产生相关事件
- 主站发现同步管理器 1 所管理缓冲区有数据写入，于是发送指令为配置地址读、地址为 0x1002、偏移量为 0x1100 的报文，获取数据并按照与从站应用相似的方式逐步解析数据，最终获得设备类型信息，完成一次 SDO 服务

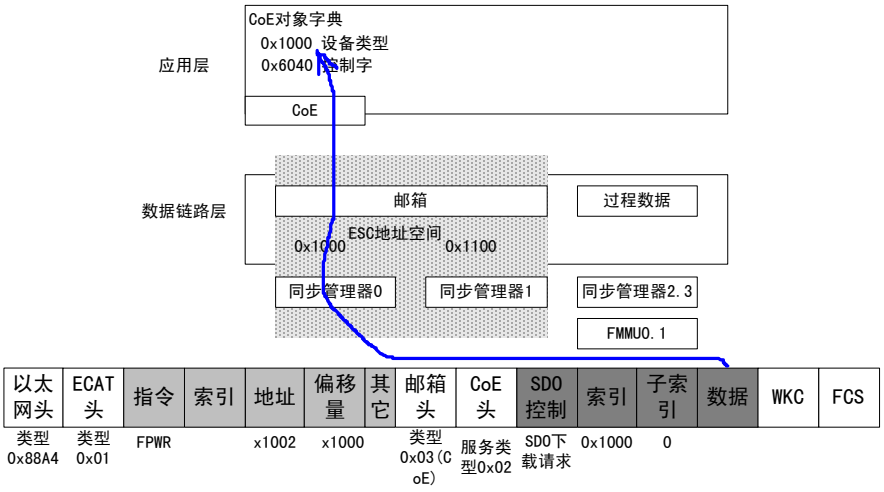

图 3.27 SDO 服务数据处理过程

Fig. 3.27 Data Processing for SDO Service

4) 过程数据对象（PDO）

在 CoE 中 PDO 映射是完全相同的，但在 EtherCAT 网络中没有了 COB-ID 的概念，那么从站如何知道接收到的数据为 PDO 对象，如何知道此数据对应那些 PDO 对象呢？同样这需要和数据链路层管理过程数据交换的同步管理器（缓冲模式）联系起来。

从站会使用专门的同步管理器（一般为同步管理器 2 和 3）来用于过程数据通信，如果从站发现对应同步管理器所管理缓冲区有数据到达，就会将其中内容解释为 PDO 对象数据。但是这些数据到底对应哪些 PDO 对象呢？为了解决此问题，CoE 引入了一个新的对象字典结构——同步管理器 PDO 分配对象（1C10h - 1C2Fh）。每个同步管理器都对应有一个此类对象，它指出每个同步管理器和哪些 PDO 对象相关联（包含有这些 PDO 的索引值），这样从站就知道接收到的数据对应于哪些 PDO 对象了，于是就可以将数据映射到具体的应用对象中去。对 PDO 的发送也是同样的道

理，只是解析工作由主站完成。根据从站的灵活度，主站可以在初始化阶段配置从站对象字典中 PDO 映射和 PDO 同步管理器分配对象，同时根据 PDO 组织配置对应同步管理器和 FMMU。

与 SDO 类似，CoE 中 PDO 协议也不必受 8 个字节长度的限制，只需要考虑以太网报文长度和从站的存储资源。另外还在对象字典中增加了与 PDO 参数管理相关的同步管理器参数对象（1C30h-1C4Fh），其中包含有同步类型、周期时间等信息。

假定从站应用需要周期性的接收指令控制字和目标位置，它们对应于对象字典中的索引值为 0x6040 和 0x607A。此时可以配置同步管理器 2 工作于缓冲模式，起始地址为 0x1800，长度为 6 个字节，同时将 FMMU0 的物理起始地址配置为 0x1800，长度 6 个字节。另外使用 RxPDO0（0x1600）作为输出过程数据映射对象，负责映射 0x6040 与 0x607A，同时将同步管理器 PDO 分配对象 0x1C12 配置为 0x1600，这样就可以开始输出过程数据的传输过程了，过程。

- 主站发送包含输出过程数据的报文，其中指令为逻辑地址写，逻辑起始地址为 FMMU0 的逻辑地址，长度为 6 字节
- 从站数据链路层负责接收和处理报文，在发现指令为逻辑地址写后，将之后的逻辑地址交 FMMU 映射，发现可以有 FMMU0 转换为物理地址（0x1800）
- 从站处理器发现物理地址对应于同步管理器 2 所管理缓冲区，于是将之后的数据写入对应缓冲区中
- 在缓冲区最后一个字节被访问且报文帧校验正确后，引发同步管理器 2 状态变化并产生对应事件标志
- 此时若同步事件到达，则从站应用开始处理过程数据，此时发现同步管理器 2 有数据到达，表明通信过程同步正常，于是通过 PDI 接口获取缓冲区中 6 个字节的过程数据
- 从站应用查询同步管理器 2 所对应同步管理器 PDO 映射对象 0x1C12，发现对应的 PDO 映射对象为 0x1600，于是查询此 PDO 配置
- 从站应用发现 0x1600 负责映射 0x6040 以及 0x607A，并且可以获得对应映射长度，于是将 6 字节数据分配到对应的控制字和目标位置中，完成指令的接收工作

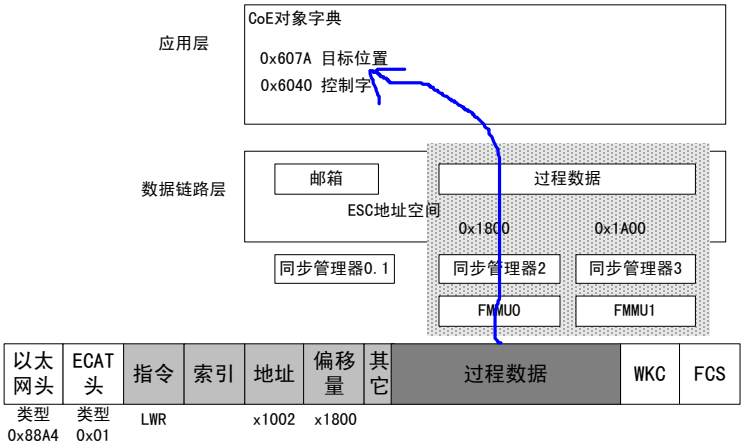

图 3.28 过程数据处理过程

Fig. 3.28 Data Processing for Process Data

5) 同步机制

CANopen 中网络的同步时钟由同步对象（SYNC）对象提供，而在 CoE 中，将直接利用 EtherCAT 的报文或分布时钟来实现，但同步 PDO 的处理与同步时钟之间的基本关系（图 3.19、图 3.20 和图 3.24）没变。由于 EtherCAT 相对于 CAN 总线在底层网络中传输速率和处理效率上的优势，对一般的应用，可以将报文同步与 CANopen 中的 SYNC 同步方式相对应，而分布时钟则可以用于实现更高等级（微秒级）的同步机制。

6) 错误处理

CANopen 中的心跳和监护机制在 CoE 中将由其它的错误处理机制来取代，包括报文中的工作计数器、同步管理器看门狗、过程数据接口看门狗以及 ESC 中丰富的错误检测寄存器。而紧急对象在 CoE 中则与 CANopen 完全兼容，而且与 SDO 类似，突破了八个字节的长度限制。

3.2.4. CANopen驱动规范

CANopen 驱动与运动控制规范（DSP402）为驱动控制器、变频器或步进电机提供标准的设备子协议。此规范是基于 CANopen 通信规范的设备应用协议，并且独立于总线进行描述，本文将以 CoE 为基础将此规范作为的应用协议实现。

总的来说，此规范定义了两部分的内容：[55, 56]

1) 设备控制

定义了一个有限状态机，用于控制设备从启动到运行的各个阶段，设备所处的

状态限制了它当前所能接收和执行的指令。此状态机通过对象字典中的控制字和状态字来维护，一般由主站发出请求，从站执行相关服务后进行回复。

2) 运行模式

定义了包括轨迹位置、速度和转矩模式，插补位置模式，回零模式，速度模式以及周期同步的位置、速度和转矩模式等多种运行模式。规范给出了模式相关的配置参数和过程数据信息，统一了对象字典的组织，定义了各种基本的功能组件，给出了模式特定的基本控制结构，为设备的标准化实现提供了丰富详尽的指导。

规范给设备的具体实现预留了很大的灵活度，对状态机和运行模式的设计与实现将非常依赖于具体应用的需求，因此本文将在设计与实现部分对此规范的细节进行描述。

4. 网络接口设计与实现

本文将集中描述为伺服驱动器提供 EtherCAT 接口而进行的软件设计与实现过程。本软件以 TI 公司 2812 系列 DSP 为平台，采用 C 语言进行开发。设计将按照“基于工业以太网接口的系统整体方案”一章中所展示的结构来进行，软件整体流程图如图 4.1 所示，分为轮询和中断两个部分。

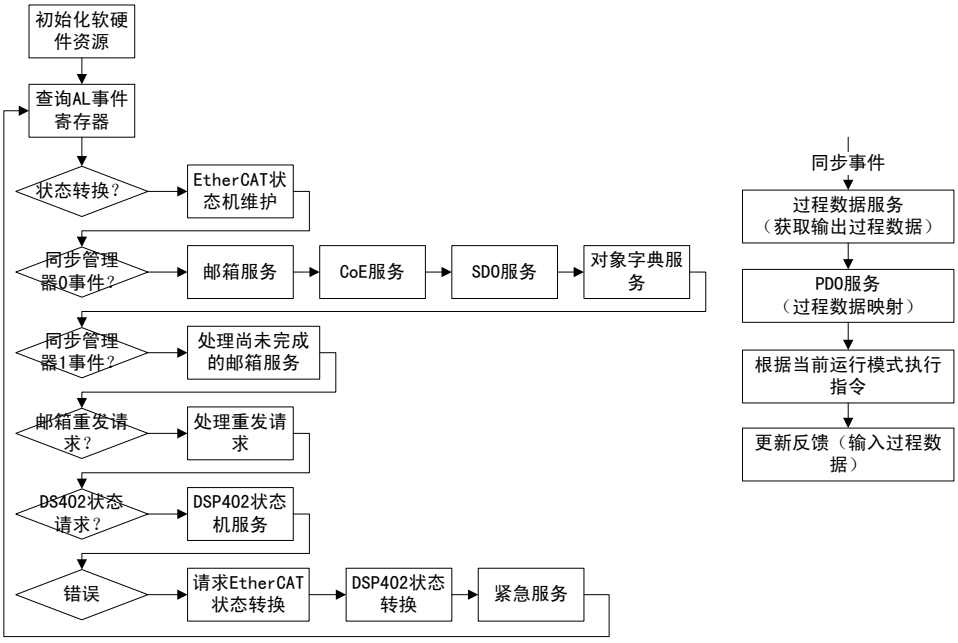

图 4.1 软件整体流程图

Fig. 4.1 Software Flow Chart

4.1. 应用层设计与实现

应用层将负责与网络通信相关的功能，为上层应用提供独立于底层网络的通信接口。本文应用层设计将分为 EtherCAT 和 CoE 两部分的内容，结构如图 2.5 所示。这两部分有着紧密的联系，EtherCAT 通信协议一方面作为 CoE 规范的实现基础，另一方面也负责对 CoE 服务进行管理。应用层服务在报文中的基本封装格式如图 4.2 所示。本文中邮箱服务只支持 CoE 类型服务，而 CoE 服务只支持 SDO 请求与紧急类型的服务。

应用层一方面为应用提供服务，一方面要充分利用数据链路层提供的基本服务和组件，因此设计相对复杂和关键。为了展现一个清晰的设计和实现过程，本文将基本按照如下的范式来进行描述：

- 服务流程：给出主站和从站利用此服务交换信息而需要执行的主要动作，描绘一个完整的服务概况
- 资源：描述此服务所需要的主要软硬件资源
- 接口：给出此服务为外界所提供的主要接口
- 要点：分析设计和实现过程的重难点和技巧

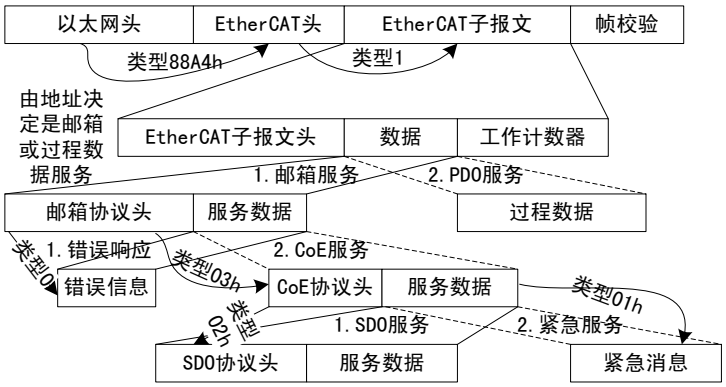

图 4.2 服务数据的封装

Fig. 4.2 Encapsulate of the Service Data

4.1.1. EtherCAT通信协议设计与实现

EtherCAT 通信协议中邮箱和过程数据服务相对独立，而 EtherCAT 通信状态机将需要使用邮箱和过程数据模块提供的服务，同时也用于指导邮箱和过程数据服务接口的设计。

4.1.1.1. 邮箱服务

1) 服务流程

一般邮箱服务都是由主站发起请求，从站进行响应并给出回复，流程如图 4.3 所示。

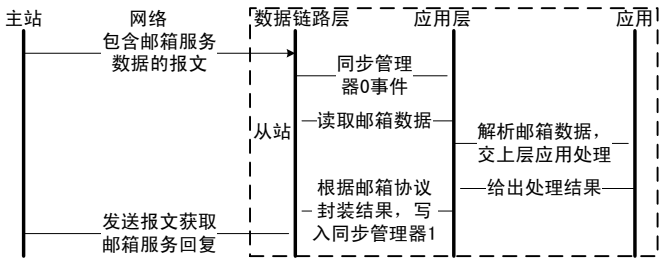

图 4.3 主站请求邮箱服务流程

Fig. 4.3 Flow of the Master Requested Mailbox Service

a) 主站通过报文（一般为配置地址写指令）将邮箱数据发送到同步管理器 0 所管理缓冲区

b) 从站轮询发现有同步管理器 0 事件，于是读取对应缓冲区数据

c) 将数据按照邮箱协议进行解析并对长度、计数、服务类型等进行校验，如果出错，给出错误回复数据并进入 e，否则按照服务类型交由上层服务（本文为 CoE 服务）处理

d) 上层服务处理完毕，返回服务回复数据

e) 从站将回复数据封装于邮箱协议中，写入同步管理器 1 所管理缓冲区

f) 主站发现有同步管理器 1 事件，发送报文读取缓冲区数据，得到邮箱请求回复，服务完成

从站也可以主动发起邮箱服务请求，本文只用于紧急消息的发送，流程如图 4.4 所示。

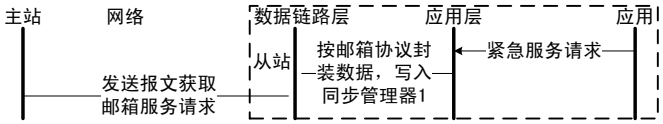

图 4.4 从站请求邮箱服务流程

Fig. 4.4 Flow of the Slave Requested Mailbox Service

a) 从站应用请求邮箱服务

b) 从站将上层服务数据封装于邮箱协议，发送至同步管理器 1 所管理缓冲区

c) 主站发现有同步管理器 1 事件，发送报文读取缓冲区数据，服务结束

2) 资源

为提供邮箱服务，从站需要掌握的资源包括：ESC 中 AL 事件寄存器信息（主要为同步管理器事件）；用于邮箱通信的同步管理器 0 和 1 的配置信息（起始地址、长度等）；两个本地缓冲区，使得邮箱服务和 ESC 缓冲区解耦，用于实现邮箱重发机制等；

3) 接口

本文为邮箱服务设计的主要接口有：

Mbx_CheckSmCfgForMbx：检测同步管理器 0 和 1 的配置信息是否符合要求（主要包括起始地址、长度、事件映射、激活等信息）

Mbx_StartMbxHandler：开启邮箱服务，主要是获取邮箱缓冲区地址和长度，通

过 PDI 使能同步管理器运行，并更新标志位、指针等软件资源

Mbx_StopMbxHandler：关闭邮箱服务，主要是通过 PDI 停止同步管理器 0 和 1 运行并初始化软件资源

Mbx_Main：查询同步管理器 0 和 1 以及邮箱重发请求事件并通过内部接口进行响应

Mbx_Init：初始化邮箱服务相关软硬件资源

4) 要点

邮箱服务一般用于传输非周期的参数变量数据，对实时性要求不高，应采用轮询的方式进行实现。同时为了方便的处理邮箱服务请求和回复数据，同时实现邮箱重发请求，本文在内部维护了两个邮箱缓冲区，其基本的状态转换过程如图 4.5 所示。图中 WR、RD 和 RP 分别表示写、读和重发邮箱指针。

主站的邮箱请求总是存入 WR 所指向区域，从站当前的回复总是由 RD 指出，如果主站有邮箱重发请求，那么有 RP 所指向的数据将被再次作为回复发送。

为了让软件设计结构更清晰，可以将邮箱服务的实现划分为函数模块供外部接口来使用，主要包括：获取同步管理器信息、检测同步管理器、获取邮箱数据、写邮箱事件响应、读邮箱事件响应和发送邮箱数据等接口。

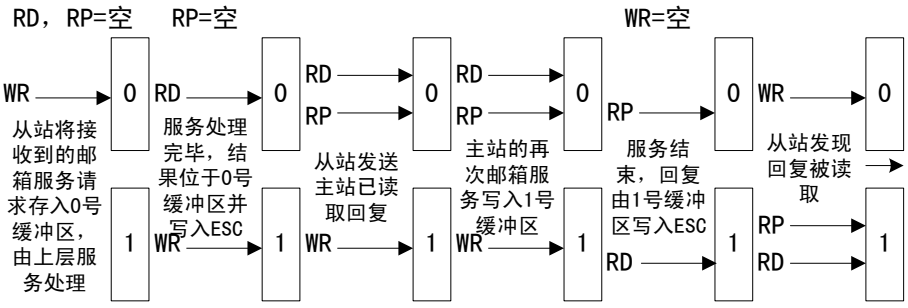

图 4.5 本地邮箱缓冲区维护

Fig. 4.5 Maintenance of the Local Mailbox Buffer

4.1.1.2. 过程数据服务

1) 服务流程

本文总是以同步的方式处理过程数据，其流程如图 4.6 所示。

a) 同步事件到达，从站从同步管理器 2 所管理缓冲区获取主站之前所发送的输出过程数据，并交由上层服务（PDO 服务）处理

b) 从站请求上层服务（PDO 服务）进行输入过程数据采样，并将结果送入同步管

理器 3 所管理的缓冲区中

c) 在下一个同步周期主站获取从站的输入过程数据（同时有新的指令到达）

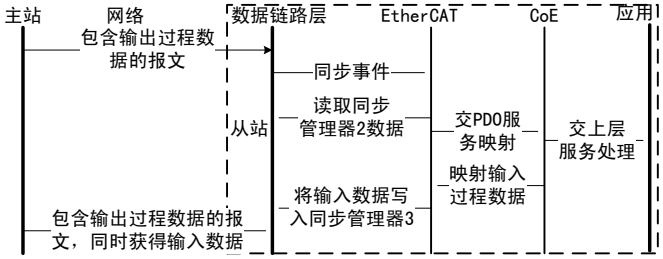

图 4.6 过程数据服务流程

Fig. 4.6 Flow of the Process Data Service

2) 资源

主要包括：同步管理器 2 和 3 的配置信息（缓冲区地址、长度）；输入和输出过程数据大小；AL 事件请求和屏蔽寄存器（同步事件和同步管理器事件）；

3) 接口

Pd_CheckSmCfgForPd：检测用于维护过程数据的同步管理器配置信息

Pd_StartInputHandler：开启输入过程数据服务，主要包括获取同步管理器 2 地址和长度信息，获取同步类型信息，配置 AL 中断屏蔽寄存器并使能同步管理器 2 和 3 工作，同时更新软件标志位。

Pd_StartOutputHandler：开启输出过程数据服务，开始更新输出过程数据

Pd_StopInputHandler：停止输入过程数据服务，通过 PDI 停止同步管理器 2 和 3 的工作从而停止过程数据更新，初始化相关软件资源

Pd_StopOutputHander：停止输出过程数据服务，将输出数据置于安全的状态

Pd_UpdateProData：同步事件到达后从同步管理器 2 缓冲区获取输出数据交由上层服务（PDO 服务）处理并通过上层服务获取输入过程数据并写入同步管理器 3 缓冲区

Pd_Init：初始化过程数据服务相关软硬件资源

4) 要点

过程数据更新属于实时性要求很高的服务，需要在中断服务程序中维护。在开启输入过程数据服务前从站应该先更新输入过程数据，同时在开启输出过程数据服务前需要判断主站是否已经更新了输出数据。

为了保证实时安全的过程数据交换，在运行状态需要对过程数据的同步性能进行监测，同时提供看门狗来监视过程数据的更新过程，如果有同步错误或看门狗超时，从站应该主动请求进入安全运行状态，将输出置于安全的状态。

为了更好的模块化设计，本文将过程数据服务独立出来，只负责过程数据的更新和维护，至于数据的具体意义和处理方式将交由 CoE 的 PDO 服务来负责。

4.1.1.3. EtherCAT通信状态机

1) 服务流程

通信状态转换请求一般由主站发起，从站进行响应，服务流程如图 4.7 所示。

a) 主站通过写 AL 控制寄存器请求状态转换

b) 从站查询 AL 事件寄存器发现有状态转换请求事件，于是读取 AL 控制寄存器内容，并根据当前自身状态，按照图 3.18 所示状态机进行转换

c) 如果状态转换成功，从站将所请求状态写入 AL 状态寄存器，将状态错误代码寄存器清零，否则从站将发生错误的状态写入 AL 状态寄存器并设置错误标识，同时将错误代码写入 AL 状态代码寄存器

d) 主站查询 AL 状态寄存器获得状态切换结果，如果转换出错，在请求其它状态转换之前需要发送错误响应报文来清除当前错误

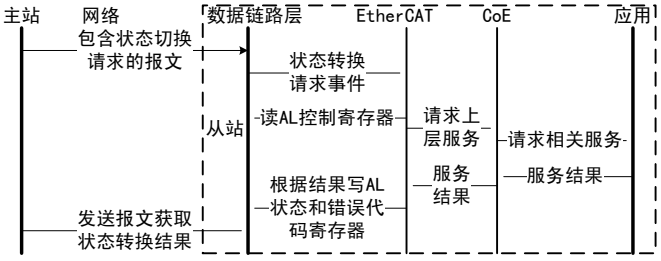

图 4.7 主站请求 EtherCAT 通信状态机服务流程

Fig. 4.7 Flow of the Master Requested EtherCAT Communication State Machine Service

当检测到严重的本地错误发生时，从站也会主动请求状态转换，流程如图 4.8 所示。

a) 从站将所请求状态写入 AL 状态寄存器，设置错误标识，提供错误代码

b) 主站查询发现有状态变化，通过写 AL 控制寄存器来响应从站状态请求

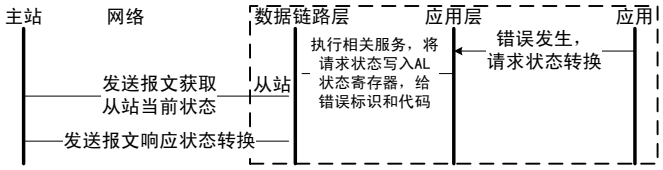

图 4.8 从站请求 EtherCAT 通信状态机服务流程

Fig. 4.8 Flow of the Slave Requested EtherCAT Communication State Machine Service

2) 资源

AL 事件寄存器，AL 状态寄存器和状态代码寄存器以及 AL 控制寄存器

3) 接口

Sm_StateReq：响应主站状态转换请求

Sm_StateChange：负责从站主动请求的状态转换

Sm_Init：初始化状态机相关软硬件资源

4) 要点

EtherCAT 协议规定的通信状态转换过程中需要完成的服务相对较多，状态机设计的关键是组织好状态切换过程中从站应该执行的操作，为此，本文将这些操作分为两部分：检测和服务。检测部分包括同步管理器配置的检测、状态转换请求的检测等，服务包括邮箱和过程数据输入输出服务的开启和关闭，只有检测成功后才会执行相关服务，状态机切换过程如表 4.1 所示，通过使用邮箱和过程数据服务所提供的接口来实现。

表 4.1 状态转换动作

Tab. 4.1 Actions for State Changing

IP	检测同步管理器 0 和 1 配置，开启邮箱服务		
PS	检测同步管理器 2 和 3 的配置，开启过程数据输入服务		
SO	开启过程数据输出服务	OS	停止过程数据输出服务
OP	停止过程数据输入输出服务	OI	停止过程数据和邮箱服务
SP	停止过程数据输入服务	SI	停止过程数据输入和邮箱服务
PI	停止邮箱服务		

为了便捷与安全性，如果状态转换错误，从站必须要在主站响应此错误后才能进行状态转换过程，但是有一个例外是如果主站请求进入 INIT 状态，那么从站总是

应该给予响应，进入初始状态并清除错误代码。

4.1.2. CoE通信规范设计与实现

4.1.2.1. 对象字典

1) 服务流程

本文中提供对象字典服务专门为 SDO 实现服务，所提供的接口非常简单，所以服务过程也非常清楚：

a) 上层应用请求访问对象字典

b) 对象字典服务根据请求所提供的索引、子索引、访问类型等方式，查找对象字典中对应对象并执行相关操作（读或写）

c) 将处理结果提交给上层服务处理

2) 接口

Obj_ReadObject：通过索引和子索引项获取对象字典内容

Obj_WriteObject：通过索引和子索引将数据写入对象字典

3) 要点

对象字典是 CoE 的核心，包含有通信和应用相关的所有参数，本文将此部分功能独立出来，主要是为了保证 SDO 服务可以独立于对象字典的组织来访问对象字典的内容。

为了保证对象字典中数据的一致性，本文在实现中对读写操作进行了较为严格的条件校验，包括数据的长度、访问权限、子索引范围等都进行了检测。

为了高效的存储和维护对象字典的内容，本文按照如下的结构体定义对象字典中每一个对象的内容，然后以有序数组（按索引排序）的形式组织整个对象字典。

```
//每一个子索引项的描述
typedef struct entryDescription_T
{
    Uint16    bitLength       : 8; //!<位长
    Uint16    objectAccess    : 8; //!<访问权限
}entryDescription_t;
//每一个索引项的描述
typedef struct object_T
{
    const entryDescription_t * const pEntryDesc; //!<子项目描述
    void * const pVarPtr;                        //!<项目变量
    const Uint16 index;                          //!<索引项
```

```
    Uint16 subindex0            :8;         //!<子索引数目
    Uint16 nonVolatileOffset    :8;         //!<在 nonVolatile 空间中的偏移量
    const Uint16 maxSubindex    :8;         //!<最大子索引数目
    const Uint16 objectCode     :8;         //!<object code
}object_t;
```

4.1.2.2. PDO服务

1) 服务流程

PDO 服务以过程数据服务为基础来实现，本文设计的 PDO 服务流程为：

a) 对由过程数据服务提供的输出过程数据，首先按照 PDO 同步管理器配置的当前内容获取有效地 PDO 映射对象，然后按照 RPDO 映射的配置将数据映射到具体的应用对象中，并交由上层应用处理

b) 通过 PDO 同步管理器配置和对应 TPDO 映射将应用对象组织为输入过程数据并交由过程数据服务处理

2) 资源

对象字典中 PDO 映射和 PDO 同步管理器分配，同步管理器配置信息

3) 接口

CoEApp_InputMapping：将具体应用对象按照 RPDO 映射的配置映射到输入过程数据

CoEApp_OutputMapping：将接收到的输出过程数据映射到具体应用对象

CoEApp_GenerateMapping：检测 PDO 映射和分配，同时获得输入和输出过程数据长度

CoEAPP_ResetOutput：重置过程数据输出相关信息，将输出置于安全状态

4) 要点

PDO 服务的主要功能是完成实际过程数据的映射工作，至于如何处理这些数据将由上层应用负责。在 CoE 中，对 PDO 的组织有不同等级的灵活度，应根据实际应用需求进行选择，以避免不必要的软件复杂度。本文将不支持对 PDO 映射对象的直接修改，但是支持通过 PDO 同步管理器分配来选择当前有效地 PDO 映射，从而支持不同的应用需求。

4.1.2.3. SDO服务

1) 服务流程

SDO 服务总是由主站发起，并通过 EtherCAT 邮箱服务来传递。

a) 从站接收到邮箱服务传来的 SDO 服务请求，将数据按照 SDO 协议进行解析，判断服务类型

b) 对 SDO 下载请求，将下载数据和长度信息交给对象字典写服务接口处理

c) 对 SDO 上传服务，通过对象字典读服务接口获取对象的数据和长度信息

d) 将对象字典服务返回数据按照 SDO 回复协议进行封装，交邮箱服务处理

2) 接口

Sdo_SdoIndicate：按照 SDO 服务协议解析数据，通过内部接口执行相应服务并给出结果，交邮箱服务处理

Sdo_Init：初始化 sdo 服务相关软件资源

3) 要点

SDO 服务的实现相对很直接，按照协议格式进行解析和处理即可。CoE 定义了多种 SDO 服务类型，但是根据对象字典中数据的特性，以及 CoE 特有的突破了八个字节长度限制的协议，可以发现 SDO 快速和正常上传和下载服务已经完全能够满足本文设计要求，为减少不必要的复杂度，完全可以不用实现分段的 SDO 服务。

因为已经将对象字典的访问进行了封装，因此 SDO 服务实现可以相对简单，在内部可以将 SDO 上传和下载服务的快速和正常模式独立实现为函数模块，供外部接口在 SDO 协议解析过程中进行调用。

4.1.2.4. 紧急服务

紧急服务用于从站发现严重的内部错误时发生紧急消息通知主站，此服务以邮箱服务为基础实现。

1) 服务流程

a) 应用发现错误，请求紧急服务，根据需要更新错误寄存器并提供错误代码

b) 紧急服务将错误消息按照紧急协议格式进行封装，交邮箱服务处理

c) 主站通过邮箱服务获取紧急消息

2) 资源

紧急消息队列，对象字典

3) 接口

Emcy_SendEmergency：用于请求发送紧急消息

Emcy_Continue：发送紧急消息队列中尚未发送的紧急消息

4) 要点

本文在内部将维护一个紧急消息队列，以免紧急消息的丢失。关键在于应用需要根据错误类型更新对象字典中的错误寄存器并提供有意义的错误代码，本文中紧急消息主要用于传输过程数据看门狗超时、同步管理器检测出错和伺服驱动器相关错误信息。

4.2. 应用设计与实现

应用按照 CANopen 驱动与运动控制规范（DSP402）进行设计实现，主要完成与伺服驱动器运动控制相关功能，包括对设备状态的控制以及对位置和速度的控制，整体结构如图 2.6。

4.2.1. 设备状态机

设备状态机定义了从站设备的控制时序，对应每一个状态限制了设备当前能够接受的指令和执行的动作，其结构如图 4.9 所示。

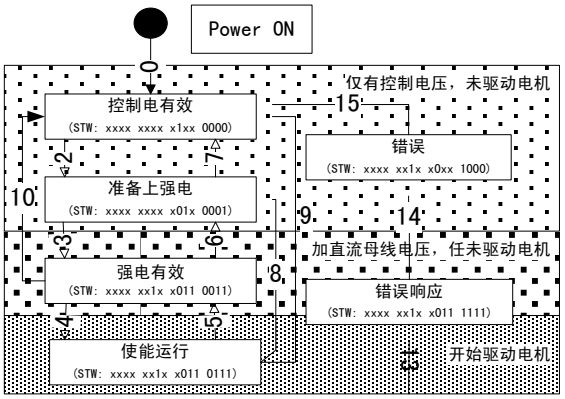

图 4.9 设备状态机

Fig. 4.9 Device State Machine

设备状态机由对象字典中的控制字和状态字进行维护，采用主从结构，主站通过写控制字请求状态转换，从站将实际状态写入状态字供主站查询状态转换结果。图中数字代表的动作如表 4.2 所示。

表 4.2 设备控制状态转换动作

Tab. 4.2 Actions according to the device control state machine of the servo driver

0,2,7	空操作	3	上强电
4	开始驱动电机	5	停止驱动电机
6,10	断强电	8,9	停止驱动电机，断强电
13	驱动器内部出错，设置错误标志，进入错误响应状态	14	错误响应完毕自动转换
15	主站发送清除错误请求，如果本地无错误则退出错误状态否则保持原状态		

在设备控制状态机的软件设计和实现中，应根据实际需求确定状态机中所需状态，这之后从站只需查询控制字检测是否有状态转换请求，按照表 4.2 执行相关动作，最后将转换后状态写入状态字即可。需要注意的是设备状态机应该在 EtherCAT 通信状态机进入运行状态之后才允许进行操作，这也就意味着 EtherCAT 通讯状态机在退出运行状态前也应该保证设备状态机处于初始状态。同时从站在响应状态转换时要在所需动作成功完成之后才能进入所请求的状态，否则应保持原状态不变。

4.2.2. 运行模式设计与实现

本文支持周期同步和轨迹两种模式，它们的主要差异在于所构建系统的控制结构。对周期同步模式，主站将作为整个闭环控制系统的一个环节，如对周期同步速度模式，意味着位置闭环控制位于主站，而对于轨迹速度模式则意味着系统将不会对位置进行控制。运行模式实现关键在于确定对应的指令和反馈，同时明确其控制结构。为支持多种运行模式，本文设计了三组内容固定的 PDO 映射对象，如表 4.3 所示，通过 PDO 同步管理器分配对象进行选择。运行模式的实现依赖于 PDO 服务，需要同步实时的处理。[57]

表 4.3 PDO 映射组织

Tab. 4.3 PDO Mapping Structure

接收 PDO（RPDO）		**发送 PDO（TPDO）**	
1600h	控制字（6040h），目标位置（607Ah）	1A00h	状态字（6041h），实际位置（6064h）
1601h	控制字（6040h），目标速度（60FFh）	1A01h	状态字（6041h），实际位置（6064h）
1602h	控制字（6040h），运行模式（6060h），目标位置（607Ah），轨迹速度（6081h），轨迹加速度（6083h），目标速度（60FFh）	1A02h	状态字（6041h），运行模式显示（6061h），实际位置（6064h），实际速度（606Ch）

4.2.2.1. 周期同步位置模式

周期同步位置模式总体控制框图如图 4.10 所示[56]。此模式下轨迹规划的功能将由上位机完成，主站在每个通讯周期都需要发送有效的目标位置值，从站根据插补

周期和自身位置环采样周期进行位置插补，得到实时位置指令用于位置控制。此模式充分利用了 EtherCAT 高速的通信能力，结合 EtherCAT 良好的同步性能，能够方便的用于对多轴的联动控制。此模式下从站需要关注的对象字典中的应用参数主要有：

int32 i32TargetPosition;//目标位置（绝对位置，单位 inc）

int32 i32PosActualVal;//位置实际值（绝对位置，单位 inc）

IpTimePeriod_t IpTimePeriod;//插补周期，结构如下所示

```
typedef struct IpTimePeriod_T
{
    Uint16 timePeriod; //!<时间周期
    int16 timeIndex; //!<单位索引值
}IpTimePeriod_t;
```

同时主站需要设置位置轨迹规划相关参数如轨迹速度和加速度等。此模式下本文实现中将主站发送的目标位置指令视为绝对位置值，单位是编码器脉冲增量，从站在检测设备状态机的当前状态后，依据插补周期以及运行模式等参数，将绝对位置转化为位置增量并进行位置插补，得到位置环指令值，最终完成位置控制。

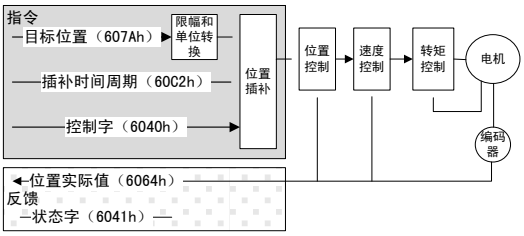

图 4.10 周期同步位置模式控制结构框图

Fig. 4.10 Cyclic Synchronous Position Mode Control Structure

4.2.2.2. 周期同步速度模式

周期同步速度模式的控制结构框图如图 4.11 所示[56]。此模式下上位机将根据目标位置和轨迹加速度等信息完成位置轨迹规划的任务，同时结合从站反馈的位置实际值来进行位置闭环控制，并输出目标速度指令供从站做速度控制。从站使用到的主要参数包括：

int32 i32TargetVel;//目标速度（0.1rpm）

int32 i32PosActualVal;//位置实际值（绝对位置，inc）

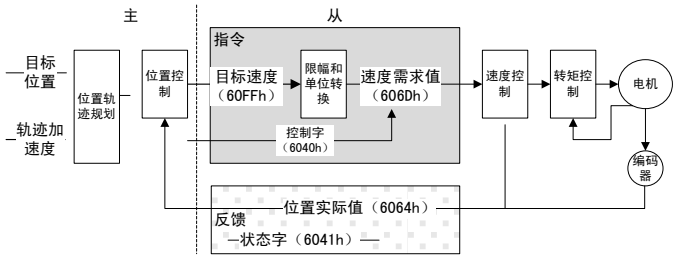

图 4.11 周期同步速度模式控制结构框图

Fig. 4.11 Cyclic Synchronous Velocity mode Control Structure

4.2.2.3. 轨迹位置模式

轨迹位置模式的大致控制框图如图 4.12 所示[55, 56]。在每一次的定位过程中，主站实际只需要发送一次目标位置以及轨迹速度和加速度的值，同时设定控制字中的相应使能位。从站自身根据这些参数进行轨迹的规划，完成位置控制。这个过程中主站和从站之间需要设置和读取控制字以及状态字中的相关标志位来进行握手，以便知道双方当前状态，比如是否有有效目标位置指令到达、指令是否已读取、目标位置是否已到达等。本文不支持对目标位置的动态改变，如果在轨迹规划过程中接收到新的目标位置，将存储新的指令，待此次目标位置到达后再执行对应指令。此模式一般用于点对点的位置控制，相关参数的定义如下：

int32 i32TargetPosition;//目标位置（单位 inc）

Uint32 u32ProfileVel;//轨迹速度（单位 inc/st（采样周期），Q16 格式）

Uint32 u32ProfileAcc;//轨迹加速度（单位 inc/st^2，Q16 格式）

int32 i32PosActualVal;//位置实际值（绝对位置，inc）

int16 i16MotionProfileType;//轨迹规划模式（梯形或 S 形加减速）

目标位置指令根据控制字中的配置可以解释为绝对位置或相对于当前位置的增量，从站每个采样周期都根据主站配置的参数进行位置的轨迹规划。

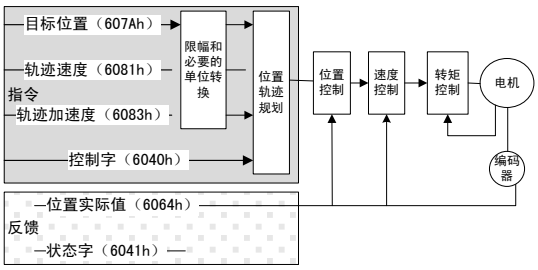

图 4.12 轨迹位置模式控制结构框图

Fig. 4.12 Profile Position Mode Control Structure

4.2.2.4. 轨迹速度模式

轨迹速度模式控制结构框图如图 4.13 所示。[55, 56]此模式与轨迹位置模式下主站和从站间的控制流程完全相同，不同的是主站直接发送目标速度指令，而从站则进行速度轨迹规划，没有位置控制单元。相关参数有：

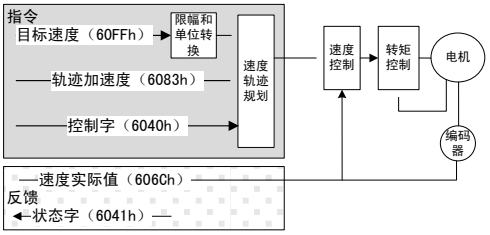

图 4.13 轨迹速度模式控制结构框图

Fig. 4.13 Profile Velocity Mode Control Structure

int32 i32TargetVel;//目标速度（0.1rpm）

Uint32 u32ProfileAcc;//轨迹加速度（0.1rpm/st，Q16）

本文将简单的根据当前速度以及轨迹加速度进行速度的线性规划，得到速度指令用于速度控制。

4.2.2.5. 模式切换

DSP402 在对象字典中定义了运行模式和运行模式显示对象，用于对设备运行模式的控制。主站通过改变运行模式对象来请求模式切换，从站查询到状态转换请求后执行相关动作并将当前实际模式写入运行模式显示对象作为回复。实现可以根据需求提供不同的灵活度，模式的切换可以规定在特定的设备状态下才能够执行，甚至可以规定只能由本地切换，也可以在运行过程中动态的进行切换。因为运行模式的请求可以通过 SDO 或者 PDO 服务来改变，为了保证运行模式与对应指令能够始终保持一致，如果决定支持动态的运行模式切换，应该通过 PDO 服务来请求模式切换，此时需要主站通过 PDO 同步的更新所请求模式的指令，同时在从站确认状态转换成功之前需要继续维持旧模式的指令，使其处于有效状态。另外，本文支持 DSP402 定义的远程和本地控制模式切换，如果 EtherCAT 通信状态机进入运行状态那么设备将进入远程控制模式，并通过设置状态字中的标志位来通知主站。[55, 56]

5. 实验结果

为利用本文开发的伺服驱动器 EtherCAT 接口组建网络化的控制系统，需要相应的准备工作。

首先本文的开发工作集中于从站，而控制系统需要主站来完成对整个网络的管理和维护，以及实时的指令发送等工作。对于本文中的伺服驱动系统而言，主站的功能相当繁重，除了对等的 EtherCAT 协议以及 CoE 协议的实现，还需要完成数控系统的相关功能，以支持本文所描述的多种运行模式，最为重要的是需要有实时性的保障。鉴于开发的周期和难度，本文将直接选用 windows 环境下倍福公司的 TwinCAT 软件来实现主站相关工作。

其次为了描述从站设备，需要为 TwinCAT 准备 xml 格式的从站描述文件，其中主要提供了同步管理器的配置、PDO 的映射和分配以及整个对象字典的组织等信息，用于主站获取从站的配置和控制参数。主站对从站的状态机的维护、指令和反馈的获取都将依赖于此配置文件。

最后还需要适当的配置 EtherCAT 从站控制器（ET1100）的 EEPROM，使得 ESC 能够和 DSP 接口相匹配。其中最重要的是前十六个字节的内容，主要包含有对 ESC 过程数据接口（PDI）类型的配置，对信号驱动类型的配置以及对分布时钟的配置。

最终，本文以普通PC机作为主站，在windows®环境下，以倍福（Beckhoff®）公司的TwinCAT®软件作为系统的上位机控制软件，以自行设计的带EtherCAT接口的伺服驱动器为从站组建控制系统进行了实验，系统整体结构如图 5.1 所示。

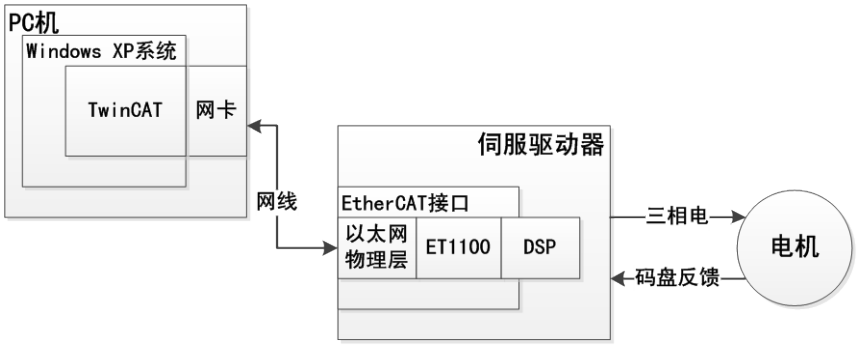

图 5.1 控制系统结构框图

Fig. 5.1 Structure of the control system

控制系统实际参数如下：通讯周期为 3 毫秒，设定电机 1 转对应 40 毫米，最大

速度限定为 1200mm/s，加速度设定为 1200mm/s^2，加速度变化设定为 2400mm/s^3，轨迹规划采用 7 段 S 型加减速模式。永磁同步电机的额定转速 1200r/min，最大转速 2000r/min，反馈编码器分辨率为 19 位。伺服驱动器位置环采样周期 300 微秒。对轨迹位置和速度模式需要在 TwinCAT 环境下开发相应 PLC 控制程序。下面各图中横坐标单位均为秒。本文实验平台如图 5.2 所示。

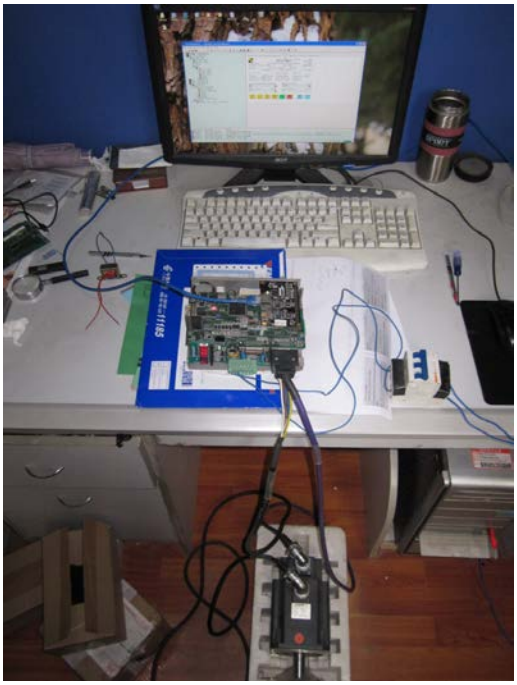

图 5.2 实际实验平台

Fig. 5.2 Experimental platform

图 5.3 为周期同步位置模式下不同目标位置和目标速度下运行结果，主站在每个通讯周期更新其规划出的位置指令值，从站实时接收该指令完成对位置的控制。实验中主站发送绝对位置值，从站将其转换为每个插补周期的位置增量进行位置插补。

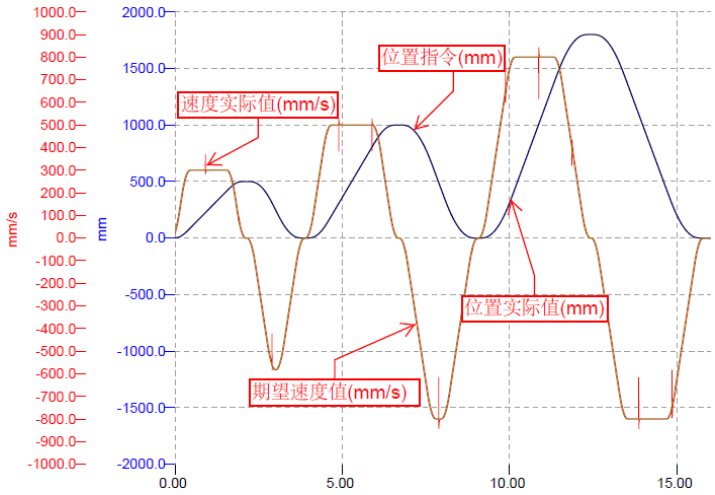

图 5.3 周期同步模式运行结果

Fig. 5.3 Results of the Cyclic Synchronous Position Mode

图 5.4 为周期同步速度模式下运行结果，在此模式下将有主站完成轨迹规划和位置控制工作。主站利用当前规划出的位置值与从站反馈的位置实际值进行闭环控制，得到速度指令值，从站直接使用主站发送的速度指令完成速度控制。为了得到较好的控制效果，需要对主站位置环的控制参数进行调节。

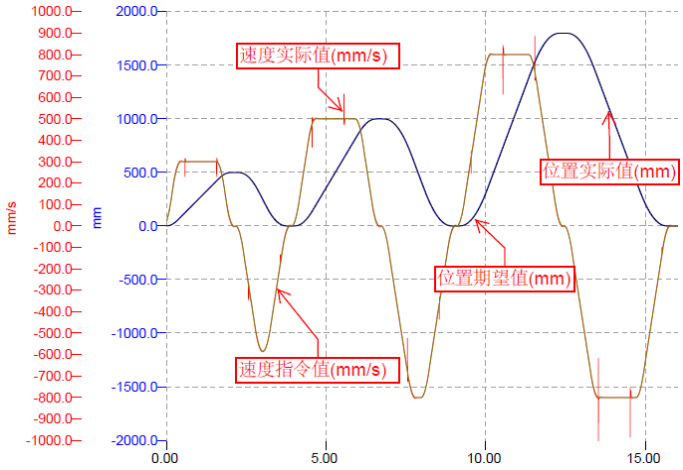

图 5.4 周期同步速度模式运行结果

Fig. 5.4 Results of the Cyclic Synchronous Velocity Mode

上述结果中，TwinCAT 根据反馈的实际位置计算反馈速度。在实际运行中，受 Windows 操作系统和主站硬件的影响，通过上位机的以太网帧捕获工具发现主站系统会接近周期性的丢失 TwinCAT 的数据发送任务，导致从站丢失位置指令，主站丢失位置反馈信息的情况，因此实际速度结果存在明显跳变。

图 5.5 为轨迹位置模式下，分别以不同的轨迹速度、加速度和目标位置进行轨迹规划的运行结果，采用 S 型加减速模式，可以看出位置响应非常平滑。

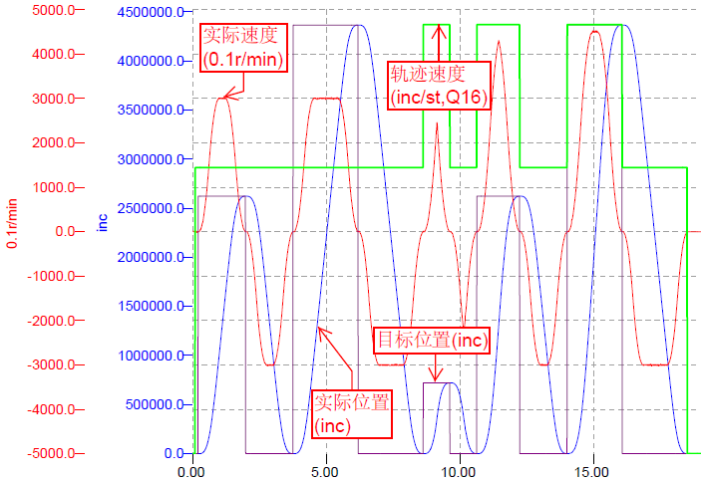

图 5.5 轨迹位置模式运行结果

Fig. 5.5 Results of the Profile Position Mode

图 5.6 为轨迹速度模式下，分别以不同的目标速度和轨迹加速度进行规划的运行结果，系统能很好的实现速度的跟踪控制。

图 5.6 轨迹速度模式运行结果

Fig. 5.6 Results of the Profile Velocity Mode

实验过程中，可以体验到 EtherCAT 在参数的管理、错误的检查等方面非常令人满意，同时网络的组建非常的简单高效，其从站自带的分布时钟机制所带来的同步性能很优越。通过分析实验数据可以发现 EtherCAT 非常适合于伺服驱动系统，能够非常高效的实现多种对位置和速度的控制模式。

因为上位机软硬件环境等不确定因素的影响，实验中一直无法保证主站时钟的准确性，从而导致整个实验阶段留有不少遗憾，无法更进一步的挖掘 EtherCAT 的性能潜力。

6. 总结与展望

工业以太网技术在国外已经有了较为广泛的应用而且展现了其优势和活力，而在国内的研究和应用则相对落后，因此本文对伺服驱动系统的工业以太网接口技术进行了研究，期待进行一些有益的探索。

本文对伺服系统工业以太网接口的设计进行了深入研究，初步构建了伺服驱动器工业以太网接口的软硬件环境，基本达成了预期的构想，完成的主要工作和取得的关键成果有：

1) 深入研究了 EtherCAT 通信协议以及 CANopen 通信和驱动规范，解析了其关键技术，研究了其工作原理
2) 设计软件实现了 EtherCAT 通信协议栈
3) 结合 CANopen 协议进一步规范应用层设计，软件实现了 CoE 通信协议栈，根据应用环境对协议进行了裁剪
4) 引入 CANopen 驱动与运动控制协议规范应用设计，软件实现了协议栈，支持多种运行模式
5) 组建了网络控制系统，进行了实际的测试。

回顾分析、设计、实现到组建网络的整个过程，可以发现 EtherCAT 协议确实有着灵活度高、开放性好等特点。结合对整个网络的实际控制以及最终结果，可以体验到基于工业以太网的伺服驱动系统有着组网简单，控制方式灵活多样，参数管理方便高效，同步和实时特性好等诸多自身优势。

本文还需要进一步开展的工作有：

1) 研究 EtherCAT 网络下多轴间协调同步的控制，测试其实际性能
2) 实际测试 EtherCAT 网络与其他现场总线或工业以太网技术应用于伺服驱动系统中的优势和不足
3) 对运行模式部分功能的扩充和完善
4) 在 Linux（或 windows）环境下开发主站控制软件

致　谢

随着论文写作的完成，我的研究生生活也要画上句号了。回顾这两年多的经历，有收获，有成长，也有遗憾，最重要的是自己不一样了。想要感谢的人有很多。

首先要感谢我的导师李叶松教授。导师在学习和生活上都给了我很多的关心，带我一步步走出了当初的迷茫，明确了自己的方向，教会了我很多做人做事的方法，使我养成了认真负责的科研态度，懂得了反思和总结的重要性，有了一个乐观积极的心态。很庆幸能遇上这样一位做事低调、自信乐观、以身作则的老师，在研究生期间为我今后的发展打下了良好的基础，老师的风范会是我永远的学习榜样。

其次要感谢自动化所优秀的同学们，从他们身上我看到了自身的不足，找到了学习的动力和方向。在生活中他们给予了我无数的帮助，让我有了太多美好的回忆。

同样要感谢自动化所沈安文、程善美、赵金、周永鹏、何顶新、李曦等老师在学习和生活中给予我的指导和帮助，感谢我的母校为我带来了这么好的学习和生活环境。

最后要感谢我的父母和兄弟姐妹，这么多年来你们无私的付出，为我营造了一个温馨的家庭生活，失败时给我安慰，成功时教我踏实，包容了我无数的错误和任性，我今天的一切一切都离不开你们的照顾。

说不尽的感激，当用一生回报。

李文虎

2011 年 1 月于华中科技大学

参考文献

[1] 缪学勤. 现场总线技术的最新进展. 自动化仪表, 2000(6):1-4.

[2] 贾东耀, 汪仁煌. 工业控制网络结构的发展趋势. 工业仪表与自动化装置, 2002(5):12-14.

[3] 姜晓林, 韩江, 夏链, 等. 现场总线与工业以太网的应用分析. 现代制造工程, 2006(3):14-16.

[4] 马世平. 现场总线标准的现状和工业以太网技术. 机电一体化, 2007(3):6-8.

[5] 黄新平. 开放式网络运动控制系统的研究与实现[硕士学位论文]. 华中科技大学, 2007.

[6] 冯冬芹, 金建祥, 褚健. “工业以太网及其应用技术”讲座 第3讲 以太网与现场总线. 自动化仪表, 2003(6).

[7] Larsson Lars, 张丹丹. 众所瞩目的十四种工业以太网解决方案. 国内外机电一体化技术, 2006(6):12-16.

[8] 杜品圣. 工业以太网技术的介绍和比较. 仪器仪表标准化与计量, 2005(5).

[9] 陈献铎, 孙国锋, 韩学岗. 工业以太网的发展及其技术特点. 山东化工, 2006(3):47-49.

[10] 王家军, 齐冬莲. 运动控制系统的发展与展望. 电气时代, 2004(10):54-56.

[11] 邵明涛. 伺服驱动系统的技术发展趋势分析. 机械工程师, 2008(4):5-7.

[12] 云利军, 孙鹤旭, 雷兆明, 等. 运动控制网络的研究现状及发展趋势. 控制工程, 2006(4):289-293.

[13] 王华兵. 交流伺服系统总线接口设计[硕士学位论文]. 华中科技大学, 2006.

[14] Lei W, Junyan Q. The Real-Time Networked Data Gathering Systems Based on EtherCAT: Environmental Science and Information Application Technology, 2009. ESIAT 2009. International Conference on, Wuhan, 2009[C].4-5 July 2009.

[15] Park J H, Lee S, Lee K C, et al. Implementation of IEC61800 based EtherCAT slave module for real-time multi-axis smart driver system: Control Automation and Systems (ICCAS), 2010 International Conference on, Gyeonggi-do, Korea (South), 2010[C].27-30 Oct. 2010.

[16] Grevsmuhl T, Penno M, Grimberg M, et al. Standard industry components as I/O extension for an Interlock System at PITZ, MTF and XFEL: Real Time Conference, 2009. RT '09. 16th IEEE-NPSS, Beijing, 2009[C].10-15 May 2009.

[17] Junyan Q, Lei W. Networked Motion Control System Design Based on EtherCAT: Intelligent Computation Technology and Automation, 2009. ICICTA '09. Second International Conference on, Changsha, Hunan, 2009[C].10-11 Oct. 2009.

[18] Yoon S S, Lee J C, Cho S J, et al. Gateway between high-performance Fieldbus and serial communication: Control Automation and Systems (ICCAS), 2010 International Conference on, Gyeonggi-do, Korea (South), 2010[C].27-30 Oct. 2010.

[19] Yongseon M, Nak Y K, Eunju K, et al. Network module design based on photonic-EthetCAT for robot drive: Control, Automation and Systems, 2008. ICCAS 2008. International Conference on, Seoul, 2008[C].14-17 Oct. 2008.

[20] Robertz S G, Nilsson K, Henriksson R, et al. Industrial robot motion control with real-time Java and EtherCAT: Emerging Technologies and Factory Automation, 2007. ETFA. IEEE Conference on, Patras, 2007[C].25-28 Sept. 2007.

[21] Jeon Y H, Kim J H, Seo S H, et al. Design of configurable network controller between Ethernet and EtherCAT: Control Automation and Systems (ICCAS), 2010 International Conference on, Gyeonggi-do, Korea (South), 2010[C].27-30 Oct. 2010.

[22] Junyan Q, Lei W, Huijuan J, et al. Design and performance evaluation of networked data acquisition systems based on EtherCAT: Information Management and Engineering (ICIME), 2010 The 2nd IEEE International Conference on, Chengdu, 2010[C].16-18 April 2010.

[23] Hou C, Jiang H, Yang Y, et al. Research on Implementing Real Time Ethernet for Ship Power System: Intelligent Systems and Applications (ISA), 2010 2nd International Workshop on, Wuhan, 2010[C].22-23 May 2010.

[24] Lei W, Mu G L, Jing W, et al. The design & performance analysis for real-time EtherCAT network data acquisition system: Intelligent Control and Information Processing (ICICIP), 2010 International Conference on, Dalian, 2010[C].13-15 Aug. 2010.

[25] Yongming C, Hua C, Mingzhong Z, et al. The relevant research of CoE protocol in

EtherCAT Industrial Ethernet: Intelligent Computing and Intelligent Systems (ICIS), 2010 IEEE International Conference on, Xiamen, China, 2010[C].29-31 Oct. 2010.

[26] Lei W, Huijuan J, Junyan Q, et al. The construction of soft servo networked motion control system based on EtherCAT: Environmental Science and Information Application Technology (ESIAT), 2010 International Conference on, Wuhan, 2010[C].17-18 July 2010.

[27] 刘艳强, 王健, 单春荣. 基于EtherCAT的多轴运动控制器研究. 制造技术与机床, 2008(6):100-103.

[28] 单春荣, 刘艳强, 郇极. 工业以太网现场总线EtherCAT及驱动程序设计. 制造业自动化, 2007(11):79-82.

[29] 阮倩茹, 王辉, 施大发, 等. 基于EtherCAT的高性能交流伺服控制系统设计. 科技导报, 2010(20):58-61.

[30] 李木国, 王磊, 王静, 等. 基于EtherCAT的工业以太网数据采集系统. 计算机工程, 2010(3):237-239.

[31] 孔丽丽. 基于EtherCAT波高数据采集系统的研究[硕士学位论文]. 大连理工大学, 2009.

[32] 谢香林. EtherCAT网络及其伺服运动控制系统研究[硕士学位论文]. 大连理工大学, 2008.

[33] 向乾亮. 实时以太网EtherCAT系统设计及在电力系统中的应用[硕士学位论文]. 华北电力大学（北京）, 2008.

[34] Prytz G. A performance analysis of EtherCAT and PROFINET IRT: Emerging Technologies and Factory Automation, 2008. ETFA 2008. IEEE International Conference on, Hamburg, 2008[C].15-18 Sept. 2008.

[35] Cena G, Scanzio S, Valenzano A, et al. Performance evaluation of the EtherCAT distributed clock algorithm: Industrial Electronics (ISIE), 2010 IEEE International Symposium on, Bari, 2010[C].4-7 July 2010.

[36] Cena G, Scanzio S, Valenzano A, et al. Performance analysis of switched EtherCAT Networks: Emerging Technologies and Factory Automation (ETFA), 2010 IEEE Conference on, Bilbao, 2010[C].13-16 Sept. 2010.

[37] Jae C L, Seong J C, Yong H J, et al. Dynamic drift compensation for the Distributed

clock in EtherCAT: Robotics and Biomimetics (ROBIO), 2009 IEEE International Conference on, Guilin, 2009[C].19-23 Dec. 2009.

[38] Seno L, Zunino C. A simulation approach to a Real-Time Ethernet protocol: EtherCAT: Emerging Technologies and Factory Automation, 2008. ETFA 2008. IEEE International Conference on, Hamburg, 2008[C].15-18 Sept. 2008.

[39] Cena G, Bertolotti I C, Scanzio S, et al. On the accuracy of the distributed clock mechanism in EtherCAT: Factory Communication Systems (WFCS), 2010 8th IEEE International Workshop on, Nancy, 2010[C].18-21 May 2010.

[40] Cereia M, Bertolotti I C, Scanzio S. Performance evaluation of an EtherCAT master using Linux and the RT Patch: Industrial Electronics (ISIE), 2010 IEEE International Symposium on, Bari, 2010[C].4-7 July 2010.

[41] Cena G, Valenzano A, Zunino C. An arbitration-based access scheme for EtherCAT networks: Emerging Technologies and Factory Automation, 2008. ETFA 2008. IEEE International Conference on, Hamburg, 2008[C].15-18 Sept. 2008.

[42] EtherCAT Technology Group | EtherCAT[EB/OL]. [2010-12-29]. http://www.ethercat.org/en/ethercat.html.

[43] Jansen D, Buttner H. Real-time ethernet the EtherCAT solution. Computing & Control Engineering Journal, 2004,15(1):16-21.

[44] EtherCAT Technology Group | Technology[EB/OL]. [2010-12-29]. http://www.ethercat.org/en/technology.html.

[45] EtherCAT技术组. EtherCAT——技术介绍及发展概貌. 国内外机电一体化技术, 2006(6):17-22.

[46] Rostan M. Industrial Ethernet Technology[Z]. EtherCAT Technology Group, 2009.

[47] Beckhoff. EtherCAT ET1100 Datasheet[Z]. 2010.

[48] IEC/SC 65C. IEC-61158-3-12 Industrial communication networks – Fieldbus specifications – Part 3-12: Data-link layer service definition – Type 12 elements[S]. 2007.

[49] IEC/SC 65C. IEC-61158-4-12 Industrial communication networks – Fieldbus specifications – Part 4-12: Data-link layer protocol specification – Type 12 elements[S]. 2007.

[50] IEC/SC 65C. IEC-61158-6-12 Industrial communication networks – Fieldbus specifications – Part 6-12: Application layer protocol specification – Type 12 elements[S]. 2007.

[51] Ethercat-Technology-Group. EtherCAT Communication[Z]. 2009.

[52] 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.

[53] CiA(CAN-in-Automation). DS301 CANopen Application Layer and Communication Profile[S]. 2002.

[54] IEC/SC 65C. IEC-61158-5-12 Industrial communication networks – Fieldbus specifications – Part 5-12: Application layer service definition – Type 12 elements[S]. 2007.

[55] CiA(CAN-in-Automation). DSP402 CANopen Device Profile Drives and Motion Control[S]. 2005.

[56] IEC/SC 22G. IEC 61800-7-200 Adjustable speed electrical power drive systems – Part 7-201: Generic interface and use of profiles for power drive systems – Profile type 1 specification[S]. 2007.

[57] Ethercat-Technology-Group. IEC 61800-7 ETG Implementation Guideline for the CiA402 Drive Profile[Z]. 2007.

附　录　攻读学位期间发表的论文

[1] 李文虎，李叶松，王江城.伺服驱动器 EtherCAT 接口设计.电气传动（已录用）

伺服驱动器工业以太网接口设计——基于EtherCAT与CANopen技术

作者： 李文虎
学位授予单位： 华中科技大学