文章编号:1671-4598(2023)01-0222-08 DOI:10.16526/j.cnki.11—4762/tp.2023.01.033 中图分类号:TP273 文献标识码:A

PROFINET IO 设备中 DCP 协议的研究与应用

荣 锋[1,2], 朱瑞华[1]
(1. 天津工业大学 电子与信息工程学院, 天津 300387;
2. 天津市光电检测技术与系统重点实验室, 天津 300387)

摘要: 为解决自动化外围现场设备接入 PROFINET 网络, 而同类设备依赖特定芯片, 成本较高, 可移植性差的问题, 利用软件设计实现普通网卡下的 PROFINET IO 通信; 在树莓派 3B+开发板上实现 IO 设备, 支持非同步实时数据交换, 可以通过槽/子槽数据模型匹配 IO 模块; 通过设计描述性文件, 使其 IO 模块类型映射模块、子模块的 ID, 实现模块化 IO 设备, 允许根据需要添加和删除模块; 系统启动时控制器根据设备名称寻址, 再基于设备名称为 IO 设备分配 IP 地址, 这一过程的实现依赖 DCP 协议, 它是 PROFINET 网络中数据交互的基础; 为了深入探究 PROFINET 协议栈, 在实现 IO 通信的基础上从 DCP 协议的原理入手, 对其功能和程序实现进行了详细研究, 并将所实现的 IO 设备与 PLC S7-1200 进行组态, 验证了 IO 设备所实现的功能和 DCP 工作的过程; 实验测试表明, 树莓派实现了 PROFINET RT 和 NRT 通信功能, 其更新时间抖动小于 50 μs, 满足控制工程上的应用要求。

关键词: PROFINET; DCP; Wireshark; PLC S7-1200; 数据交互; 设备名称

Research and Application of DCP in PROFINET IO Device

RONG Feng[1,2], ZHU Ruihua[1]
(1. School of Electronic and Information Engineering, Tianjin Polytechnic University, Tianjin 300387, China;
2. Tianjin Key Laboratory of Optoelectronic Detection Technology and System, Tianjin 300387, China)

Abstract: In order to solve the problems of similar devices relying on specific chips, high cost, and poor portability when the automatic peripheral field devices are connected to PROFINET network, the software design is used to realize PROFINET IO communication under the common network card. The IO device is implemented on the Raspberry Pi 3B+ development board, which supports asynchronous real-time data exchange. The IO module can be matched through the slot or sub-slot data model. Through design descriptive file, the IO module type is mapped to the ID of the module and sub-module to realize the modular IO devices, and modules can be added and deleted as required; When the system is started, the controller addresses the IO devices by the device name, and then assigns IP addresses to the IO devices based on the device name. The implementation of this process depends on DCP protocol, which is the basis of data interaction in PROFINET network. In order to deeply explore the PROFINET protocol stack, based on the implementation of PROFINET, starting with the principle of DCP protocol, the function and program implementation are studied in detail, and the implemented IO devices are configured with PLC S7-1200 to verify the function of IO devices and working process of DCP. Experimental tests show that the Raspberry Pi realizes the communication functions of PROFINET RT and NRT, and the update time jitter is less than 50 μs, which meets the application requirements of control engineering.

Keywords: PROFINET; DCP; Wireshark; PLC S7-1200; data interaction; equipment name

0 引言

PROFINET 是工业控制领域广泛应用的一种工业以太网, 具备成熟的解决方案[1]。在 HMS Networks 对工业网络市场研究报告中发现 PROFINET 的市场份额位居首位, 尤其是在工厂自动化、过程自动化、安全应用领域[2]。截止到 2022 年初, PROFINET 约已连接 1 600 万个节点, 足以证明 PROFINET 的应用是如此广泛。而国内在“中国制造 2025”的国家战略推动下, PROFINET 也将有更大的发展, 因此对 PROFINET 的研究也应该更深层次和多方面。

对于 PROFINET 的研究可大致分为性能分析、产品开发、诊断、安全等。张国栋等人对 PROFINET RT 和 IRT 的数据转发方式进行了分析, 并利用公式推导出了 IO 设备的刷新时间[3]。KLEINES H 等人通过数值模拟 PROFINET IO 网络性能[4]。刘柯等人对 RT 通信帧的结构、周期组成以及通信周期的相关参数计算进行了研究[5]。对于

收稿日期:2022-06-20; **修回日期**:2022-07-15。
基金项目:天津市科委青年基金项目(15JCQNJC42100);天津市科技特派员项目(16JCTPJC48100,16JCTPJC47200)。
作者简介:荣 锋(1979-),男,山东潍坊人,博士,副教授,主要从事工业以太网,信息采集与处理,智能控制等方向的研究。
通讯作者:朱瑞华(1997-),女,河南周口人,硕士研究生,主要从事工业以太网,嵌入式系统等方向的研究。
引用格式:荣 锋,朱瑞华.PROFINET IO 设备中 DCP 协议的研究与应用[J].计算机测量与控制,2023,31(1):222-229.

PROFINET IO 设备的开发，从原理上普通以太网网卡不能完全实现 PROFINET IO 功能，目前市场上广泛应用的解决方案是依赖西门子公司的 ERTEC 芯片、瑞萨公司的 TPS-1 芯片、赫优讯公司的 Netx51/52 芯片等专用网络控制芯片来实现 PROFINET IO 通信。如闫菲等人用 ERTEC200P 芯片开发了 PROFINET IO 工业以太网接口[6]。徐建明等人基于 TPS-1 芯片设计了一种 PROFINET 耦合器设备[7]。刘振杰以 NETX51 芯片设计了一款 PROFINET 从站接口模块[8]。但这些方案开发成本较高，对于一些非运动控制领域有一定的冗余性。因此本文利用普通网卡通过软件设计实现 PROFINET IO 通信的 NRT、RT 功能。为证明广泛性，在树莓派 3B+硬件平台上移植协议栈，实现树莓派接入 PROFINET 网络。PROFINET IO 协议栈依照设备类型 Class B 进行设计，支持非同步实时数据交换，可以实现 LLDP、DCP、SNMP 协议。本设计可以实现对实时性要求不苛刻的非运动控制领域的工业控制。

PROFINET 通信系统中，DCP 在系统启动时分配设备名称和 IP 地址，是 PROFINET 网络中数据交互的基础，对于 PROFINET 的研究和实现通信功能离不开 DCP。因此本文在实现 PROFINET IO 通信功能的基础上对协议栈中的 DCP 子协议进行深入研究，从 DCP 协议的原理出发，对 DCP 在 PROFINET 系统中的作用及软件实现进行研究和分析。利用西门子 PLC S7-1200 控制器搭建通信测试平台，验证 DCP 协议在 PROFINET 中的作用及树莓派 RT 和 NRT 通信的实现，并将 Wireshark 软件抓取到的数据帧解析，得出 IO 设备的更新时间抖动。本文的研究将会帮助自动化工程师加深对 PROFINET 协议栈的理解，从而更好的去应用 PROFINET。

1　PROFINET IO

PROFINET 根据不同的应用类型提供了两种解决方案，PROFINET CBA 和 PROFINET IO[9]。其中 PROFINET CBA 主要用于实现控制器之间的通信，而 PROFINET IO 主要用于实现控制器与 IO 设备之间的通信。

PROFINET IO 定义了 3 个设备角色：IO 控制器、IO 监控器和 IO 设备。

1）IO 设备是构成自动化过程接口的外围现场设备，分为块 IO 设备和模块化 IO 设备两种类型。块 IO 设备具有固定模块配置，而模块化 IO 设备可灵活添加和删除模块。

2）IO 控制器用于控制自动化过程，负责其相关的现场设备的 IP 地址分配、配置、参数设置、警报处理。

3）IO 监视器用于完成对设备的远程控制、检查、维护或参数化 IO 设备的客户端功能，通过在监视器禁用 IO 控制器来临时强制过程输出值，通常是 HMI、工程工具充当监视器。

PROFINET IO 设备在应用层定义了插槽/子插槽的设备模型，其中插槽表示设备的物理或逻辑模块，又被分为多个子插槽，子插槽相当于过程数据的接口，每个插槽的 IO 数据、报警、诊断数据都被分配到这些子插槽中[10]。IO 设备不同模块和子模块在通用站点描述文件（GSD，general station description）中描述。块 IO 设备具有通用站点描述文件中描述的固定模块配置，而模块化 IO 设备允许根据需要添加和删除模块，GSD 文件中包含所有可能的模块，但 IO 设备的实际模块需要在工程工具中定义，下载到控制器后，由控制器告诉 IO 设备需要的配置。

PROFINET IO 针对数据对实时性不同的要求，划分为标准通道，实时通道和等实时通道[11]。标准通道是基于 UDP/IP，用于上下文管理，处理 IO 设备的配置和诊断。实时通道（RT）用于传输循环过程数据、事件和警报，它在标准通道的基础上，将实时数据帧嵌入到以太网帧中，不使用任何高层协议。也就是直接在数据链路层进行封装，在以太网报文头后加上 PROFINET 报文头，因此能绕过 UDP/IP 堆栈，直接由 PROFINET 的协议栈处理，以此来减少数据的开销量，高效利用带宽，提高 PROFINET 的实时性能。等实时通道用于对时间要求严格的通信，如运动控制，它通过特殊的 IRT 硬件使 IRT 数据在预先配置和计划的时隙中发送，从而达到更高实时性。

2　DCP 发现和基本配置协议

DCP 为“发现和基本配置协议”是一种数据链路层协议，它为 PROFINET 提供多种服务，例如用于 PROFINET 网络中的发现识别设备，配置设备名称、配置 IP 地址等[12]。

2.1　DCP 的主要功能

为实现这些服务 DCP 提供了“Identify All”、“Identify”、“Set”、“Set-Flash”、“Set-Reset to Factory”、“Get”和“Hello”作为主要功能。PROFINET 工程工具、控制器和设备中都集成了 DCP 服务，但他们所侧重的功能不同，如在 IO 设备上要能对控制器所发布的命令做出响应，要能主动利用 Hello 功能向控制器发送消息，而 IO 控制器要能利用 Identify 功能来查找具体设备。主要功能具体描述如下。

Identify All：识别全部设备。以广播的方式向整个网络发送消息，所有设备收到消息都要做出响应。工程工具中利用此功能可以获得所有设备信息列表来确定网络中是否存如下问题：1）设备有无连接；2）设备名称是否设置；3）设备中有无重复的 IP 地址或重复的设备名称；4）设备名称和 IP 地址设置是否合规；5）设备类型或供应商是否正确，利用 Identify All 功能工程工具可以轻松实现网络管理。

Identify：查找具体设备和检查设备参数设置。系统启动前，IO 控制器会用它来识别设备，通过设备名称来进行查找，具有该设备名称的设备进行响应，但如果查找的设备名称长度为零则所有未分配名称的设备都要做出响应。

Get：获取设备信息。比如获取设备名称、IP 地址和制造商信息等。

Set：向设备写入参数。将设备名称和 IP 地址写入寻址到的设备中，可以在一个帧中依次请求多个条件，用来找到相匹配的设备。

Set-Flash：让指定 IO 设备的 LED 灯闪烁，当同一网络中有多个同类设备时，可以通过闪烁 LED 的方式来确定要操作的对象。

Hello：IO 设备主动给 IO 控制器发送信息。主要在快速启动时使用 Hello 功能，表示 IO 设备已经准备就绪，从而缩短控制器查找 IO 设备的时间，实现系统的快速启动。

2.2 DCP 在 PROFINET 中的典型用法

DCP 是 PROFINET 重要的组成部分，实现 PROFINET IO 系统启动时设备名称、IP 地址的分配，是 PROFINET 网络中数据交互的基础[13]。同时 DCP 使 PROFINET IO 设备实现无需组态工具替换成为可能。

2.2.1 启动时的 DCP

DCP 协议是实现 PROFINET IO 通信的条件，在系统组态时，工程工具会先给 IO 控制器分配 IP 地址，然后给组态好的 IO 设备分配设备名称。随后将组态好的信息下载到 IO 控制器中，这样 IO 控制器就拥有寻址 IO 设备和数据交换所需的所有信息。IO 控制器在根据设备名称为 IO 设备分配 IP 地址。IP 地址和设备名称的分配都依赖于 DCP 子协议，在系统通电后，IO 控制器与 IO 设备建立通信关系和应用关系，然后 IO 控制器和 IO 设备交换过程数据、报警和非循环数据。从组态到系统启动的步骤如图 1 所示。

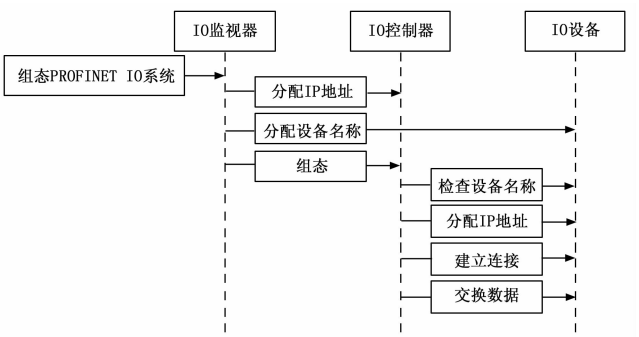

图 1 组态到系统启动

PROFINET 系统中，指定 DCP 协议分配设备名称和 IP 地址。每个 PROFINET 设备会根据 DNS 和 IP 地址分配唯一的设备名称。这个设备名称用于系统启动时的 PROFINET IO 设备寻址。设备名称和 IP 地址的成功分配是控制器和设备之间正常通信的前提。DCP 协议分配 IO 设备名称和 IP 地址的具体步骤如下。

1）IO 设备分配名称。

自动化系统启动时需要进行基于符号名称的设备地址解析，在 IO 控制器和 IO 设备真正建立连接之前，IO 设备就已经分配了名字，该名称是由 IO 监视器分配并保存在 IO 设备中。这个名称用作设备运行期间对它明确的标识。具体流程如图 2 所示。

2）IO 设备分配 IP 地址。

IO 控制器以设备名称作为寻址标准向 IO 设备发送请求，具有该名字的设备发送响应。ARP 请求在子网内以广播的方式发送，用于寻找对应 IP 地址，该 IP 地址的设备发

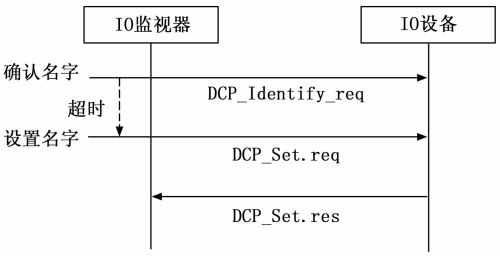

图 2 DCP 分配设备名称

送自己的 MAC 地址作为响应，以此来检查 IP 地址所对应的 MAC 地址。如果没有接收到 ARP 响应，则代表该 IP 地址的设备在此子网内不存在或者没有处在活动的状态。然后利用 DCP _ Set 设置 IP 地址并等待 IO 设备发送相应响应进行确认。具体步骤如图 3 所示。

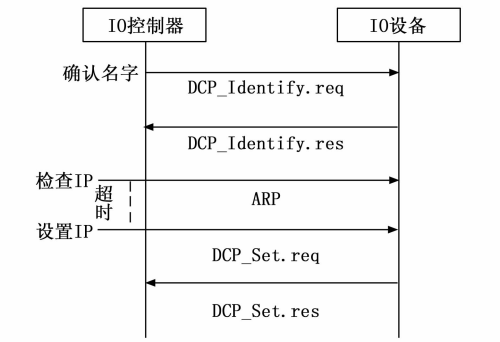

图 3 DCP 分配 IP 地址

PROFINET 网络中也可以选用动态主机配置协议（DHCP）对 IO 设备进行地址分配[14]。但 DHCP 在地址分配时，依赖一个可以使用的服务器，通常在自动化应用的范围外，服务器会因为各种原因关闭导致地址分配不成功。PROFINET 网络依赖于静态地址，而 DHCP 提供的是动态地址这使设备更换困难。MAC 地址通常与 DHCP 服务器进行绑定，如果需要更换一个设备，MAC 地址即发生改变，这会阻止新设备获取地址，并且使进程处于停滞状态，直到重新配置 DHCP 服务器。而使用 DCP 可以本地化管理地址，IP 地址不会过期，并且设备名称对每个设备来说都是唯一的，DCP 可以在不使用组态工具的情况下完成设备替换。

2.2.2 设备替换

DCP 和 LLDP 结合使用可以实现“无需组态工具的设备替换”[15]。LLDP 通过交换机端口交换邻居设备的信息，LLDP 帧中包括发送端口的端口 ID、发送设备的 MAC 地址或设备名称、目的端口等。现场设备接通后，相邻设备之间会周期性的发送 LLDP 帧用于交换设备信息。设备解析收到的 LLDP 报文，将端口 ID 和设备名称结合形成设备别名，这个别名将用于替换设备的寻址。

如果设备发生故障，则将停止与其邻居设备循环交换 LLDP 帧，这样可以检测出设备故障。当响应超时，邻居设备会删除故障设备的信息，控制器也将检测出这个设备不

再“可寻址”。更换设备后，控制器会循环的尝试用原设备名称和新设备建立联系。因为新设备的设备名称为空，所以不能成功建立联系，此时控制器使用 DCP _ Identify 服务，设置请求参数“Alias Name Of Station”，通过设备别名询问设备，新设备进行应答，控制器再通过 DCP _ Set 将新设备重新命名。

3　PROFINET IO 协议栈设计与实现

3.1　PROFINET IO 软件总体架构

PROFINET IO 提供了主要用于 IO 控制器和 IO 设备之间通信的服务和协议[16]。PROFINET IO 协议是由一组协议机器定义，协议栈总架构如图 4 所示[17]。图 3 描绘了 IO 设备内提供应用关系应用服务元素（AR ASE，Application Relationship Application Service Element）行为的协议机器及协议机器之间的交互。灰色框为 PROFINET IO 协议中主要的协议机器，箭头代表协议机器之间的主要交互。本文所实现的 PROFINET IO 协议栈以 RT 为最高通信标准实现数据收发和诊断报警功能。

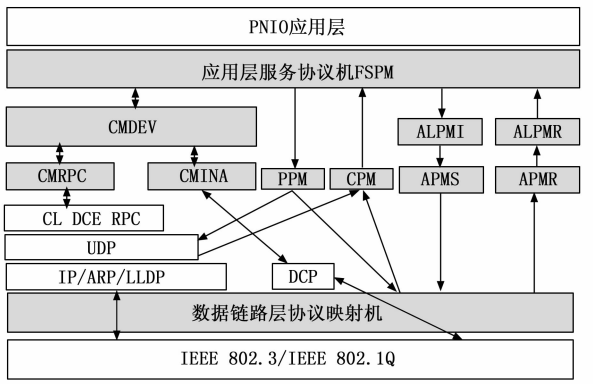

图 4　PROFINET IO 协议总架构

PROFINET IO 用户通过 IO 设备的应用层服务协议映射机（FSPM，FAL service protocol machines）发布或接收服务原语。IO 设备上下文协议机（CMDEV，context management protocol machine device）用于控制通信关系的建立和启动停止其他协议机。上下文管理 RPC 设备协议机（CMRPC，context management RPC protocol machine）中映射并封装了与远程过程调用协议（DCE RPC，distributed computing environment remote procedure call）相关的服务。管理 IP 和名称分配协议机器（CMINA，context management Ip and name assignment）主要用于处理站名和 IP 地址的分配，帮助处理 DCP _ Set 和 DCP _ Identify 请求和管理发送 DCP _ HELLO 请求。消费者协议机器（CPM，consumer protocol machine）处理接收循环数据，为传入的实时数据帧注册处理程序并监视传入的数据帧。提供者协议机器（PPM，provider protocol machine）处理数据的循环发送并初始化传输缓冲区。报警协议启动机器（ALPMI，alarm protocol machine initiator）用于触发警报。报警协议应答机器（ALPMR，alarm protocol machine responder）用于响应传入的警报。非循环协议接收机器（APMR，acyclic protocol machine receiver）用于接收传入的报警帧。非循环协议发送机器（APMS，acyclic protocol machine sender）用于向 IO 控制器发送报警以太网帧。

3.2　PROFINET IO 数据交换模块设计

在 PROFINET IO 通信中，发送数据调度是周期进行，网络中的每个设备的数据传输开始于循环 RT 数据，随后是传输非循环 RT 数据，例如警报数据，最后才传输 NRT 数据。

PROFINET 配置阶段，控制器告诉 IO 设备实际的模块和子模块。IO 设备会在 app _ plug _ dap（）函数中利用 app _ get _ module _ cfg（）函数、app _ get _ submodule _ cfg（）函数、app _ get _ parameter _ cfg 函数获取所需要模块、子模块、参数配置，再调用 pnet _ plug _ module（）、pnet _ plug _ submodule（）将特定模块插入特定的插槽，特定子插槽插入特定的子模块，最终实现 IO 设备的实际应用模块所有与组态信息相一致的模块信息。配置完成后，IO 设备和控制器之间连续交换循环数据。

PROFINET IO 设备和 IO 控制器之间的实际数据传输只能通过应用关系（AR，application relationship）建立通信，而一个 AR 中可以包含一个或者多个用于交换数据的通信关系（CR，communication relationship）。CR 在多个队列或缓冲区之间建立，用于交换不同的数据，循环 RT 数据使用 IO 数据 CR，配置和其他非循环数据在记录数据 CR 中交换，实时报警在报警 CR 中交换。控制器与 IO 设备之间的数据交互根据不同事件调用不同的回调函数进行处理，反复调用 Events _ Stata（）函数判断不同事件。PROFINET

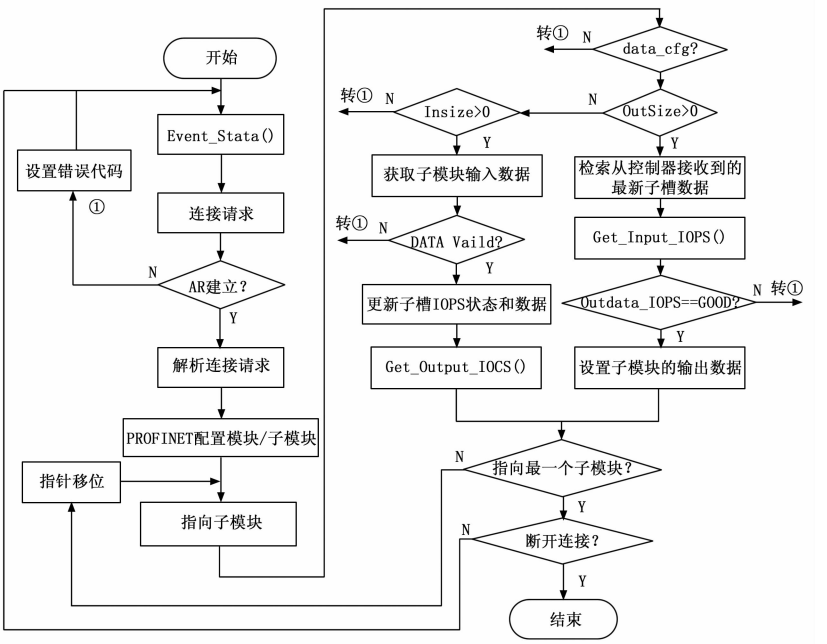

图 5　PROFINET IO 数据交互流程

数据交互流程如图 5 所示。

PROFINET IO 通信是按照生产者/消费者的模式进行，在循环数据交换时，不仅要包含数据单元还要包含两个数据状态属性，输入输出生产者状态（IOPS，input output provider status）和输入输出消费者状态（IOCS，input output consumer status）。IOCS 和 IOPS 状态指示收到或发送的数据是否有效。

当 IO 设备向 IO 控制器发送输入数据时，IO 设备即是生产者也是消费者。作为生产者时，Set _ Output _ Data（）函数发送过程数据并利用 Set _ Output _ IOPS（）函数为输入数据设置 IOPS 状态，然后通过输出数据 CR 将过程数据提供给 IO 控制器。作为消费者时，用 Set _ Intput _ IOCS（）函数向控制器报告自己之前接收到的数据是否可以使用。

当 IO 控制器向 IO 设备发送输出数据时，IO 设备调用 Get _ Intput _ Data（）函数获取输入缓冲区中数据，Get _ Intput _ IOPS（）函数读取指定模块的 IOPS 状态，Get _ Output _ IOCS（）函数获取 IO 控制器报告的接收到的数据是否可用信息。Data _ Status _ Changed（）函数通知应用程序接收到的数据状态已更改。循环数据交换过程如图 6 所示。

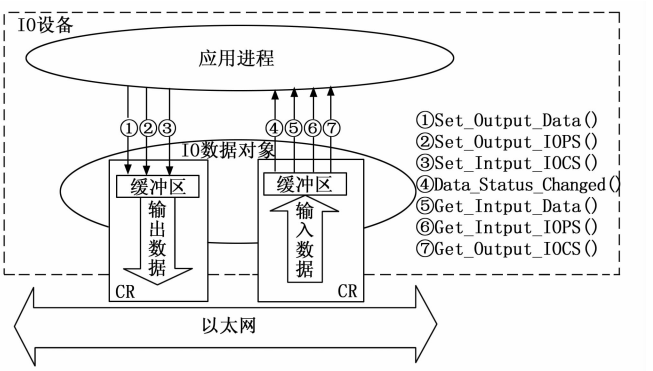

图 6 循环数据交互过程

4 DCP 协议程序实现

PROFINET IO 设备中集成的 DCP 协议实现程序主要包括处理 DCP _ Identify 请求程序、处理 DCP _ Set 请求程序、处理 DCP _ Get 请求程序、处理 DCP _ Hello 请求程序、闪烁信号灯、重置设置等。要求 IO 设备能够对控制器发出的 DCP 服务请求作出正确的响应。IO 控制器和 IO 设备上电后，IO 设备接收到来自控制器的数据包，先判断此包是否为 PROFINET 数据包，数据类型是否等于 0x8892。再将接收到的数据包解析，根据 frame ID 发送到不同处理程序。本节主要分析 DCP _ Identify 请求程序和处理 DCP _ Set 请求程序的具体实现过程。

4.1 DCP 中使用的选项和子选项

DCP 是一种通过不同过滤判据进行读取和写入设备参数及发现设备的协议。DCP 服务的实现是根据设备选项进行过滤判断再执行，IO 控制器中发送的请求命令中包含筛选条件，满足筛选条件的设备才会作出响应。因此协议中定义了很多选项和子选项，选项及子选项如表 1 所示。

表 1 DCP 用到的选项及子选项

选项	子选项
IP	IP 参数 MAC 地址
设备属性	设备 ID 设备角色详细信息 站点名、站点别名 设备选项、设备子选项
控制	开始、停止、信号 恢复出厂设置
制造商特定	制造商特定子选项 制造商 OUI 制造商特定的字符串
寻址所有设备	寻址所有设备子选项

4.2 处理 DCP _ Identify 请求程序

DCP _ Identify _ Req（）是处理传入的 DCP _ Identify 请求函数，是实现构造响应并发送响应的过程。请求中包含筛选条件，只有所有的条件都匹配的设备才发送响应。接收到请求中的 frame ID=0xfefe 时，表示接收到的是 DCP _ Identify 请求需要发送 DCP _ Identify 响应。

先获取本设备的 MAC 地址并给响应分配缓冲区，从接收到的请求帧中读取信息准备构造响应帧，其中响应帧中的目的地址就是请求帧中的源地址，响应帧中的源地址就是本设备的 MAC 地址。构造响应帧的报文头，从请求帧的报文头中修改所需要的内容，包括以太网报文头、PROFINET 报文头、DCP 报文头，DCP block 报文头。通过 DCP _ Get _ Req _（）函数来读取请求帧中 block 头中的选项和子选项。如果 Option 为 PF _ DCP _ OPT _ ALL 并且子选项为 PF _ DCP _ SUB _ ALL 时，则将变量 Identify _ all 赋值为 ture，表示请求帧是要寻址所有设备，不过滤此条信息，IO 设备也准备做出响应。根据读取请求帧中的 Option 的值来改变变量 match 和 ret 的值，当 ret=0 并且 match=true 时，PF _ dcp _ Get _ Req（）函数插入 Device _ Option 的元素个数个 block 构造响应帧。造响应帧的流程图如图 7 所示。

4.3 处理 DCP _ Set 请求程序

DCP _ Get _ Set（）是处理 DCP _ Set 请求程序函数，发送 DCP _ Set 响应的流程图如图 8 所示。使用 Get _ Device _ Macaddr（）函数检查请求帧中的目的地址与本设备的 MAC 地址是否相匹配，如果匹配则修改请求帧为构造响应帧做准备。包括在响应帧中插入 frame ID，设置以太网报头，并从请求帧中复制 DCP 头等。然后判断请求帧中的 Service _ ID 是否为 PF _ DCP _ SERVICE _ SET，在响应中插入块构造完成响应后发送响应，再通过 DCP _ Set _ Commit（）函数提交对 ip _ suite 的更改。

5 GSD 描述文件的设计

GSD 是由元素和属性构成，基于可扩展标记语言（XML，eXtensible markup language）的通用工作站标识语言（GSDML，general station description markup language）

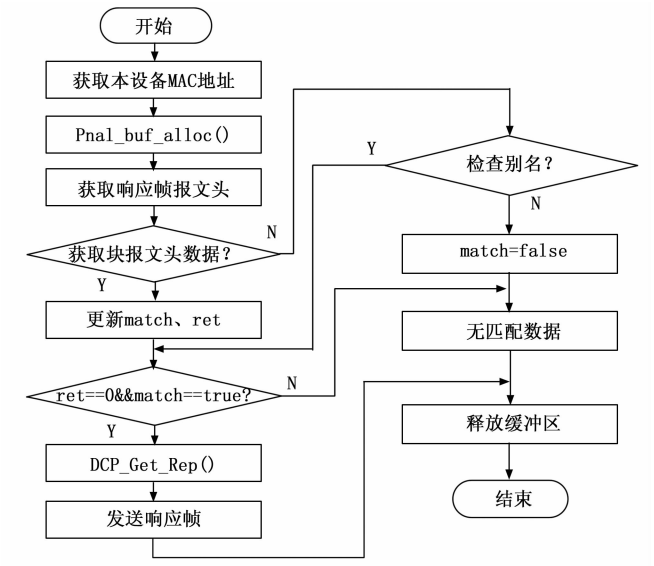

图 7　处理 DCP _ Identify 请求程序

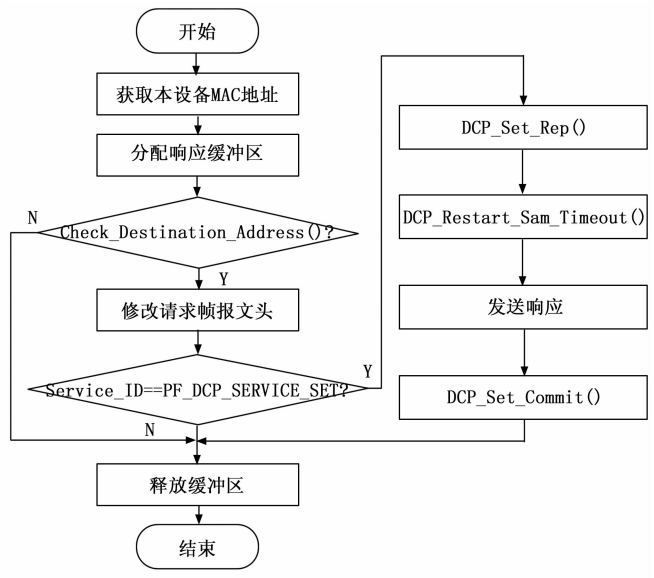

图 8　处理 DCP _ Set 请求程序

编写，用来描述 PROFINET 设备功能的文件[18]。该文件主要描述了可在 PROFINET IO 设备中使用的可插拔硬件模块类型，但设备实际要使用的硬件模块需要在工程工具中进行配置。配置好的组态信息下载到控制器中，控制器会告诉 IO 设备需要插入的 IO 模块的类型，只有 IO 设备中所插入的模块与组态信息中的配置模块一致才能控制器和 IO 设备才能组态成功。GSD 文件由＜ProfileHeader＞、＜ProfileBody＞两部分构成，＜ProfileHeader＞头里的内容一般是固定不变的，＜ProfileBody＞描述设备的信息。＜ProfileBody＞里的＜DeviceIdentity＞用于设置制造商 ID、设备 ID 和制造商的名称，＜DeviceFunction＞设置为“I/0”，＜ApplicationProcess＞主要用于处理通信设置。GSD 文件的结构图如图 9 所示。

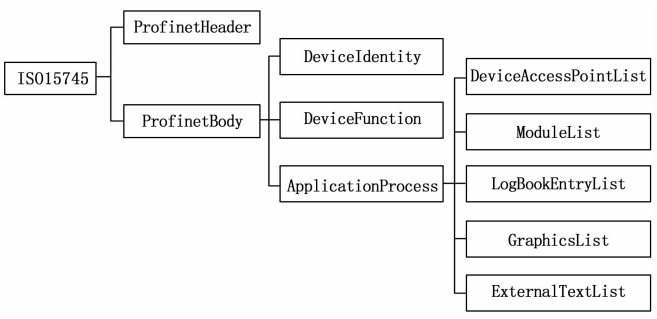

图 9　GSD 文件结构图

＜ApplicationProcess＞中的＜ModuleList＞描述 IO 设备中所有的模块的信息。图 10 是一字节输入一字节输出模块 GSD 文件描述。

图 10　GSD 文件

＜ModuleItem ID＝“IDM _ 32”ModuleIdentNumber＝“0x00000032”＞用来声明一个模块，控制器与 IO 设备通信时通过模块的 ID 来建立 AR 关系，所以每个模块的 ID 需是唯一；＜ModuleInfo＞用来描述模块名称和硬软件版本信息；一些子模块是模块固定不变的部分称之为虚拟子模块，＜VirtualSubmoduleList＞就是描述这些子模块的输入输出数据和记录数据。＜VirtualSubmoduleItem ID＝“IDSM _ 132”SubmoduleIdentNumber＝“0x0132”MayIssueProcess Alarm＝“true”＞用来声明一个子模块，其中“IDSM _ 132”用于外部文本列表中的赋值，“0x0132”为子模块的 ID。＜IOData＞是对输入输出数据的描述，Type＝“Unsigned8”表示输入的数据类型为 8 位无符号数；UseAsBits＝“true”表示可以在工程工具中显示每一位。＜BitDataItem＞用来命名各个位，最低有效位的偏移量为 0，之后偏移量加一。＜RecordDataList＞对子模块参数进行描述，其中 Index＝“123”利用 Index＝“123”的索引值可以找到对应的模块进行非实时数据的读写操作，Length＝“4”定义了参数长度为四字节。

6　系统测试

PROFINET 是在标准以太网的基础上精简协议栈，不

仅提高了实时性还使 PROFINET 的实现更加简单[19]。这种优化是在软件层面，因此主要要求硬件平台能够接受和发送以太网二层的数据帧就可以实现作为 IO 设备接入 PROFINET 网络中。本文选用 Raspberry Pi 3B＋（树莓派）作为硬件平台，它是基于 Linux 系统的 Cortex-A53 架构微型电脑主板，搭配千兆网卡 LAN7515，不仅可以实现数据帧的收发还可以实现 PC 的基本功能[20]。树莓派和 PLC 通过交换机连接，但此次测试采用的交换机不是镜像交换机，因此需要在树莓派上运行 Wireshark 软件来抓取硬件平台和 IO 控制器之间的数据交互。系统测试连接如图 11 所示。

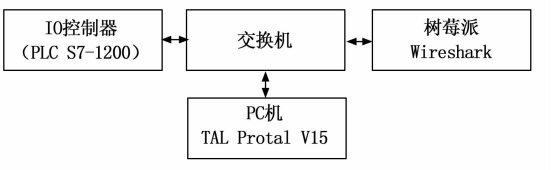

图 11 实验平台结构

6.1 从组态到运行系统

在工程工具 TAL Portal V15 中对 PROFINET IO 系统进行组态，将 IO 设备的 GSD 文件加载到工程工具中[21]。组态连接如图 12 所示。

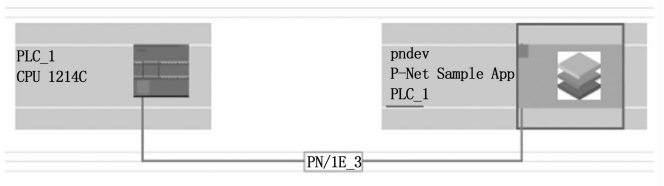

图 12 TAL Portal 组态

6.2 启动期间的数据交换

控制器上电后发送 DCP 广播消息，子网上的所有 IO 设备用其 MAC 地址应答。控制器向 IO 设备发送带有特定 MAC 地址的 DCP 消息，其中包含 IO 设备应使用的 IP 地址和站名。IO 设备相应地设置自己的 IP 地址和站名。然后，PLC 使用在 IP 上的 UDP 运行的 DCE/RPC 协议启动 IO 设备的实际配置。配置完成后，IO 设备和控制器之间连续交换循环数据。

树莓派上运行 Wireshark 对设备和控制器之间的通信进行抓包分析。在设备刚启动时，控制器先通过设备名称“pndev”寻址设备并通过 DCP _ Set 服务分配 IP 地址，ARP 确认过设备后开始建立连接。系统的初始化数据是 NRT 非实时数据，通过 IP 地址传输，其中 192.168.0.1 为 PLC 控制器的 IP 地址，192.168.0.50 是树莓派的 IP 地址。IO 控制器和 IO 设备之间的连接建立分为四步：第一步建立 AR 和 CR，控制器发送连接请求，树莓派对建立连接的请求进行响应，只有在能保证建立连接的时候才会发送响应，然后开始建立数据交互所需的 AR 和 CR。第二步传输参数，控制器发送写请求，树莓派进行响应后给 IO 设备传输参数，每个模块或子模块的参数都要进行传输。第三步传输结束，控制器发出“DControl”请求也就代表着控制器对参数的传输结束，树莓派进行响应后连接就完全建立起来了。第四步确认，树莓派在发送“CControl”请求对连接的建立进行确认，收到控制器的响应后就开始进行数据交换。Wireshark 在启动期间的抓包如图 13 所示。

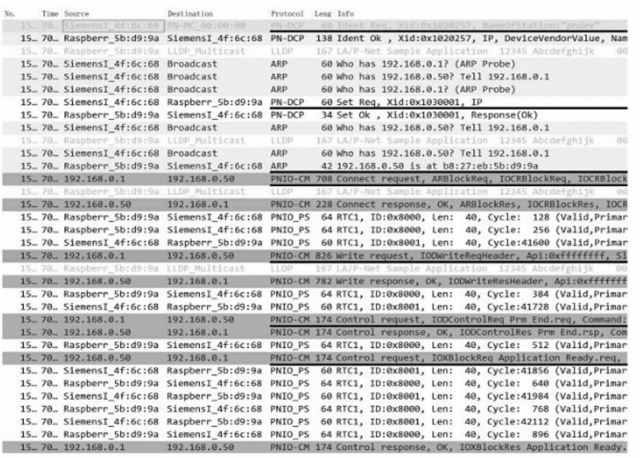

图 13 启动期间的数据交换

6.3 设备发送周期及抖动性分析

在工程工具中组态时设置 IO 设备的更新时间为 4 ms，利用 Wireshark 软件对系统数据交互进行抓包，组态工具中的更新时间如图 14 所示。将抓取到的数据进行筛选，筛选出 IO 设备发送给控制器的 RT 数据并导出为 CSV 文件。在 Excel 中处理数据，计算出相邻两个数据的发送时间差值得到更新时间，再计算相邻两个更新时间的差值得到更新时间抖动值。利用 Matlab 绘制更新时间折线图和抖动折线图，如图 15 所示发送周期在所设置的 4 ms 上下浮动，并且可以看到抖动值不超过 50 μs，所实现的 IO 设备符合 PROFINET RT 实时性的要求。

图 14 工程工具中的更新时间

7 结束语

PROFINET 是工业实时以太网络中的典型代表，在 PROFINET 应用广泛的今天，对 PROFINET 的研究应该更加深入和多面化。针对同类设备依赖特定芯片，成本较高，可移植性差的问题，通过软件设计在普通网卡下实现 PROFINET IO 的 RT 和 NRT 通信。为证明广泛性，在树莓派硬件平台上移植协议栈实现树莓派接入 PROFINET 网络。本文在实现通信功能的基础上对 PROFIENT 协议栈中

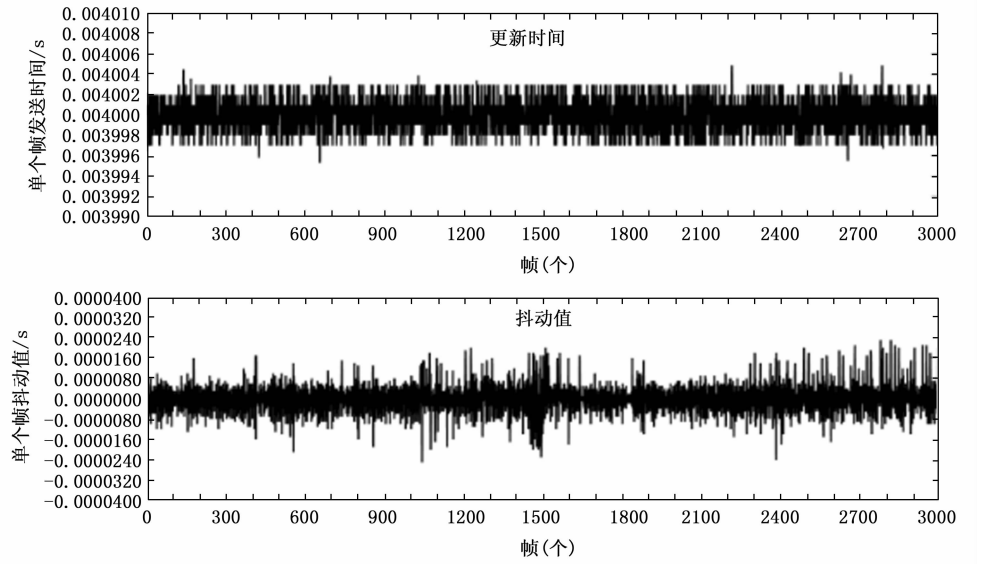

图 15　更新时间及抖动值

的 DCP 协议进行重点研究。阐述了 DCP 协议的功能、原理和在 PROFINET 网络中所起的作用，对 DCP 的软件程序进行了设计与实现。最后为所设计的 IO 设备搭建了一套测试系统，并完成硬件组态。验证了从组态到系统启动中 DCP 的作用和 IO 设备所实现的 RT 和 NRT 通信功能。通过在树莓派上运行 Wireshark 抓包软件对 IO 设备和控制器 PLC 建立通信前的流程和 DCP 的功能实现过程进行分析，对所抓取的数据进行了筛选分析，得出了 IO 设备更新时间的抖动值，设备的实时性及可靠性满足自动化生产控制中的需要，具有广泛的应用前景。本文对 PROFINET IO 通信的实现和对集成协议 DCP 的分析与研究，可以帮助从事相关方向的工程师掌握 PROFINET 技术的应用。

参考文献：

[1] DIAS A L，SESTITO G S，TURCATO A C，et al. Panorama，challenges and opportunities in PROFINET protocol research [J]. 2018 13th IEEE International Conference on Industry Applications，2018：186－193.

[2] 李中胜，乔　枫，杨志家，等. PROFINET IO-PROFIBUS PA 网关设计与实现方法 [J]. 计算机测量与控制，2015，207 (12)：4153－4155，4160.

[3] 张国栋，王有春. PROFINET 的实时性及其协议分析 [J]. 计算机测量与控制，2017，25 (3)：187－190.

[4] KLEINES H，DETERT S，DROCHNER M，et al. Performance Aspects of PROFINET IO [J]. IEEE Transactions on Nuclear Science，2008，55 (1)：290－294.

[5] 刘　柯，耿春明，夏继强. PROFINET RT 通信技术研究 [J]. 电气自动化，2018，40 (4)：40－42.

[6] 闫　菲，李　腾，韩　松，等. 基于 ERTEC200P 芯片的 PROFINET IO 工业以太网接口开发 [J]. 电气传动，2021，51 (4)：76－80.

[7] 徐建明，管意军. 基于 TPS-1 的 PROFINET 耦合器设计 [J]. 浙江工业大学学报，2021，49 (5)：473－481.

[8] 刘振杰. PROFINET 从站模块开发 [D]. 北京：北方工业大学，2016.

[9] FERRARI P，FLAMMINI A，VITTURI S. Performance analysis of PROFINET networks [J]. Computer Standards & Interfaces，2006：369－385.

[10] DUERKOP L，TRSEK H，JASPERNEITE J，et al. Towards autoconfiguration of industrial automation systems：A case study using PROFINET IO [C] // Proceedings of 2012 IEEE 17th International Conference on Emerging Technologies & Factory Automation. IEEE，2013：1－8.

[11] 李永生，杜文博. PROFINET IO 系统的实时技术架构分析 [J]. 自动化仪表，2021，42 (2)：33－39，44.

[12] 陈　曦. PROFINET IO 技术中的 DCP 协议 [J]. 中国仪器仪表，2016 (10)：27－30.

[13] 和淑芬，沈　勇，范雄涛，等. 深入解析 PROFINET IO 系统启动前寻址的实时报文 [J]. 电气技术，2018 (2)：96－101.

[14] profinet university [Z/OL]. https：//profinetuniversity.com/naming-addressing/ip-addresses/

[15] 陈　曦，张桂红. PROFINET IO 技术中的 LLDP 协议 [J]. 中国仪器仪表，2015 (12)：45－47.

[16] 周　侗，张哲睿，于　洋. 基于 STM32 的 PROFINET IO 协议栈框架的实现 [J]. 沈阳理工大学学报，2019 (3)：61－66.

[17] NEUMANN P，POSCHMANN A. Ethernet-based Real-Time communication with PROFINET IO [J]. WSEAS Transactions on Communications. 2005，4 (5)：235－245.

[18] DURKO L，IMTIAZ J，TRSEK，et al. Using OPC-UA for the autoconfiguration of real-time Ethernet systems [J]. IEEE International Conference on Industrial Informatics，2013：248－253.

[19] ROHR D，FELSER M，RENTSCHLER M. Simplifying the engineering of modular PROFINET IO devices [C] // Emerging Technologies & Factory Automation. IEEE，2011：1－4.