

基于 IgH 的工业机器人多 EtherCAT 从站自动配置系统

刘 生

(南京熊猫电子装备有限公司, 江苏 南京 210046)

摘 要: 为了提升工业机器人系统中 EtherCAT 总线的配置效率与灵活性, 解决传统方法在设备增减和连接顺序变化时所带来的繁琐操作问题, 提出一种基于 EtherCAT 主站协议栈 IgH 的自动配置方法。在系统启动前, 自动扫描总线设备, 生成配置文件和从站设备描述文件。通过读取这些文件, 动态配置各从站的 PDO 索引地址、参数地址及参数属性, 最终生成 EtherCAT 总线上所有从站设备的映射对象表。该方法成功实现了总线的自动初始化, 避免了手动配置的复杂性。通过自动配置, 系统可以灵活应对设备数量和连接顺序的变化, 大幅提高了系统的适应性。所提出的自动配置方法有效提高了工业机器人系统在 EtherCAT 总线中的配置效率和灵活性, 具有重要的实际应用价值。

关键词: EtherCAT; IgH 主站; 工业机器人; 从站自动配置; 伺服电机驱动器

DOI: 10.11907/rjtk.241812

中图分类号: TP242.2

文献标识码: A

文章编号: 1672-7800(2025)010-0040-07

扫描二维码阅读全文:



Industrial Robot Multi-EtherCAT Slave Automatic Configuration System Based on IgH

LIU Sheng

(Nanjing Panda Electronic Equipment Co., Ltd., Nanjing 210046, China)

Abstract: To enhance the configuration efficiency and flexibility of the EtherCAT bus in industrial robot systems, and to address the cumbersome operations associated with changes in the number of devices and connection order in traditional methods, a dynamic configuration method based on the EtherCAT master stack IgH is proposed. Before system startup, the bus devices are automatically scanned, generating configuration files and description files for the slave devices. By reading these files, the PDO index addresses, parameter addresses, and parameter attributes of each slave device are dynamically configured, ultimately creating a mapping object table for all slave devices on the EtherCAT bus. This method successfully achieves automated bus initialization, eliminating the complexity of manual configuration. Through dynamic configuration, the system can flexibly adapt to changes in device quantity and connection order, significantly enhancing its adaptability. The proposed dynamic configuration method effectively improves both the configuration efficiency and flexibility of EtherCAT bus systems in industrial robots, offering significant practical value.

Key Words: EtherCAT; IgH master; industrial robot; slave automatic configuration; servo motor drives

0 引言

EtherCAT (Ethernet for Control Automation Technology) 是由倍福 (Beckhoff) 公司开发的一种基于以太网的实时通信协议, 专为工业自动化领域设计, 具有高效、灵活和可靠的特点^[1]。在机器人领域, EtherCAT 的应用尤为广泛。目前, 支持 EtherCAT 的从站设备众多, 包括伺服电机驱动器、IO 模块、传感器、变频器等, 这些设备都可以接入支持 EtherCAT 的机器人控制器。

由于 EtherCAT 技术协会 (ETG) 确保 EtherCAT 技术对所有潜在用户开放, 第三方公司推出了 EtherCAT 在不同操作系统及处理器平台的不同实现^[2]。商业版主站软件有 acontis 公司的 EC-Master 产品^[3]、koenig-pa 公司的 KPA

收稿日期: 2024-11-13

作者简介: 刘生 (1988-), 男, 博士, 南京熊猫电子装备有限公司工程师, 研究方向为工业机器人系统。

EtherCAT Master^[4]。开源的主站软件有 IgH 公司的 EtherCAT-Master^[5]、RT Labs 公司的 (SOME)^[6]。IgH 是 EtherCAT 协议主站协议栈的开源实现,是一个完全符合 EtherCAT 标准的库,能够在实时操作系统上运行。相对于其他开源主站软件,IgH 实现了更多 EtherCAT 协议中的功能,而且支持更多平台,本文采用 IgH 以构建工业机器人 EtherCAT 的主站。

EtherCAT 主站启动过程中,需要先初始化配置寄存器,包括从站配置、寄存器映射配置、同步时钟配置等^[7-8]。一般从站设备厂商会提供与其产品配套的从站描述 ESI (EtherCAT Slave Information) 文件,一般为 xml 格式,可以通过主站程序载入该描述文件以实现配置寄存器的初始化。文件中的内容主要有设备 ID、设备名称、接收过程数据对象 (Receive Process Data Object, RPDO) 和发送过程数据对象 (Transmit Process Data Object, TPDO) 映射对象的地址等。

在工业机器人系统中,需要接入多种类型的 EtherCAT 从站设备,在添加或移除设备后,设备连接情况会发生变化,需要重新配置^[9-10]。这在实际应用中经常存在,比如:工业机器人外接了一个或多个 EtherCAT IO 模块以满足外部设备的控制要求,或者需要添加一个或多个电机驱动器作为外部轴,以控制走形轨道或者变位机^[11]。这种情况下,主站需要重新配置其寄存器才可以正常与各从站建立通讯。对于 Acontis 公司的 EC-Master,需要根据实际总线设备及连接情况,基于 ENI 文件手动生成网络信息 ENI (EtherCAT Network Information) 文件,然后导入到主站中使用。如果总线设备或连接情况发生了变化,需要重新编写对应的 ENI 文件并更新到 EtherCAT 主站协议栈中^[12-13]。对于开源的 IgH 及 SOME 主站,则需要根据 ESI 文件去修改相关配置代码以匹配从站连接情况,这在实际应用中存在诸多不便。

文献[14-15]基于 IgH 搭建了 EtherCAT 主站,并对其实时性进行了研究,但未关注多从站配置问题。文献[16-17]利用 IgH 主站结合 ROS 实现了机器人控制系统,虽然对伺服驱动器的 PDO 数据配置进行了描述,但均未涉及自动配置相关内容。目前,许多案例主要集中在系统平台搭建、通讯实时性方面,且所涉及的从站设备通常为同品牌、同型号。然而,在实际工业应用中,常常会遇到不同类型的 EtherCAT 从站设备,而对此类多样化情况的研究仍显不足,这表明自动配置问题在多种设备接入的背景下亟待深入探讨。

针对上述问题,本文提出了一种基于 EtherCAT 主站协议栈 IgH 的工业机器人多从站自动配置方法。该自动配置系统能够灵活应对设备数量和连接顺序变化,自动完成总线扫描及初始化,从而显著提高系统适应性,加快开发与应用速度。该方法为多样化设备接入提供了有效解决方案。

1 系统整体设计

为了实现 IgH 的自动配置,首先对其默认的启动流程进行梳理。IgH 启动阶段的默认配置流程为:①创建主站域,一般只创建一个主站域;②创建从站配置,创建指定数量的从站配置,同时初始化供应商 ID 和产品 ID;③配置 PDO,包含对从站 PDO 映射对象地址的配置和输入输出寄存器入口地址的配置;④配置分布时钟。

当总线设备的连接情况发生变化时,其影响主要集中在第 2 和第 3 步操作上,需要对步骤②和步骤③加以改造,添加扫描及自动配置步骤。借助命令行工具对总线设备进行扫描,生成包含从站信息及其 PDO (进程数据对象) 的相关数据。在获得这些数据后,进入自动配置阶段。EtherCAT 多从站动态配置方法流程如图 1 所示,主要包括扫描和自动配置两个阶段。

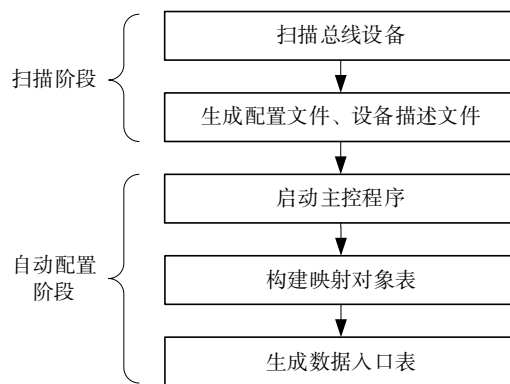


Fig. 1 The dynamic configuration method process of EtherCAT multi-slave stations

图 1 EtherCAT 多从站动态配置方法流程

在系统启动后,首先进入设备扫描阶段,使用 IgH 提供的命令行工具,运行命令 `bin/ethercat slaves` 对总线进行扫描,以获取所有连接的从站数量及类型;其次,通过命令 `bin/ethercat cstruct` 获取各从站的 PDO 信息;再次,基于这些数据,自动生成新的配置文件和设备描述文件,以确保系统能够适应新的连接配置。

自动配置阶段的功能在机器人控制器主程序中,主控程序启动,会先读取总线扫描阶段生成的配置文件和设备描述文件。根据文件中的信息,开始构建映射对象表。

映射对象表是根据配置文件中从站的数量、种类和参数而构建,包含所有从站接收 RPDO 的映射对象表和发送 TPDO 的映射对象表。映射对象表分三级结构,依次为从站设备、映射区、映射对象。他们之间的关系为:映射对象表包含一个或多个从站设备、每个从站设备包含一个或多个映射区、每个映射区包含一个或多个映射对象。

每个从站设备的映射区类型分为 RPDO 映射区和 TPDO 映射区。常用的 RPDO 映射区有 0x1600、0x1601、0x1602、0x1603 等,常用的 TPDO 映射区有 0x1A00、

0x1A01、0x1A02、0x1A03 等。其中,每个映射区都有 10 组索引,最多可以对应 10 个映射对象。

在 EtherCAT 通讯过程中,每个周期都会与所有从站进行读写交互,读写数据将统一放到一个内存区域。为了方便机器人主控程序能够快速找到某站的特定数据,需要生成 EtherCAT 总线上所有从站设备各数据入口的地址表。该地址表包含了从站编号、站内多个输入数据偏移地址、站内多个输出数据偏移地址。机器人控制器可以使用这些偏移地址以读写 EtherCAT 通讯数据,完成与 EtherCAT 总线的数据交互。

2 关键技术实现

2.1 扫描阶段

通过创建 Linux 的开机自动运行脚本,在脚本中扫描总线设备并将执行结果处理后分别存储到配置文件和设备描述文件中。

需要注意的是,由于从站启动需要一定的时间,如果主站协议栈启动较快,则需要等待从站启动完毕再扫描。在扫描时,要根据主站从站的启动速度调整等待时间。如果在从站启动完毕前就发起了扫描,则得到的结果不准确。

2.2 自动配置功能结构

自动配置阶段需要具有配置文件解析、映射对象表构建和数据入口表构建等功能。为了更好地设计该部分功能,使用 UML 建模工具创建相关类图,如图 2 所示^[18]。

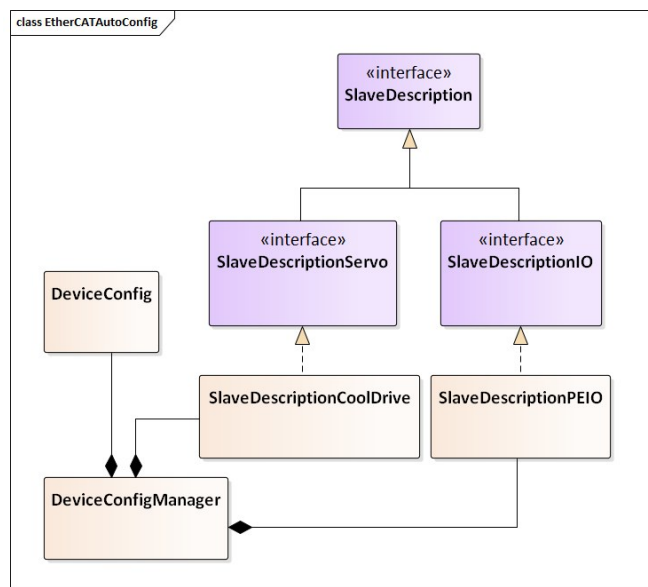


Fig. 2 EtherCAT automatic configuration UML class diagram

图2 EtherCAT 自动配置 UML 类图

DeviceConfig 类读取配置文件和设备描述文件,解析出所有的从站设备列表和 PDO 信息。UML 类图给出了两种类型的 EtherCAT 从设备示例,一种是伺服电机驱动器,另一种是 IO 设备。SlaveDescription 类定义了一些设备描

述类通用的接口,主要包括:①供应商和产品 ID 的读取和设置接口;②TxPDOEntry 映射对象地址的读取和设置接口;③RxPDOEntry 映射对象地址的读取和设置接口;④Tx-PDOMapping 映射区的读取和设置接口;⑤RxPDOMapping 映射区的读取和设置接口;⑥TxPDORegistrations 数据入口的读取和设置接口;⑦RxPDORegistrations 数据入口的读取和设置接口。

接口类 SlaveDescriptionServo 和 SlaveDescriptionIO 主要针对设备类型的不同,定义了设备相关接口。对于驱动器设备,主要有控制字、操作模式、目标位置、实际位置、实际力矩、报警码等功能接口。对于 IO 设备主要是数字输入和输出功能接口。

SlaveDescriptionCoolDrive 是针对某品牌驱动器的实现类,其中对 SlaveDescription 和 SlaveDescriptionServo 中定义的接口进行了具体实现。SlaveDescriptionPEIO 是针对某品牌 EtherCAT IO 设备的实现类,其中对 SlaveDescription 和 SlaveDescriptionIO 中定义的接口进行了具体实现。

DeviceConfigManager 是设备配置管理器,是该自动配置功能的中枢,DeviceConfigManager 根据 DeviceConfig 解析到的从站设备列表,依次对各从设备进行初始化。

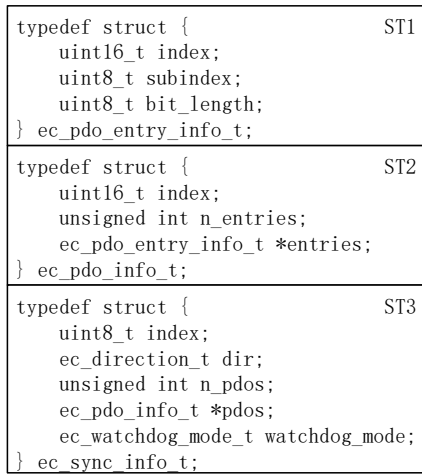
2.3 映射对象表管理

主站和从站进行数据交互的方式包括数据服务对象 (Service Data Object, SDO) 和过程数据对象 (Process Data Object, PDO) 两种不同的方式。SDO 基于请求响应机制,在需要读写时发起请求并进行数据传输。PDO 通过映射的方式,将对象字典里需要传输的数据直接映射到 PDO 数据,可以实现周期性实时的数据传输。相比 SDO, PDO 具有更高的通讯效率。为了实现 PDO 数据传输,在启动阶段需要对 PDO 的映射区域进行配置,使其映射到对应的对象字典。

在设备扫描阶段生成了设备描述文件,存储的是从站的 PDO 信息。经过 DeviceConfig 类读取后,会解析出 3 种类型的数据: ec_pdo_entry_info_t、ec_pdo_info_t、ec_sync_info_t。PDO 相关结构体及对应关系如图 3 所示。为方便理解,下文用 ST1、ST2 和 ST3 加以表示。解析完毕后,会根据不同的设备类型以创建从站描述对象 SlaveDescriptionCoolDrive 或 SlaveDescriptionPEIO。每个对象由对应设备的 ST1、ST2、ST3 组成,存储了该设备类型的 PDO 相关信息。

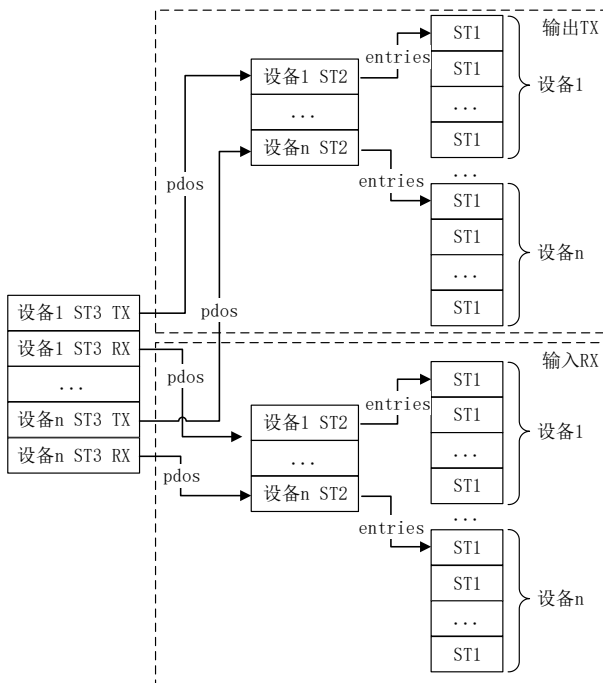
ST1 是 PDO 映射的对象字典信息,包括主索引、次索引和位长。例如:伺服电机驱动器中状态字对应的主索引是 0x6041,次索引是 0x00,位长是 16 位。

ST2 是 PDO 映射区配置信息,用来配置将映射的一组对象字典放入对应的映射区,使用成员 entries 指针指向一组 ST1 类型 ST。根据类别,又可分为接收和发送两种对象。常用的接收映射区有 0x1600 和 0x1700,用来存放接收对象,比如驱动器中的控制字、目标位置等。常用的发



(a) PDO related structures

(a) PDO 相关结构体



(b) The relationship among PDO related structures

(b) PDO 相关结构体之间的关系

Fig. 3 PDO related structures and their corresponding relationships

图 3 PDO 相关结构体及对应关系

送映射区有 0x1A00 和 0x1B00,用来存放发送对象,比如驱动器中的状态字、当前位置等。

ST3 用来组织多个映射区以实现周期性的数据同步,内部使用指针 pdos 指向 ST2 类型结构体。在 ST3 中,需要配置映射区的方向和映射区的索引地址。三者之间的数量关系是:一个 ST3 对应一个或多个 ST2,一个 ST2 对应一个或多个 ST1。

在有多从站时,需要对多个从站的 PDO 信息重新进行组合设置,可以得到其集合所对应的 5 个列表,分别是 ST1 的 Tx 表、ST1 的 Rx 表、ST2 的 Tx 表、ST2 的 Rx 表、ST3 的表。具体过程如下:针对各设备的多个 ST1 数据,Devi-

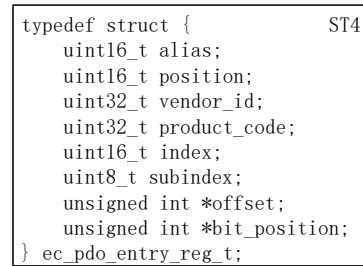
ceConfigManager 会遍历所有从站描述对象 SlaveDescriptionCoolDrive 或 SlaveDescriptionPEIO,并依次将设备对象的对象字典信息根据类别存储在 ST1 的 Tx 表、ST1 的 Rx 表中。同时,ST2 的 Tx 表、ST2 的 Rx 表和 ST3 表也会被更新。由于有多从站,ST2 列表中的各映射区地址要与 ST1 列表中的各映射区重新建立关联,ST2 的结构体成员 entries 需要依次指向 ST1。同样地,ST3 也需要根据实际设备连接情况,设置结构体成员 dir 为输入或者输出方向,设置结构体成员 pdos 指针依次指向对应的 ST2 对象。

当完成上述 3 种类型数据的设置之后,即可以调用 IgH 的从站配置接口 ecrt_slave_config_pdos 以配置所有从站的 PDO。

2.4 数据入口表管理

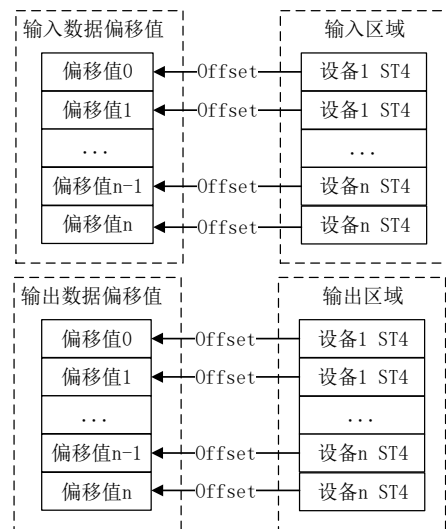
数据入口管理主要实现对 EtherCAT 中各从站数据的访问,提供从站 PDO 映射中对应对象字典的读写入口。DeviceConfigManager 根据 DeviceConfig 完成从站设备列表和 PDO 信息解析后,会对所有从站的输入输出区域,分别创建 ec_pdo_entry_reg_t 列表并初始化。数据入口结构体及其与本地数据关系如图 4 所示。

为方便理解,用 ST4 表示结构体 ec_pdo_entry_reg_t。



(a) Data entry structure

(a) 数据入口结构体



(b) The relationship between local data and data entry structure

(b) 本地数据与数据入口结构体间的关系

Fig. 4 Data entry structure and its relationship with local data

图 4 数据入口结构体及其与本地数据关系

ST4除主索引、次索引,还有 position 和 offset 需要设置。position 为从站的顺序编号,offset 指针为对应数据在数据区域中的偏移地址。

主站管理数据入口的方式是将输入和输出数据的偏移值分别放在两块连续的区域“输入数据偏移值”和“输出数据偏移值”中,该系统对应代码中的该索引数组名称为 m_inPDOOffsets 和 m_outPDOOffsets。访问 EtherCAT 数据时,根据此索引即可方便访问。具体实现过程如下:如果是读取 EtherCAT 数据,需从“输入数据偏移值”中找到该数据在主站数据中的偏移值,利用该偏移值即可更新从站的对象字典。比如:读取第一个站偏移值为 1 的无符号 16 位数据时,可使用 EC_READ_U16(m_inProcessData + m_inPDOOffsets[1]),其中 m_inProcessData 是输入 PDO 数据区的起始地址。写 EthercatCAT 数据与读数据类似,本文不再赘述。

3 关键代码

3.1 扫描阶段

将扫描到的设备类型和数量写入配置文件,将设备具体信息写入设备描述文件。关键代码如图 5 所示。

```
driver_num=`/opt/etherlab/bin/ethercat slaves | grep -w "CoolDrive" | wc -l`
ecio_num=`/opt/etherlab/bin/ethercat slaves | grep -w "PECATIO" | wc -l`
sed -i -e "${driver_line}a\NUMBER = ${driver_num}" ${INI_PATH}
sed -i -e "${ecio_line}a\NUMBER = ${ecio_num}" ${INI_PATH}
/opt/etherlab/bin/ethercat cstruct > ${DES_PATH}
```

Fig. 5 The scanned script key code

图 5 扫描脚本关键代码

前面两行完成了对驱动器和 IO 设备的计数,第 3 行、第 4 行完成了对配置文件的更新,最后一行将所有设备信息写入了配置文件。在代码实现中,对不同设备的名称进行了限定,只有当设备名称匹配成功时,才会对其进行计数并在后续流程中加以处理。对于无法识别名称的设备,则不会建立连接。这一机制在一定程度上提升了系统安全性,防止了恶意设备随意接入,减少了潜在安全风险。

3.2 自动配置阶段

IgH 主站通过调用从站配置接口 ecrt_slave_config_pdos 以配置所有从站的 PDO, DeviceConfigManager 为 IgH 主站提供了获得所有从站 ec_pdo_info_t 类型的 TxPDO 信息的调用接口,如图 6 所示。

代码中对所有从站 ec_pdo_info_t (ST2) 信息进行组合,并通过 vector 类型返回给 IgH 主站。类似地,

```
void DeviceConfigManager::GetTxPDOMapping(std::vector<std::vector<ec_pdo_info_t> >> &map)
{
    // All sub GetTxPDORegistrations will push_back ec_pdo_info_t to _reg
    for (auto itr = m_manipulatorDescription.begin(); itr != m_manipulatorDescription.end(); ++itr) {
        std::vector<ec_pdo_info_t> deviceMap;
        (*itr)->GetTxPDOMapping(deviceMap);
        _map.push_back(deviceMap);
    }
    for (auto itr = m_IODescription.begin(); itr != m_IODescription.end(); ++itr) {
        std::vector<ec_pdo_info_t> deviceMap;
        (*itr)->GetTxPDOMapping(deviceMap);
        _map.push_back(deviceMap);
    }
}
```

Fig. 6 API function that obtained TxPDO mapping

图 6 获得 TxPDO 映射的 API 函数

ec_pdo_entry_info_t (ST1)、ec_sync_info_t (ST3) 采用同样的方式进行访问管理。通过调用对应的接口可以获取所有 PDO 相关信息,最后调用 ecrt_slave_config_pdos 完成所有从站 PDO 的配置。

对于设备数据入口,根据 TX 和 RX 分类,将其依次存储到不同对象中。图 7 展示了数据入口初始化关键代码。

```
for (int indexDev = 0; indexDev < m_deviceNum; indexDev++) {
    std::vector<ec_pdo_entry_reg_t> deviceOutReg = outputRegs[indexDev];
    std::vector<ec_pdo_entry_reg_t> deviceInReg = inputRegs[indexDev];

    // Assign array
    for (size_t i = 0; i < deviceOutReg.size(); i++) {
        deviceOutReg[i].position = indexDev;
        deviceOutReg[i].offset = &m_outPDOOffsets[outArrayOffset];
        outputRegsArray[outArrayOffset] = deviceOutReg[i];
        outArrayOffset++;
    }
    for (size_t i = 0; i < deviceInReg.size(); i++) {
        deviceInReg[i].position = indexDev;
        deviceInReg[i].offset = &m_inPDOOffsets[inArrayOffset];
        inputRegsArray[inArrayOffset] = deviceInReg[i];
        inArrayOffset++;
    }
}
ecrt_domain_reg_pdo_entry_list(m_outDomain, outputRegsArray);
ecrt_domain_reg_pdo_entry_list(m_inDomain, inputRegsArray);
```

Fig. 7 IO module data entry initialization key code

图 7 IO 模块数据入口初始化关键代码

首先,遍历所有从站设备,针对每一个设备的输入输出区域,创建 ec_pdo_entry_reg_t (ST4) vector;其次,设置其 position 和 offset,并将列表内容赋值给 inputRegsArray 和 outputRegsArray;最后,调用 ecrt_domain_reg_pdo_entry_list 以完成 IgH 主站的 PDO 入口表初始化。

4 功能验证及测试

本文实验平台为熊猫工业机器人 PR07DL 机型,其负载能力为 7 kg,臂展为 0.9 m。该机器人环境为嵌入式 Linux 环境,使用的是 IgH EtherCAT 主站,驱动器选用清能德创 CoolDrive RC 系列 6 合一一体化驱动器。此外,系统外接了 3 个 Panda EtherCAT IO 模块,每个模块具备 16 路输入和 16 路输出功能,以测试自动配置功能和通讯性能。实验平台如图 8 所示。



Fig. 8 EtherCAT automatic configuration experimental platform

图 8 EtherCAT 自动配置实验平台

4.1 功能验证

启动机器人,手动运行命令“bin/ethercat slaves”,显示总线上的所有从站列表,得到结果如图 9 所示。

```
[root@G4Robot:~]# /opt/etherlab/bin/ethercat slaves
0 0:0 OP + CoolDrive RC
1 0:1 OP + CoolDrive RC
2 0:2 OP + CoolDrive RC
3 0:3 OP + CoolDrive RC
4 0:4 OP + CoolDrive RC
5 0:5 OP + CoolDrive RC
6 0:6 OP + PECATIO-1616 (Digital 16-Input/16-Output)
7 0:7 OP + PECATIO-1616 (Digital 16-Input/16-Output)
8 0:8 OP + PECATIO-1616 (Digital 16-Input/16-Output)
```

Fig. 9 Slave list example

图 9 从站列表实例

可以看到,当前 EtherCAT 总线上共有 9 个从站设备,前面 6 个是驱动器,后面 3 个是输入输出 IO 模块。启动机器人后,在扫描阶段会自动扫描以获取上述信息,可以查看其生成的配置文件以验证扫描阶段的功能。更新后的配置文件内容如图 10 所示。

```
; *****
; EtherCAT Device settings
; *****

[ManipulatorMotor]          ; Manipulator Motor settings
NUMBER = 6
TYPE = Cooldrive

[IOModule]                  ; IO Module settings
NUMBER = 3
NUM_1_TYPE = PECATIO_1616 ; PANDA EtherCAT IO module 16 DI 16 DO
NUM_2_TYPE = PECATIO_1616 ; PANDA EtherCAT IO module 16 DI 16 DO
NUM_3_TYPE = PECATIO_1616 ; PANDA EtherCAT IO module 16 DI 16 DO
```

Fig. 10 Configuration files

图 10 配置文件

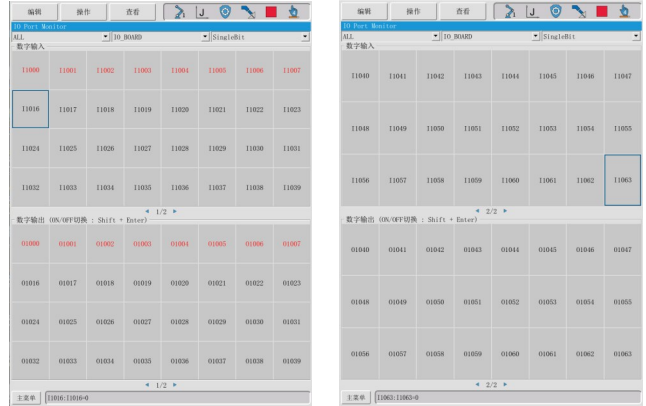
等待大约 40 s,机器人系统启动完毕,IO 板卡建立 EtherCAT 通讯,如图 11 所示。可以看到,3 块 IO 板卡方形框中的 RUN 指示灯都常亮,代表 EtherCAT 总线已建立正常通讯。



Fig. 11 EtherCAT communication established by IO board

图 11 IO 板卡建立的 EtherCAT 通讯

通过示教器界面查看外部 IO 监视界面,如图 12 所示。可以看到,增加了编号为 I1016-I1063 的 48 个输入 IO,增加了编号为 O1016-O1063 的 48 个输出 IO,与 3 块板卡的输入输出数量匹配。由此可以验证,3 块增加的 EtherCAT IO 板卡完成了扫描及自动配置流程,并正常运行。



(a) Monitor interface 1

(b) Monitor interface 2

(a) 监视界面 1

(b) 监视界面 2

Fig. 12 Robot teaching pendant IO port monitor interface

图 12 机器人示教器 IO 监视界面

4.2 通讯性能测试

高精度的时间同步对于保证 EtherCAT 通信的实时性能至关重要。同步周期决定了从站设备之间数据交换的频率和实时控制系统的响应速度,而同步周期的抖动时间则是衡量通信稳定性的重要指标。如果抖动时间过大,数据同步无法在规定的响应时间内完成,可能导致机器人系统中机械臂的定位误差增大,甚至引起机器人抖动。严重情况下,还可能造成机器人停机,增加机械损耗,影响设备寿命。

一般而言,从站时钟较为精确。本文实验中,第一个从站的时钟被用作参考时钟,主站与其同步。主站通过对每个周期进行计时,测量实际周期与标准周期的差值,以获取通信周期的抖动时间。实验中采用 2 ms 的同步周期,进行了 3 min 的采集,共采集了 9 万次的同步周期抖动数据。结果如图 13 所示,横轴为通信周期次数(单位:万次),纵轴为通信周期的抖动时间(单位:μs)。可以看到,绝大多数抖动时间保持在 ±6 μs 范围内,最大抖动时间约为 25 μs。该结果表明,完成自动配置的系统具备良好的

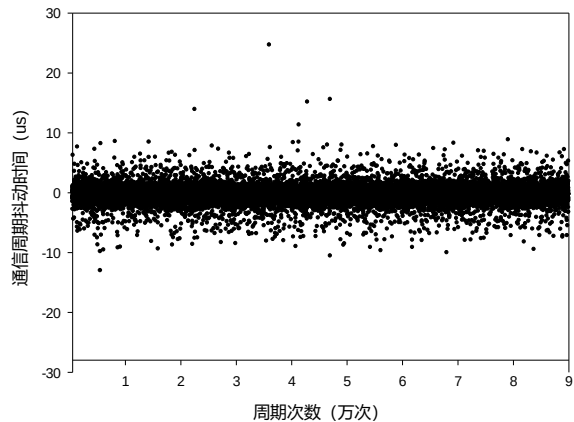


Fig. 13 Communication cycle jitter time

图 13 通信周期抖动时间

实时性和稳定性,能够满足机器人运动控制和外部数字 IO 控制的要求。

5 结语

本文基于 IgH 主站,针对机器人需要接入不同种类 EtherCAT 从站的需求,设计并实现了工业机器人 EtherCAT 多从站自动配置方法。通过扫描阶段获得在线的从站信息,并更新配置文件和设备描述文件。在自动配置阶段完成设备 PDO 映射信息管理和设备数据入口管理。最后, IgH 主站利用这些信息,完成主站初始化并提供数据访问接口。

尽管本文技术细节依赖于 IgH 主站的具体实现,但该流程同样适用于其他商业或开源的 EtherCAT 主站。首先进行总线扫描以生成配置信息,然后根据不同协议栈的具体实现方式完成自动配置。其主要差别体现在所使用的扫描工具、生成的配置文件,以及自动配置的流程细节上。本文案例的实现方法可为其他 EtherCAT 协议栈,甚至其他类型的总线通信协议提供参考和借鉴。

在从站的设备配置管理器中,需要预先将各种待接入设备的描述信息编写好。未来,可考虑进一步简化流程,直接从命令“bin/ethercat cstruct”输出的 PDO 信息中提取所需内容,这样无需提前编写设备的描述文件,从而进一步提高系统的通用和灵活性。

参考文献:

- [1] EtherCAT Ethernet fieldbus [EB/OL]. <https://www.beckhoff.com.cn/zh-cn/products/i-o/ethercat>.
EtherCAT 以太网现场总线 [EB/OL]. <https://www.beckhoff.com.cn/zh-cn/products/i-o/ethercat>.
- [2] Technology group [EB/OL]. https://www.ethercat.org/en/tech_group.html.
- [3] EtherCAT® main device stack for EtherCAT device protocol (EDP) [EB/OL]. <https://www.acontis.com/en/ethercat-master.html>.
- [4] KPA EtherCAT master [EB/OL]. <https://koenig-pa.de/products/ethercat/kpa-ethercat-master/>.
- [5] EtherCAT master [EB/OL]. https://etherlab.org/en_GB/ethercat.
- [6] Simple open EtherCAT master or SOEM [EB/OL]. <https://openethercatso-ciety.github.io/doc/soem/index.html>.
- [7] QING P Z, LIU J Q, LI Q, et al. Real time EtherCAT master development for servo motion control [J]. Journal of Mechanical & Electrical Engineering, 2019, 36(11): 1216-1220.
庆鹏展, 刘建群, 李强, 等. 面向伺服运动控制的实时 EtherCAT 主站开发 [J]. 机电工程, 2019, 36(11): 1216-1220.
- [8] KANG S J. A study on implementation of real-time EtherCAT master [J]. Journal of the Semiconductor & Display Technology, 2021, 20(2): 131-136.
- [9] DAI H C, SUN D N, ZHAO W B. A review of the development and application of industrial robot technology [J]. The Journal of New Industrialization, 2021, 11(4): 5-6.
代浩岑, 孙丹宁, 赵文博. 工业机器人技术的发展与应用综述 [J]. 新型工业化, 2021, 11(4): 5-6.
- [10] ARENTS J, GREITANS M. Smart industrial robot control trends, challenges and opportunities within manufacturing [J]. Applied Sciences, 2022, 12(2): 937.
- [11] MALPE P N, PANGAONKAR S. Porting an EtherCAT slave stack for PLC I/O module [J]. International Journal of Recent Development in Engineering and Technology, 2014, 2(2): 97-101.
- [12] DONG H T, ZHANG S T, FENG J Q. Design of multi-axis EtherCAT protocol based on analog servo [J]. Instrumentation Technology and Sensors, 2023(2): 69-72.
董海涛, 张帅涛, 冯建强. 基于模拟伺服的多轴 EtherCAT 协议设计 [J]. 仪表技术与传感器, 2023(2): 69-72.
- [13] JIANG J, GAO J L, XU C L, et al. Design of EtherCAT master based on X86 platform [J]. Machine Tool & Hydraulics, 2019, 47(7): 89-93.
蒋杰, 高军礼, 徐超蓝, 等. 基于 X86 平台的 EtherCAT 主站设计 [J]. 机床与液压, 2019, 47(7): 89-93.
- [14] QIU C H, GUO Y J, LI Z P, et al. Development and real-time optimization of EtherCAT master based on Xenomai [J]. Automation Applications, 2021(12): 82-86.
邱昌华, 郭永杰, 李宗坡, 等. 基于 Xenomai 的 EtherCAT 主站开发与实时性优化 [J]. 自动化应用, 2021(12): 82-86.
- [15] LI Z P, HE J L, QIU C H, et al. Implementation and real-time research of EtherCAT master based on Linux platform [J]. Information System Engineering, 2022(1): 52-56.
李宗坡, 贺加乐, 邱昌华, 等. 基于 Linux 平台 EtherCAT 主站的实现和实时性研究 [J]. 信息系统工程, 2022(1): 52-56.
- [16] ZHANG H, HUANG Z, ZHANG X Y, et al. Design of multi-joint robot control system based on ROS and IgH master [J]. Instrument Technology and Sensor, 2022(3): 57-62.
张洪, 黄昭, 张小勇, 等. 基于 ROS 和 IgH 主站的多关节机器人控制系统设计 [J]. 仪表技术与传感器, 2022(3): 57-62.
- [17] XU J M, WU S W, WU X W, et al. SCARA robot control system based on ROS and IgH EtherCAT master [J]. High Technology Communications, 2019, 29(9): 876-885.
徐建明, 吴蜀巍, 吴小文, 等. 基于 ROS 和 IgH EtherCAT 主站的 SCARA 机器人控制系统 [J]. 高技术通讯, 2019, 29(9): 876-885.
- [18] XUE L, TANG J P, GAO J. UML modeling technology [M]. Shanghai: Shanghai Jiao Tong University Press, 2022.
薛岚, 唐金鹏, 高杰. UML 建模技术 [M]. 上海: 上海交通大学出版社, 2022.

(责任编辑: 孙娟)