Target GPS-coordinates Calculation Protocol

(Baudrate 115200, 8, 1, N, little-endian)

1. Mode settings command (hex): (just send one time to gimbal)

AA 55 0F 31 FF. Blue byte is mode byte, the definitions are as follow:

Byte0	AA	Frame header1
Byte1	55	Frame header2
Byte2	0F	SET ID
Byte3 Mode byte	Bit0: OSD display position type 0: display UAV GPS-coordinates on OSD 1: display Target GPS-coordinates on OSD Bit1~bit2: reserved Bit3: output OUTx data to RJ45 Net port (just for T serial that with Net port) 0: gimbal RJ45 Net port do not output OUTx data. 1: gimbal RJ45 Net port output OUTx data.(OUT1 ~OUT4, optional) Bit4: output OUTx data to serial port (all serial) 0: gimbal serial port do not output OUTx data. 1: gimbal serial port output OUTx data.(OUT1 ~OUT4 , optional) Bit5~7: OUTx data 000: OUT1 format; 001: OUT2 format; 010: OUT3 format; 011: OUT4 format;	
Byte4	FF	Frame end

Command examples:

1.1)AA 55 0F 71 FF. Display target GPS-coordinates, serial port of gimbal output OUT4 data.

1.2) AA 55 0F 59 FF. Display target GPS-coordinates,serial port and RJ45 output OUT3 data.

1.3) AA 55 0F 30 FF.Display UAV GPS-coordinates, serial port of gimbal output OUT2 data.

1.4)AA 55 0F 11 FF.Display target GPS-coordinates, serial port of gimbal output OUT1 data.

1.5)AA 55 0F 79 FF.Display target GPS-coordinates, RJ45 of gimbal output OUT4 data.

1.6)AA 55 0F 01 FF.Display target GPS-coordinates, do not output data.(**default settings**)

2) Important attentions :

2.1) For models with LRF, Target GPS-coordinates is calculated by Laser Ranger Finder data.

2.2) For models without LRF. Target GPS-coordinates is calculated by UAV relative altitude -altitude above ground, If target and take-off point of UAV are in the same horizontal plane, the Target GPS-coordinates calculation result is basically accurate; Otherwise, the error will increase with the height error.

2.3) Headings of gimbal (yaw 0 degree, home position) and UAV should be same direction.

2.4) Gimbal need get following data for Target GPS-coordinates calculation:

2.4.1) Vehicle heading (yaw angle,0.00...359.99 degrees.north is 0.00)

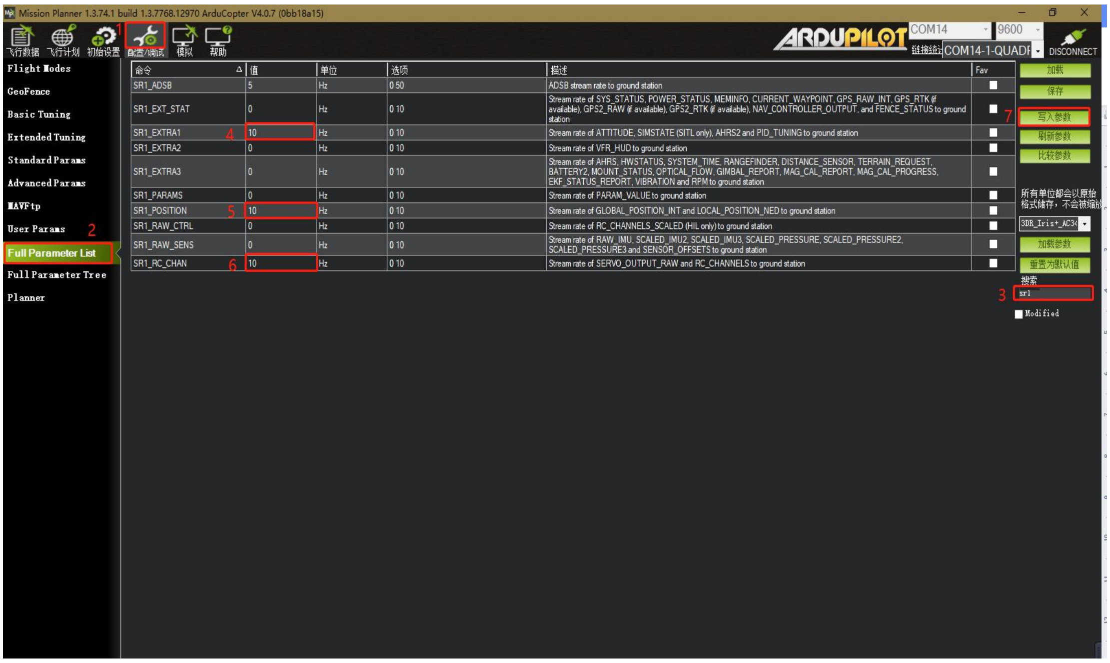
2.4.2) Vehicle latitude and longitude.

2.4.3) Vehicle Ground X Speed,Ground Y Speed,Ground Z Speed.(optional)

3) How to connect gimbal to Pixhawk for calculation (APM,PIX4).

3.1) set output messages. connect TELEMx port of flight controller to serial port of gimbal, make sure baud-rate of flight controller is 115200,8,1,none, gimbal need get following Messages from flight controller for target GPS-coordinates calculation.

Set SRx_EXTRA1 10hz and SRx_POSITION 10hz. x is Telem port number of FC, for example, if use telem1,just set SR1_EXTRA1 10hz and SR1_POSITION 10hz are enough for target GPS-coordinates calculation. (as following image, item4, item5)



3.2) Extended reading, how does gimbal parse mavlink messages for attitude, position,time and RC_channels.

- Gimbal get ATTITUDE(SRx_EXTRA1) and POSITION (SRx_POSITON) from FC for target GPS-coordinates calculation.
  (https://mavlink.io/en/messages/common.html#ATTITUDE)

ATTITUDE (#30)

[Message] The attitude in the aeronautical frame (right-handed, Z-down, X-front, Y-right).

Field Name	Type	Units	Description
time_boot_ms	uint32_t	ms	Timestamp (time since system boot).
roll	float	rad	Roll angle (-pi..+pi)
pitch	float	rad	Pitch angle (-pi..+pi)
yaw	float	rad	Yaw angle (-pi..+pi)
rollspeed	float	rad/s	Roll angular speed
pitchspeed	float	rad/s	Pitch angular speed
yawspeed	float	rad/s	Yaw angular speed

(https://mavlink.io/en/messages/common.html#GLOBAL_POSITION_INT)

GLOBAL_POSITION_INT (#33)

[Message] The filtered global position (e.g. fused GPS and accelerometers). The position is in GPS-frame (right-handed, Z-up). It is designed as scaled integer message since the resolution of float is not sufficient.

Field Name	Type	Units	Description
time_boot_ms	uint32_t	ms	Timestamp (time since system boot).
lat	int32_t	degE7	Latitude, expressed
lon	int32_t	degE7	Longitude, expressed
alt	int32_t	mm	Altitude (MSL). Note that virtually all GPS modules provide both WGS84 and MSL.
relative_alt	int32_t	mm	Altitude above ground
vx	int16_t	cm/s	Ground X Speed (Latitude, positive north)
vy	int16_t	cm/s	Ground Y Speed (Longitude, positive east)
vz	int16_t	cm/s	Ground Z Speed (Altitude, positive down)
hdg	uint16_t	cdeg	Vehicle heading (yaw angle), 0.0..359.99 degrees. If unknown, set to: UINT16_MAX

- Gimbal get time from message SYSTEM_TIME (SRx_EXTRA3). (https://mavlink.io/en/messages/common.html#SYSTEM_TIME)

SYSTEM_TIME (#2)

[Message] The system time is the time of the master clock, typically the computer clock of the main onboard computer.

Field Name	Type	Units	Description
time_unix_usec	uint64_t	us	Timestamp (UNIX epoch time).
time_boot_ms	uint32_t	ms	Timestamp (time since system boot).

- Gimbal get PWM value of RC_IN signal from message RC_CHANNELS(SRx_RC_CHN). (https://mavlink.io/en/messages/common.html#RC_CHANNELS)

RC_CHANNELS (#65)

[Message] The PPM values of the RC channels received. The standard PPM modulation is as follows: 1000 microseconds: 0%, 2000 microseconds: 100%. A value of UINT16_MAX implies the channel is unused. Individual receivers/transmitters might violate this specification.

Field Name	Type	Units	Description
time_boot_ms	uint32_t	ms	Timestamp (time since system boot).
chancount	uint8_t		Total number of RC channels being received. This can be larger than 18, indicating that more channels are available but not given in this message. This value should be 0 when no RC channels are available.
chan1_raw	uint16_t	us	RC channel 1 value.
chan2_raw	uint16_t	us	RC channel 2 value.
chan3_raw	uint16_t	us	RC channel 3 value.
chan4_raw	uint16_t	us	RC channel 4 value.
chan5_raw	uint16_t	us	RC channel 5 value.
chan6_raw	uint16_t	us	RC channel 6 value.
chan7_raw	uint16_t	us	RC channel 7 value.
chan8_raw	uint16_t	us	RC channel 8 value.
chan9_raw	uint16_t	us	RC channel 9 value.
chan10_raw	uint16_t	us	RC channel 10 value.
chan11_raw	uint16_t	us	RC channel 11 value.
chan12_raw	uint16_t	us	RC channel 12 value.
chan13_raw	uint16_t	us	RC channel 13 value.
chan14_raw	uint16_t	us	RC channel 14 value.
chan15_raw	uint16_t	us	RC channel 15 value.
chan16_raw	uint16_t	us	RC channel 16 value.
chan17_raw	uint16_t	us	RC channel 17 value.
chan18_raw	uint16_t	us	RC channel 18 value.
rssi	uint8_t		Receive signal strength indicator in device-dependent units/scale. Values: [0-254], UINT8_MAX: invalid/unknown.

4) Viewpro private protocol for target GPS-coordinate calculation.

Gimbal can parse input data, IN1 data or IN2 data for Target GPS-coordinates calculation, details as 4.1

Gimbal can output data OUT1 data , or OUT2 data , or OUT3 data , or OUT4 data as settings ,details as 4.2

4.1) Incoming command format: Gimbal get ATTITUDE and POSITION of UAV. Input data Select IN1 or IN2. (or IN3 for test)

4.1.1) IN1 packet format

input data without speed information (User serial port send to Gimbal)

byte 0: 0xF9 uint8 frame header

Byte 1: 0xFB uint8 command ID

Byte 2~byte3: uint16 year;

Byte 4: uint8 mon;

Byte 5: uint8 day;

Byte 6: uint8 hour;

Byte 7: uint8 min;

Byte 8: uint8 sec;

Byte9 ~ byte12: float UAV roll; /*< Roll angle (rad, -pi..+pi)*/

Byte13~ byte16: float UAV pitch; /*< Pitch angle (rad, -pi..+pi)*/

Byte17 ~ byte20: float UAV yaw; /*< Yaw angle (rad, -pi..+pi)*/

Byte21~byte24: int32 UAV latitude; /*<Latitude (WGS84, in degrees * 1E7*/

Byte25 ~ byte28: int32 UAV longitude; /*< Longitude (WGS84 in degrees * 1E7*/

Byte29~byte32 int32 UAV alt; /*< [mm] Altitude (MSL)*/

Byte 33: uint8 checksum = sum from byte0 to byte 32.

4.1.2) IN2 command

input data with speed information (NOTE: speed data format should be same as mavlink MSG: #33 GLOBAL_POSITION_INT.)(User serial port send to Gimbal)

byte 0： 0xF9 uint8 frame header

Byte 1： 0xFC uint8 command ID

Byte 2~byte3： uint16 year;

Byte 4： uint8 mon;

Byte 5: uint8 day;

Byte 6: uint8 hour;

Byte 7: uint8 min;

Byte 8: uint8 sec;

Byte9 ~ byte12: float UAV roll; /*< Roll angle (rad, -pi..+pi)*/

Byte13~ byte16: float UAV pitch; /*< Pitch angle (rad, -pi..+pi)*/

Byte17 ~ byte20: float UAV yaw; /*< Yaw angle (rad, -pi..+pi)*/

Byte21~byte24: int32 UAV latitude; /*<Latitude (WGS84, in degrees * 1E7*/

Byte25 ~ byte28: int32 UAV longitude; /*< Longitude (WGS84 in degrees * 1E7*/

Byte29~byte32 int32 UAV altitude; /*< [mm] Altitude (MSL)*/

Byte33~byte34 int16 vx cm/s Ground X Speed (Latitude, positive north)

Byte35~byte36 int16 vy cm/s Ground Y Speed (Longitude, positive east)

Byte37~byte38 int16 vz cm/s Ground Z Speed (Altitude, positive down)

Byte39 uint8 checksum (sum of byte0~byte38, mod 256)

For example: F9 FC E5 07 08 07 11 1B 3A 00 00 00 00 *00 00 00 00* 00 00 00 00 07 5C C8 0D 07 45 6D 43 F8 2A 00 00 00 00 00 00 00 00 AC

4.1.2) IN3 command(input LRF distance just for test)

Input data with speed information and LRF test data (NOTE: speed data format should be same as mavlink MSG: #33 GLOBAL_POSITION_INT.) (User serial port send to Gimbal)(this IN3 command can be used for test)

byte 0： 0xF9 uint8 frame header

Byte 1： 0xFC uint8 command ID

Byte 2： uint8 year; //0x15 + 2000 = 2021

byte3: 0XFF uint8 IN3 flag
Byte 4: uint8 mon;
Byte 5: uint8 day;
Byte 6: uint8 hour;
Byte 7: uint8 min;
Byte 8: uint8 sec;
Byte9 ~ byte12: uint32 **LRF distance; /* expressed as * 1000 (millimeters)*/**
Byte13~ byte16: int32 UAV pitch; /*< Pitch angle in degrees * 1E3*/
Byte17 ~ byte20: int32 UAV yaw; /*< Yaw angle in degrees * 1E3*/
Byte21~byte24: int32 UAV latitude; /*<Latitude (WGS84, in degrees * 1E7*/
Byte25 ~ byte28: int32 UAV longitude; /*< Longitude (WGS84 in degrees * 1E7*/
Byte29~byte32 int32 UAV alt; /*< [mm] Altitude (MSL)*/
Byte33~byte34 int16 vx cm/s Ground X Speed (Latitude, positive north)
Byte35~byte36 int16 vy cm/s Ground Y Speed (Longitude, positive east)
Byte37~byte38 int16 vz cm/s Ground Z Speed (Altitude, positive down)
Byte39 uint8 checksum (sum of byte0~byte38, mod 256)

4.2) Outgoing Command: gimbal output data. out1~out4, optional,

4.2.1) Out1 data (Gimbal output data to user serial port,set command AA 55 0F 11 FF)

byte 0: 0xFE uint8 frame header
Byte 1: 0xFB uint8 command ID
Byte2 ~byte5 float gimbal_pitch_angle; /*degree*/
Byte6 ~byte9 float gimbal_yaw_angle; /*degree*/
Byte10 ~ byte13 float distance (laser range finder data); /*m*/
Byte14 ~ byte 17 int32 Target _pos_lon; /*< Longitude (WGS84, in degrees * 1E7*/
Byte18~byte 21 int32 Target _pos_lat; /*< Latitude (WGS84, in degrees * 1E7*/
Byte22 uint8 checksum = sum from byte0 to byte 21
Feedback example:
FE FB 00 00 D8 41 00 00 80 41 00 00 00 00 1E E9 9C 2F 35 66 23 04 67

4.2.2) Out2 data:(Gimbal output data to user serial port ,set command AA 55 0F 31 FF)

Byte 0: 0xFE Header
Byte 1: 0XFC CMD ID
Byte 2~byte3: uint16 year;
Byte 4: uint8 month;
Byte 5: uint8 day;
Byte 6: uint8 hour;
Byte 7: uint8 minutes;
Byte 8: uint8 second;
Byte9~byte10: uint16 Zoom position value;
Byte11~byte14: float Gimbal ENC Roll; /*< Pitch angle (rad, -pi..+pi)*/
Byte15~byte18: float Gimbal ENC Pitch;/*< Pitch angle (rad, -pi..+pi)*/
Byte19~byte22: float Gimbal ENC Yaw; /*< YAW angle (rad, -pi..+pi)*/
Byte23~byte26: float Laser ranger finder data; /* unit : meter
Byte27~byte30: float UAV Rollrad; /*< Pitch angle (rad, -pi..+pi)*/

Byte31~byte34: float UAV Pitchrad;/*< Pitch angle (rad, -pi..+pi)*/
Byte35~byte38: float UAV YawRad;/*< YAW angle (rad, -pi..+pi)*/
Byte39:~byte42: int32 UAV alt; /*< [mm] Altitude (MSL)*/
Byte43~byte50: double UAVlatrad; /*< Latitude */
Byte51~byte58: double UAVlonrad; /*< Longitude */
Byte59~byte66: double Target lat rad; /*(rad, -pi..+pi)*/
Byte67~byte74: double Target lon rad; /*(rad, -pi..+pi)*/
Byte75: uint8 checksum= sum（byte0...byte74）mod 256

Totally 76 bytes, checksum is lower 8 bit of the sum of first 75 bytes.

Example: FE FC E3 07 08 15 07 16 00 60 26 00 00 00 00 3A 46 F1 3E 35 FA 8E 3E 00 00 00 00 CD CC 4C 3E CD CC 4C 3F 00 00 00 3F B0 BF FE FF 9C BB 70 46 A3 03 BF 3F 18 23 12 39 7F 4E F6 3F 1C 52 6C 9A 65 05 BF 3F 60 FA E1 FA 9A 4E F6 3F A7

4.2.3) Out3 data : (Gimbal output data to user serial port ,set command AA 55 0F 51 FF)

Byte 0: 0xFE Header
Byte 1: 0XFD CMD ID
Byte 2~byte3: uint16 year;
Byte 4: uint8 month;
Byte 5: uint8 day;
Byte 6: uint8 hour;
Byte 7: uint8 minutes;
Byte 8: uint8 second;
Byte9~byte10: uint16 Zoom position value;
Byte11~byte14: float Gimbal Roll; /*< Pitch angle (rad, -pi..+pi)*/
Byte15~byte18: float Gimbal Pitch;/*< Pitch angle (rad, -pi..+pi)*/
Byte19~byte22: float Gimbal Yaw; /*< YAW angle (rad, -pi..+pi)*/
Byte23~byte26: float Laser ranger finder data; /* unit : meter
Byte27~byte30: float UAV Roll; /*< Pitch angle (rad, -pi..+pi)*/
Byte31~byte34: float UAV Pitch; /*< Pitch angle (rad, -pi..+pi)*/
Byte35~byte38: float UAV Yaw; /*< YAW angle (rad, -pi..+pi)*/
Byte39:~byte42: int32 UAV alt; /*< [mm] Altitude (MSL)*/
Byte43~byte46: int32 UAV latitude ; /*< Latitude (WGS84, in degrees * 1E7*/
Byte47~byte50: int32 UAV longitude; /*< Longitude (WGS84, in degrees * 1E7*/
Byte51~byte54: int32 Target latitude ; /*< Latitude (WGS84, in degrees * 1E7*/
Byte55~byte58: int32 Target longitude; /*< Longitude (WGS84, in degrees * 1E7*/
Byte59: uint8 checksum= sum（byte0...byte58）mod 256

Totally 60 bytes, checksum is lower 8 bit of the sum of all 59 bytes.

For example: FE FD E5 07 02 05 0D 13 31 00 00 00 00 00 00 50 77 56 BD C2 B8 32 3F 7C 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 2F 07 00 ED D3 5C 14 13 1F F9 40 3F F5 5C 14 DE 40 F9 40 4C

4.2.4) Out4 data (Gimbal output data to user serial port ,set command AA 55 0F 71 FF)

Byte 0: 0xFE Header
Byte 1: 0XFD CMD ID

Byte 2:	uint8	year; //if 21 means + 2000 = 2021
Byte3:	0XFF	flag
Byte 4:	uint8	month;
Byte 5:	uint8	day;
Byte 6:	uint8	hour;
Byte 7:	uint8	minutes;
Byte 8:	uint8	second;
Byte9~byte10:	uint16	Zoom position value;
Byte11~byte14:	int32	Gimbal Roll; /*< Roll angle in degrees * 1E3*/
Byte15~byte18:	int32	Gimbal Pitch;/*< Pitch angle in degrees * 1E3*/
Byte19~byte22:	int32	Gimbal Yaw; /*< YAW angle in degrees * 1E3*/
Byte23~byte26:	uint32	Laser ranger finder data; /*expressed as * 1000 (millimeters)*/
Byte27~byte30:	int32	Target alt; /*< [mm] Altitude (MSL)*/
Byte31~byte34:	int32	UAV Pitch; /*< Pitch angle in degrees * 1E3 */
Byte35~byte38:	int32	UAV Yaw; /*< YAW angle in degrees * 1E3*/
Byte39:~byte42:	int32	UAV alt; /*< [mm] Altitude (MSL)*/
Byte43~byte46:	int32	UAV latitude ; /*< Latitude (WGS84, in degrees * 1E7*/
Byte47~byte50:	int32	UAV longitude; /*< Longitude (WGS84, in degrees * 1E7*/
Byte51~byte54:	int32	Target latitude ; /*< Latitude (WGS84, in degrees * 1E7*/
Byte55~byte58:	int32	Target longitude; /*< Longitude (WGS84, in degrees * 1E7*/
Byte59:	uint8	checksum = sum（byte0...byte58）mod 256

Totally 60 bytes, checksum is lower 8 bit of the sum of all 59 bytes.

5) adjust angle shift value to solve calculation error.

5.1）Adjust yaw angle command : AA 55 06 XX FF (HEX)

XX: angle for shift step . int8 , unit : 0.01degree (- left shift--- + right shift),

For example: when calculation point is at left 1 degree of target actual point.

Send command : AA 55 06 9c FF . 0x9c= -100 , -100*0.01 = -1degree.

For example: when calculation point is at right 1 degree of actual point.

Send command : AA 55 06 64 FF . 0x64= 100 , 100*0.01 = +1 degree.

For example: when calculation point is at right 5 degree of actual point.

Send 5 times 1-degree right command:

AA 55 06 64 FF
AA 55 06 64 FF
AA 55 06 64 FF
AA 55 06 64 FF
AA 55 06 64 FF

5.2）Adjust pitch angle command : AA 55 36 XX FF (HEX)

XX: pitch angle for step shift . int8 , unit : 0.01degree (-down shift--- +up shift),

For example: when calculation point is at down 1 degree of target actual point.

Send command : AA 55 36 9c FF . 0x9c= -100 , -100*0.01 = -1degree.

For example: when calculation point is at up 1 degree of actual point.

Send command : AA 55 36 64 FF . 0x64= 100 , 100*0.01 = 1 degree.
For example: when calculation point is at up 5.5 degree of actual point.
Send commands 1~6 :

1) AA 55 36 64 FF
2) AA 55 36 64 FF
3) AA 55 36 64 FF
4) AA 55 36 64 FF
5) AA 55 36 64 FF
6) AA 55 36 32 FF   0x32= 50 , 50*0.01 = 0.5 degree.