G8T Serial Communication Protocol

1 Communication Format

The communication format is shown in Table 1.3.

Table 1.3 Protocol format

SOF	VER_LEN	CHECKSUM	SEQ	STS	SENDER	RECEIVER	CMD_SET	CMD_ID	DATA	CRC16
1-byte	2-byte	1-byte	2-byte	1-byte	1-byte	1-byte	1-byte	1-byte	N-byte	2-byte

In the v1 communication format shown in the table above, the definitions of each field are as follows：

SOF: 1-byte，Frame Header，Fixed to 0xAA；

VER_LEN: 2-byte，the protocol version number and frame length, where：

bit9~bit0：frame length, the number of bytes in the entire frame；

bit15~bit10：protocol version number, currently fixed at 1；

CHECKSUM: 1-byte，check byte，checksum of "SOF+VER&LEN"；

SEQ: 2-byte， command sequence number, number the commands to avoid repeated execution of commands；

STS: 1-byte，status word, where：

bit3~bit0：reserved；

bit4：0 - DATA field is not encrypted, 1 - DATA field is encrypted；

bit7~bit5：reserved；

SENDER: 1-byte, sender number, where:

bit2~bit0: index;

bit7~bit3: device；

RECEIVER: 1-byte, receiver number, where:

bit2~bit0: index;

bit7~bit3: device；

CMD_SET: 1-byte, command set, indicating which command set the current command belongs to, where:

0x00: general command set;

0x01: gimbal command set;

0x02: flight control command set;

0xff~0x03: reserved；

CMD_ID: 1-byte，corresponding to the command ID in the command set, the specific command to be executed

DATA: N-byte，command data；

CRC16: 2-byte, Check byte, perform CRC16 check on the data

"SOF+VER_LEN+CRC8+SEQ+STS+SEND+RECEIVE
+CMD_SET+CMD_ID+DATA"；

Define "SOF+VER_LEN+CRC8+SEQ+STS+SEND+RECEIVE+CMD_SET+CMD_ID" as "V1_HEADER", then Table 1.3 can be simplified to：

Table 1.4 simplified v1 protocol format

V1_HEADER	DATA	CRC16
11-byte	N-byte	2-byte

2 General Command Set(0x00)

2.1 Soft reset(0x00)

CMD_SET: 0x00

CMD_ID: 0x00

Structure definition：

```
typedef struct
{
    header_v1_struct header;
    uint16_t crc16;
}reboot_request_struct;
```

2.2 Request version number(0x01)

CMD_SET: 0x00

CMD_ID: 0x01

Structure definition：

```
typedef struct
{
    header_v1_struct header;
    uint16_t crc16;
}version_request_struct;
```

2.3 Response to get version number(0x02)

CMD_SET: 0x00

CMD_ID: 0x02

Structure definition：

```
typedef struct
{
    header_v1_struct header;
    uint32_t loader_ver;
    uint32_t firmware_ver;
    uint32_t firmware_size;
    uint32_t firmware_crc32;
    uint16_t crc16;
}version_reply_struct;
```

DATA Field Definition：

loader_ver: 4-byte，boot version number；

firmware_ver: 4-byte，firmware version number；

firmware_size: 4-byte，firmware size, in bytes；

firmware_crc32: 4-byte，firmware checksum；

2.4 Request to erase firmware(0x03)

CMD_SET: 0x00

CMD_ID: 0x03

Structure definition：

```
typedef struct
{
    header_v1_struct header;
    uint16_t crc16;
}erase_app_request_struct;
```

2.5 Response Erase Firmware(0x04)

CMD_SET: 0x00

CMD_ID: 0x04

Structure definition：

```
typedef struct
{
    header_v1_struct header;
    uint16_t crc16;
}erase_app_reply_struct;
```

2.6 Request to start upgrade(0x0b)

CMD_SET: 0x00

CMD_ID: 0x0b

Structure definition：

```
typedef struct
{
    header_v1_struct header;
    uint8_t type;
    uint32_t data_size;
    uint16_t crc16;
}start_upgrade_request_struct;
```

DATA Field Definition：

type: 1-byte，reserved；

data_size: 4-byte，the size of the .bin file used for upgrading, in bytes；

2.7 Response to start upgrading(0x0c)

CMD_SET: 0x00

CMD_ID: 0x0c

Structure definition：

```
typedef struct
{
    header_v1_struct header;
    uint8_t data_size_limit;
    uint16_t crc16;
}start_upgrade_reply_struct;
```

DATA Field Definition：

data_size_limit: 1-byte，whether the upgrade file size exceeds the limit, where：

0 - no limit;

1 - exceeded the limit；

2.8 Request to upgrade data(0x0d)

CMD_SET: 0x00

CMD_ID: 0x0d

Structure definition：

```
typedef struct
{
    header_v1_struct header;
    uint8_t data[64];
    uint16_t crc16;
}upgrading_request_struct;
```

DATA Field Definition：

data: 64-byte，Upgrade file package；

2.9 Response upgrade data(0x0e)

CMD_SET: 0x00

CMD_ID: 0x0e

Structure definition：

```
typedef struct
{
    header_v1_struct header;
    uint16_t crc16;
}upgrading_reply_struct;
```

2.10 Request to end upgrade(0x0f)

CMD_SET: 0x00

CMD_ID: 0x0f

Structure definition：

```
typedef struct
```

```
{
    header_v1_struct header;
    struct
    {
        uint8_t signature : 1;
        uint8_t verify : 1;
        uint8_t reserved : 6;
    }cfg;
    uint16_t crc16;
}stop_upgrade_request_struct;
```

DATA Field Definition：

cfg: 1-byte, upgrade configuration word, where:

bit0: 0 - without digital signature, 1 - with digital signature;

bit1: 0 - without checksum, 1- with checksum;

bit7~bit2: reserved；

2.11 Response end upgrade(0x10)

CMD_SET: 0x00

CMD_ID: 0x10

Structure definition：

```
typedef struct
{
    header_v1_struct header;
    struct
    {
        uint8_t signature_matched : 1;
        uint8_t md5_matched : 1;
        uint8_t reserved : 6;
    }status;
    uint16_t crc16;
}stop_upgrade_reply_struct;
```

DATA Field Definition：

status: 1-byte, upgrade status word, where:

bit0: 0 - digital signature does not match, 1 - digital signature matches;

bit1: 0 - verification does not match, 1- verification matches;

bit7~bit2: reserved;

3 Gimbal Command Set(0x01)

3.1 Heartbeat packet(0x00)

CMD_SET: 0x01

CMD_ID: 0x00

Structure definition：

```
typedef struct
{
    header_v1_struct header;
    int16_t gb_pitch;
    int16_t gb_roll;
    int16_t gb_yaw;
    struct
    {
        uint32_t atti_ready : 1;
        uint32_t joint_init_ok : 1;
        uint32_t under_volt : 1;
        uint32_t mems_lost : 1;
        uint32_t actuator : 2;
        uint32_t soft_limit : 1;
        uint32_t ctrl_mode : 2;
        uint32_t state_machine : 4;
        uint32_t reserve : 19;
    }status;
    int16_t joint_outter_angle;
    int16_t joint_middle_angle;
    int16_t joint_inner_angle;
    int16_t joint_outter_speed;
    int16_t joint_middle_speed;
    int16_t joint_inner_speed;
    int16_t imu_w_x;
    int16_t imu_w_y;
    int16_t imu_w_z;
    uint8_t reserve[10];
    uint16_t crc16;
}gb_heartbeat_push_struct;
```

DATA Field Definition：

gb_pitch: 2-byte， gimbal attitude angle, unit: 0.01 degrees, the coordinate system and Euler angle sequence must be agreed upon in advance；

gb_roll: 2-byte， gimbal attitude angle, unit: 0.01 degrees, the coordinate system and Euler angle sequence must be agreed upon in advance；

gb_yaw: 2-byte， gimbal attitude angle, unit: 0.01 degrees, the coordinate system and Euler angle sequence must be agreed upon in advance；

status: 4-byte, gimbal status,

bit0: 0-attitude not ready, 1-attitude ready;

bit1: 0-joint angle initialization not completed, 1-joint initialization completed;

bit2: 0-voltage normal, 1-undervoltage;

bit3: 0-IMU normal, 1-IMU missing;

bit5~bit4: 0-joint normal drive, 1-joint stalled, 2-joint closed;

bit6: 0-not reached software limit, 1-reached software limit;

bit8~bit7: 0-torque mode, 1-speed mode, 2-position mode, 3-attitude mode;

bit12~bit9: state machine;

bit31-bit13: reserved;

joint_outter_angle: 2-byte，outer frame angle, unit: 0.01 degree；

joint_middle_angle: 2-byte，middle frame angle, unit: 0.01 degree；

joint_inner_angle: 2-byte，inner frame angle, unit: 0.01 degree；

joint_outter_speed: 2-byte，outer frame angular velocity, unit: 0.01 degree/s；

joint_middle_speed: 2-byte，middle frame angular velocity, unit: 0.01 degree/s；

joint_inner_speed: 2-byte，inner frame angular velocity, unit: 0.01 degree/s；

imu_w_x: 2-byte，gyroscope x-axis angular velocity, unit: 0.01 degree/s；

imu_w_y: 2-byte，gyroscope y-axis angular velocity, unit: 0.01 degree/s；

imu_w_z: 2-byte，gyroscope z-axis angular velocity, unit: 0.01 degree/s；

reserve: 10-byte，reserved；

3.2 Log package(0xff)

CMD_SET: 0x01

CMD_ID: 0xff

Structure definition：

```
typedef struct
{
    header_v1_struct header;
    uint16_t data[240];
    uint16_t crc16;
}gb_log_push_struct;
```

DATA Field Definition：

data: 480-byte，log data；

4 Flight Control Command Set(0x02)

4.1 Heartbeat packet(0x00)

CMD_SET: 0x02

CMD_ID: 0x00

Structure definition：

```
typedef struct
{
    header_v1_struct header;
    int16_t fcs_pitch;
    int16_t fcs_roll;
    int16_t fcs_yaw;
    int16_t fcs_yaw_ref;
    int16_t fcs_motion_accx;
    int16_t fcs_motion_accy;
    int16_t fcs_motion_accz;
    uint8_t gb_ctrl_mode;
    int16_t gb_pitch_ref;
    int16_t gb_roll_ref;
    int16_t gb_yaw_ref;
    uint8_t fcs_state;
    int16_t fcs_yaw_w_ref;
    int32_t latitude;     //Actual latitude and longitude*10000000
    int32_t longitude;    //Actual latitude and longitude*10000000
    float altitude;       //Float
    uint8_t gb_yaw_lock_en;
    uint16_t crc16;
}fcs_heartbeat_push_struct;
```

DATA Field Definition：

fcs_pitch: 2-byte，Flight control attitude angle, unit: 0.01 degrees, the coordinate system and Euler angle order must be agreed in advance；

fcs_roll: 2-byte，Flight control attitude angle, unit: 0.01 degrees, the coordinate system and Euler angle order must be agreed in advance；

fcs_yaw: 2-byte，Flight control attitude angle, unit: 0.01 degrees, the coordinate system and Euler angle order must be agreed in advance；

fcs_yaw_ref: 2-byte，Flight control heading command, unit: 0.01 degrees, this command will be regarded as the gimbal heading command；

fcs_motion_accx: 2-byte，Flight control motion acceleration, unit: 0.001m2/s, the coordinate system needs to be agreed in advance；

fcs_motion_accy: 2-byte，Flight control motion acceleration, unit: 0.001m2/s, the coordinate system needs to be agreed in advance；

fcs_motion_accz: 2-byte，Flight control motion acceleration, unit: 0.001m2/s, the coordinate system needs to be agreed in advance；

gb_ctrl_mode: 1-byte, gimbal control mode, 0-no control, 1-speed control (attitude mode), 2-angle control (attitude mode), 3-return to center (attitude mode), 4-gimbal release, 5-angle control (position mode)；

gb_pitch_ref: 2-byte, when the gimbal is in angle control (attitude mode), it indicates the angle command, unit: 0.01 degree; when the gimbal is in speed control (attitude mode), it indicates the speed command, unit: 0.01 degree/s; when the gimbal is in angle control (position mode), it indicates the internal frame angle command, unit: 0.01 degree；

gb_roll_ref: 2-byte, when the gimbal is in angle control (attitude mode), it indicates the angle command, unit: 0.01 degree; when the gimbal is in speed control (attitude mode), it indicates the speed command, unit: 0.01 degree/s; when the gimbal is in angle control (position mode), it indicates the mid-frame angle command, unit: 0.01 degree；

gb_yaw_ref: 2-byte, when the gimbal is in angle control (attitude mode), it indicates the angle command, unit: 0.01 degree; when the gimbal is in speed control (attitude mode), it indicates the speed command, unit: 0.01 degree/s; when the gimbal is in angle control (position mode), it indicates the outer frame angle command, unit: 0.01 degree；

fcs_state: 1-byte, when bit0 is 0, the aircraft has not taken off, when bit0 is 1, the aircraft has taken off;

bit1~bit7, reserved；

fcs_yaw_w_ref: byte，Flight control heading speed command, unit: 0.01 degrees/s；

latitude: 4-byte，reserved；

longitude: 4-byte，reserved；

altitude: 4-byte，reserved；

gb_yaw_lock_en: 1-byte，reserved；

5 Infrared Camera Command Set(0x03)

5.1 Color palette switch(0x00)

CMD_SET: 0x03

CMD_ID: 0x00

Structure definition：

```
typedef struct
{
    header_v1_struct header;
    uint8_t color_palette;
    uint16_t crc16;
}thermal_camera_switch_color_palette_struct;
```

DATA Field Definition：

color_palette	0x00: White heat (default)
	0x01: Black heat
	0x03: Rainbow
	0x06: Iron red
	0x09: Lava
	0x0a: Ice and fire

5.2 Color palette switch reply(0x01)

CMD_SET: 0x03

CMD_ID: 0x01

Structure definition：

```
typedef struct
{
    header_v1_struct header;
    uint8_t ack;
    uint16_t crc16;
}thermal_camera_switch_color_palette_ack_struct;
```

DATA Field Definition：

ack	0x00: The color palette switch command was successfully received
	0x01: The color palette switch command was successfully executed
	0x02: The color palette switch command failed to execute

6 Tracking Camera Command Set(0x05)

6.1 Take a photo(0x00)

CMD_SET: 0x05

CMD_ID: 0x00

Structure definition：

```
typedef struct
{
```

```
    header_v1_struct header;
    uint16_t crc16;
}tracking_camera_photo_struct;
```

6.2 Reply take a photo (0x01)

CMD_SET: 0x05

CMD_ID: 0x01

Structure definition：

```
typedef struct
{
    header_v1_struct header;
    uint8_t ack;
    uint16_t crc16;
}tracking_camera_photo_ack_struct;
```

DATA Field Definition：

ack	0x00: Successfully received the photo command
	0x01: Photo execution successful
	0x02: Photo execution failed

6.3 Record(0x02)

CMD_SET: 0x05

CMD_ID: 0x02

Structure definition：

```
typedef struct
{
    header_v1_struct header;
    uint8_t action;
    uint16_t crc16;
}tracking_camera_record_struct;
```

DATA Field Definition：

action	0x00: Stop recording
	0x01: Start recording

6.4 Reply Record(0x03)

CMD_SET: 0x05

CMD_ID: 0x03

Structure definition：

```
typedef struct
{
    header_v1_struct header;
```

```
    uint8_t ack;
    uint16_t crc16;
}tracking_camera_record_ack_struct;
```

DATA Field Definition：

ack	0x00: Recording command received successfully
	0x01: Recording executed successfully
	0x02: Recording executed failed

6.5 Visible Light Electronic Zoom(0x04)

CMD_SET: 0x05

CMD_ID: 0x04

Structure definition：

```
typedef struct
{
    header_v1_struct header;
    uint8_t mode;
    uint16_t zoom;
    uint16_t crc16;
}tracking_camera_visible_zoom_struct;
```

DATA Field Definition：

mode	0: Thermal image electronic zoom +
	1: Thermal image electronic zoom －
	2: Thermal image electronic zoom Specified parameter
zoom	Indicates the specified multiple value, the specified zoom multiple = input value * 0.1, range (10~80)

6.6 Reply visible light electronic zoom(0x05)

CMD_SET: 0x05

CMD_ID: 0x05

Structure definition：

```
typedef struct
{
    header_v1_struct header;
    uint8_t ack;
    uint16_t crc16;
}tracking_camera_visible_zoom_ack_struct;
```

DATA Field Definition：

ack	0x00: Visible light zoom command received successfully
	0x01: Visible light zoom command executed successfully

0x02: Visible light zoom command failed to execute

6.7 Infrared electronic zoom(0x06)

CMD_SET: 0x05

CMD_ID: 0x06

Structure definition：

```
typedef struct
{
    header_v1_struct header;
    uint8_t mode;
    uint16_t zoom;
    uint16_t crc16;
}tracking_camera_thermal_zoom_struct;
```

DATA Field Definition：

mode	0: Thermal image electronic zoom +
	1: Thermal image electronic zoom －
	2: Thermal image electronic zoom Specified parameter
zoom	Indicates the specified multiple value, the specified zoom multiple = input value * 0.1, range (10~80)

6.8 Reply infrared electronic zoom (0x07)

CMD_SET: 0x05

CMD_ID: 0x07

Structure definition：

```
typedef struct
{
    header_v1_struct header;
    uint8_t ack;
    uint16_t crc16;
}tracking_camera_thermal_zoom_ack_struct;
```

DATA Field Definition：

ack	0x00: Successfully received the infrared zoom command
	0x01: The infrared zoom command was successfully executed
	0x02: The infrared zoom command failed to execute

6.9 Video output type settings(0x08)

CMD_SET: 0x05

CMD_ID: 0x08

Structure definition：

```
typedef struct
{
    header_v1_struct header;
    uint8_t video_type;
    uint16_t crc16;
}tracking_camera_video_type_struct;
```

DATA Field Definition：

video_type	0x00: White light
	0x01: Thermal imaging
	0x02: White light (full screen) + Thermal imaging (small)
	0x03: Thermal imaging (full screen) + White light (small)

6.10 Reply video output type settings(0x09)

CMD_SET: 0x05

CMD_ID: 0x09

Structure definition：

```
typedef struct
{
    header_v1_struct header;
    uint8_t ack;
    uint16_t crc16;
}tracking_camera_video_type_ack_struct;
```

DATA Field Definition：

ack	0x00: Successful video output type command
	0x01: Successful execution of video output type command
	0x02: Failed execution of video output type command

6.11 IP address settings(0x0A)（Restart to take effect）

CMD_SET: 0x05

CMD_ID: 0x0A

Structure definition：

```
typedef struct
{
    header_v1_struct header;
    uint8_t ip1;
    uint8_t ip2;
    uint8_t ip3;
    uint8_t ip4;
    uint16_t crc16;
```

```
}tracking_camera_set_ip_struct;
```

DATA Field Definition：

ip1	IPV4 the first field of the IP address, such as 192
Ip2	IPV4 the second field of the IP address, such as 168
Ip3	IPV4 the third field of the IP address, such as 1
Ip4	IPV4 the fourth field of the IP address, such as2

6.12 Reply IP address settings(0x0B)

CMD_SET: 0x05

CMD_ID: 0x0B

Structure definition：

```
typedef struct
{
    header_v1_struct header;
    uint8_t ack;
    uint16_t crc16;
}tracking_camera_set_ip_ack_struct;
```

DATA Field Definition：

ack	0x00: The set IP command was received successfully
	0x01: The set IP command was executed successfully
	0x02: The set IP command failed to execute

6.13 Query infrared zoom(0x0C)

CMD_SET: 0x05

CMD_ID: 0xC

Structure definition：

```
typedef struct
{
    header_v1_struct header;
    uint16_t crc16;
}tracking_camera_query_thermal_zoom_struct;
```

6.14 Reply query infrared zoom(0x0D)

CMD_SET: 0x05

CMD_ID: 0x0D

Structure definition：

```
typedef struct
{
    header_v1_struct header;
```

```
    uint8_t ack;
    uint16_t zoom;
    uint16_t crc16;
}tracking_camera_query_thermal_zoom_ack_struct;
```

DATA Field Definition：

ack	0x00: Successfully received the infrared zoom query command
	0x01: Successfully executed the infrared zoom query command
	0x02: Failed to execute the infrared zoom query command
zoom	Multiple*10

6.15 Query visible light zoom(0x0E)

CMD_SET: 0x05

CMD_ID: 0x0E

Structure definition：

```
typedef struct
{
    header_v1_struct header;
    uint16_t crc16;
}tracking_camera_query_visable_zoom_struct;
```

6.16 Reply query visible light zoom(0x0F)

CMD_SET: 0x05

CMD_ID: 0x0F

Structure definition：

```
typedef struct
{
    header_v1_struct header;
    uint8_t ack;
    uint16_t zoom;
    uint16_t crc16;
}tracking_camera_query_visable_zoom_ack_struct;
```

DATA Field Definition：

ack	0x00: The visible light zoom query command was successfully received
	0x01: The visible light zoom query command was successfully executed
	0x02: The visible light zoom query command failed to execute
zoom	Multiple*10

6.17 Query SD card status(0x10)

CMD_SET: 0x05

CMD_ID: 0x10

Structure definition：

```
typedef struct
{
    header_v1_struct header;
    uint16_t crc16;
}tracking_camera_query_sd_struct;
```

6.18 Reply query SD card status(0x11)

CMD_SET: 0x05

CMD_ID: 0x11

Structure definition：

```
typedef struct
{
    header_v1_struct header;
    uint8_t ack;
    uint16_t sd_state;
    uint16_t crc16;
}tracking_camera_query_sd_ack_struct;
```

DATA Field Definition：

ack	0x00: Successfully received the query SD card status command
	0x01: Successfully executed the query SD card status command
	0x02: Failed to execute the query SD card status command
sd_state	0x00 SD card not detected
	0x01 SD card is present and normal
	0x02 SD card full
	0x03 SD card abnormal error

6.19 Query recording status(0x12)

CMD_SET: 0x05

CMD_ID: 0x12

Structure definition：

```
typedef struct
{
    header_v1_struct header;
    uint16_t crc16;
}tracking_camera_query_record_struct;
```

6.20 Reply query recording status(0x13)

CMD_SET: 0x05

CMD_ID: 0x13

Structure definition：

```
typedef struct
{
    header_v1_struct header;
    uint8_t ack;
    uint16_t record_second;
    uint16_t crc16;
}tracking_camera_query_record_ack_struct;
```

DATA Field Definition：

ack	0x00: Successfully received the query video status command
	0x01: The query video status command was executed successfully
	0x02: The query video status command failed to execute
record_second	The number of seconds of recording, 0 means recording has not started

6.21 Query the video output mode(0x14)

CMD_SET: 0x05

CMD_ID: 0x14

Structure definition：

```
typedef struct
{
    header_v1_struct header;
    uint8_t color_palette;
    uint16_t crc16;
}tracking_camera_query_video_type_struct;
```

6.22 Reply query the video output mode(0x15)

CMD_SET: 0x05

CMD_ID: 0x15

Structure definition：

```
typedef struct
{
    header_v1_struct header;
    uint8_t ack;
    uint16_t video_type;
    uint16_t crc16;
}tracking_camera_video_type_ack_struct;
```

DATA Field Definition：

ack	0x00: The query video output command was received successfully
	0x01: The query video output command was executed successfully
	0x02: The query video output command failed to execute
video_type	0x00: White light
	0x01: Thermal imaging
	0x02: White light (full screen) + Thermal imaging (small)
	0x03: Thermal imaging (full screen) + White light (small)

6.23 Start tracking(0x16)

CMD_SET: 0x05

CMD_ID: 0x16

Structure definition：

```
typedef struct
{
    header_v1_struct header;
    uint16_t x;
    uint16_t y;
    uint16_t w;
    uint16_t h;
    uint16_t crc16;
}tracking_camera_start_track_struct;
```

DATA Field Definition：

x	Tracking coordinate x, the upper left corner is 0. Note: x value range [0,1900]
y	Tracking coordinate y, the upper left corner is 0. Note: y value range [0,1060]
w	Tracking box width, value range [10,260]
h	Tracking box height, value range [10,260]

6.24 Reply start tracking(0x17)

CMD_SET: 0x05

CMD_ID: 0x17

Structure definition：

```
typedef struct
{
    header_v1_struct header;
    uint8_t ack;
    uint16_t crc16;
}tracking_camera_start_track_ack_struct;
```

DATA Field Definition：

ack	0x00: Start tracking command received successfully
	0x01: Start tracking command executed successfully
	0x02: Start tracking command executed failed

6.25 Stop tracking(0x18)

CMD_SET: 0x05

CMD_ID: 0x18

Structure definition：

```
typedef struct
{
    header_v1_struct header;
    uint16_t crc16;
}tracking_camera_stop_track_struct;
```

6.26 Reply stop tracking(0x19)

CMD_SET: 0x05

CMD_ID: 0x19

Structure definition：

```
typedef struct
{
    header_v1_struct header;
    uint8_t ack;
    uint16_t crc16;
}tracking_camera_stop_track_ack_struct;
```

DATA Field Definition：

ack	0x00: Stop tracking command received successfully
	0x01: Stop tracking command executed successfully
	0x02: Stop tracking command executed failed

6.27 Tracking result feedback(0x1A)

CMD_SET: 0x05

CMD_ID: 0x1A

Structure definition：

```
typedef struct
{
    header_v1_struct header;
    uint8_t state;
    uint16_t x;
```

```
    uint16_t y;
    uint16_t w;
    uint16_t h;
}tracking_camera_track_result_struct;
```

DATA Field Definition：

state	0x01: Tracking in progress 0x02: Tracking lost, trying to find 0x03: Tracking lost
x	Tracking coordinate x, upper left corner is 0(center point coordinate of tracking frame (x, y)）
y	Tracking coordinate y, upper left corner is 0(center point coordinate of tracking frame (x, y))
w	Detection width of tracking target
h	Detection height of tracking target

6.28 Target recognition feedback(0x1B)

CMD_SET: 0x05

CMD_ID: 0x1B

Structure definition：

```
typedef struct
{
    header_v1_struct_t header;
    uint8_t target_num;
    int16_t data[60];
    uint16_t crc16;
}tracking_camera_target_identifacation_struct_t;
```

DATA Field Definition：

target_num	Number of identifications

data	typedef struct
	{
	short num; //Test result serial number
	short class; //Detection category
	shortx; //Detection starting coordinates
	short y;
	short width;
	short height;
	}targetInfo t;

6.29 Set up automatic storage(0x1C)

CMD_SET: 0x05

CMD_ID: 0x1C

Structure definition：

```
typedef struct
{
    header_v1_struct_t header;
    uint8_t status;//0 Off    1 On
    uint16_t crc16;
}tracking_camera_auto_save_struct_t;
```

DATA Field Definition：

	status	0 Off 1 On

6.30 Reply set up automatic storage(0x1D)

CMD_SET: 0x05

CMD_ID: 0x1D

Structure definition：

```
typedef struct
{
    header_v1_struct_t header;
    uint8_t ack;
    uint16_t crc16;
}tracking_camera_auto_save_ack_struct_t;
```

DATA Field Definition：

	ack	0x00: Automatic storage command received successfully
		0x01: Stop automatic storage executed successfully
		0x02: Stop automatic storage executed failed

Upgrade Process

Take the use of the Assistant software "ZIN_Assistant" to upgrade the gimbal as an example for explanation:

step1：The Assistant software sends a "Request to start upgrading (CMD_SET=0x00, CMD_ID =0x0b)" message to the gimbal；

step2：After receiving the Assistant software command, the gimbal returns a "Response to start upgrading (CMD_SET=0x00, CMD_ID =0x0c)" message;

step3：After the Assistant software receives the response from the gimbal, if the response message tells that the firmware size is within the limit, it will start to split the upgrade .bin file into 64 bytes (fill with zeros if it is less than 64 bytes), send the "Request Upgrade Data (CMD_SET = 0x00, CMD_ID = 0x0d)" message, and set the ".header.seq" of the message to zero. "seq" is used to represent the unpacking sequence number of the upgrade file, indicating that it is the first package. In addition, ".header.status.encrypt" needs to be set to 1, indicating that it is an encrypted firmware.；

step4： After receiving the command from the Assistant software, the gimbal will perform FLASH operation. After the operation is completed, it will send the message "Response Upgrade Data (CMD_SET = 0x00, CMD_ID = 0x0e)" and assign the ".header.seq" of the message to the ".header.seq" of the received message;

step5： After the Assistant software receives the gimbal response, if the ".header.seq" of the response message is consistent with the ".header.seq" of the message sent at the last moment, it will send the next packet of message and increase the ".header.seq" of the message by 1. It also needs to set ".header.status.encrypt" to 1, indicating that it is encrypted firmware；

step6：Repeat step 4 to step 5 on the Assistant software and gimbal until all upgrade files are sent；

step7：The Assistant software sends a message "Request to end upgrade (CMD_SET = 0x00, CMD_ID = 0x0f)" to the gimbal, and sets ".cfg.signature" and ".cfg.verify" in the message to 1.；

step8：After receiving the command from the Assistant software, the gimbal verifies the FLASH content. After the operation is completed, it will send the message "Response to end upgrade (CMD_SET = 0x00, CMD_ID = 0x10)". If the FLASH verification passes, the ".status.signature_matched" and ".status.md5_matched" of the message will be set to 1；

step9：After receiving the response from the gimbal, the Assistant software sends a "soft reset (CMD_SET = 0x00, CMD_ID = 0x00)" message, and the gimbal will restart and execute the new firmware；

7 Appendix

7.1 V1_HEADERStructure definition

```
typedef struct
{
    uint8_t sof;                    /* Start sign */
    struct
    {
        uint16_t len : 10;
        uint16_t ver : 6;
    }ver_len;                       /* Frame version number and total data length */
    uint8_t check_sum;          /* Checksum of the first 3 bytes */
    uint16_t seq;                   /* Frame sequence number */
    struct
    {
        uint8_t reserved : 4;
        uint8_t encrypt : 1;            /* 0 - data segment is not encrypted, 1 - data segment is encrypted (usually used for upgrades) */
        uint8_t reserved1 : 3;
    }status;
    struct
    {
        uint8_t index : 3;              /* Low 3 bits */
        uint8_t device : 5;             /* High 5 bits */
    }sender;
    struct
    {
        uint8_t index : 3;              /* Low 3 bits */
        uint8_t device : 5;             /* High 5 bits */
    }receiver;
    uint8_t cmd_set;
    uint8_t cmd_id;
}header_v1_struct;
```

7.2 crc16 Calculation

```
/* CRC high byte value table, x16+x15+x2+1, if const is cancelled, this table will be stored in RAM, and the execution speed
will be faster */
uint8_t crc16_high_table[] =
{
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
    0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
    0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1,
    0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
```

```
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1,
    0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
    0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40,
    0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1,
    0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
    0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40,
    0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
    0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
    0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
    0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
    0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
    0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40,
    0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1,
    0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
    0x80, 0x41, 0x00, 0xC1, 0x81, 0x40
};

/* CRC low byte value table. If const is cancelled, this table will be stored in RAM, and the execution speed will be faster */
uint8_t crc16_low_table[] =
{
    0x00, 0xC0, 0xC1, 0x01, 0xC3, 0x03, 0x02, 0xC2, 0xC6, 0x06,
    0x07, 0xC7, 0x05, 0xC5, 0xC4, 0x04, 0xCC, 0x0C, 0x0D, 0xCD,
    0x0F, 0xCF, 0xCE, 0x0E, 0x0A, 0xCA, 0xCB, 0x0B, 0xC9, 0x09,
    0x08, 0xC8, 0xD8, 0x18, 0x19, 0xD9, 0x1B, 0xDB, 0xDA, 0x1A,
    0x1E, 0xDE, 0xDF, 0x1F, 0xDD, 0x1D, 0x1C, 0xDC, 0x14, 0xD4,
    0xD5, 0x15, 0xD7, 0x17, 0x16, 0xD6, 0xD2, 0x12, 0x13, 0xD3,
    0x11, 0xD1, 0xD0, 0x10, 0xF0, 0x30, 0x31, 0xF1, 0x33, 0xF3,
    0xF2, 0x32, 0x36, 0xF6, 0xF7, 0x37, 0xF5, 0x35, 0x34, 0xF4,
    0x3C, 0xFC, 0xFD, 0x3D, 0xFF, 0x3F, 0x3E, 0xFE, 0xFA, 0x3A,
    0x3B, 0xFB, 0x39, 0xF9, 0xF8, 0x38, 0x28, 0xE8, 0xE9, 0x29,
    0xEB, 0x2B, 0x2A, 0xEA, 0xEE, 0x2E, 0x2F, 0xEF, 0x2D, 0xED,
    0xEC, 0x2C, 0xE4, 0x24, 0x25, 0xE5, 0x27, 0xE7, 0xE6, 0x26,
    0x22, 0xE2, 0xE3, 0x23, 0xE1, 0x21, 0x20, 0xE0, 0xA0, 0x60,
    0x61, 0xA1, 0x63, 0xA3, 0xA2, 0x62, 0x66, 0xA6, 0xA7, 0x67,
    0xA5, 0x65, 0x64, 0xA4, 0x6C, 0xAC, 0xAD, 0x6D, 0xAF, 0x6F,
    0x6E, 0xAE, 0xAA, 0x6A, 0x6B, 0xAB, 0x69, 0xA9, 0xA8, 0x68,
    0x78, 0xB8, 0xB9, 0x79, 0xBB, 0x7B, 0x7A, 0xBA, 0xBE, 0x7E,
    0x7F, 0xBF, 0x7D, 0xBD, 0xBC, 0x7C, 0xB4, 0x74, 0x75, 0xB5,
    0x77, 0xB7, 0xB6, 0x76, 0x72, 0xB2, 0xB3, 0x73, 0xB1, 0x71,
```

```
    0x70, 0xB0, 0x50, 0x90, 0x91, 0x51, 0x93, 0x53, 0x52, 0x92,
    0x96, 0x56, 0x57, 0x97, 0x55, 0x95, 0x94, 0x54, 0x9C, 0x5C,
    0x5D, 0x9D, 0x5F, 0x9F, 0x9E, 0x5E, 0x5A, 0x9A, 0x9B, 0x5B,
    0x99, 0x59, 0x58, 0x98, 0x88, 0x48, 0x49, 0x89, 0x4B, 0x8B,
    0x8A, 0x4A, 0x4E, 0x8E, 0x8F, 0x4F, 0x8D, 0x4D, 0x4C, 0x8C,
    0x44, 0x84, 0x85, 0x45, 0x87, 0x47, 0x46, 0x86, 0x82, 0x42,
    0x43, 0x83, 0x41, 0x81, 0x80, 0x40
};
/*****************************************************************************************
* *Function name: calc_crc16
*   Input parameter: uint8_t* data pointer
                     uint16_t data length
* Return parameter: uint16_t calculated crc16 result
* Function description: Calculate the crc16 of the specified data length
* Modification record:
*****************************************************************************************/
uint16_t calc_crc16(uint8_t *pbuf, uint16_t len)
{
    uint8_t crc16_high = 0xff;      /* High CRC byte initial */
    uint8_t crc16_low = 0xff;       /* Low CRC byte initial */
    uint8_t index = 0;              /* Index in CRC loop */

    while(len--) /* Transmit message buffer 区 */
    {
        index = crc16_low ^ (*pbuf++); /* Calculating CRC */
        crc16_low = crc16_high ^ crc16_high_table[index];
        crc16_high = crc16_low_table[index];
    }

    return (crc16_high << 8 | crc16_low);
}
```

7.3 Module Number

The module number is represented by 1 byte, where the upper 5 bits represent "device" and the lower 3 bits represent "index", as shown in Table 4.1.

Table 4.1 Module number

Device	Index	Bit field assignment	Byte assignment
PC	PC	.device = 0x00, .index = 0x00,	0x00
gimbal	gimbal	.device = 0x01, .index = 0x00,	0x08

Flight Control	Flight Control	.device = 0x02, .index = 0x00,	0x10

8. Serial Port Gimbal Control Example

No control：

AA 32 04 E0 00 00 00 10 08 02 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 25 37

Speed control　　Yaw 30°/sec:

AA 32 04 E0 00 00 00 10 08 02 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 00 00 00 00 B8 0B 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 3B 1C

Angle control　Rotate to 30°：

AA 32 04 E0 00 00 00 10 08 02 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 02 00 00 00 00 B8 0B 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 6F F9

Gimbal return to center：

AA 32 04 E0 00 00 00 10 08 02 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 03 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 71 D2