幻思创新 FPM 光流模块编程开发指南

光流数据结构定义如下：

```
typedef   struct   optical_flow_data
{
int16_t     flow_x_integral;      // X 像素点累计时间内的累加位移(radians*10000)
                                  // [除以 10000 乘以高度(mm)后为实际位移(mm)]
int16_t     flow_y_integral;      // Y 像素点累计时间内的累加位移(radians*10000)
                                  // [除以 10000 乘以高度(mm)后为实际位移(mm)]
uint16_t integration_timespan; // 上一次发送光流数据到本次发送光流数据的累计时间（us）
uint16_t   ground_distance;   // 预留。默认为 999（0x03E7）
uint8_t   valid;              // 状态值:0(0x00)为光流数据不可用
                              //245(0xF5)为光流数据可用
uint8_t    version;           //版本号
} Upixels_OpticalFlow;
```

光流解析源码如下：

```
typedef struct
{
    int16_t   flow_x_integral;
    int16_t   flow_y_integral;
    uint16_t integration_timespan;
    uint16_t ground_distance;
    uint8_t  quality;
    uint8_t  version;
}LC302_Data;
LC302_Data lc302_data;

typedef struct {
   bool healthy;
   float flow_dt;
   float vel_x;
   float vel_y;
   float pos_x;
   float pos_y;
} Opticalflow_state;
Opticalflow_state opticalflow_state;
```

```
uint8_t get_lc302_data(uint8_t buf);
void opticalflow_update(void);

static int16_t s  = 0, p = 0;
static uint8_t Xor_r = 0x00, Xor_c = 0x00;
uint8_t get_lc302_data(uint8_t buf)
{
    uint8_t ret = 1;

    switch(s)
    {
        case 0:
            if(buf == 0xFE)
            {
                s = 1;
                //printf("Got FE\n");
            }
        break;
        case 1:
            if(buf == 0x0A)
            {
                s = 2;
                p = 0;
                Xor_c = 0x00;
                //printf("Got 0A\n");
            }else
                s = 0;
        break;
        case 2:
            ((char *)&lc302_data)[p++] = buf;
            Xor_c ^= buf;
            if(p == 10){
                s = 3;
                p = 0;
            }
        break;
        case 3: //crc
            s = 4;
            Xor_r = buf;
        break;
        case 4://end
            if(buf == 0x55){
               //printf("Got 0x55\n");
```

```
                if(Xor_r == Xor_c){
                    ret = 0;
                    opticalflow_update();//解算函数具体实现在下面
                }
                else
                    ret = 2;
            }
            s = 0;
        break;
        default:
        break;
    }
    return ret;
}


//光流解算函数
static float flow_alt, flow_bf_x, flow_bf_y;
void opticalflow_update(void){
   //首先根据激光传感获取对地高度
   flow_alt=（你的激光测距数据,单位cm）;

   if(lc302_data.quality==245){
      //光流数据正常
      opticalflow_state.healthy=true;
   }else{
      //光流数据异常
      opticalflow_state.healthy=false;
      return;
   }
   //光流像素->机体角速度
   flow_bf_x=(float)lc302_data.flow_y_integral*0.0001f+（你的旋转补偿）;
   flow_bf_y=-(float)lc302_data.flow_x_integral*0.0001f+（你的旋转补偿）;
```

```
    //然后用户自己处理，对光流测出的机体角速度进行限幅！


    //由机体角速度转换为机体速度
    opticalflow_state.flow_dt=(float)lc302_data.integration_timespan*0.000001f;
    opticalflow_state.vel_x=opticalflow_state.rads_x*flow_alt/opticalflow_state.flow_dt;
    opticalflow_state.vel_y=opticalflow_state.rads_y*flow_alt/opticalflow_state.flow_dt;


    //下一步用户自己处理,对光流测速数据进行低通滤波


    //光流测速的积分得到光流位移
    opticalflow_state.pos_x+=opticalflow_state.vel_x*opticalflow_state.flow_dt;
    opticalflow_state.pos_y+=opticalflow_state.vel_y*opticalflow_state.flow_dt;

}
```