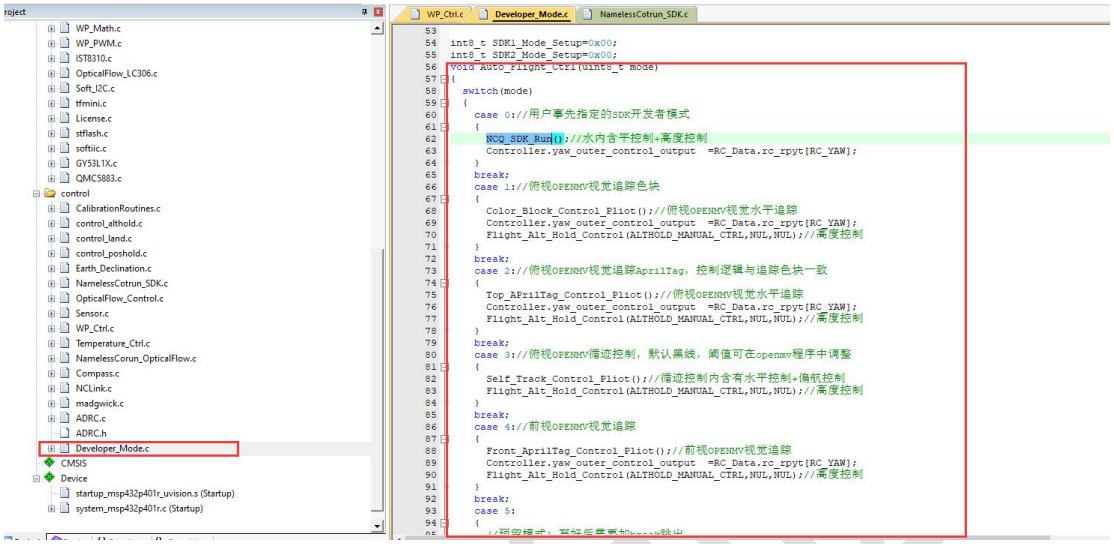
八、MSP432 飞控 SDK 模式开发教程

本节将详细介绍飞控常用的 API 函数应用与利用 SDK 视觉数据实现自主飞行控制。下面分别以自定义轨迹飞行、追踪物块、APrilTag 定位、自主循迹为例，了解飞控相关的 API 函数在不同任务中的使用，提供用户二次开发的 SDK 控制函数 Auto_Flight_Ctrl()函数，后续二次开发任务可以均写在此函数中。



① 自定义轨迹飞行 NCQ_SDK_Run():

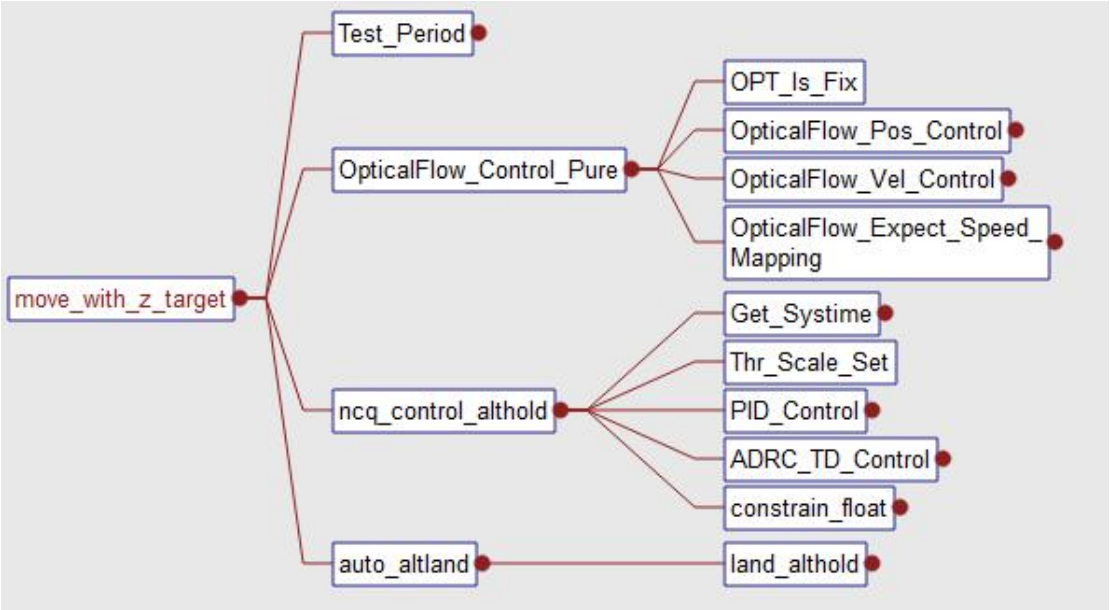
默认例程是分三个 SDK 子任务，分别是向上飞行 100cm 高度，然后前进 50cm，最后下降 150cm，如果是以地面初始起飞条件，飞机再执行第三个任务时会降落到地面，地面检测函数检测到满足地面状态时，飞机会自动上锁。

WP_Ctrl.c | Developer_Mode.c | NamelessCotrun_SDK.c

```
//#define NCQ_SDK_DUTY1 move_with_speed_target(10,0,2000 ,&SDK_Duty_Status,1-1)//左
//#define NCQ_SDK_DUTY2 move_with_speed_target(0,10,2000 ,&SDK_Duty_Status,2-1)//前
//#define NCQ_SDK_DUTY3 move_with_speed_target(-10,0,2000,&SDK_Duty_Status,3-1)//右
//#define NCQ_SDK_DUTY4 move_with_speed_target(0,-10,2000,&SDK_Duty_Status,4-1)//后

#define NCQ_SDK_DUTY_MAX   3
#define NCQ_SDK_DUTY1 move_with_z_target(100,0,0,&SDK_Duty_Status,1-1)
#define NCQ_SDK_DUTY2 move_with_xy_target(0,50,&SDK_Duty_Status,2-1)
#define NCQ_SDK_DUTY3 move_with_z_target(-150,0,0,&SDK_Duty_Status,3-1)

SDK_Status SDK_Duty_Status;
uint16_t SDK_Duty_Cnt=0;
uint16_t SDK_Transition_Time=0;
void NCQ_SDK_Run(void)
{
  if(SDK_Duty_Status.Transition_Time[SDK_Duty_Cnt]>=1)
    SDK_Duty_Status.Transition_Time[SDK_Duty_Cnt]--;//安排过渡时间

  if(SDK_Duty_Status.Status[SDK_Duty_Cnt].Start_Flag==1
     &&SDK_Duty_Status.Status[SDK_Duty_Cnt].Execute_Flag==1
       &&SDK_Duty_Status.Status[SDK_Duty_Cnt].End_flag==1
         &&SDK_Duty_Status.Transition_Time[SDK_Duty_Cnt]==0)
    SDK_Duty_Cnt++;

  if(SDK_Duty_Cnt>=NCQ_SDK_DUTY_MAX) SDK_Duty_Cnt=NCQ_SDK_DUTY_MAX;

  if(SDK_Duty_Cnt==0)         NCQ_SDK_DUTY1;
  else if(SDK_Duty_Cnt==1)    NCQ_SDK_DUTY2;
  else if(SDK_Duty_Cnt==2)    NCQ_SDK_DUTY3;
  //else if(SDK_Duty_Cnt==3)    NCQ_SDK_DUTY4;
  //else if(SDK_Duty_Cnt==4)    NCQ_SDK_DUTY5;
  //else if(SDK_Duty_Cnt==5)    NCQ_SDK_DUTY6;
  //else if(SDK_Duty_Cnt==6)    NCQ_SDK_DUTY7;
  else
  {
    ncq_control_althold();//高度控制
    OpticalFlow_Control_Pure(0);//位置控制
  }
}
```

- NCQ_SDK_DUTY1\NCQ_SDK_DUTY3 中

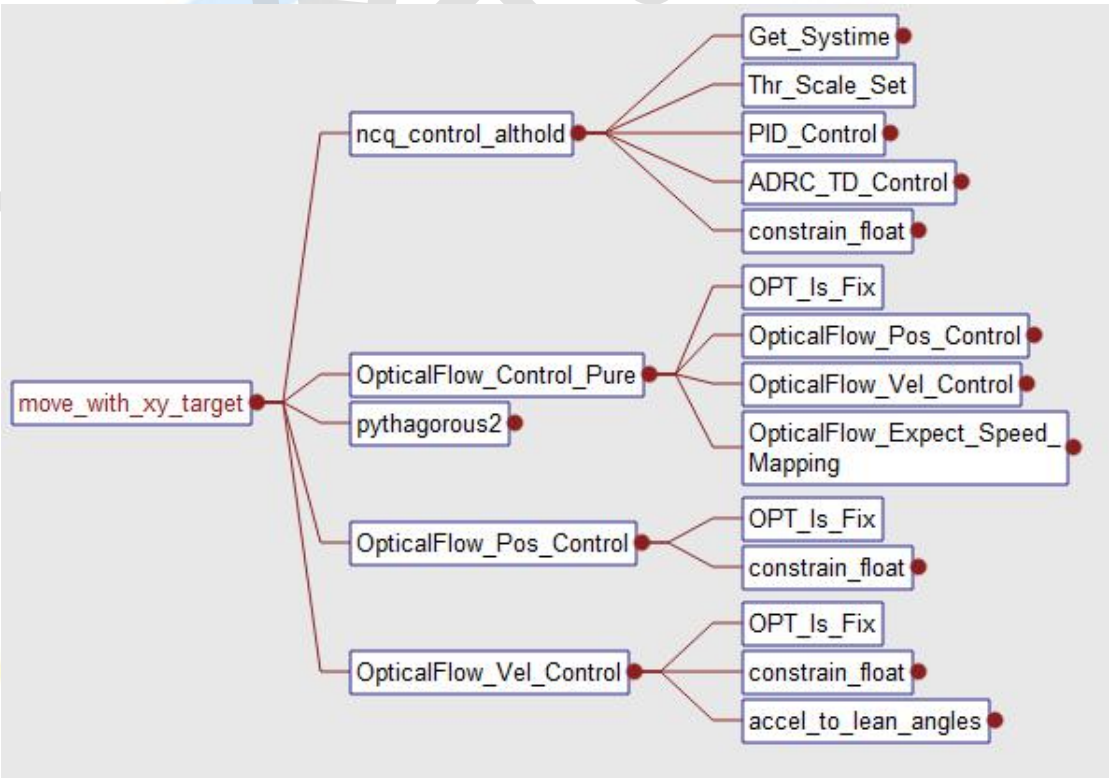
uint8_t move_with_z_target(float z_target,float z_vel,float delta,SDK_Status *Status,uint16_t number)



其中函数入口参数：z_target 表示目标位移，z_vel 表示期望速度，delta 表示运行总时间（和 z_vel 结合使用，即以某一速度爬升/下降多长时间），*Status 表示 SDK 任务结构体，number 表示 SDK 子任务序号。move_with_z_target 任务执行完毕后会返回完成标志 1，用户可以依据此标志位来判断是否完成子任务，过渡到下一子任务。

- NCQ_SDK_DUTY2 中

uint8_t move_with_xy_target(float pos_x_target,float pos_y_target,SDK_Status *Status,uint16_t number)



其中函数入口参数：pos_x_target 表示 X 方向目标位移，pos_y_target 表示 Y 方向目标位移，*Status 表示 SDK 任务结构体，number 表示 SDK 子任务序号。move_with_xy_target 任务执行完毕后会返回完成标志 1，用户可以依据此标志位来判断是否完成子任务，过渡到下一子任务。需要注意的是在本 SDK 任务子任务执行中间安排了过渡过程，原地悬停 1S 中再执行下一任务。

- 其它 API 接口

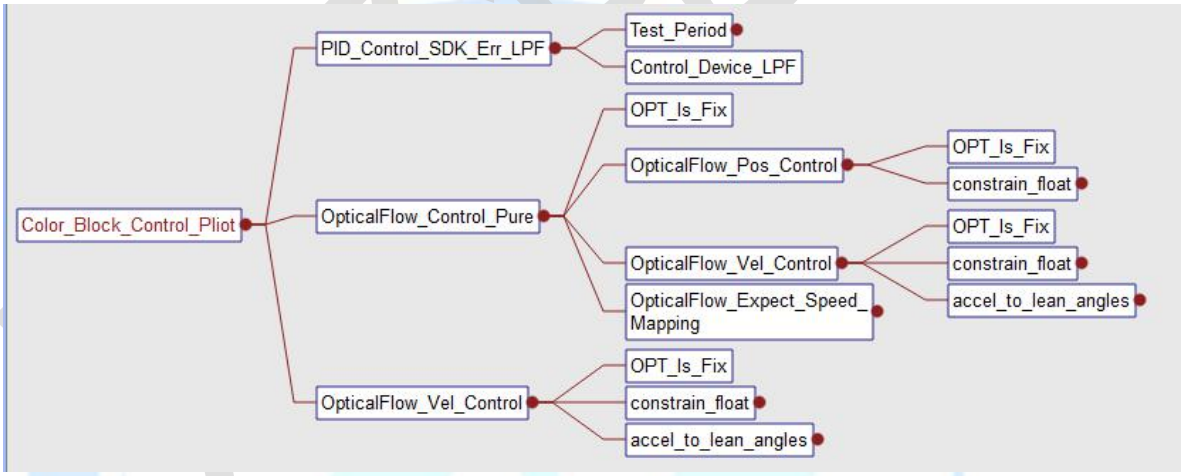
  uint8_t move_with_speed_target(float x_target,float y_target,float delta,SDK_Status *Status,uint16_t number)

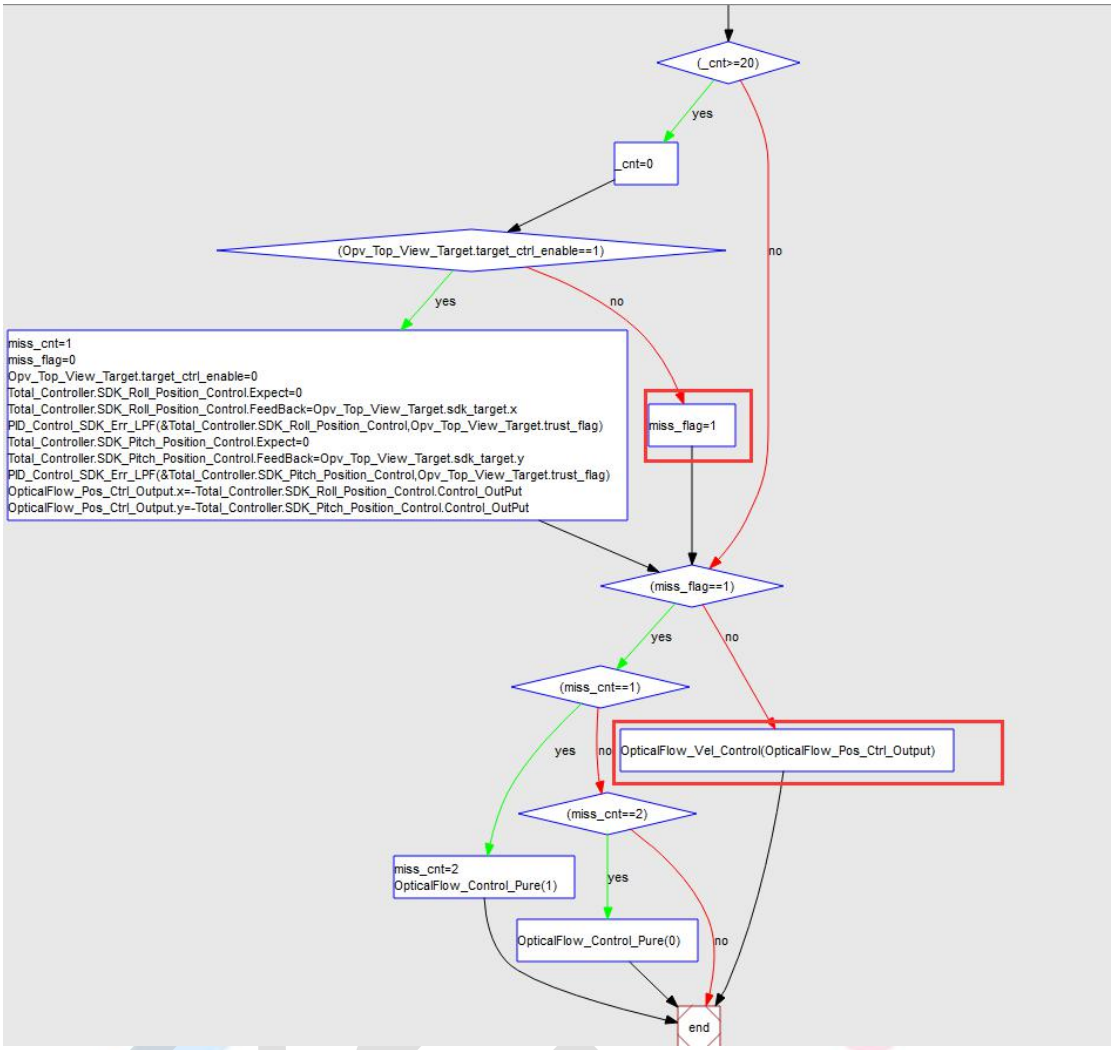
其中函数入口参数：x_target 表示 X 方向期望速度，y_target 表示 Y 方向期望速度，delta 表示运行总时间（和期望速度结合使用，即以某一速度飞行多长时间），*Status 表示 SDK 任务结构体，number 表示 SDK 子任务序号。move_with_speed_target 任务执行完毕后会返回完成标志 1，用户可以依据此标志位来判断是否完成子任务，过渡到下一子任务。

② OPENMV 视觉水平追踪 Color_Block_Control_Pliot():

俯视 OPENMV 视觉水平追踪 Top_APrilTag_Control_Pliot()——同理

俯视 OPENMV 视觉水平追踪任务中，根据 OPENMV 识别到的水平目标位置，运行 PID_Control_SDK_Err_LPF 函数得到水平速度期望，然后运行光流速度控制函数，最后得到姿态角期望。当目标丢失时会自动退出追踪控制，保持原地悬停。

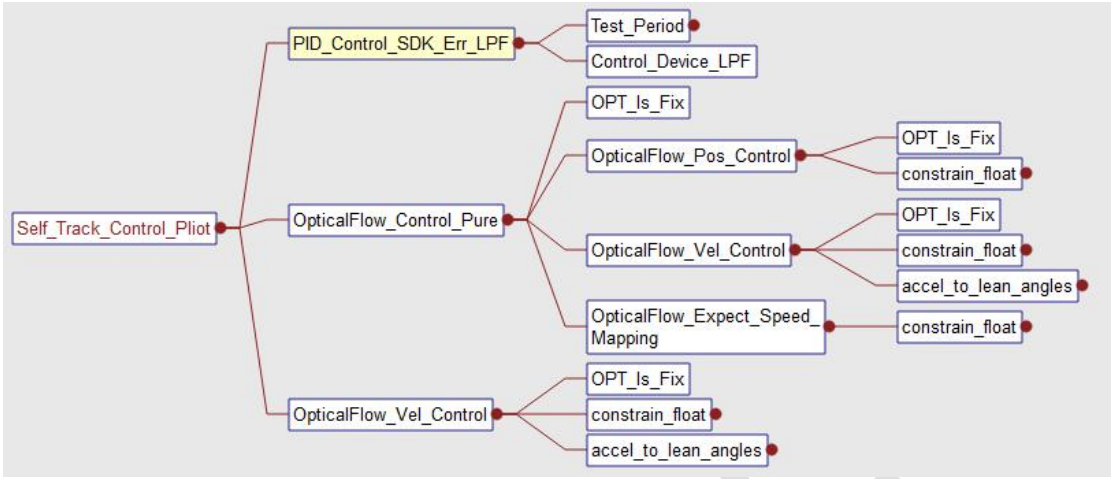




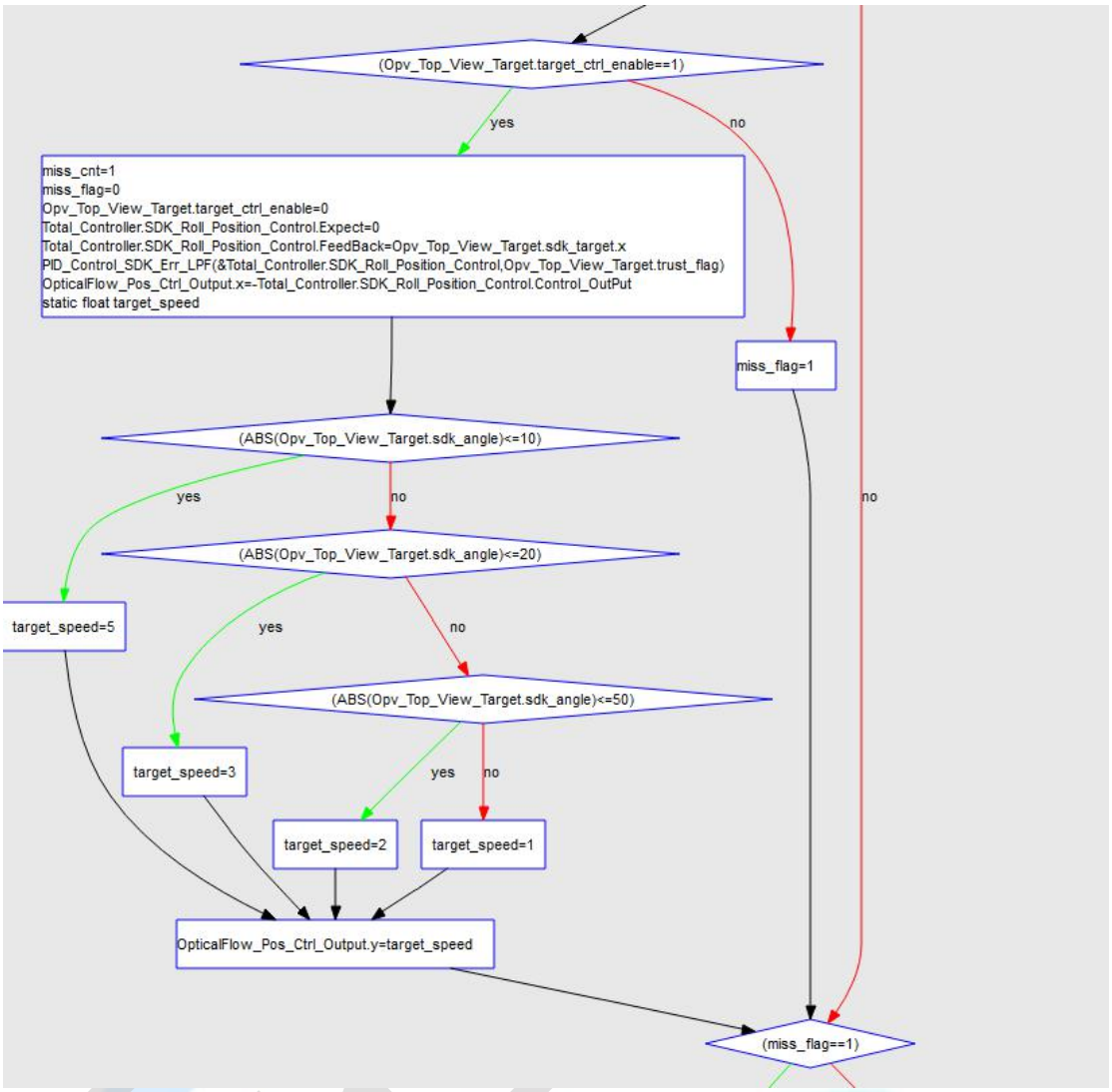
默认例程是分三个 SDK 子任务，分别是向上飞行 100cm 高度，然后前进 50cm，最后下降 150cm，如果是以地面初始起飞条件，飞机再执行第三个任务时会降落到地面，地面检测函数检测到满足地面状态时，飞机会自动上锁。

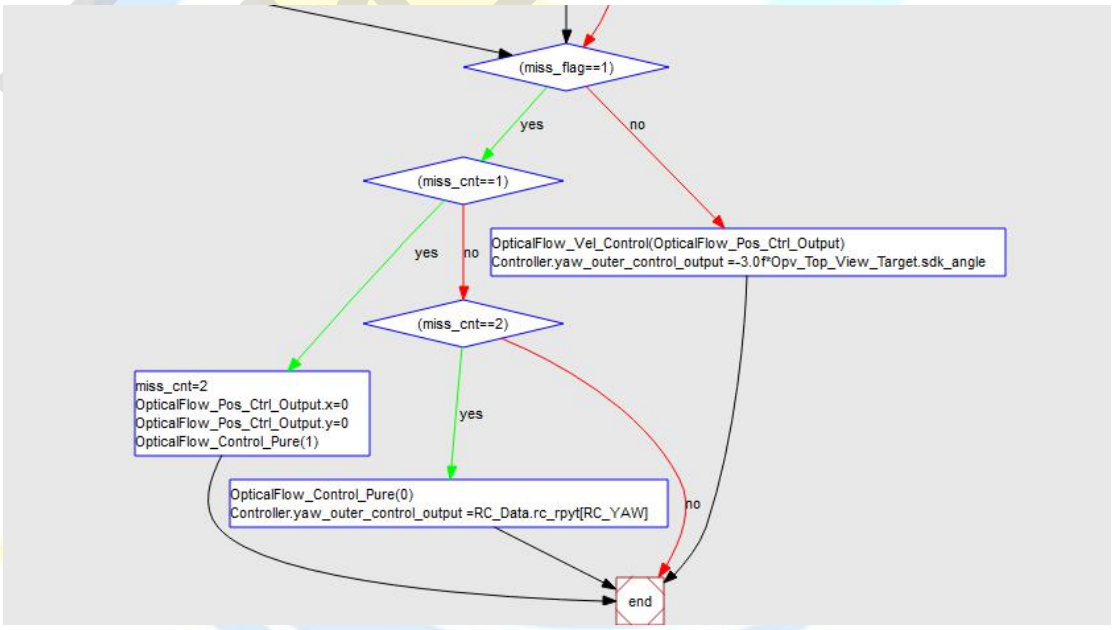
③ 俯视 OPENMV 循迹控制 Self_Track_Control_Pliot():

俯视 OPENMV 循迹控制任务中，根据 OPENMV 识别到的底部黑线轨迹，运行 PID_Control_SDK_Err_LPF 函数得到水平 X 方向速度期望，然后运行 X 方向光流速度控制函数，最后得到横滚方向姿态角期望。偏航方向根据识别到轨迹的角度，运用单比例 P 控制得到期望偏航的角速度，Y 方向根据当前轨迹的斜率（角度）大小，决策合适的 Y 方向期望速度。同理当目标丢失时会自动退出追踪控制，保持原地悬停。




```
604     {
605         _cnt=0;
606       if(Opv_Top_View_Target.target_ctrl_enable==1)//目标点检测跟踪
607         {
608             miss_cnt=1;
609             miss_flag=0;
610             Opv_Top_View_Target.target_ctrl_enable=0;
611
612             Total_Controller.SDK_Roll_Position_Control.Expect=0;
613             Total_Controller.SDK_Roll_Position_Control.FeedBack=Opv_Top_View_Target.sdk_target.x;
614             PID_Control_SDK_Err_LPF(&Total_Controller.SDK_Roll_Position_Control,Opv_Top_View_Target.trust_flag);
615             OpticalFlow_Pos_Ctrl_Output.x=-Total_Controller.SDK_Roll_Position_Control.Control_OutPut;
616
617             static float target_speed;
618             if(ABS(Opv_Top_View_Target.sdk_angle)<=10)      target_speed=5;
619             else if(ABS(Opv_Top_View_Target.sdk_angle)<=20) target_speed=3;
620             else if(ABS(Opv_Top_View_Target.sdk_angle)<=50) target_speed=2;
621             else target_speed=1;
622
623 //          Total_Controller.SDK_Pitch_Position_Control.Expect=Forward_Keep_Distance;//期望为前向保持距离，具体可根据实际自行定义
624 //          Total_Controller.SDK_Pitch_Position_Control.FeedBack=Opv_Top_View_Target.apriltag_distance;
625 //          PID_Control_SDK_Err_LPF(&Total_Controller.SDK_Pitch_Position_Control,Opv_Top_View_Target.trust_flag);
626
627             OpticalFlow_Pos_Ctrl_Output.y=target_speed;
628         }
629         else//丢失目标
630         {
631           miss_flag=1;
632         }
633     }
634
635
636     if(miss_flag==1)//目标丢失
637     {
638         if(miss_cnt==1)//初始丢失跟踪目标后，锁定当前位置后，进行普通光流控制
639         {
640             miss_cnt=2;
641             OpticalFlow_Pos_Ctrl_Output.x=0;
642             OpticalFlow_Pos_Ctrl_Output.y=0;
643             OpticalFlow_Control_Pure(1);
644         }
645         else if(miss_cnt==2)//丢失跟踪目标后，进行普通光流控制
646         {
647             OpticalFlow_Control_Pure(0);
648             Controller.yaw_outer_control_output  =RC_Data.rc_rpyt[RC_YAW];
649         }
650     }
651   else//目标未丢失
652     {
653         OpticalFlow_Vel_Control(OpticalFlow_Pos_Ctrl_Output);//速度控制周期20ms
654         Controller.yaw_outer_control_output  =-3.0f*Opv_Top_View_Target.sdk_angle;
655     }
656 }
```

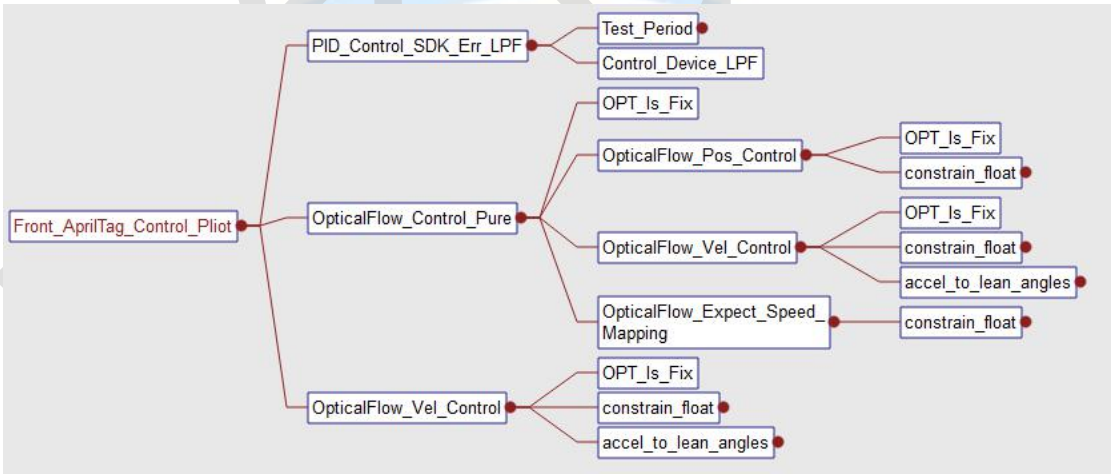

(Opv_Top_View_Target.target_ctrl_enable==1)
yes
no
miss_cnt=1
miss_flag=0
Opv_Top_View_Target.target_ctrl_enable=0
Total_Controller.SDK_Roll_Position_Control.Expect=0
Total_Controller.SDK_Roll_Position_Control.FeedBack=Opv_Top_View_Target.sdk_target.x
PID_Control_SDK_Err_LPF(&Total_Controller.SDK_Roll_Position_Control,Opv_Top_View_Target.trust_flag)
OpticalFlow_Pos_Ctrl_Output.x=-Total_Controller.SDK_Roll_Position_Control.Control_OutPut
static float target_speed
miss_flag=1
(ABS(Opv_Top_View_Target.sdk_angle)<=10)
yes
no
no
(ABS(Opv_Top_View_Target.sdk_angle)<=20)
target_speed=5
yes
no
(ABS(Opv_Top_View_Target.sdk_angle)<=50)
target_speed=3
yes
no
target_speed=2
target_speed=1
OpticalFlow_Pos_Ctrl_Output.y=target_speed
(miss_flag==1)

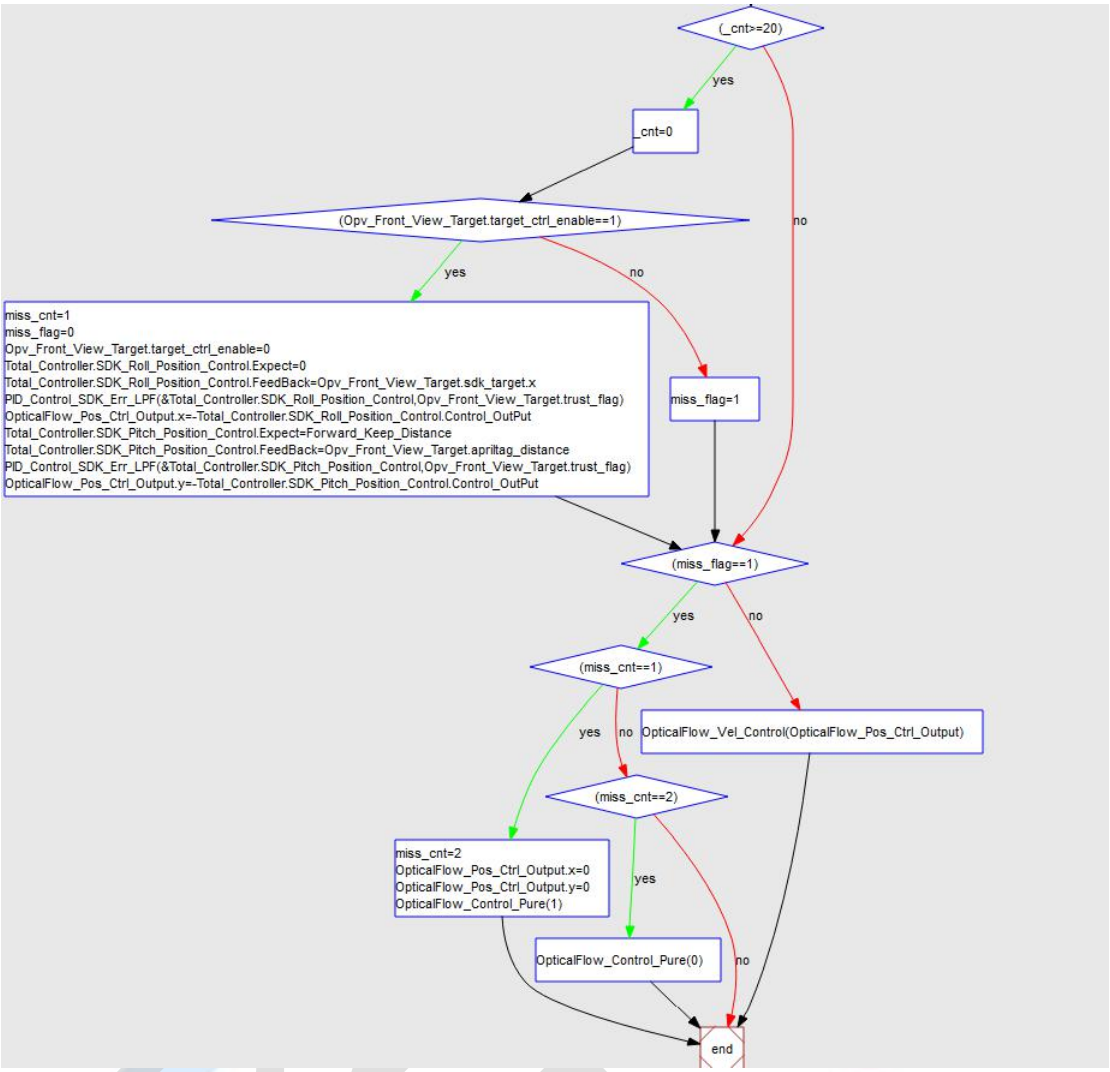

(miss_flag==1)
yes
no
(miss_cnt==1)
yes
no
OpticalFlow_Vel_Control(OpticalFlow_Pos_Ctrl_Output)
Controller.yaw_outer_control_output =-3.0f*Opv_Top_View_Target.sdk_angle
(miss_cnt==2)
miss_cnt=2
OpticalFlow_Pos_Ctrl_Output.x=0
OpticalFlow_Pos_Ctrl_Output.y=0
OpticalFlow_Control_Pure(1)
yes
OpticalFlow_Control_Pure(0)
Controller.yaw_outer_control_output =RC_Data.rc_rpyt[RC_YAW]
no
end

④ 前视 OPENMV 视觉追踪控制 Front_AprilTag_Control_Pliot():

前视 OPENMV 视觉追踪控制任务中，运行 PID_Control_SDK_Err_LPF 函数得到水平 X 方向速度期望，然后运行 X 方向光流速度控制函数，最后得到横滚方向姿态角期望。Y 方向根据 AprilTag 到飞机的距离，运行 PID_Control_SDK_Err_LPF 函数得到水平 Y 方向速度期望，然后运行 Y 方向光流速度控制函数。同理当目标丢失时会自动退出追踪控制，保持原地悬停。

```
  _cnt++;
  if(_cnt>=20)//100ms
  {
    _cnt=0;
    if(Opv_Front_View_Target.target_ctrl_enable==1)//目标点检测跟踪
    {
      miss_cnt=1;
      miss_flag=0;
      Opv_Front_View_Target.target_ctrl_enable=0;

      Total_Controller.SDK_Roll_Position_Control.Expect=0;
      Total_Controller.SDK_Roll_Position_Control.FeedBack=Opv_Front_View_Target.sdk_target.x;
      PID_Control_SDK_Err_LPF(&Total_Controller.SDK_Roll_Position_Control,Opv_Front_View_Target.trust_flag);
      OpticalFlow_Pos_Ctrl_Output.x=-Total_Controller.SDK_Roll_Position_Control.Control_OutPut;



      Total_Controller.SDK_Pitch_Position_Control.Expect=Forward_Keep_Distance;//期望为前向保持距离，具体可根据实际自行定义
      Total_Controller.SDK_Pitch_Position_Control.FeedBack=Opv_Front_View_Target.apriltag_distance;
      PID_Control_SDK_Err_LPF(&Total_Controller.SDK_Pitch_Position_Control,Opv_Front_View_Target.trust_flag);
      OpticalFlow_Pos_Ctrl_Output.y=-Total_Controller.SDK_Pitch_Position_Control.Control_OutPut;
    }
    else//丢失目标
    {
      miss_flag=1;
    }
  }


  if(miss_flag==1)//目标丢失
  {
    if(miss_cnt==1)//初始丢失跟踪目标后，锁定当前位置后，进行普通光流控制
    {
      miss_cnt=2;
      OpticalFlow_Pos_Ctrl_Output.x=0;
      OpticalFlow_Pos_Ctrl_Output.y=0;
      OpticalFlow_Control_Pure(1);
    }
    else if(miss_cnt==2)//丢失跟踪目标后，进行普通光流控制
    {
      OpticalFlow_Control_Pure(0);
    }
```





⑤ 预留用户控制与降落任务控制:

Auto_Flight_Ctrl()函数中预留了用户控制函数与降落控制函数，用二次开发任务时编写具体项目任务函数，执行完毕后用降落函数落地至地面怠速后自动上锁。同时针对某些赛题结束时需要对准降落目标，存在末端摄像头丢失视野目标时，可以调用降落模式，用光流辅助自动下降，短时内基本不会偏移。

```
125     case 13:
126     {
127       //预留模式9,写好后需要加break跳出
128     }
129     case 14:
130     {
131       //预留模式10,写好后需要加break跳出
132     }
133     case 15://前面预留case不满足情况下执行此情形
134     {
135       Controller.roll_outer_control_output =RC_Data.rc_rpyt[RC_ROLL];
136       Controller.pitch_outer_control_output=RC_Data.rc_rpyt[RC_PITCH];
137       Controller.yaw_outer_control_output  =RC_Data.rc_rpyt[RC_YAW];
138       Flight_Alt_Hold_Control(ALTHOLD_MANUAL_CTRL,NUL,NUL);//高度控制
139     }
140     break;
141     case 16://SDK模式中原地降落至地面怠速后停桨,用于任务执行完成后降落
142     {
143       OpticalFlow_Control(0);
144       Controller.yaw_outer_control_output  =RC_Data.rc_rpyt[RC_YAW];
145       Flight_Alt_Hold_Control(ALTHOLD_AUTO_VEL_CTRL,NUL,-50);//高度控制
146     }
147     break;
148     default:
149     {
150       Controller.roll_outer_control_output =RC_Data.rc_rpyt[RC_ROLL];
151       Controller.pitch_outer_control_output=RC_Data.rc_rpyt[RC_PITCH];
152       Controller.yaw_outer_control_output  =RC_Data.rc_rpyt[RC_YAW];
153       Flight_Alt_Hold_Control(ALTHOLD_MANUAL_CTRL,NUL,NUL);//高度控制
154     }
```