 DOI: https://doi.org/10.12345/dlynyqy.v3i11.33454

Embedded Device Protocol and Hardware Decoupled Architecture Based on Standardized Data Interfaces

Shucheng Huang[1] HaoWang[2] Yichang Huang[1] Yisi Su[1] Kun Zhang[1] Xiaodong Huang[1]

1. Tancy Instrument Group Co.,Ltd., Wenzhou, Zhejiang, 325800, China
2. Goldcard Smart Group Co.,Ltd Hangzhou Zhejiang 310018

Absrtact

Addressing issues in current embedded device protocol development such as tight coupling between protocol and hardware layers, high cross-platform adaptation costs, and maintenance difficulties, this paper proposes a hardware-decoupled architecture for embedded device protocols based on standardized data interfaces. Through an innovative three-tier design (protocol parsing layer, protocol data layer, and standardized data layer) with strictly defined standardized data interface specifications, this architecture achieves complete separation between protocol logic and hardware resources. Experimental results demonstrate that the architecture significantly improves code reusability, reduces adaptation costs and cycles for new hardware platforms, and enhances system stability and maintainability. Particularly suitable for metering devices such as water and gas meters requiring multi-hardware platform adaptation, this architecture provides an effective solution for addressing the diversified and rapid iteration needs of embedded devices in the IoT era.

Keywords

Embedded Devices; Protocol Architecture; Hardware Decoupling; Standardized Data Interfaces; Metering Devices

基于标准数据接口的嵌入式设备协议与硬件解耦架构

黄书成[1] 王浩[2] 黄益昌[1] 苏义思[1] 章坤[1] 黄孝栋[1]

1. 天信仪表集团有限公司，中国 · 浙江 温州 325800
2. 金卡智能集团有限公司，中国 · 浙江 杭州 310018

摘 要

针对当前嵌入式设备协议开发中存在的协议层与硬件层强耦合、跨平台适配成本高、维护困难等问题，本文提出了一种基于标准数据接口的嵌入式设备协议与硬件解耦架构。该架构通过创新的三层设计（协议解析层、协议数据层、标准数据层），定义严格的标准数据接口规范，实现了协议逻辑与硬件资源的彻底分离。实验结果表明，该架构显著提高了代码复用率，降低了新硬件平台的适配成本和周期，增强了系统的稳定性和可维护性。该架构特别适用于水表、气表等计量设备中多硬件平台适配的协议架构设计，为物联网时代嵌入式设备多样化、快速迭代的发展需求提供了有效的解决方案。

关键词

嵌入式设备；协议架构；硬件解耦；标准数据接口；计量设备

1 引言

嵌入式设备，诸如水表、气表等计量装置，在工业自动化、智慧城市及物联网等领域中的应用日益广泛。随着技术迭代加速与应用场景的不断拓展，这类设备在硬件平台升级、功能扩展与协议更新等方面面临持续挑战。当前，嵌入式设备协议开发普遍采用的“协议层 - 协议数据层”两层架构，易导致协议层与硬件层紧密耦合、跨平台适配成本高昂、数据表示不一致、系统维护复杂以及扩展性受限等问题。

在传统两层架构中，协议数据层直接绑定并操作特定硬件资源，协议层与硬件层存在强耦合关系，协议逻辑直接依赖于具体的硬件特性。当面临不同硬件平台或基线变更时，需要对协议层代码进行大量重写或修改，无法实现协议逻辑的跨平台复用。这种架构在实际应用中导致了设备厂商在硬件升级、平台迁移时面临巨大的开发成本和维护挑战。

为了解决这些问题，本文提出了一种基于标准数据接口的嵌入式设备协议与硬件解耦架构。该架构通过创新的三层设计模式，在传统两层架构的基础上增加了独立的标准数据层，并定义了严格的标准数据接口规范，实现了协议逻辑与硬件资源的彻底解耦。本文详细阐述了该架构的设计思想、各层功能、标准数据接口定义以及实验验证结果，展示了该架构在提高代码复用率、降低适配成本、增强系统稳定性等方面的显著优势。

【作者简介】黄书成（1993–），男，中国浙江温州人，本科，工程师，燃气仪表。

2 系统架构设计

2.1 整体架构概述

本文提出的基于标准数据接口的嵌入式设备协议与硬件解耦架构采用创新的三层设计模式，整体架构如图 1 所示。该架构在传统 " 协议层 - 协议数据层 " 两层架构的基础上，增加了独立的 " 标准数据层 "，通过标准数据接口实现了协议逻辑与硬件资源的双向解耦 [1]。

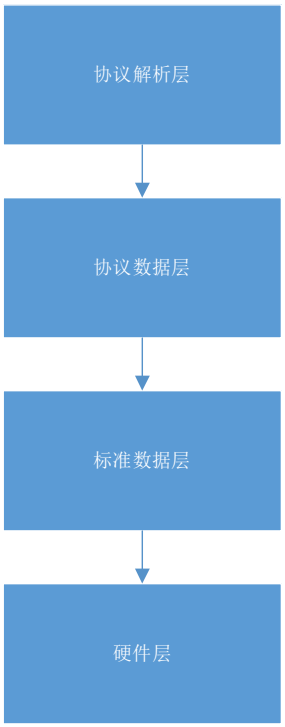

图 1 基于标准数据接口的嵌入式设备协议与硬件解耦架构图

该架构的核心设计思想是：协议解析层仅依赖协议数据层，协议数据层仅依赖标准数据层，标准数据层负责与具体硬件交互。通过这种分层解耦模式，实现了协议逻辑与硬件资源的彻底分离 [2]。

2.2 各层功能设计

2.2.1 协议解析层

协议解析层是架构的最上层，主要负责处理协议格式相关的逻辑，实现与外部通信系统的数据交互。其主要功能包括：

处理各类协议指令的解析与打包，包括指令的加密、解密、校验等安全相关操作；

严格遵循 " 协议与数据分离 " 原则，处理过程中仅涉及协议格式的操作，不涉及具体的业务逻辑处理；

支持客户协议数据对标准协议数据的继承和扩展，当客户协议数据结构与标准协议数据兼容时，可直接调用标准协议数据接口；

根据不同的上报类型控制上报帧的格式和内容，处理特殊格式帧（如注册帧）与标准协议格式不一致的情况；

负责下行指令的协议框架解析，包括帧头、帧尾、校验码等字段的验证和提取 [3]；

负责上行指令的协议框架打包，按照协议规范组装完整的数据包，确保数据传输的正确性和完整性。

2.2.2 协议数据层

协议数据层是连接协议解析层与标准数据层的中间层，主要负责协议数据与标准数据之间的转换和业务逻辑处理。其主要功能包括：

实现设备参数与协议参数格式之间的转换（如数据类型转换、单位转换、精度处理等），确保数据在不同层之间的正确传递；

根据设备的业务特性，在参数操作过程中执行相应的业务逻辑（如配置累积量时触发液晶显示、参数修改时记录操作日志等）；

作为客户定制协议与标准数据接口之间的桥梁，所有客户定制协议指令中的相关记录、参数、数据都通过该层接口进行统一的读写操作；

维护协议特定的数据结构和状态，确保协议逻辑的一致性和完整性；

提供统一的错误处理机制，对数据访问和业务操作中的异常情况进行规范化处理和反馈。

2.2.3 标准数据层

标准数据层是架构的最底层，直接与硬件交互，是实现协议与硬件解耦的核心层。其主要功能包括：

定义统一的标准数据接口，屏蔽不同硬件平台的实现差异，提供一致的参数访问方式；

实现设备自身参数与标准数据格式之间的转换，处理不同硬件平台的数据表示差异（如字节序、数据精度等）；

通过标准化的 Get/Set 语义化接口操作设备硬件资源（如寄存器读写、传感器数据采集、执行器控制等）；

负责硬件资源的抽象和管理，包括资源的初始化、配置、状态监控等；

提供硬件平台适配能力，针对不同的硬件平台，只需实现标准数据接口，无需修改上层协议逻辑。

2.3 数据流转过程

在本架构中，数据流转遵循严格的分层传递原则。当需要上行数据时，通信管理组件调用协议解析层的接口获取上行数据；当接收到下行指令时，通信管理组件处理客户协议下行数据，并通过三层架构逐层传递和处理指令，最终由标准数据层执行硬件操作，处理成功后将结果逐层返回并生成应答数据。

具体数据流转过程如下：

上行数据流程：硬件层数据采集 → 标准数据层数据转换 → 协议数据层业务处理 → 协议解析层协议打包 → 外部通信系统；

下行指令流程：外部通信系统 → 协议解析层协议解析 → 协议数据层业务处理 → 标准数据层硬件操作 → 硬件层执行指令。

这种分层的数据流转机制确保了协议逻辑与硬件资源的彻底分离，同时也提高了系统的可维护性和扩展性 [4]。

3 标准数据接口定义

3.1 接口设计原则

标准数据接口是本文实现协议与硬件解耦的核心 [5]。为了确保接口的通用性和可扩展性，接口设计遵循以下原则：

统一性原则：定义统一的参数访问方式和数据格式，确保协议层能够以一致的方式访问不同硬件平台的数据和资源；

语义化原则：采用语义明确的 Get/Set 操作，使接口的使用更加直观和易于理解；

可扩展性原则：接口设计应具备良好的可扩展性，能够方便地添加新的参数和功能；

安全性原则：接口应提供必要的权限控制和数据校验机制，确保数据访问的安全性；

高效性原则：接口实现应高效，避免不必要的性能开销。

3.2 接口规范

标准数据接口通过参数 ID、参数名称、单位、字节长度、数据类型和访问类型等属性来定义设备参数。以下为规范的数据接口定义示例：

参数 ID	采集数据	单位	字节长度	数据类型	访问类型
E_ST_DATETIME	时间		6	hex	读写
E_ST_T	温度	℃	4	float32_t	只读
E_ST_P	压力	kPa	4	float32_t	只读
E_ST_QB	标况瞬时流量	Nm3/h	4	float32_t	只读
E_ST_QM	工况瞬时流量	m3/h	4	float32_t	只读
E_ST_VB	标况累计流量	Nm3	8	float64_t	只读
E_ST_VM	工况累计流量	m3	8	float64_t	只读
E_ST_QE	瞬时能量	MJ/h	4	float32_t	只读
E_ST_E	累计能量	MJ	8	float64_t	只读
E_ST_VALVE_STA	阀门状态	0：开 1：普关 2：强关 3：开关阀中	1	uint8_t	只读
E_ST_MAIN_BAT_VOL	电池电压	V	4	float32_t	只读
...					

标准数据接口通过统一的参数 ID、数据类型、访问方式等规范，确保了协议层能够以一致的方式访问不同硬件平台的数据和资源。同时，接口的定义也考虑了不同硬件平台的数据表示差异，通过数据转换机制确保了数据在不同层之间的正确传递。

3.3 接口实现机制

标准数据接口的实现采用了基于函数指针的注册机制。具体来说，每个硬件平台需要实现一组标准的接口函数，并通过注册机制将这些函数与参数 ID 关联起来。当上层调用标准数据接口时，系统根据参数 ID 查找对应的实现函数，并调用该函数执行具体的硬件操作。

这种基于函数指针的注册机制具有以下优势：

灵活性：可以根据不同的硬件平台注册不同的接口实现函数；

可扩展性：可以方便地添加新的参数和接口实现；

高效性：通过函数指针直接调用，避免了不必要的性能开销；

解耦性：上层协议逻辑只需与接口定义交互，无需了解具体的硬件实现细节 [6]。

4 实验验证与分析

4.1 实验目的

为了验证本文提出的基于标准数据接口的嵌入式设备协议与硬件解耦架构的有效性和优势，我们进行了一系列实验。实验的主要目的包括：

验证该架构在不同硬件平台上的适配能力；

评估该架构在代码复用率、开发效率、维护成本等方面的优势；

测试该架构在实际应用中的稳定性和可靠性。

4.2 实验环境与方法

4.2.1 实验环境

我们选择了三种不同的硬件平台作为实验对象，分别记为平台 A、平台 B 和平台 C。这三种平台在 MCU 型号、传感器类型、寄存器映射等方面存在显著差异，具有较好的代表性。

实验中使用的软件环境包括：

集成开发环境：Keil uVision5；

编程语言：C 语言；

调试工具：J-Link 调试器。

4.2.2 实验方法

我们分别在三种硬件平台上实现了基于本文架构的协

议系统，并与传统两层架构的实现进行了对比分析。具体实验方法如下：

在平台 A 上实现基于本文架构的协议系统，包括协议解析层、协议数据层和标准数据层；

将平台 A 上的协议解析层和协议数据层代码直接移植到平台B和平台C上，仅为平台B和平台C实现标准数据层；

在三种平台上分别实现基于传统两层架构的协议系统；

对比分析两种架构在代码复用率、开发周期、维护成本等方面的差异；

对两种架构的系统进行功能测试和性能测试，评估其稳定性和可靠性。

4.3 实验结果与分析

4.3.1 代码复用率分析

实验结果表明，本文提出的架构在代码复用率方面具有显著优势。具体数据如下：

架构类型	平台 A 代码量 (行)	平台 B 代码量 (行)	平台 C 代码量 (行)	代码复用率 (%)
传统两层架构	5000	4800	5200	0
本文架构	5500	1200	1300	78.2

从表中可以看出，传统两层架构在不同平台上的代码复用率为 0，每个平台都需要重新开发几乎所有的协议代码。而本文架构的代码复用率达到了 78.2%，平台 B 和平台 C 只需实现标准数据层的代码，极大地减少了开发工作量。

4.3.2 开发周期对比

在开发周期方面，本文架构也表现出了明显的优势。具体数据如下：

架构类型	平台 A 开发周期 (天)	平台 B 开发周期 (天)	平台 C 开发周期 (天)	总开发周期 (天)
传统两层架构	30	28	32	90
本文架构	35	8	9	52

从表中可以看出，虽然本文架构在平台 A 上的开发周期略长于传统两层架构（主要是因为需要设计和实现标准数据层），但在平台 B 和平台 C 上的开发周期显著缩短。总体而言，本文架构的总开发周期比传统两层架构缩短了 42.2%，大大提高了开发效率。

4.3.3 系统稳定性和可靠性测试

我们对两种架构的系统进行了为期 3 个月的稳定性和可靠性测试。测试结果表明，本文架构的系统在测试期间未出现任何因协议与硬件耦合导致的故障，而传统两层架构的系统出现了 3 次因硬件平台差异导致的协议功能异常。

此外，在系统维护方面，本文架构的优势也非常明显。当硬件平台出现故障或需要升级时，只需维护和修改标准数据层，而协议层和协议数据层无需修改，大大降低了维护成本和系统性风险。

5 结语

本文提出了一种基于标准数据接口的嵌入式设备协议与硬件解耦架构。该架构通过引入标准数据层及其接口规范，构建了清晰的三层模型，有效隔离了协议逻辑与硬件资源。 实验验证表明，该架构展现出多方面的显著优势。它不仅实现了协议逻辑代码在不同硬件平台间的高度复用，极大提升了代码复用率，还显著降低了新增硬件平台的适配成本与开发周期。协议与硬件的彻底解耦，硬件变更或升级的影响被限制在标准数据层内，降低了系统性风险。同时，标准化的数据接口确保了跨平台数据表示的一致性，而明确的分层职责与清晰的边界设计，则赋予了系统良好的扩展能力与更高的整体稳定性。

综上所述，本研究为解决嵌入式设备协议与硬件耦合这一传统难题提供了一种有效方案。该架构特别契合物联网时代下嵌入式设备多样化、快速迭代的发展趋势，在水表、气表等计量设备领域已显现出良好的应用价值，并具备向更广泛嵌入式应用领域推广的潜力。

参考文献

[1] 杨旭，周维，徐彬，等.NFV网络云智能网卡关键技术方案及引入策略[J]. 电信工程技术与标准化, 2023(10).

[2] 张亚聪. 软件化雷达应用软件的设计与实现[D]. 西安: 西安电子科技大学, 2025.

[3] 李通. 基于全可编程SoC的飞机线缆自动检测系统的设计与实现[D]. 西安: 西安电子科技大学, 2025.

[4] 王树梅. 基于PLC的病房智能呼叫系统研究与设计[D]. 南京: 南京理工大学, 2014.

[5] 邢卓异，朱舜杰，黄晓峰，等.软件定义航天器系统架构设计[J]. 航天器工程, 2021(10).

[6] 胡英男，冒亚明.一个基于SOA的石油化工HSE信息系统的设计[J]. 石油化工安全环保技术, 2010(6).