###### 整流柜和桩端功率分配之间的解释：

物理模型是整流柜内部有多个模块，每个模块最大输出功率30kw；柜可以控制将模块组的一个或者几个模块分配给某个桩端（可以认为是接线的通断）；

功率需求的实现是通过“模块之间的组合”+“每个模块在0~30kw之间调整自己的功率输出（给定电压，变化电流）”；比如某个桩需要40kw，那么功率供给方案一般是分2个模块给该桩端，然后将每个模块的的功率输出调整为20kw。

车端需求在充电过程中是动态变化的（SOC：效率较高的充电曲线），车端需求一般包含电压和电流，动态变化的是电流需求（SOC充电曲线，桩端在接收报文不断动态满足的也是电流需求）；车端需求和车端电池需求一开始一般是不一样的，比如车端需求400V，电池可能是390V，实际测的也是这个电压，之后这个电压会最终在400V附近；

车端充电过程的电流需求是通过报文告诉整流柜，整流柜计算处理并调整分配和输出；

除了SOC充电曲线，还有上面提到的模块曲线（刚才提到的模块电压电流曲线，图中的电流是“该电压下的模块可以输出最大电流，然后在该电压可以调节出不同的电流大小，如果电流需求超过最大电流，那么就要增加模块，因为电压需求在充电过程中是不变的”）；

**整流柜**：

1. 与桩端进行充电报文交互（当然还有心跳报文，升级报文等等）；
2. 向模块发出电流需求并与模块交互（了解模块可以提供的最大电流和电压）；
3. 控制模块与桩端之间的物理连接（给各个桩端进行模块分配）；
4. 运行功率分配算法：结合各种资源进行计算分析得出分配方案，告诉车端自己可以提供的功率（电压定，主要是可以输出的电流大小），车端认可，部署实施分配方案；

与桩端的接口（以太网，代码是zmq，函数集和chgctrl差不多）；与模块的接口（对应的代码是？函数集合是？）；与“模块分配”单元的接口（对应的代码是？）

“通过debus和其他进程交互（如web，ocpp），直接调用底层封装好的驱动接口”，“维护那些全局变量？”进程自己内部的线程之间的通信（全局变量）；**如何理清一个进程的代码架构并清晰的将其记录下来**？

1. 多看几遍；
2. 设计更好的看代码思路（层级接口）；
3. 把上层逻辑理清楚，别被底层代码拖住（感兴趣或者有需要了才去看下一层的实现，**“先相信下层代码”,着重看上层的业务逻辑，理解甚至发现其错误**）；
4. 很多代码是共用，花时间看，它们为什么是共用的；

###### 总结

1. 整流柜可以控制模块和桩端之间的接线通断，决定桩端可以连那几个模块（一个模块同一时间只能连一个桩端）；
2. 整流柜可以给模块发指令，控制模块的输出电流和输出电压：
   1. 模块有一个电压电流曲线，该曲线受限于模块的功率，比如30kw时，要求模块输出1000V电压，那么模块可输出电流范围是0~30A，具体的输出以及控制指令；
   2. 当电流不够时（一开始就计算好够不够），整流柜会再让一个模块连接终端，然后调整相应的电流需求指令；
   3. 思想主要是：整流柜收到桩端的充电电压和充电电流需求（充电电压在充电过程是不变的），然后根据当前模块的空闲情况以及模块的实际情况（模块电压电流输出曲线不是一成不变的，可能会受外界温度等影响会稍有浮动），决定“连接那几个模块以及各模块的输出电压和输出电流是多少”；
   4. 充电过程中，汽车的电流需求是变化的（SOC曲线），但电压不变，所以整流柜也会跟着调整各模块的电流输出以满足车端需求（一开始的模块分配的根据是车端最大功率即满功率充电时的电压电流需求，充电开始时车端会发送一个最大允许电流）；
3. SOC充电曲线：充电开始一般是接近满功率输出，之后慢慢下降，快充满时（60%~90%）出现拐点，下降更快直至充满（类似于往杯里倒水）：所以一开始的充电电流是最大的，之后会越来越小；

###### 问题

###### 临时

typedef struct

{

unsigned int gunId; // 枪号

unsigned int volNeed; // lhm:应该是实时的电压需求

unsigned int curNeed; // lhm:应该是实时的电流需求

unsigned int pwrNeed; // lhm:应该是实时的功率需求

unsigned long startTime; // 开始充电时间

}CHG\_POLICY\_NEED\_STRUCT;

typedef struct{

unsigned char gunId;

unsigned char gunUsed; // 1-已使用 0-未使用

int pwrMax; // 最大提供的功率

CHG\_POLICY\_NEED\_STRUCT gun;

}GUN\_STRUCT;

typedef struct

{

int gunId; // 枪ID

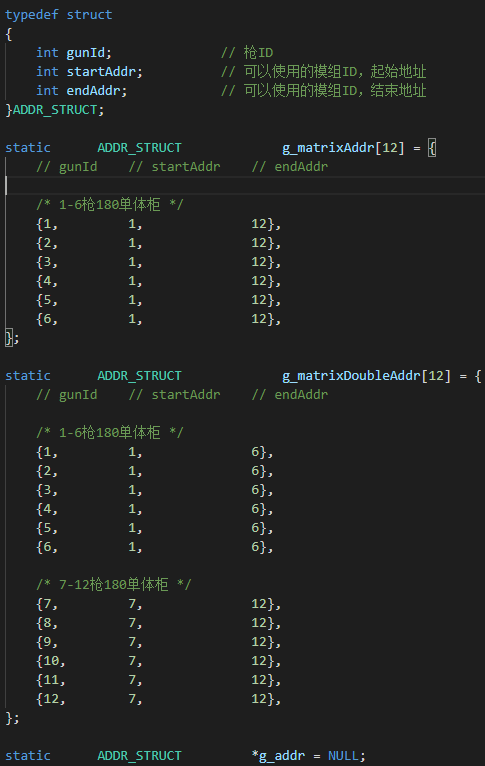
int startAddr; // 可以使用的模组ID，起始地址

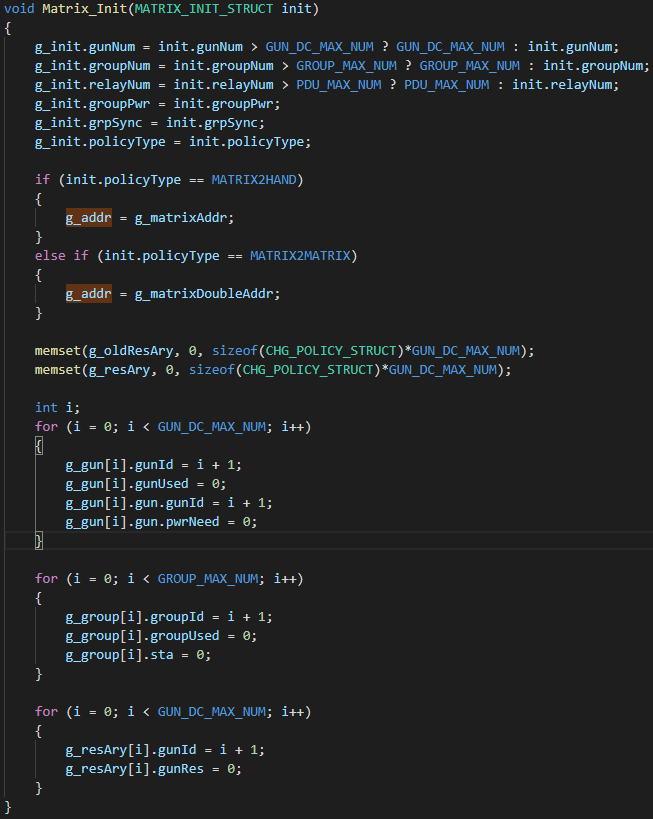
int endAddr; // 可以使用的模组ID，结束地址

}ADDR\_STRUCT;//lhm:某枪当前可以使用的模块（是可以使用，不是正在使用，另外它们是连续的模块，应该还需//要检查每个模块是否在线）

//lhm: 本代码中group是模组的意思，模组就是模块的意思，即group是模块，一个group就是一个模块，groupNum

//就是模块的数量





#### 桩端