PVMesh CV主控机的Python程序架构设计

2016-10-20 吕海峰

1. Preface

CV在PVMesh体系中, 相当于”集中器”在抄表系统中的位置的设备. 上接主控云, 下面管理现场的诸多SS, 负责规划网络, 定时读取SS状态, 响应SS的报警信息等等.

CV应该有如下功能接口:

\* 收集现场所有的模块UID信息;

\* 指定一个模块的UID, 对其Diag;

\* 指定一个模块的UID, 对其配置域ID, VID, GID;

\* 指定UID或VID, 做任何设置, 抄读;

\* 将收集的信息保存到数据库, 数据库应按日新建, 不应在一个大数据库文件里;

1. 需求

\* 需要现场长期运行, 按嵌入式设计方法学设计, 不能随便退出

1. 功能模块设计

|  |  |  |  |
| --- | --- | --- | --- |
| 对象 | 功能需求 | 特殊要求 |  |
| Interface() | \* 封装串口对象;  \* 向串口发送数据;  \* 使用独立线程接收串口返回的数据, 将返回的完整数据包放在queue中 | 任何时候退出都要保证串口IO关闭 |  |
| CV\_Protocol() | \* |  |  |
| Powermesh() |  |  |  |
|  |  |  |  |

1. 对象设计
   1. 总论 Pandect

现场总控单元(FPU - Field Processing Unit , 一般是一台运行linux + python的树莓派或工控机)有多个tty com口, 每个com口至多通过RS485连接一个协议转换器CV.

CV负责将FPU com口的通信数据包转换成PLC信号发送到电力线上, 同时将PLC上接收的数据解调发送到FPU的com口上.

设计时, FPU的每个com口和对应CV共同看成一个独立的对象Powermesh实例, 有多个com连接多个cv就建立多个powermesh实例, 每个powermesh实例都包括以下接口:

\* diag(u8 uid[6], u8 rate, bool scan=True): //指定目标uid, 对其进行链路通信特性诊断

\* explore():

\* ptp\_send();

* 1. 类, 方法设计

class Phy ():

{

u8 xmode; //频率, 速率, default: 0x10

bool scan; // default: False

bool srf; // default: False, if true, ac\_update=False, psdu只发前两字节;

bool ac\_update; //携带相位信息, default:False

u32 delay; //延迟毫秒, default:20

str psdu[]; //发送的物理帧

def send():

// 控制CV发送数据包

}

class Dll(Phy):

{

u8 dest[6];

u8 src[6]; //初始化时通过com口获取cv的uid

static u8 index; //类变量

def frame(target\_uid[6])

}