**即插即用demo总体说明**

创建时间： 2021/04/24 19：30

作者：[1099023759@qq.com](mailto:1099023759@qq.com)

# 环境配置

Goland、edgex-go、rt内核、Igh Ethercat主站、edgex的appservice-sdk-go

均在Tower上有文档。

# 前置知识

IO-Link，Igh EtherCAT，OPC UA，Edgex设备微服务以及APPservice，JS。

IO-Link：本工程仅仅利用Ethercat主站配置IO-Link主站为唤醒模式，IO-Link主站和从站之间将建立通信。

以上知识不在本文件里详细描述，请从三份可执行文件的源码入手开始。

# 网络拓扑图

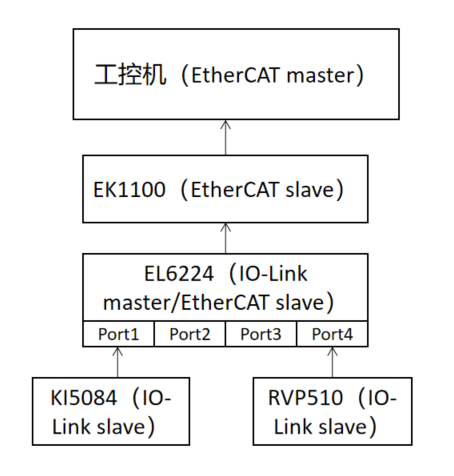


图1-网络拓扑图

# 软件架构

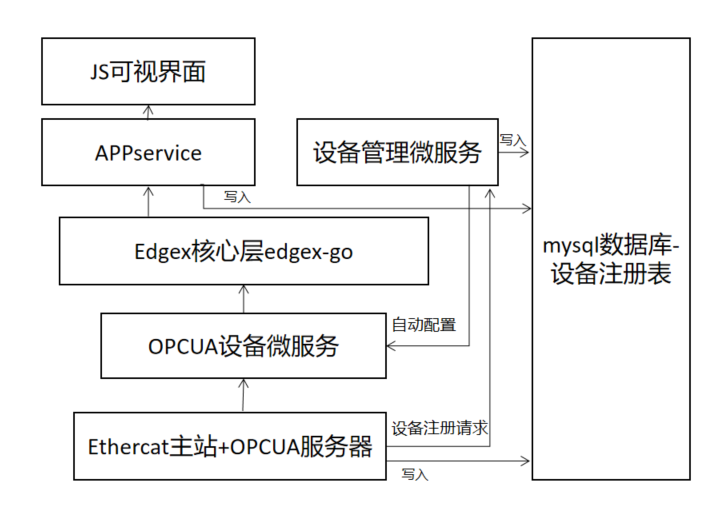


图2-软件架构

# Plug\_and\_play文件包目录树

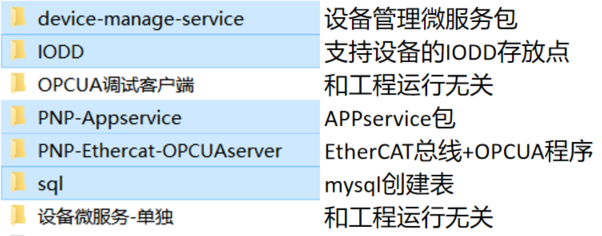


图3-Plug\_and\_play文件包目录树

# 文件及其说明

项目的根目录是Plug\_and\_play，其中可执行文件作用如下表：

表1-即插即用可执行文件说明

|  |  |  |
| --- | --- | --- |
| 表名 | 角色 | 作用 |
| PNP-Ethercat-OPCUAServer/EL6224 | Ethercat总线+OPCUA服务器 | Monitor线程：监控IO-Link端口设备插入，加载OPCUA信息模型，配置EtherCAT  Ethercat\_thread线程：EtherCAT总线数据读取  Main线程：运行OPCUA服务器 |
| Device-manage-service/device-manage-service | 设备管理模块 | 接口1：IODD解析  接口2：接受设备edgex注册请求，配置toml和yaml，启动设备微服务。 |
| PNP-Appservice/appservice | 数据导出+可视化组件 | 线程1：过滤Edgex数据流，选择注册的IO-Link设备数据可视化。  向js界面提供：  接口1：n个注册设备的设备id  接口2：n个IO-Link设备的过程数据 |

以上三个文件细节，请看源码完整注释。

# 演示即插即用流程

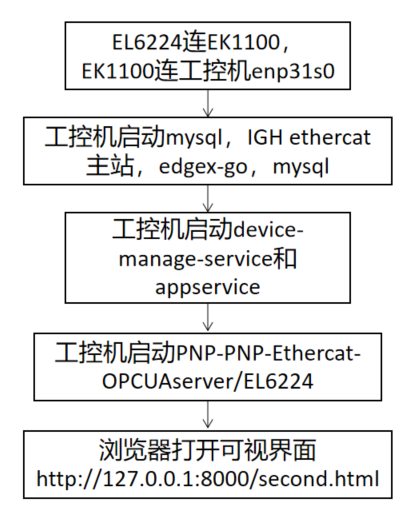


图4-即插即用运行流程

具体运行指令可以参考目录下的readme.txt文件。

注意先在EL6224的port4上接入RVP510，port1不接入KI5084。

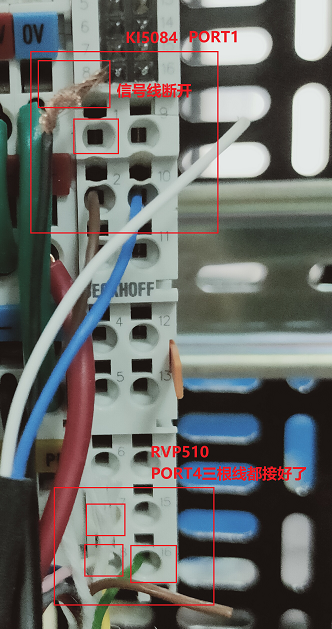


图5-EL6224与IO-Link Slave接线图

至此运行起来了程序环境。此时EtherCAT是OP状态。RVP5084已经被自动化配置，在可视界面可以看到RVP5084的四个过程数据。



图6-JS可视界面

在EL6224（IO-Link主站）上的port1插口1号插入KI5084的信号线，本程序系统，将自动对其进行配置，在可视界面可以看到新增了KI5084的两个过程数据。



图7-JS可视界面-新设备388插入

即实现了设备的即插即用。

# 数据库表

为了实现设备即插即用系统中几个微服务的协作，设计了一张数据库表，如下。

CREATE TABLE IO\_Link\_Device(

`device\_id` BIGINT NOT NUll AUTO\_INCREMENT COMMENT 'IO-Link设备id',

`deviceservice\_create\_flag` int NOT NULL COMMENT '设备微服务开启标志',

`appservice\_monitor\_flag` int NOT NULL COMMENT 'APP微服务监控设备标志',

`js\_show\_flag` int NOT NULL COMMENT 'JS可视化标志',

`io\_link\_port` int NOT NULL COMMENT 'IO-Link主站Port',

`edgex\_port` int NOT NULL COMMENT '设备微服务Port',

`create\_time` TIMESTAMP NOT NULL DEFAULT CURRENT\_TIMESTAMP COMMENT '创建时间',

PRIMARY KEY (device\_id)

)ENGINE=INNODB AUTO\_INCREMENT=1000 DEFAULT CHARSET=utf8 COMMENT='IO-Link设备表';

设备插入时，总线程序将开启一行，设置设备id和创建时间，IO-Link主站。

设备管理微服务获取注册请求，设置设备微服务标志，设备微服务port。

APPservice周期性查询数据库表，获得没被APP监控的设备id，APP微服务监控设备标开启。

# 程序交互时序图

以行时间递增的时序图。

表2-程序交互时序图

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 程序外部 | EL6224内部 | EL6224与device-manage-service之间 | Device-manage-service | Appservice | Appservie与js之间 | JS界面 | Mysql表 |
|  | 启动monitor线程 |  |  |  |  |  |  |
|  | 初始化ethercat主站 |  |  |  |  |  |  |
|  | 启动ethercat\_thread线程(OP) |  |  |  |  |  |  |
|  | 启动OPCUA服务器 |  |  |  |  |  |  |
| 设备插入 |  |  |  |  |  |  |  |
|  | Monitor线程提取出设备id |  |  |  |  |  |  |
|  | monitor线程关闭ethercat\_thread线程（PREOP） |  |  |  |  |  |  |
|  | Monitor线程 | POST Deviceid 过程数据OPCUA节点---> |  |  |  |  |  |
|  | 数据库表插入设备id为主键的一行 |  | 解析IODD |  |  |  | Deviceid |
|  |  |  | 1.配置toml、选择yaml。  2.以过程数据OPCUA节点为参数，启动设备微服务。  3.数据库表设备id行置deviceservice标志为1。 | 1.查mysql表，启动对设备名的监控（选择edgex数据流）。  2.Mysql表APPservice标志位置1 |  |  | Deviceid和deviceservice标志位1 |
|  |  |  |  |  |  |  | Deviceid、deviceservice标志位1和APPservice标志位1 |
|  |  |  |  | 得到IO-Link设备的数据流 |  |  |  |
|  |  |  |  | 长度n的双向链表存储n个IO-Link设备的数据流 |  |  |  |
|  |  |  |  |  | <-- 获取注册设备id |  |  |
|  |  |  |  |  |  | 可视化设备窗口 |  |
|  |  |  |  |  | <-- 获取注册设备过程数据流 |  |  |
|  |  |  |  |  |  | 设备窗口呈现过程数据 |  |