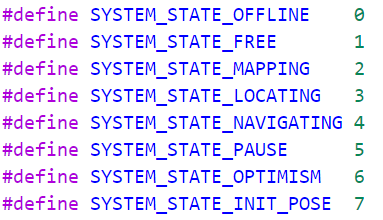
# class AGVInstance

AGV主控模块，单例，使用函数void init\_global\_agv\_instance()初始化，使用函数AGVInstance \* get\_global\_agv\_instance()获取实例指针；

## int get\_system\_state();

获取当前系统的状态码



## void update\_last\_position(Position &, int, int);

更新系统位姿并修改系统位姿等级。当系统位姿处于高等级时，会拒绝低等级的更新请求。当输入位姿的时间戳小于当前系统位姿时，不更新

参数1：输入位姿

参数2：发起位姿更新请求的等级

参数3：更新结束后将位姿等级改为此值

## void update\_last\_position(Position &);

同void update\_last\_position(arg1, 1, 1);

## Position get\_last\_position();

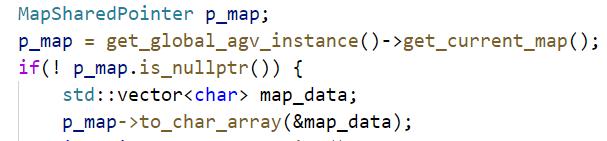
获取最新的系统位姿。可能不是实时位姿

## Position expolate\_current\_position();

估计当前的位姿。从最新的系统位姿开始，使用里程计估计

## MapSharedPointer get\_current\_map()

获取当前的地图数据的共享指针。MapSharedPointer重载了->和\*运算符，用法约等于SImpleGridMap\*。但不可在声明的时候赋值



## void load\_grid\_map(SimpleGridMap &map)

使用一个新的地图对象代替当前的系统地图。不会影响之前获取到的地图指针。

## void load\_grid\_map(char \*buf, int size)

使用一组二进制数据构造地图，并替换当前的系统地图。同上

## MsgQueue \* get\_message\_queue()

获取系统的消息队列对象指针。消息队列用法见其相关文档。

## void recieve\_message(const int, char \*, const int);

作为消息队列的监听者实现的函数，不需要手动调用。

## bool start\_mapping()

开始建图。成功则返回1

## bool stop\_mapping()

停止建图

## void start\_global\_locating()

开始全局定位。全局定位要求AGV处于free状态，定位过程中会暂停当前的移动动作，暂停后台的跟踪定位。全局定位结束后会将位姿更新到等级为1的系统位姿。

## PointCloudData get\_last\_radar\_data()

获取最新的一帧激光雷达数据