robomaster机器人代码学习

**问题：**

cmd\_task.c 452行中为什么只有y的值

底盘x，y，旋转的计算公式

为什么底盘的角度等于云台相对初始位置 cmd\_task.c 469行

小陀螺是指哪里的小陀螺

cmd\_task.c 545行memset的作用

kb\_state\_e，按键按下的状态ONCE是什么意思

发射状态的两个发射错误分别是指什么

cmd\_task.c 706行几个扳机状态是什么意思

堵转是一个什么状态

发布消息，接收消息具体是怎么实现的

static publisher\_t \*pub\_gim, \*pub\_chassis, \*pub\_shoot,\*pub\_ui;

中发布者定义在cmd\_task.c中，是怎么实现的向外传递信息

*GIMBAL\_AUTO 云台控制模式，这个时怎样的一个模式*

*//推测是云台和底盘分开运动的状态*

typedef enum  
{  
 *KEY\_RELEASE* = 0, //没有按键按下  
 *KEY\_WAIT\_EFFECTIVE*, //等待按键按下有效，防抖  
 *KEY\_PRESS\_ONCE*, //按键按下一次的状态  
 *KEY\_PRESS\_DOWN*, //按键已经被按下  
 *KEY\_PRESS\_LONG*, //按键长按状态  
} kb\_state\_e;

//按键按下一次的状态是什么意思  
cmd\_task.c621行，小陀螺的开与关，为什么开和关用到的结构体不一样，为什么要用不同的结构体来进行开关，为什么好像有两个结构体的输入  
关于左摇杆左右的作用，同时控制小车旋转，云台运动，好像没有添加条件限制，两行代码好像会打架

未发现wasd键盘输入控制小车移动

注册斜坡是个什么东西

**cmd\_task.c：**

static publisher\_t \*pub\_gim, \*pub\_chassis, \*pub\_shoot,\*pub\_ui;  
static subscriber\_t \*sub\_gim, \*sub\_shoot,\*sub\_trans,\*sub\_ins,\*sub\_referee;

创建发布者，订阅者

sub\_get\_msg(sub\_gim, &gim\_fdb);

该函数将sub\_gim复制到gim\_fdb，其他同理，

后续仍需继续深入理解线程之间的收发消息的机制，暂时理解在这

订阅者和发布者相关的函数的深入实现

在什么地方专门实现信息的传递

uint8\_t pub\_push\_msg(publisher\_t \*pub, void \*data){  
 if(rt\_sem\_take(pub->tp->sem, RT\_WAITING\_NO)){  
 LOG\_W("take sem failed!");  
 return 0;  
 }  
 rt\_memcpy(pub->tp->msg, data, pub->len);  
 rt\_sem\_release(pub->tp->sem);  
 return 1;  
}

该函数使用了信号量，安全操作，将data存入pub里面，便于函数发布信息

uint8\_t sub\_get\_msg(subscriber\_t \*sub, void \*data){  
 rt\_memcpy(data, sub->tp->msg, sub->len);  
 return 1;  
}

将订阅的sub的数据复制到data中

左摇杆开控制云台运动，右摇杆控制底盘运动

gim\_cmd.yaw = trans\_fdb.yaw+gyro\_yaw\_inherit + mouse\_accumulate\_x  
 /\* + 150 \* rc\_now->ch3 \* RC\_RATIO \* GIMBAL\_RC\_MOVE\_RATIO\_YAW\*/;//上位机自瞄  
gim\_cmd.pitch = trans\_fdb.pitch+gyro\_pitch\_inherit + mouse\_accumulate\_y  
 /\* +100 \* rc\_now->ch4 \* RC\_RATIO \* GIMBAL\_RC\_MOVE\_RATIO\_PIT \*/;//上位机自瞄

自瞄模式下云台的操作命令值来自三个地方，自瞄，鼠标输入

还有普通模式下的历史记录，之前的角度，摇杆好像没起作用

推测自瞄模式摇杆不好控制特定的角度

普通模式*GIMBAL\_GYRO下只有摇杆输入，和键盘输入*

云台除了断开模式以外就两种状态，自瞄模式，跟随模式 *GIMBAL\_GYRO*

*GIMBAL\_AUTO，通过一下代码实现*

*if (rc\_now->mouse.r==0)  
{  
 if (gim\_cmd.last\_mode == GIMBAL\_RELAX)  
 {/\* 判断上次状态是否为RELAX，是则先归中 \*/  
 gim\_cmd.ctrl\_mode = GIMBAL\_INIT;  
 chassis\_cmd.ctrl\_mode=CHASSIS\_RELAX;  
 }  
 //先进行归中  
 else  
 {  
 if (gim\_fdb.back\_mode == BACK\_IS\_OK) /\*!是否归中完成\*/  
 {  
 gim\_cmd.ctrl\_mode = GIMBAL\_GYRO;  
 chassis\_cmd.ctrl\_mode = CHASSIS\_FOLLOW\_GIMBAL;  
 memset(r\_buffer\_point,0,sizeof (\*r\_buffer\_point));  
 ////作用：指向自瞄模块使用的环形缓冲区，用于清除旧数据。  
 ////场景：手动模式下清空自瞄数据，避免残留数据干扰。  
 }  
 else if(gim\_fdb.back\_mode==BACK\_STEP)  
 {  
 chassis\_cmd.ctrl\_mode=CHASSIS\_RELAX;  
 gim\_cmd.ctrl\_mode=GIMBAL\_INIT;  
 }  
 }  
}  
// uint8\_t sw1; //左侧拨杆  
// uint8\_t sw2; //右侧拨杆  
if (rc\_now->mouse.r==1||rc\_now->sw2==RC\_DN) /\*!如果鼠标按下右键或者遥控器选择自瞄模式\*/  
{  
 //右键也是，按下一次就直接赋值1，后续不会自动置0  
 if (gim\_cmd.last\_mode == GIMBAL\_RELAX)  
 {/\* 判断上次状态是否为RELAX，是则先归中 \*/  
 gim\_cmd.ctrl\_mode = GIMBAL\_INIT;  
 chassis\_cmd.ctrl\_mode=CHASSIS\_RELAX;  
 }  
 else  
 {  
 if (gim\_fdb.back\_mode == BACK\_IS\_OK)  
 {/\* 判断归中是否完成 \*/  
 //应该只会归中一次，后续直接进入  
 gim\_cmd.ctrl\_mode = GIMBAL\_AUTO;  
 chassis\_cmd.ctrl\_mode = CHASSIS\_FOLLOW\_GIMBAL;  
 }  
 else if(gim\_fdb.back\_mode==BACK\_STEP)  
 {  
 chassis\_cmd.ctrl\_mode=CHASSIS\_RELAX;  
 gim\_cmd.ctrl\_mode=GIMBAL\_INIT;  
 }  
 }  
}*

在自瞄模式下底盘的运动状态是跟随云台的由一下代码实现

*if (gim\_fdb.back\_mode == BACK\_IS\_OK)  
{/\* 判断归中是否完成 \*/  
 //应该只会归中一次，后续直接进入  
 gim\_cmd.ctrl\_mode = GIMBAL\_AUTO;  
 chassis\_cmd.ctrl\_mode = CHASSIS\_FOLLOW\_GIMBAL;//底盘模式为跟随云台  
}*

关于UI的结构体目前只看到是否初始化

chassis\_cmd.ctrl\_mode=*CHASSIS\_RELAX*;  
gim\_cmd.ctrl\_mode=*GIMBAL\_INIT*;

底盘失能，云台初始化，为底盘云台的一般初始化

***遥控器的控制：***

初始化时需要将两个拨杆拨到上位

右摇杆来控制小车的移动

左摇杆的左右控制小车旋转，但需要开启小陀螺，关闭小陀螺时，左侧摇杆控制云台运动

右侧拨杆处于下位时，自瞄模式

左侧拨杆处于下位时，打开小陀螺，中位时打开摩擦轮，不是中位关闭摩擦轮

左侧拨杆调整开火

***键鼠控制：***

单击右键开启自瞄模式

左键开火

b键初始UI

e键打开小陀螺

f键打开摩擦轮

g键关闭摩擦轮

v键更改射击模式 连发模式，单发模式