

# 云台

## 私有协议

# 阅读提示 - 符号说明



重要注意事项



操作提示



词汇解释及参考信息

# 版本历史

日期	文档版本	固件版本	协议版本
2024.12.13	V1.0.0	V3.5 及以上	V1.0

日期	文档版本	固件版本	协议版本
2025.01.03	V1.0.1	V3.5 及以上	V1.0

日期	文档版本	固件版本	协议版本
2025.02.26	V1.0.2	V3.5 及以上	V1.0

1.       上位机发送的数据包结构体：
- 1.1     详细了命令触发计数 (注释 2) 的说明文字；[P3]
- 1.2     增加了启动云台与停止云台命令的说明。[P3]

日期	文档版本	固件版本	协议版本
2025.06.09	V1.0.3	V3.5 及以上	V1.0

1.       增加附录 3 示例代码；[P10]
2.       增加附录 4 示例数据包及转换。[P21]

# 目录

接口配置	1
串口配置	1
姿态角与姿态角速度	1
上位机发送的数据包结构体	2
云台返回的数据包结构体	5
CRC 校验函数	7
附录 1 载机坐标系定义	8
附录 2 相机坐标系定义及旋转顺序	9
附录 3 示例代码	10
附录 4 示例数据包及转换	21



## 接口配置

### 串口配置

- 串口电平：TTL
- 数据位：8
- 停止位：1
- 校验位：无
- 通信模式：全双工
- 通信逻辑：应答式。上位机首先发送数据包，云台接收到正确的数据包后会发送返回数据包
- 波特率：115200
- 通信频率： $\geq 50\text{Hz}$ ，频率越高，控制效果越好。通信频率不可过低或停发数据
- 字节序：小端序（CRC 除外）

## 姿态角与姿态角速度

本文中：

相机姿态角	相机在大地坐标系 (NED) 下的欧拉角
载机姿态角	载机在大地坐标系 (NED) 下的欧拉角
相机姿态角速度	相机在大地坐标系 (NED) 下的欧拉角速度
载机姿态角速度	载机在大地坐标系 (NED) 下的欧拉角速度
相机相对角	相机在载机坐标系下的欧拉角，又名框架角或关节角
相机载机姿态差角	相机姿态角与载机姿态角的差值

载机坐标系与相机坐标系详见附录 1 与附录 2。


## 上位机发送的数据包结构体


```
#pragma pack(1)
typedef struct
{
    uint8_t sync[2]; // 注释 1
    struct
    {
        uint8_t trig:3; // 注释 2
        uint8_t value:5; // 注释 3
    } cmd;
    struct
    {
        uint8_t:3;
        int8_t fl_sens:5; // 注释 4
    } aux;
    struct
    {
        uint8_t:3;
        uint8_t go_zero:1; // 注释 5
        uint8_t wk_mode:2; // 注释 6
        uint8_t op_type:2; // 注释 7
        int16_t op_value; // 注释 8
    } gbc[3]; // 注释 9
    struct
    {
        uint8_t:7;
        uint8_t valid:1; // 注释 10
        int16_t angle[3]; // 注释 11
        int16_t accel[3]; // 注释 12
    } uav; // 注释 13
    struct
    {
        uint32_t vert_fov1x:7; // 注释 14
        uint32_t zoom_value:24; // 注释 15
        uint32_t reserved:1; // 注释 16
        float target_angle[2]; // 注释 17
    } cam;
    uint8_t crc[2]; // 注释 18
} Gcu2GbcPkt_t;
#pragma pack()
```


注释 1: 协议头 0xA9 0x5B

注释 2: 命令触发计数, 当发送的命令码与上一数据包相同时, 需改变计数值, 方可使云台重复执行此命令; 若命令码与上一数据包不同, 无论是否改变计数值, 云台都将立刻执行此命令。

注释 3: 命令码, 只发送一次, 上位机收到云台返回数据包后, 需将命令码置为 0  
0- 未定义; 1- 陀螺仪校准; 2- 启动云台; 3- 停止云台; 4- 手动控制; 5- 指点平移

 需等待云台回报温控就绪后, 方可校准陀螺仪。校准陀螺仪时需确保云台静止, 校准过程将持续数秒。

 启动云台: 启动云台电机, 增稳生效, 云台进入正常工作状态, 并非使云台开始转动或开始执行命令。云台上电后会自动启动, 仅在发送停止云台命令后, 才需要发送本命令使云台恢复正常工作。

 停止云台: 关闭云台电机, 增稳失效, 并非使云台停止转动或停止正在执行的命令。仅在需要云台处于通电状态下存储时, 才需要发送本命令。

注释 4: FPV 跟随灵敏度, [-16,15]。仅在 FPV 模式下有效。灵敏度越大, 相机随载机姿态变化的响应越快, 但消除载机晃动的幅度越小


注释 5: 此数据变化触发云台回中

注释 6: 工作模式

0- 跟随模式, 当载机姿态改变时, 相机随载机转动, 但可消除一定程度的晃动。跟随模式下控制相机载机姿态差相对角或相机姿态角速度。跟随模式下倾角保护有效;

1- 锁定模式, 当载机姿态改变时, 相机不随载机转动, 锁定当前姿态角。锁定模式下控制相机姿态角或相机姿态角速度。锁定模式下倾角保护有效;

2-FPV 模式, 当载机姿态改变时相机随载机转动, 但可消除一定程度的晃动。FPV 模式下控制相机相对角。任意一轴设为 FPV 模式时, 其余轴会自动设为 FPV 模式。FPV 模式下倾角保护无效

 倾角保护: 当云台安装平台倾斜超过保护倾角时, 云台将触发保护模式并回中, 此时云台不可控。保护倾角可在云台调试软件 GimbalConfig 上进行修改。

 俯仰轴与滚转轴请勿设置为跟随模式。



## 注释 7:控制模式

0- 角度控制 (跟随、锁定及 FPV 模式可用), 控制相机姿态角、相机载机姿态差角或相机相对角;

1- 比例角速度控制 (跟随及锁定模式可用), 控制相机姿态角速度, 但角速度与随相机倍率增大而减小, 使得不同倍率下画面移动速度一致;

2- 真实角速度控制 (跟随及锁定模式可用), 控制相机姿态角速度, 不随相机倍率变化

## 注释 8:云台控制量

跟随 + 角度控制模式下, 期望相机载机姿态差角  $= op\_value * 0.01 (deg)$ ;

跟随 + 比例角速度控制模式下, 期望相机姿态角速度  $= op\_value * 0.1 * \text{倍率系数} (deg/s)$ 。 $op\_value=0$  时, 相机跟随载机转动;

跟随 + 真实角速度控制模式下, 期望相机姿态角速度  $= op\_value * 0.1 (deg/s)$ 。 $op\_value=0$  时, 相机跟随载机转动;

锁定 + 角度控制模式下, 期望相机姿态角  $= op\_value * 0.01 (deg)$ ;

锁定 + 比例角速度控制模式下, 期望相机姿态角速度  $= op\_value * 0.1 * \text{倍率系数} (deg/s)$ 。 $op\_value=0$  时, 相机锁定当前角度;

锁定 + 真实角速度控制模式下, 期望相机姿态角速度  $= op\_value * 0.1 (deg/s)$ 。 $op\_value=0$  时, 相机锁定当前角度;

FPV+ 角度控制模式下, 期望相机相对角  $= op\_value * 0.01 (deg)$

**?** 倍率系数: 由相机 1x 倍率的垂直视场角 ( $vert\_fov1x$ ) 与相机倍率 ( $zoom\_value$ ) 计算得出,  $\text{倍率系数} = \text{arccot}(zoom\_value * \cot(0.5 * vert\_fov1x)) / 18$ 。

## 注释 9:0- 滚转;1- 俯仰;2- 偏航

gbc 结构体内的数据仅在手动控制模式下生效

## 注释 10:载机惯导数据有效标志

0- 无效;1- 有效

## 注释 11:载机姿态角, 单位 0.01deg

0- 滚转,  $[-18000, 18000]$ ; 1- 俯仰,  $[-9000, 9000]$ ; 2- 偏航,  $[-18000, 18000]$

注释 12:载机加速度, 单位 0.01m/s<sup>2</sup>

0- 北向 (北向为正); 1- 东向 (东向为正); 2- 天向 (天向为正)

注释 13:uav 结构体为云台所需的载机数据, 不提供或提供错误的数据会导致载机机动飞行时, 云台的姿态角不准确

## 注释 14:相机 1x 倍率的垂直视场角, 单位 1deg

## 注释 15:相机倍率, 单位 0.001x

## 注释 16:预留

注释 17: 目标偏移角度, 原点为画面中心, 向右 / 向下为正, 单位 1deg。为指点平移功能的控制量

0- 水平偏移角度; 1- 垂直偏移角度

注释 18: CRC16 校验

## 云台返回的数据包结构体

```
#pragma pack(1)
typedef struct
{
    uint8_t sync[2]; // 注释 19
    uint8_t fw_ver; // 注释 20
    uint8_t hw_err; // 注释 21
    uint8_t inv_flag; // 注释 22
    uint8_t gbc_stat; // 注释 23
    uint8_t tca_flag; // 注释 24
    uint8_t t;
    struct
    {
        uint8_t stat; // 注释 25
        uint8_t value; // 注释 26
    } cmd;
    int16_t cam_rate[3]; // 注释 27
    int16_t cam_angle[3]; // 注释 28
    int16_t mtr_angle[3]; // 注释 29
    uint8_t crc[2]; // 注释 30
} Gbc2GcuPkt_t;
#pragma pack()
```

注释 19:协议头 0xB5 0x9A

注释 20:固件版本号

注释 21:硬件故障, 需返厂检修

注释 22:吊装 / 立装标志

0- 吊装;1- 立装

注释 23:云台状态

0- 未定义;1- 初始化中;2- 云台停止;3- 云台保护;4- 手动控制;5- 指点平移

注释 24:温控就绪标志

0- 未就绪;1- 已就绪

 温控就绪后, 方可校准陀螺仪。

注释 25:命令执行状态

0- 执行中;1- 执行成功;2- 执行失败

注释 26:命令码回报

注释 27:相机本体角速度, 即云台 IMU 输出的角速度, 单位 0.1deg/s

0- 俯仰;1- 滚转;2- 偏航

注释 28:相机姿态角, 单位 0.01deg。角度转换详见附录 2

0- 滚转;1:俯仰;2:偏航

注释 29: 相机相对角, 单位 0.01deg。此数据由电机编码器角度解算得出, 不依赖载机惯导数据

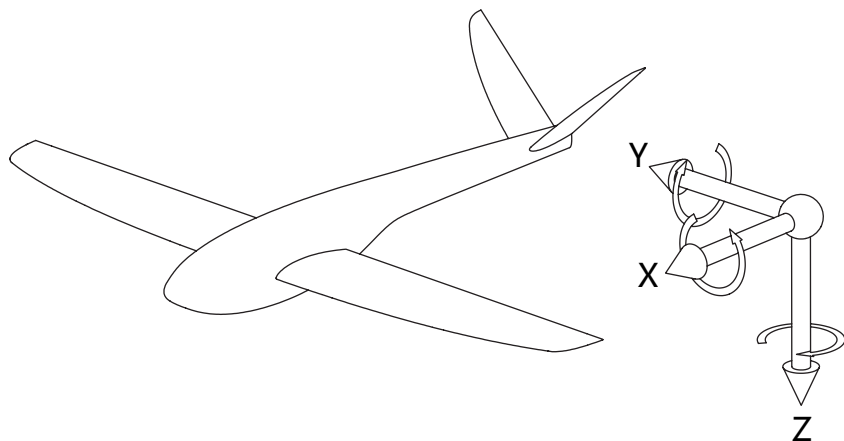
0- 俯仰;1- 滚转;2- 偏航

注释 30: CRC16 校验

## CRC 校验函数

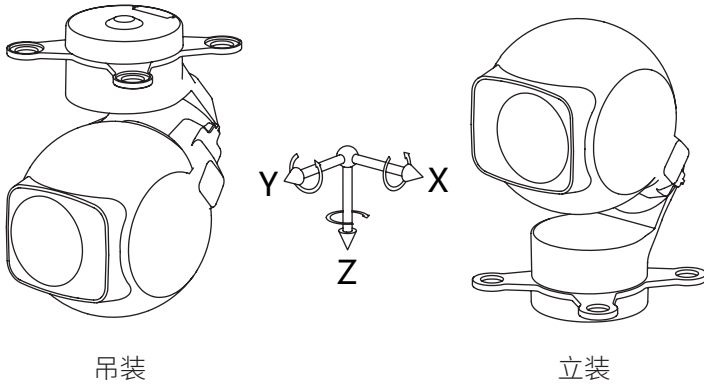
```
uint16_t CalculateCrc16(uint8_t *ptr,uint8_t len)
{
    uint16_t crc;
    uint8_t da;
    uint16_t crc_ta[16]={
        0x0000,0x1021,0x2042,0x3063,0x4084,0x50a5,0x60c6,0x70e7,
        0x8108,0x9129,0xa14a,0xb16b,0xc18c,0xd1ad,0xe1ce,0xf1ef,
    };
    crc=0;
    while(len--!=0)
    {
        da=crc>>12;
        crc<<=4;
        crc^=crc_ta[da^(*ptr>>4)];
        da=crc>>12;
        crc<<=4;
        crc^=crc_ta[da^(*ptr&0x0F)];
        ptr++;
    }
    return(crc);
}
```


## 附录 1 载机坐标系定义



## 附录 2 相机坐标系定义及旋转顺序

1. 坐标系定义: 欧拉角  $\phi$ 、 $\theta$ 、 $\psi$  的定义如下图坐标系所示。



 云台减震平台需与载机 XOY 平面平行，云台安装时请尽量靠近载机质心。

2. 旋转顺序:  $Z \rightarrow Y \rightarrow X$

3. 角度转换:

```

 $\psi += 90;$ 
cam_angle[0] =  $+\theta$ ;
cam_angle[1] =  $-\phi$ ;
cam_angle[2] =  $+\psi$ ;

```

## 附录 3 示例代码

### 1. 开发环境

示例运行环境为 Keil5 V5.25，使用 V5.06 update 6 (build 750) 编译器。

### 2. 主控芯片

型号：STM32F103C8T6（Cortex-M3，72MHz 主频）。

### 3. 外设配置

- UART1（PA9-TX，PA10-RX）：与云台通信，波特率 115200，8N1。
- UART2（PA2-TX，PA3-RX）：输出调试信息，波特率 115200，8N1。
- TIM2（50Hz 中断，用于周期性发送控制命令）。

### 4. 通信连接

UART1 连接云台：

- STM32 PA9（TX）接云台 UART\_Rx。
- STM32 PA10（RX）接云台 UART\_Rx。
- 共地。

UART2 连接电脑（调试输出）：

- STM32 PA2（TX）接 USB-TTL 模块 UART\_RX。
- 共地。

## 5. 示例代码

```

#include "stm32f10x.h"
#include <stdio.h>
#include <string.h>

// ----- 协议数据包定义 -----
#pragma pack(1)
typedef struct
{
    uint8_t sync[2]; // 协议头 0xA9 0x5B
    struct
    {
        uint8_t trig : 3; // 命令触发计数
        uint8_t value : 5; // 命令码
    } cmd;
    struct
    {
        uint8_t : 3;
        int8_t fl_sens : 5; // FPV 跟随灵敏度
    } aux;
    struct
    {
        uint8_t : 3;
        uint8_t go_zero : 1; // 回中触发
        uint8_t wk_mode : 2; // 工作模式
        uint8_t op_type : 2; // 控制模式
        int16_t op_value; // 云台控制量
    } gbc[3]; // 0- 滚转, 1- 俯仰, 2- 偏航
    struct
    {
        uint8_t : 7;
        uint8_t valid : 1; // 载机惯导数据有效标志
        int16_t angle[3]; // 载机姿态角
        int16_t accel[3]; // 载机加速度
    } uav;
    struct
    {
        uint32_t vert_fovlx : 7; // 相机 1x 垂直视场角
        uint32_t zoom_value : 24; // 相机倍率
        uint32_t reserved : 1;
        float target_angle[2]; // 目标偏移角度
    } cam;
    uint8_t crc[2]; // CRC16 校验
} Gcu2GbcPkt_t;

```



```

typedef struct
{
    uint8_t sync[2]; // 协议头 0xB5 0x9A
    uint8_t fw_ver; // 固件版本号
    uint8_t hw_err; // 硬件错误代码
    uint8_t inv_flag: 1; // 吊装 / 立装标志
    uint8_t gbc_stat: 3; // 云台状态
    uint8_t tca_flag: 1; // 温控就绪标志
    uint8_t : 3;
    struct
    {
        uint8_t stat: 3; // 命令执行状态
        uint8_t value: 5; // 命令码回报
    } cmd;
    int16_t cam_rate[3]; // 相机姿态角速度
    int16_t cam_angle[3]; // 相机姿态角
    int16_t mtr_angle[3]; // 相机相对角
    uint8_t crc[2]; // CRC16 校验
} Gbc2GcuPkt_t;
#pragma pack()
// ----- 全局变量 -----
volatile uint8_t send_flag = 0; // 50Hz 发送标志
uint8_t debug_output = 0; // 调试输出标志
Gbc2GcuPkt_t g_response; // 接收数据缓存
float ned_angle[3] = {0}; // 姿态角, 单位 1deg
// 模拟载机数据
float simulated_ned_angle[3] = {0.0f, 0.0f, 0.0f}; // 载机姿态角, 单位 1deg
float simulated_ned_accel[3] = {0.0f, 0.0f, 0.0f}; // 载机加速度, 单位 1m/s2

// ----- 函数声明 -----
void UART1_Init(void);
void UART2_Init(void);
void TIM2_Init(void);
void SendControlPacket(void);
uint16_t CalculateCrc16(uint8_t *ptr, uint8_t len);
void PrintSendFrame(const Gcu2GbcPkt_t *pkt);
void PrintGimbalStatus(const Gbc2GcuPkt_t *pkt);
int fputc(int ch, FILE *f);

```

```
// ----- 主程序 -----
int main(void)
{
    SystemInit(); // 初始化系统时钟 (72MHz)
    UART1_Init(); // UART1:云台通信
    UART2_Init(); // UART2:调试输出
    TIM2_Init(); // TIM2:50Hz 中断

    while (1)
    {
        if (send_flag)
        {
            SendControlPacket(); // 发送控制命令
            send_flag = 0;
            if (debug_output)
            {
                PrintGimbalStatus(&g_response);
                debug_output = 0;
            }
        }
    }
}

// ----- 定时器中断 (50Hz) -----
void TIM2_IRQHandler(void)
{
    if (TIM_GetITStatus(TIM2, TIM_IT_Update) != RESET)
    {
        static uint8_t count = 0;
        if (count++ % 25 == 0)
        {
            debug_output = 1; // 每 500ms 输出一次调试信息
            count %= 25; // 重置计数器
        }
        send_flag = 1; // 触发发送
        TIM_ClearITPendingBit(TIM2, TIM_IT_Update);
    }
}
```

```
// ----- 发送控制命令, 按需要设置云台模式, 发送载机数据 -----
-----
void SendControlPacket(void)
{
    Gcu2GbcPkt_t packet = {0};
    packet.sync[0] = 0xA9;
    packet.sync[1] = 0x5B;
    packet.cmd.value = 4; // 命令码 4: 手动控制
    packet.cmd.trig = 0; // 无需重复触发手动控制命令, 只需执行一次即可

    // 滚转轴: 锁定模式 + 角度控制 (0deg)
    packet.gbc[0].wk_mode = 1; // 锁定模式
    packet.gbc[0].op_type = 0; // 角度控制
    packet.gbc[0].op_value = 0;

    // 俯仰轴: 锁定模式 + 真实角速度控制 (0deg/s)
    packet.gbc[1].wk_mode = 1;
    packet.gbc[1].op_type = 2;
    packet.gbc[1].op_value = 0;

    // 偏航轴: 跟随模式 + 真实角速度控制 (0deg/s)
    packet.gbc[2].wk_mode = 0;
    packet.gbc[2].op_type = 2;
    packet.gbc[2].op_value = 0;

    // 载机数据 (模拟值)
    packet.uav.valid = 0; // 载机数据无效。如无准确真实的载机姿态角和加速度, 此数据务必置 0, 否则错误的载机数据会导致错误的云台的姿态角
    // 欧拉角转换 (deg-> 0.01deg)
    packet.uav.angle[0] = (int16_t)(simulated_ned_angle[0] * 100); // 滚转
    packet.uav.angle[1] = (int16_t)(simulated_ned_angle[1] * 100); // 俯仰
    packet.uav.angle[2] = (int16_t)(simulated_ned_angle[2] * 100); // 偏航
    // 加速度转换 (m/s2-> 0.01m/s2)
    packet.uav.accel[0] = (int16_t)(simulated_ned_accel[0] * 100); // 北向
    packet.uav.accel[1] = (int16_t)(simulated_ned_accel[1] * 100); // 东向
    packet.uav.accel[2] = (int16_t)(simulated_ned_accel[2] * 100); // 天向
}
```

```

// CRC16 校验
uint16_t crc = CalculateCrc16((uint8_t *)&packet, sizeof(packet) - 2);
packet.crc[0] = (crc>> 8) & 0xFF;
packet.crc[1] = crc& 0xFF;

for (uint8_t i = 0; i<sizeof(packet); i++)
{
    USART_SendData(USART1, *((uint8_t *)&packet + i));
    while (USART_GetFlagStatus(USART1, USART_FLAG_TXE) == RESET)
        ;
}
if (debug_output)
{
    PrintSendFrame(&packet);
}
}

// ----- UART1 接收中断 -----
void USART1_IRQHandler(void)
{
    static uint8_t rx_buffer[sizeof(Gbc2GcuPkt_t)];
    static uint8_t rx_index = 0;

    if (USART_GetITStatus(USART1, USART_IT_RXNE) != RESET)
    {
        rx_buffer[rx_index++] = USART_ReceiveData(USART1);
        // 完整接收一帧数据
        if (rx_index == sizeof(Gbc2GcuPkt_t))
        {
            Gbc2GcuPkt_t *pkt = (Gbc2GcuPkt_t *)rx_buffer;
            // 校验协议头和 CRC
            if (pkt->sync[0] == 0xB5 && pkt->sync[1] == 0x9A)
            {
                uint16_t crc = CalculateCrc16(rx_buffer, sizeof(Gbc2GcuPkt_t) - 2);
                if (pkt->crc[0] == (crc>> 8) && pkt->crc[1] == (crc& 0xFF))
                {
                    memcpy(&g_response, pkt, sizeof(Gbc2GcuPkt_t));
                }
            }
            rx_index = 0;
        }
    }
}
}

```

```
// ----- CRC16 校验函数 -----
uint16_t CalculateCRC16(uint8_t *ptr, uint8_t len)
{
    uint16_t crc = 0;
    const uint16_t crc_ta[16] = {
        0x0000, 0x1021, 0x2042, 0x3063, 0x4084, 0x50A5, 0x60C6, 0x70E7,
        0x8108, 0x9129, 0xA14A, 0xB16B, 0xC18C, 0xD1AD, 0xE1CE, 0xF1EF};
    while (len--)
    {
        uint8_t da = (crc>> 12);
        crc = (crc<< 4) ^ crc_ta[da ^ (*ptr>> 4)];
        da = (crc>> 12);
        crc = (crc<< 4) ^ crc_ta[da ^ (*ptr& 0x0F)];
        ptr++;
    }
    return crc;
}

// ----- 外设初始化 -----
void UART1_Init(void)
{
    GPIO_InitTypeDefGPIO_InitStruct;
    USART_InitTypeDefUSART_InitStruct;
    NVIC_InitTypeDefNVIC_InitStruct;

    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA | RCC_APB2Periph_
    USART1, ENABLE);

    // PA9(TX): 复用推挽输出
    GPIO_InitStruct.GPIO_Pin = GPIO_Pin_9;
    GPIO_InitStruct.GPIO_Mode = GPIO_Mode_AF_PP;
    GPIO_InitStruct.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_Init(GPIOA, &GPIO_InitStruct);

    // PA10(RX): 浮空输入
    GPIO_InitStruct.GPIO_Pin = GPIO_Pin_10;
    GPIO_InitStruct.GPIO_Mode = GPIO_Mode_IN_FLOATING;
    GPIO_Init(GPIOA, &GPIO_InitStruct);
}
```

```

// USART1 配置:115200, 8N1
USART_InitStruct.USART_BaudRate = 115200;
USART_InitStruct.USART_WordLength = USART_WordLength_8b;
USART_InitStruct.USART_StopBits = USART_StopBits_1;
USART_InitStruct.USART_Parity = USART_Parity_No;
USART_InitStruct.USART_Mode = USART_Mode_Tx | USART_Mode_Rx;
USART_InitStruct.USART_HardwareFlowControl = USART_
HardwareFlowControl_None;
USART_Init(USART1, &USART_InitStruct);

// 使能接收中断
USART_ITConfig(USART1, USART_IT_RXNE, ENABLE);
USART_Cmd(USART1, ENABLE);

// 配置中断优先级
NVIC_InitStruct.NVIC_IRQChannel = USART1_IRQn;
NVIC_InitStruct.NVIC_IRQChannelPreemptionPriority = 0;
NVIC_InitStruct.NVIC_IRQChannelSubPriority = 0;
NVIC_InitStruct.NVIC_IRQChannelCmd = ENABLE;
NVIC_Init(&NVIC_InitStruct);
}

void UART2_Init(void)
{
    GPIO_InitTypeDefGPIO_InitStruct;
    USART_InitTypeDefUSART_InitStruct;

    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE);
    RCC_APB1PeriphClockCmd(RCC_APB1Periph_USART2, ENABLE);

    // PA2(TX): 复用推挽输出
    GPIO_InitStruct.GPIO_Pin = GPIO_Pin_2;
    GPIO_InitStruct.GPIO_Mode = GPIO_Mode_AF_PP;
    GPIO_InitStruct.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_Init(GPIOA, &GPIO_InitStruct);

    // USART2 配置:115200, 8N1
    USART_InitStruct.USART_BaudRate = 115200;
    USART_InitStruct.USART_WordLength = USART_WordLength_8b;
    USART_InitStruct.USART_StopBits = USART_StopBits_1;
    USART_InitStruct.USART_Parity = USART_Parity_No;
    USART_InitStruct.USART_Mode = USART_Mode_Tx;
    USART_InitStruct.USART_HardwareFlowControl = USART_
HardwareFlowControl_None;
    USART_Init(USART2, &USART_InitStruct);
    USART_Cmd(USART2, ENABLE);
}

```

```

void TIM2_Init(void)
{
    TIM_TimeBaseInitTypeDefTIM_InitStruct;
    NVIC_InitTypeDefNVIC_InitStruct;

    RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM2, ENABLE);

    // 配置定时器为 50Hz (72MHz 主频)
    TIM_InitStruct.TIM_Prescaler = 7200 - 1; // 72MHz / 7200 = 10kHz
    TIM_InitStruct.TIM_Period = 200 - 1; // 10kHz / 200 = 50Hz
    TIM_InitStruct.TIM_CounterMode = TIM_CounterMode_Up;
    TIM_InitStruct.TIM_ClockDivision = TIM_CKD_DIV1;
    TIM_TimeBaseInit(TIM2, &TIM_InitStruct);

    // 使能中断
    TIM_ITConfig(TIM2, TIM_IT_Update, ENABLE);
    TIM_Cmd(TIM2, ENABLE);

    // 配置中断优先级
    NVIC_InitStruct.NVIC_IRQChannel = TIM2_IRQn;
    NVIC_InitStruct.NVIC_IRQChannelPreemptionPriority = 1;
    NVIC_InitStruct.NVIC_IRQChannelSubPriority = 0;
    NVIC_InitStruct.NVIC_IRQChannelCmd = ENABLE;
    NVIC_Init(&NVIC_InitStruct);
}

// ----- 工具函数 -----
int fputc(int ch, FILE *f)
{
    USART_SendData(USART2, (uint8_t)ch);
    while (USART_GetFlagStatus(USART2, USART_FLAG_TXE) == RESET)
        ;
    return ch;
}

// ----- 调试信息输出 -----
void PrintSendFrame(const Gcu2GbcPkt_t *pkt)
{
    printf("\nSend frame: ");
    for (uint8_t i = 0; i < sizeof(Gcu2GbcPkt_t); i++) {
        printf("%02X ", ((uint8_t *)pkt)[i]);
    }
}

```

```

printf("\n");
printf("Parse send frame: \n");
printf("cmdTrig : %-1d\n", pkt->cmd.trig);
printf("cmdValue : %-2d\n", pkt->cmd.value);
printf("FPV Sens : %-2d\n", pkt->aux.fl_sens);

// 三轴控制信息
const char *axis_names[] = {" 滚转 ", " 俯仰 ", " 偏航 "};
const char *wk_mode[] = {" 跟随模式 ", " 锁定模式 ", "FPV 模式 "};
const char *op_type[] = {" 角度控制 ", " 比例角速度控制 ", " 真实角速度控制 "};

for (uint8_t i = 0; i < 3; i++) {
    printf("%-5s : ", axis_names[i]);
    printf("go_zero=%-1d ", pkt->gbc[i].go_zero);
    printf("wk_mode=%-10s ", wk_mode[pkt->gbc[i].wk_mode < 2 ? pkt-
>gbc[i].wk_mode : 2]);
    printf("op_type=%-10s ", op_type[pkt->gbc[i].op_type < 3 ? pkt->gbc[i].op_
type : 3]);
    printf("op_value=%-2d \n", pkt->gbc[i].op_value);
}

// 载机信息
printf("uav valid: %-1d\n", pkt->uav.valid);
printf("uav Angle: [%6.2f, %6.2f, %6.2f] deg\n",
    pkt->uav.angle[0]*0.01f, pkt->uav.angle[1]*0.01f, pkt->uav.angle[2]*0.01f);
printf("uav Accel: [%6.2f, %6.2f, %6.2f] m/s²\n",
    pkt->uav.accel[0]*0.01f, pkt->uav.accel[1]*0.01f, pkt->uav.accel[2]*0.01f);

// 相机信息
printf("cam vert_fovlx : %-3d\n", pkt->cam.vert_fovlx);
printf("cam zoom_value : %-6d\n", pkt->cam.zoom_value);
printf("cam target_angle : [%6.2f, %6.2f] deg\n",
    pkt->cam.target_angle[0], pkt->cam.target_angle[1]);
}

void PrintGimbalStatus(const Gbc2GcuPkt_t *pkt)
{
    printf("\nRecv frame: ");
    for (uint8_t i = 0; i < sizeof(Gbc2GcuPkt_t); i++)
    {
        printf("%02X ", ((uint8_t *)pkt)[i]);
    }
}

```



```
printf("\n");
printf("Parse recv frame: \n");
printf("FW Ver: %-2d\n", pkt->fw_ver);
printf("HW Err: %-2d\n", pkt->hw_err);
printf("Inv Flag: %-1d\n", pkt->inv_flag);
printf("GBC Stat: %-1d\n", pkt->gbc_stat);
printf("TCA Flag: %-1d\n", pkt->tca_flag);
printf("Cmd Stat: %-1d\n", pkt->cmd.stat);
printf("Cmd Value: %-2d\n", pkt->cmd.value);

printf("Cam Rate: [%6d , %6d , %6d]\n",
       pkt->cam_rate[0],
       pkt->cam_rate[1],
       pkt->cam_rate[2]);

printf("Cam Angle: [%6d , %6d , %6d]\n",
       pkt->cam_angle[0],
       pkt->cam_angle[1],
       pkt->cam_angle[2]);

printf("Mtr Angle: [%6d , %6d , %6d]\n",
       pkt->mtr_angle[0],
       pkt->mtr_angle[1],
       pkt->mtr_angle[2]);

for (uint8_t i = 0; i < 3; i++)
    ned_angle[i] = pkt->cam_angle[i] * 0.01f;

printf("NED Angle: [%6.2f , %6.2f , %6.2f]\n\n",
       ned_angle[0],
       ned_angle[1],
       ned_angle[2]);
}
```

## 附录 4 示例数据包及转换

### 示例 1

上位机发送的完整数据包为：

A9 5B 20 00 10 00 00 90 00 00 80 00 61 BA

字节	内容	原始数据	转换精度 或转为二进制	转为 16 进制 (小端序)
0	协议头	0xA9	-	A9
1		0x5B	-	5B
2	[7:3] 命令码	4- 手动控制	0b00100	20
	[2:0] 命令触发计数		0b000	
3	[7:3] FPV 跟随灵敏度	0	0b000000	00
	[2:0]		0b	
4	[7:6] 云台滚转轴控制模式	0- 角度控制	0b00	10
	[5:4] 云台滚转轴工作模式	1- 锁定模式	0b01	
	[3] 云台滚转轴回中触发计数		0b0	
	[2:0]		0b000	
5~6	云台滚转轴控制量	0deg	0	00 00
7	[7:6] 云台俯仰轴控制模式	2- 真实角速度控制	0b10	90
	[5:4] 云台俯仰轴工作模式	0- 锁定模式	0b01	
	[3] 云台俯仰轴回中触发计数	-	0b0	
	[2:0]	-	0b000	
8~9	云台俯仰轴控制量	0deg/s	0	00 00
10	[7:6] 云台偏航轴控制模式	2- 真实角速度控制	0b10	80
	[5:4] 云台偏航轴工作模式	0- 跟随模式	0b00	
	[3] 云台偏航轴回中触发计数	-	0b0	
	[2:0]	-	0b000	
11~12	云台偏航轴控制量	0deg/s	0	00 00
13	[7] 载机惯导数据有效标志	无效	0b0	00
	[6:0]	-	0b00000000	
14~15	载机滚转姿态角	0deg	0	00 00
16~17	载机俯仰姿态角	0deg	0	00 00
18~19	载机偏航姿态角	0deg	0	00 00
20~21	载机北向加速度	0m/s <sup>2</sup>	0	00 00
22~23	载机东向加速度	0m/s <sup>2</sup>	0	00 00
24~25	载机天向加速度	0m/s <sup>2</sup>	0	00 00
	[31] 预留	-	0b0	00 00 00 00
26~29	[30:7] 相机倍率	0x	0b000000000000 000000000000	
	[6:0] 相机 1x 倍率垂直视场角	0deg	0b00000000	
30~33	目标水平偏移角度	0deg	0	00 00 00 00
34~37	目标垂直偏移角度	0deg	0	00 00 00 00
38~39	CRC	-	-	61 BA

## 示例 2

上位机发送的完整数据包为：

A9 5B 20 00 90 00 00 90 0A 00 90 00 00 80 F4 01 F4 01 F4 01 0A 00 0A 00 0A 00 00 00 00 00 00 00 00 00 00 00 98 DB

字节	内容	原始数据	转换精度 或转为二进制	转为 16 进制 (小端序)
0	协议头	0xA9	-	A9
1		0x5B	-	5B
2	[7:3] 命令码	4- 手动控制	0b00100	20
	[2:0] 命令触发计数		0b000	
3	[7:3] FPV 跟随灵敏度	0	0b000000	00
	[2:0]		0b000	
4	[7:6] 云台滚转轴控制模式	2- 真实角速度控制	0b10	90
	[5:4] 云台滚转轴工作模式	1- 锁定模式	0b01	
	[3] 云台滚转轴回中触发计数		0b0	
	[2:0]		0b000	
5~6	云台滚转轴控制量	0deg/s	0	00 00
7	[7:6] 云台俯仰轴控制模式	2- 真实角速度控制	0b10	90
	[5:4] 云台俯仰轴工作模式	1- 锁定模式	0b01	
	[3] 云台俯仰轴回中触发计数	-	0b0	
	[2:0]	-	0b000	
8~9	云台俯仰轴控制量	1deg/s	10	0A 00
10	[7:6] 云台偏航轴控制模式	2- 真实角速度控制	0b10	90
	[5:4] 云台偏航轴工作模式	1- 锁定模式	0b01	
	[3] 云台偏航轴回中触发计数	-	0b0	
	[2:0]	-	0b000	
11~12	云台偏航轴控制量	0deg/s	0	00 00
13	[7] 载机惯导数据有效标志	有效	0b1	80
	[6:0]	-	0b00000000	
14~15	载机滚转姿态角	5deg	500	F4 01
16~17	载机俯仰姿态角	5deg	500	F4 01
18~19	载机偏航姿态角	5deg	500	F4 01
20~21	载机北向加速度	0.1m/s <sup>2</sup>	10	0A 00
22~23	载机东向加速度	0.1m/s <sup>2</sup>	10	0A 00
24~25	载机天向加速度	0.1m/s <sup>2</sup>	10	0A 00
	[31] 预留	-	0b0	00 00 00 00
26~29	[30:7] 相机倍率	0x	0b000000000000 00000000000000	
	[6:0] 相机 1x 倍率垂直视场角	0deg	0b00000000	
30~33	目标水平偏移角度	0deg	-	00 00 00 00
34~37	目标垂直偏移角度	0deg	-	00 00 00 00
38~39	CRC	-	-	98 DB

## 示例 3

云台返回的完整数据包为：

B5 9A 24 00 19 21 FF FF 00 00 FE FF 09 00 8C 35 B9 01 52 C9 5F 00 FE FF 52 CD

字节	内容	原始数据（16 进制）	解析数据
0	协议头	B5	0xB5
1		9A	0x9A
2	固件版本号	24	36
3	硬件错误代码	00	0
4	[7:5]	19	0b000
	[4] 温控就绪标志		0b1 1- 已就绪
	[3:1] 云台状态		0b100 4- 手动控制
	[0] 吊装 / 立装标志		0b0 吊装
5	[7:3] 命令码回报	21	0b00100 4- 手动控制
	[2:0] 命令执行状态		0b001 1- 执行成功
6~7	相机俯仰轴本体角速度	FF FF	-0.1deg/s
8~9	相机滚转轴本体角速度	00 00	0deg/s
10~11	相机偏航轴本体角速度	FE FF	-0.2deg/s
11~12	相机滚转轴姿态角	09 00	0.09deg
13~14	相机俯仰轴姿态角	8C 35	137.08deg
15~16	相机偏航轴姿态角	B9 01	4.41deg
17~18	相机俯仰轴相对角	52 C9	-139.98deg
19~20	相机滚转轴相对角	5F 00	0.95deg
21~22	相机偏航轴相对角	FE FF	-0.02deg
38~39	CRC	52 CD	0x52 0xCD