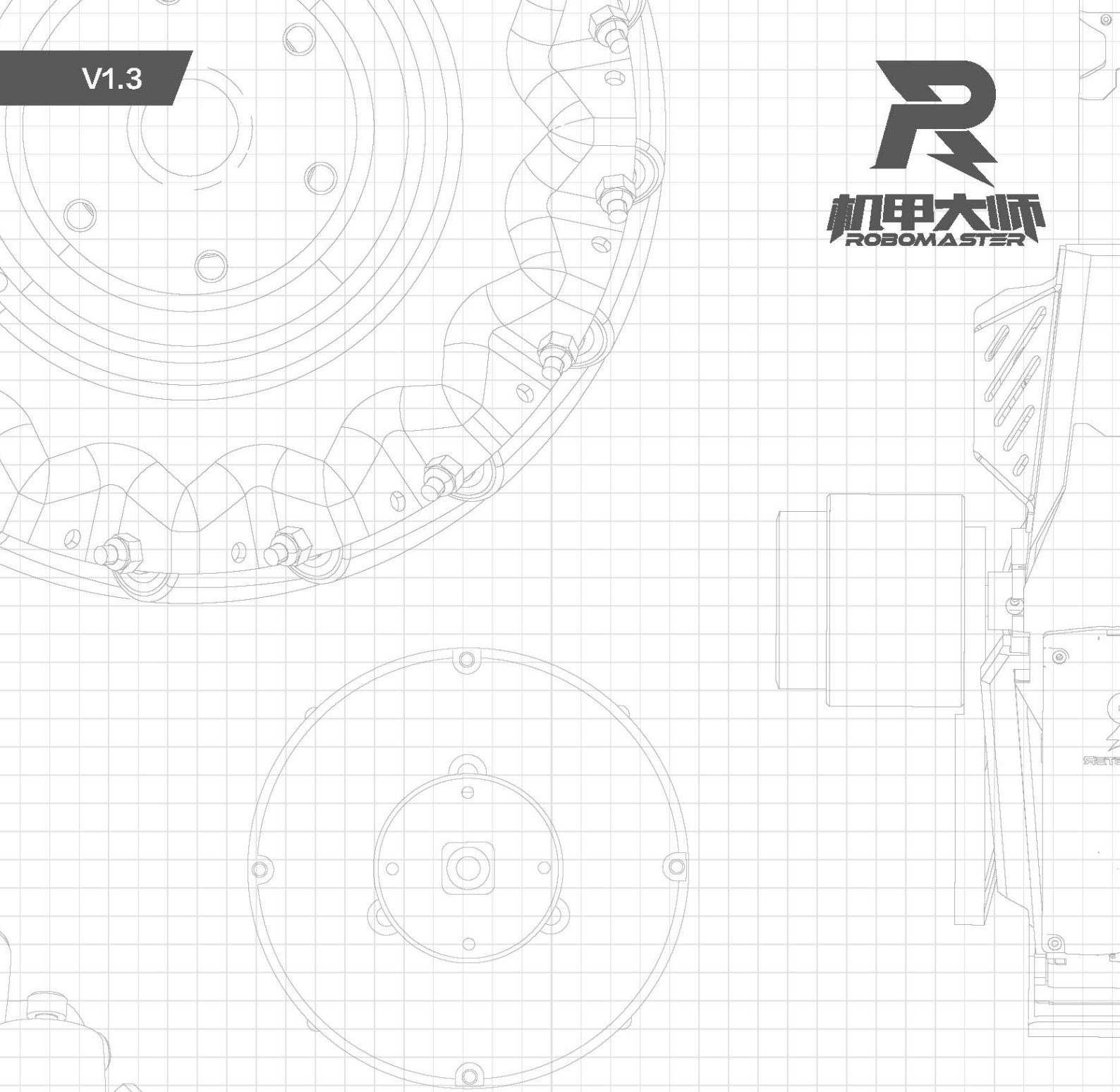
V1.3



Referee System Serial Port Protocol Appendix

Prepared by RoboMaster Organizing Committee

Released in December, 2021

Release Notes

Date	Version	Revision Records
2021.12.31	V1.3	Updated AI Challenge buff and debuff zone distribution and lurking mode: 0x0005
2021.04.30	V1.2	Fixed some bugs
2021.04.19	V1.1	Fixed some bugs
2021.03.19	V1.0	First release

Contents

1. Serial Port Configuration

The serial port is used for communication and is configured with a Baud rate 115200, an 8-bit data bit, and 1-bit stop bit without hardware flow control and check bit.

2. Interface Protocol

Table 2-1 Communication protocol format

frame_header (5-byte)	cmd_id (2-byte)	data (n-byte)	frame_tail (2-byte, CRC16, full packet check)

Table 2-2 Frame header format

SOF	data_length	seq	CRC8
1-byte	2-byte	1-byte	1-byte

Table 2-3 Frame header definitions

Domain	Offset position	Size (byte)	Detailed description
SOF	0	1	Data frame start byte; the fixed value is 0×A5
data_length	1	2	Length of data in the data frame
seq	3	1	Sequence number
CRC8	4	1	Frame header CRC8

Table 2-4 Command ID

Command code	Data segment length	Function Description
0x0001	11	Competition status data; transmitted periodically at a frequency of 1 Hz
0x0002	1	Competition result data; transmitted after the competition ends
0x0003	28	Robot HP data in competition; transmitted periodically at a frequency of 1 Hz
0x0004	3	Dart launching status; transmitted after the dart is launched

Command code	Data segment length	Function Description
0x0005	11	AI Challenge buff and debuff status; transmitted periodically at a frequency of 1 Hz
0x0101	4	Site event data; transmitted periodically at a frequency of 1 Hz
0x0102	3	Action identifier data of the site official projectile supplier; transmitted after the action changes
0x0103	2	Data of projectile supply requested from the official projectile supplier; transmitted by the competing team periodically at a frequency of 10 Hz at most (not available for RoboMaster Robotics Competition)
0x0104	2	Referee's warning data; transmitted after the warning occurs
0x0105	1	Dart Launch Opening countdown; transmitted periodically at a frequency of 1 Hz
0x0201	15	Robot status data; transmitted periodically at a frequency of 10 Hz
0x0202	14	Real-time power and heat data; transmitted periodically at a frequency of 50 Hz
0x0203	16	Robot position data; transmitted at a frequency of 10 Hz
0x0204	1	Robot buff data; transmitted after the buff status changes
0x0205	3	Aerial robot’s energy status data; transmitted only by the aerial robot’s main controller periodically at a frequency of 10 Hz
0x0206	1	Damage status data; transmitted after the damage occurs
0x0207	6	Real-time launching data; transmitted after the projectile is launched
0x0208	2	Remaining projectiles; transmitted by the aerial robot and sentry robot periodically at a frequency of 1 Hz
0x0209	4	Robot RFID status; transmitted periodically at a frequency of 1 Hz

Command code	Data segment length	Function Description
0x020A	12	Letter of instructions of the dart robot client; transmitted periodically at a frequency of 10 Hz
0x0301	n	Interaction data between robots; transmitted at a maximum frequency of 10 Hz when triggered by the sender
0x0302	n	Interaction data interface of the custom controller; transmitted at a maximum frequency of 30 Hz when triggered via the client
0x0303	15	Interaction data of the client small map; transmitted when triggered
0x0304	12	Keyboard and mouse information; transmitted via the image transmission serial port
0x0305	10	Data receipt of the client small map

3. Details

Table 3-1 Competition status data: 0x0001. Transmission frequency: 1 Hz

Byte offset	Size	Description
0	1	0-3 bit: Competition type • 1: RoboMaster Competitions • 2 : RoboMaster University Technical Challenge • 3: ICRA RoboMaster University AI Challenge • 4: RoboMaster University League 3V3 Confrontation • 5: RoboMaster University League Standard Confrontation 4-7 bit: Current stage • 0: Pre-competition stage • 1: Setup period • 2 : Initialization stage • 3: 5-second countdown

Byte offset	Size	Description
		• 4: In combat • 5: calculating competition results
1	2	Remaining time in the current round (seconds)
3	8	The exact Unix time at which the robot receives the instruction, which is effective when the on-board terminal receives a valid NTP server time service

```
typedef __packed struct
{
  uint8_t game_type : 4;
  uint8_t game_progress : 4;
  uint16_t stage_remain_time;
  uint64_t SyncTimeStamp;
} ext_game_status_t;
```

Table 3-2 Competition result data: 0x0002. Transmission frequency: Transmitted after the competition ends

Byte offset	Size	Description
0	1	0 Draw 1 Red team victory 2 Blue team victory

```
typedef __packed struct
{
   uint8_t winner;
} ext_game_result_t;
```

Table 3-3 Robot HP data: 0x0003. Transmission frequency: 1Hz

Byte offset	Size	Description
0	2	RED 1 hero robot HP, 0 when not playing or ejected
2	2	RED 2 engineer robot HP
4	2	RED 3 standard robot HP
6	2	RED 4 standard robot HP
8	2	RED 5 standard robot HP
10	2	RED 7 sentry robot HP

Byte offset	Size	Description
12	2	Red Team outpost HP
14	2	Red Team base HP
16	2	BLUE 1 hero robot HP
18	2	BLUE 2 engineer robot HP
20	2	BLUE 3 standard robot HP
22	2	BLUE 4 standard robot HP
24	2	BLUE 5 standard robot HP
26	2	BLUE 7 sentry robot HP
28	2	Blue Team outpost HP
30	2	Blue Team base HP

```
typedef __packed struct
{
  uint16_t red_1_robot_HP;
  uint16_t red_2_robot_HP;
  uint16_t red_3_robot_HP;
  uint16_t red_4_robot_HP;
  uint16_t red_5_robot_HP;
  uint16_t red_7_robot_HP;
  uint16_6 red_outpost_HP;
  uint16_t red_base_HP;
  uint16_t blue_1_robot_HP;
  uint16_t blue _2_robot_HP;
  uint16_t blue _3_robot_HP;
  uint16_t blue _4_robot_HP;
  uint16_t blue _5_robot_HP;
  uint16_t blue _7_robot_HP;
  uint16_t blue_outpost_HP;
  uint16_t blue _base_HP;
} ext_game_robot_HP_t;
```

Table 3-4 AI Challenge buff and debuff zone distribution and lurking mode: 0x0005. Transmission frequency: Transmitted periodically at a frequency of 1 Hz. Transmission range: All robots

Byte offset	Size	Description
0	3	Bits [0, 4, 8, 12, 16, 20]: F1-F6 activated states

Byte offset	Size	Description
		0: not activated 1: available Bits [1-3, 5-7, 9-11, 13-15, 17-19, 21-23]: F1-F1 status information 1: Red Team Restoration Zone 2: Red Team Supplier Zone 3: Blue Team Restoration Zone 4: Blue Team Supplier Zone 5: No shooting zone 6: No moving zone
3	2	RED 1 remaining projectiles
5	2	RED 2 remaining projectiles
7	2	BLUE 1 remaining projectiles
9	2	BLUE 2 remaining projectiles
11	1	0: normal 1: about to enter the lurking phase 2: lurking phase
12	1	Reserved byte

```
typedef __packed struct
{
  uint8_t F1_zone_status:1;
  uint8_t F1_zone_buff_debuff_status:3;
  uint8_t F2_zone_status:1;
  uint8_t F2_zone_buff_debuff_status:3;
  uint8_t F3_zone_status:1;
  uint8_t F3_zone_buff_debuff_status:3;
  uint8_t F4_zone_status:1;
  uint8_t F4_zone_buff_debuff_status:3;
  uint8_t F5_zone_status:1;
  uint8_t F5_zone_buff_debuff_status:3;
  uint8_t F6_zone_status:1;
  uint8_t F6_zone_buff_debuff_status:3;
```



```
  uint16_t red1_bullet_left;
  uint16_t red2_bullet_left;
  uint16_t blue1_bullet_left;
  uint16_t blue2_bullet_left;
  uint8_t lurk_mode;
  uint8_t res;
} ext_ICRA_buff_debuff_zone_and_lurk_status_t;
```

Table 3-5 Site event data: 0x0101. Transmission frequency: 1Hz

Byte offset	Size	Description
0	4	bit 0-2: bit 0: Occupation status of No. 1 restoration zone of own official projectile supplier, 1: occupied bit 1: Occupation status of No. 2 restoration zone of own official projectile supplier, 1: occupied bit 2: Occupation status of No. 3 restoration zone of own official projectile supplier, 1: occupied bits 3-5: Status of own Power Rune: • bit 3: occupation status of the attack point, 1: occupied • bit 4: Activation status of the small Power Rune, 1: activated • bit 5: Activation status of the large Power Rune, 1: activated bit 6: Occupation status of own R2/B2 Ring-Shaped Elevated Ground, 1: occupied bit 7: Occupation status of own R3/B3 Trapezoid-Shaped Elevated Ground, 1: occupied bit 8: Occupation status of own R4/B4 Trapezoidal Elevated Ground, 1: occupied bit 9: Status of own base shield • 1: The base has virtual shield HP • 0: The base has no virtual shield HP bit 10: Status of own outpost • 1: The outpost survives • 0: The outpost is destroyed bits 10-31: Kept.

Byte offset	Size	Description

```
typedef __packed struct
{
  uint32_t event_type;
} ext_event_data_t;
```

Table 3-6 Action identifier of the official projectile supplier: 0x0102. Transmission frequency: Transmitted after the action changes. Transmission range: Own team's robots

Byte offset	Size	Description
0	1	Official projectile supplier ID: 1: No. 1 official projectile supplier 2 : No. 2 official projectile supplier
1	1	Robot ID for projectile reload: 0: No robot is supplied with projectiles, 1: Red Team hero robot is supplied with projectiles, 2: Red Team engineer robot is supplied with projectiles, 3/4/5: Red Team standard robot is supplied with projectiles, 101: Blue Team hero robot is supplied with projectiles, 102: Blue Team engineer robot is supplied with projectiles, 103/104/105: Blue Team standard robot is supplied with projectiles
2	1	Status of the projectile outlet: 0: closed, 1: preparing projectiles, 2: the projectile is dropping
3	1	Number of supplied projectiles: 50: 50 projectiles 100: 100 projectiles 150: 150 projectiles 200: 200 projectiles

```
typedef __packed struct
{
  uint8_t supply_projectile_id;
  uint8_t supply_robot_id;
  uint8_t supply_projectile_step;
  uint8_t supply_projectile_num;
} ext_supply_projectile_action_t;
```

Table 3-7 Referee's warning information: cmd_id (0x0104). Transmission frequency: Transmitted after your own

team is warned

Byte offset	Size	Description
0	1	Warning level: 1: Yellow Card 2 : Red Card 3: Forfeiture
1	1	Foul robot ID: When receiving a punitive forfeiture, the robot ID is 0. When receiving Yellow Card or Red Card, the robot ID is the foul robot ID.

```
typedef __packed struct
{
  uint8_t level;
  uint8_t foul_robot_id;
} ext_referee_warning_t;
```

Table 3-8 Dart launching opening countdown: cmd_id (0x0105). Transmission frequency: Transmitted periodically at a frequency of 1 Hz. Transmission range: Own team's robots

Byte offset	Size	Description
0	1	15-second countdown

```
typedef __packed struct
{
  uint8_t dart_remaining_time;
} ext_dart_remaining_time_t;
```

Table 3-9 Robot status in competition: 0x0201. Transmission frequency: 10Hz

Byte offset	Size	Description
0	1	Current robot ID: 1: Red Team hero robot 2 : Red Team engineer robot 3/4/5: Red Team standard robot 6: Red Team aerial robot 7: Red Team sentry robot 8: Red Team dart robot

Byte offset	Size	Description
		9: Red Team radar station 101: Blue Team hero robot 102: Blue Team engineer robot 103/104/105: Blue Team standard robot 106: Blue Team aerial robot 107: Blue Team sentry robot 108: Blue Team dart robot 109: Blue Team radar station
1	1	Robot level: 1: Level 1, 2: Level 2, 3: Level 3
2	2	The robot's remaining HP
4	2	Robot maximum HP
6	2	Robot 1 17mm barrel cooling value per second
8	2	Robot 1 17mm barrel heat limit
10	2	Robot 1 17mm barrel speed limit (m/s)
12	2	Robot 2 17mm barrel cooling value per second
14	2	Robot 2 17mm barrel heat limit
16	2	Robot 2 17mm barrel speed limit (m/s)
18	2	Robot 42mm barrel cooling value per second
20	2	Robot 42mm barrel heat limit
22	2	Robot 42mm barrel speed limit (m/s)
24	2	Robot chassis power consumption limit

Byte offset	Size	Description
26	1	Main controller power supply condition: 0 bit: output from gimbal port, 1: 24V output, 0: no 24V output 1 bit: output from chassis port, 1: 24V output, 0: no 24V output 2 bit: output from shooter port 1: 24V output, 0: no 24V output

```
typedef __packed struct
{
  uint8_t robot_id;
  uint8_t robot_level;
  uint16_t remain_HP;
  uint16_t max_HP;
  uint16_t shooter_id1_17mm_cooling_rate;
  uint16_t shooter_id1_17mm_cooling_limit;
  uint16_t shooter_id1_17mm_speed_limit

  uint16_t shooter_id2_17mm_cooling_rate;
  uint16_t shooter_id2_17mm_cooling_limit;
  uint16_t shooter_id2_17mm_speed_limit;

  uint16_t shooter_id1_42mm_cooling_rate;
  uint16_t shooter_id1_42mm_cooling_limit;
  uint16_t shooter_id1_42mm_speed_limit;


  uint16_t chassis_power_limit;
  uint8_t mains_power_gimbal_output : 1;
  uint8_t mains_power_chassis_output : 1;
  uint8_t mains_power_shooter_output : 1;
} ext_game_robot_status_t;
```

Table 3-10 Real-time power and heat data: 0x0202. Transmission frequency: 50 Hz

Byte offset	Size	Description
0	2	Chassis output voltage (mV)
2	2	Chassis output current (mA)
4	4	Chassis output power (W)
8	2	Chassis power buffer (J) Notes: Launch ramp increases to 250 J according to rules
10	2	No. 1 17mm barrel heat

12	2	No. 2 17mm barrel heat
14	2	42mm barrel heat

```
typedef __packed struct
{
  uint16_t chassis_volt;
  uint16_t chassis_current;
  float chassis_power;
  uint16_t chassis_power_buffer;
  uint16_t shooter_id1_17mm_cooling_heat;
  uint16_t shooter_id2_17mm_cooling_heat;
  uint16_t shooter_id1_42mm_cooling_heat;
} ext_power_heat_data_t;
```

Table 3-11 Robot position: 0x0203. Transmission frequency: 10Hz

Byte offset	Size	Description
0	4	Coordinate x (m)
4	4	Coordinate y (m)
8	4	Coordinate z (m)
12	4	Barrel (degree)

```
typedef __packed struct
{
  float x;
  float y;
  float z;
  float yaw;
} ext_game_robot_pos_t;
```

Table 3-12 Robot buffs: 0x0204. Transmission frequency: 1 Hz

Byte offset	Size	Description
0	1	bit 0: Robot HP restoration status bit 1: Barrel heat cooling acceleration bit 2: Robot defense buff bit 3: Robot attack buff Other bits reserved

```
typedef __packed struct
{
```



```
  uint8_t power_rune_buff;
}ext_buff _t;
```

Table 3-13 Aerial robot's energy status: 0x0205. Transmission frequency: 10Hz

Byte offset	Size	Description
0	1	Attack time (s) 30s decreases to 0s

```
typedef __packed struct
{
    uint8_t attack_time;
} aerial_robot_energy_t;
```

Table 3-14 Damage status: 0x0206. Transmission frequency: Transmitted after the damage occurs

Byte offset	Size	Description
0	1	bits 0-3: When the HP change type is armor damage, it indicates the armor ID, and 0-4 indicate five armor plates of the robot; when it is other HP change types, the variable is 0. bits 4-7: type of HP changes 0x0 HP deduction as a result of armor damage 0x1 HP deduction as a result of module offline 0x2 HP deduction as a result of exceeding the speed limit 0x3 HP deduction as a result of exceeding the barrel heat 0x4 HP deduction as a result of exceeding the chassis power 0x5 HP deduction as a result of armor collision

```
typedef __packed struct
{
  uint8_t armor_id : 4;
  uint8_t hurt_type : 4;
} ext_robot_hurt_t;
```

Table 3-15 Real-time launching information: 0x0207. Transmission frequency: Transmitted after launching

Byte offset	Size	Description
0	1	Projectile type: 1: 17mm projectiles, 2: 42 mm projectile
1	1	Launching mechanism ID: 1: 17mm launching mechanism No. 1 2 : 17mm launching mechanism No. 2

Byte offset	Size	Description
		3: 42mm launching mechanism
2	1	Launch frequency (Hz)
3	4	Launch speed (m/s)

```
typedef __packed struct
{
  uint8_t bullet_type;
  uint8_t shooter_id;
  uint8_t bullet_freq;
  float bullet_speed;
} ext_shoot_data_t;
```

Table 3-16 Remaining projectiles: 0x0208. Transmission frequency: Transmitted by all robots periodically at a frequency of 10 Hz

Description of remaining 17mm projectiles	RoboMaster University League	Robotics University Championship
Standard Robot	Total number of remaining 17mm projectiles of the team's standard and hero robots	The number of remaining changeable 17mm projectiles of the team
Hero Robot	Total number of remaining 17mm projectiles of the team's standard and hero robots	The number of remaining changeable 17mm projectiles of the team
Aerial, Sentry	Total number of remaining 17mm projectiles of the robot	Total number of remaining 17mm projectiles of the robot

Byte offset	Size	Description
0	2	Number of remaining 17mm projectiles
2	2	Number of remaining 42mm projectiles
4	2	Number of remaining coins

```
typedef __packed struct
```



```
{
  uint16_t bullet_remaining_num_17mm;
  uint16_t bullet_remaining_num_42mm;
  uint16_t coin_remaining_num;
} ext_bullet_remaining_t;
```

Table 3-17 Robot RFID status: 0x0209. Transmission frequency: 1 Hz. Transmission range: Single robot

Byte offset	Size	Description
0	4	bit 0: RFID status of base gain zone bit 1: RFID status of elevated ground gain zone bit 2: RFID status of power rune activation point bit 3: RFID status of launch ramp gain zone bit 4: RFID status of outpost gain zone bit 6: RFID status of restoration zone buff bit 7: RFID status of engineer robot recovery card bits 8-31: Kept.

```
typedef __packed struct
{
  uint32_t rfid_status
} ext_rfid_status_t;
```

Table 3-18 Instruction data of dart robot client: 0x020A. Transmission frequency: 10 Hz. Transmission range: Single robot

Byte offset	Size	Description
0	1	Current status of dart launch opening 1: Closed 2: Is opening or closing 0: Opened
1	1	Attack target of dart, which is outpost by default 0: Outpost 1: Base
2	2	Remaining competition time when changing the attack target (s), which is 0 by default if no switch is made

Byte offset	Size	Description
4	1	Detected speed of the first dart (0.1m/s/LSB), which is 0 if no detection is performed

```
typedef __packed struct
{
  uint8_t dart_launch_opening_status;
  uint8_t dart_attack_target;
  uint16_t target_change_time;
  uint16_t operate_launch_cmd_time;
} ext_dart_client_cmd_t;
```

4. Interaction Data Between Robots

The interaction data includes a unified data segment header structure. The data segment includes the content ID, the sender and receiver IDs, and the content data segment. The total length of the entire interaction data packet has a maximum of 128 bytes, and nine bytes of frame_header, cmd_id, and frame_tail and six bytes of the data segment header structure are subtracted. Therefore, the content data segment sent has a maximum length of 113 bytes. A byte limit of the entire interaction data 0x0301 is shown in the following table. The data volume includes the bytes of frame-header, cmd_id, frame_tail, and the data segment header structure.

The total uplink and downlink bandwidth of each robot's interaction data and custom controller data does not exceed 5000 bytes. The uplink and downlink transmission frequencies each do not exceed 30 Hz.

Table 4-1 Interaction data receiving information: 0x0301

Byte offset	Size	Description	Remarks
0	2	Data segment content ID	-
2	2	Sender ID	The sender ID needs to be verified. For example, if RED 1 sends data to RED 5, RED 1 needs to be verified.
4	2	Receiver ID	The receiver ID needs to be verified. For example, data cannot be sent to an enemy robot ID.
6	x	Content data segment	The maximum value of x is 113

```
typedef __packed struct
{
  uint16_t data_cmd_id;
  uint16_t sender_ID;
```



```
  uint16_t receiver_ID;
}ext_student_interactive_header_data_t;
```

Content ID	Length (header length + content segment length)	Function Description
0x0200~0x02FF	6+n	Communication between own robots
0x0100	6+2	The client deletes a graphic.
0x0101	6+15	The client draws one graphic.
0x0102	6+30	The client draws two graphics.
0x0103	6+75	The client draws five graphics.
0x0104	6+105	The client draws seven graphics.
0x0110	6+45	The client draws a character graphic.

Ensure the bandwidth is properly configured since there are multiple content IDs and the maximum uplink frequency of the entire cmd_id is 10 Hz.

5. ID Description

Robot ID: 1. hero (red); 2. engineer (red); 3/4/5. standard (red); 6. aerial (red); 7. sentry (red); 9. radar station (red); 101. hero (blue); 102. engineer (blue); 103/104/105, standard (blue); 106. aerial (blue); 107. sentry (blue); 109. radar station (blue).

Client ID: 0x0101: hero operator client (red); 0x0102: engineer operator client (red); 0x0103/0x0104/0x0105: standard operator client (red); 0x0106: aerial operator client (red); 0x0165: hero operator client (blue); 0x0166: engineer operator client (blue); 0x0167/0x0168/0x0169: standard operator client (blue); 0x016A: aerial operator client (blue).

Communication between students' robots: cmd_id 0x0301; content ID: 0x0200~0x02FF

Table 5-1 Communication of interaction date between robots: 0x0301

Byte offset	Size	Description	Remarks
0	2	Data content ID	0x0200~0x02FF The above ID segments can be selected. The specific ID meaning shall be defined by the participating teams.

Byte offset	Size	Description	Remarks
2	2	Sender ID	The sender ID needs to be verified.
4	2	Receiver ID	The receiver ID needs to be verified. For example, data cannot be sent to an enemy robot ID.
6	n	Data segment	n needs to be less than 113.

```
typedef __pack struct
{
uint8_t data[]
} robot_interactive_data_t
```

Table 5-2 Communication between robots when client deletes a graphic: 0x0301

Byte offset	Size	Description	Remarks
0	2	Data content ID	0x0100
2	2	Sender ID	The sender ID needs to be verified.
4	2	Receiver ID	The receiver ID needs to be verified. For example, data can be sent to only a client corresponding to the robot.
6	1	Graphic operation	Including: 0: No operation 1: Delete a graphic layer 2: Delete all
7	1	Number of graphic layers	Number of graphic layers: 0~9

```
typedef __packed struct
{
uint8_t operate_tpye;
uint8_t layer;
} ext_client_custom _graphic_delete_t
```


Table 5-3 Graphic data

Byte offset	Size	Description	Remarks
0	3	Graphic name	Used as the index of the client in operations such as deleting and modifying
3	4	Graphic configuration	bit 0-2: Graphic operation: 0: No operation 1: add 2 : modify 3: delete bits 3-5: graphic type: 0: straight line 1: rectangle 2 : circle 3: ellipse 4: arc 5: floating number 6: integer 7: character bits 6-9: Number of graphic layers, 0-9 bits 10-13: color: 0: red and blue 1: yellow 2 : green 3: orange 4: purplish red 5: pink 6: cyan

Byte offset	Size	Description	Remarks
			7: black 8: white bits 14-22: Start angle (°), range [0, 360] bits 23-31: End angle (°), range [0, 360]
7	4	Graphic configuration 2	bits 0-9: Line width bits 10-20: Coordinate x of start point bits 21-31: Coordinate y of start point
11	4	Graphic configuration 3	bits 0-9: Font size or radius bits 10-20: Coordinate x of end point bits 21-31: Coordinate y of end point

```
typedef __packed struct
{
uint8_t graphic_name[3];
uint32_t operate_tpye:3;
uint32_t graphic_tpye:3;
uint32_t layer:4;
uint32_t color:4;
uint32_t start_angle:9;
uint32_t end_angle:9;
uint32_t width:10;
uint32_t start_x:11;
uint32_t start_y:11;
uint32_t radius:10;
uint32_t end_x:11;
uint32_t end_y:11;
} graphic_data_struct_t
```

For details about the graphic configuration, see the following table. Null indicates that the data of the field has no impact on the graphic. The recommended font size and line width ratio is 10:1.

Type	start_angle	end_angle	width	start_x	start_y	radius	end_x	end_y
Straight line	Null	Null	Line width	Coordinate x of start point	Coordinate y of start point	Null	Coordinate x of end point	Coordinate y of end point
Rectangle	Null	Null	Line width	Coordinate x of start point	Coordinate y of start point	Null	Coordinate x of diagonal vertex	Coordinate y of diagonal vertex
Circle	Null	Null	Line width	Coordinate x of center of a circle	Coordinate y of center of a circle	Radius	Null	Null
Ellipse	Null	Null	Line width	Coordinate x of center of a circle	Coordinate y of center of a circle	Null	Length of x axis	Length of y axis
Arc	Start angle	End angle	Line width	Coordinate x of center of a circle	Coordinate y of center of a circle	Null	Length of x axis	Length of y axis
Floating number	Font size	Number of valid decimal places	Line width	Coordinate x of start point	Coordinate y of start point	Multiply by 1000, 32-bit integer, int32_t		
Integer	Font size	Null	Line width	Coordinate x of start point	Coordinate y of start point	32-bit integer, int32_t		
Character	Font size	Character length	Line width	Coordinate x of start point	Coordinate y of start point	Null	Null	Null

Table 5-4 Communication between robots when client draws one graphic: 0x0301

Byte offset	Size	Description	Remarks
0	2	Data content ID	0x0101

Byte offset	Size	Description	Remarks
2	2	Sender ID	The sender ID needs to be verified.
4	2	Receiver ID	The receiver ID needs to be verified. For example, data can be sent to only a client corresponding to the robot.
6	15	Graphic 1	See graphic data introduction for details

```
typedef __packed struct
{
  graphic_data_struct_t   grapic_data_struct;
} ext_client_custom_graphic_single_t;
```

Table 5-5 Communication between robots when client draws two graphics: 0x0301

Byte offset	Size	Description	Remarks
0	2	Data content ID	0x0102
2	2	Sender ID	The sender ID needs to be verified.
4	2	Receiver ID	The receiver ID needs to be verified. For example, data can be sent to only a client corresponding to the robot.
6	15	Graphic 1	See graphic data introduction for details
21	15	Graphic 2	See graphic data introduction for details

```
typedef __packed struct
{
   graphic_data_struct_t   grapic_data_struct[2];
} ext_client_custom_graphic_double_t;
```

Table 5-6 Communication between robots when client draws five graphics: 0x0301

Byte offset	Size	Description	Remarks
0	2	Data content ID	0x0103
2	2	Sender ID	The sender ID needs to be verified.
4	2	Receiver ID	The receiver ID needs to be verified. For example, data can be sent to only a client corresponding to the robot.

Byte offset	Size	Description	Remarks
6	15	Graphic 1	See graphic data introduction for details
21	15	Graphic 2	See graphic data introduction for details
36	15	Graphic 3	See graphic data introduction for details
51	15	Graphic 4	See graphic data introduction for details
66	15	Graphic 5	See graphic data introduction for details

```
typedef __packed struct
{
  graphic_data_struct_t   grapic_data_struct[5];
} ext_client_custom_graphic_five_t;
```

Table 5-7 Communication between robots when client draws seven graphics: 0x0301

Byte offset	Size	Description	Remarks
0	2	Data content ID	0x0104
2	2	Sender ID	The sender ID needs to be verified.
4	2	Receiver ID	The receiver ID needs to be verified. For example, data can be sent to only a client corresponding to the robot.
6	15	Graphic 1	See graphic data introduction for details
21	15	Graphic 2	See graphic data introduction for details
36	15	Graphic 3	See graphic data introduction for details
51	15	Graphic 4	See graphic data introduction for details
66	15	Graphic 5	See graphic data introduction for details
81	15	Graphic 6	See graphic data introduction for details
96	15	Graphic 7	See graphic data introduction for details

```
typedef __packed struct
{
```



```
  graphic_data_struct_t   grapic_data_struct[7];
} ext_client_custom_graphic_seven_t;
```

Table 5-8 Communication between robots when client draws a character: 0x0301

Byte offset	Size	Description	Remarks
0	2	Data content ID	0x0110
2	2	Sender ID	The sender ID needs to be verified.
4	2	Receiver ID	The receiver ID needs to be verified. For example, data can be sent to only a client corresponding to the robot.
6	15	Character configuration	See graphic data introduction for details
21	30	Character	

```
typedef __packed struct
{
  graphic_data_struct_t   grapic_data_struct;
  uint8_t data[30];
} ext_client_custom_character_t;
```

6. Custom Controller Interaction Data

Custom controller data includes a unified segment header structure. The data segment is a content data segment. The total length of the entire interaction data packet has a maximum of 39 bytes, and nine bytes of frame_header, cmd_id, and frame_tail are subtracted. Therefore, the content data segment sent has a maximum of 30 bytes. The downlink frequency of the entire interaction data packet 0x0302 is 30 Hz.

Table 6-1 Interaction data receiving information: 0x0302. Transmission frequency: Transmitted at a frequency of 30 Hz at most

Byte offset	Size	Description	Remarks
0	x	Content segment data	The maximum value of x is 30.

```
typedef __pack struct
{
uint8_t data[]
} robot_interactive_data_t
```


7. Small Map Interaction Information

The small map interaction information includes a unified data segment header structure.

7.1 Client's Transmission of Information

Small map information transmission identifier: 0x0303. Transmission frequency: Transmitted when triggered

Table 7-1 Client's transmission of information

Byte offset	Size	Description	Remarks
0	4	Coordinate x of target (m)	When the target robot ID is sent, the entry is 0.
4	4	Coordinate y of target (m)	When the target robot ID is sent, the entry is 0.
8	4	Coordinate z of target (m)	When the target robot ID is sent, the entry is 0.
12	1	Keyboard information pressed by the aerial gimbal operator when sending the instruction	0 if no key is pressed
13	2	Target robot ID to be used	0 when the position information is sent

```
typedef __pack struct
{
float target_position_x;
float target_position_y;
float target_position_z;
uint8_t commd_keyboard;
uint16_t target_robot_ID;
} ext_robot_command_t
```

Robot ID: Robot ID: 1. hero (red); 2. engineer (red); 3/4/5. standard (red); 6. aerial (red); 7. sentry (red); 9. radar station (red); 10. outpost (red); 11. base (red); 101. hero (blue); 102. engineer (blue); 103/104/105. standard (blue); 106. aerial (blue); 107. sentry (blue); 109. radar station (blue); 110: outpost (blue); 111. base (blue).

7.2 Client's Receipt of Information

Small map information transmission identifier: 0x0305. Transmission frequency: 10Hz

The coordinate information sent by the radar station can be viewed by all operators of your own team through the first person view of the small map.

Table 7-2 Client's receipt of information

Byte offset	Size	Description	Remarks
0	2	Target robot ID	not displayed when x and y are out of boundaries
2	4	Coordinate x of target (m)	not displayed when x and y are out of boundaries
6	4	Coordinate y of target (m)	not displayed when x and y are out of boundaries

```
typedef __pack struct
{
uint16_t target_robot_ID;
float target_position_x;
float target_position_y;
} ext_client_map_command_t
```

8. Image Transmission Remote Control Information

The image transmission remote control information is transmitted through the image transmission module.

Image transmission remote control information identifier: 0x0304. Transmission frequency: 30 Hz

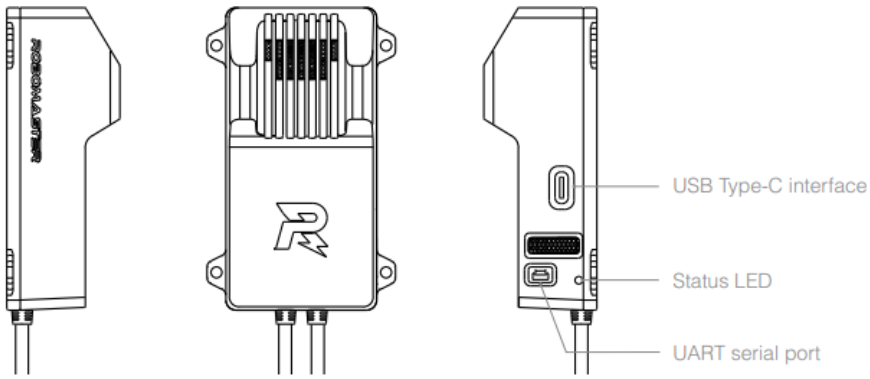


Figure 8-1 Sender

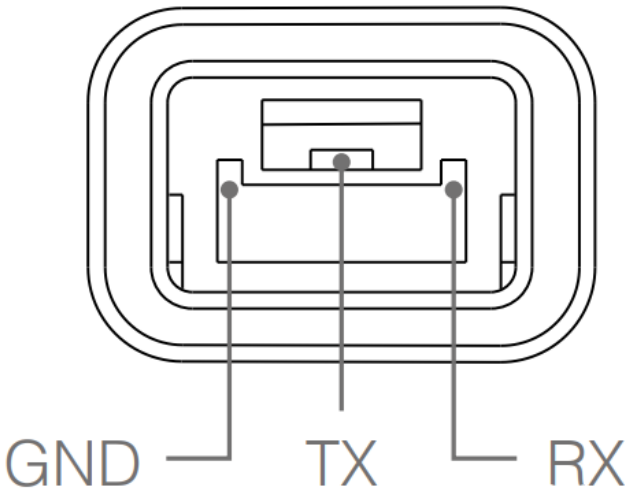


Figure 8-2 UART serial port sequence of sender

8.1 Client's Transmission of Information

Table 8-1 Client's transmission of information

Byte offset	Size	Description
0	2	X-axis information of mouse
2	2	Y-axis information of mouse
4	2	Mouse wheel information
6	1	Left mouse button
7	1	Right mouse button pressed
8	2	Keyboard information bit 0: Whether W is pressed bit 1: Whether S is pressed bit 2: Whether A is pressed bit 3: Whether D is pressed bit 4: Whether SHIFT is pressed bit 5: Whether CTRL is pressed bit 6: Whether Q is pressed bit 7: Whether E is pressed

Byte offset	Size	Description
		bit 8: Whether R is pressed bit 9: Whether F is pressed bit 10: Whether G is pressed bit 11: Whether Z is pressed bit 12: Whether X is pressed bit 13: Whether C is pressed bit 14: Whether V is pressed bit 15: Whether B is pressed
10	2	Reserved

```
typedef __pack struct
{
int16_t mouse_x;
int16_t mouse_y;
int16_t mouse_z;
int8 left_button_down;
int8 right_button_down;
uint16_t keyboard_value;
uint16_t reserved;
} ext_robot_command_t
```

9. CRC Example Code

```
//crc8 generator polynomial:G(x)=x8+x5+x4+1
const unsigned char CRC8_INIT = 0xff;
const unsigned char CRC8_TAB[256] =
{
0x00, 0x5e, 0xbc, 0xe2, 0x61, 0x3f, 0xdd, 0x83, 0xc2, 0x9c, 0x7e, 0x20, 0xa3, 0xfd, 0x1f, 0x41,
0x9d, 0xc3, 0x21, 0x7f, 0xfc, 0xa2, 0x40, 0x1e, 0x5f, 0x01, 0xe3, 0xbd, 0x3e, 0x60, 0x82, 0xdc, 0x23, 0x7d, 0x9f,
0xc1, 0x42, 0x1c, 0xfe, 0xa0, 0xe1, 0xbf, 0x5d, 0x03, 0x80, 0xde, 0x3c, 0x62, 0xbe, 0xe0, 0x02, 0x5c, 0xdf, 0x81,
0x63, 0x3d, 0x7c, 0x22, 0xc0, 0x9e, 0x1d, 0x43, 0xa1, 0xff, 0x46, 0x18, 0xfa, 0xa4, 0x27, 0x79, 0x9b, 0xc5, 0x84,
0xda, 0x38, 0x66, 0xe5, 0xbb, 0x59, 0x07, 0xdb, 0x85, 0x67, 0x39, 0xba, 0xe4, 0x06, 0x58, 0x19, 0x47, 0xa5,
0xfb, 0x78, 0x26, 0xc4, 0x9a, 0x65, 0x3b, 0xd9, 0x87, 0x04, 0x5a, 0xb8, 0xe6, 0xa7, 0xf9, 0x1b, 0x45, 0xc6,
0x98, 0x7a, 0x24, 0xf8, 0xa6, 0x44, 0x1a, 0x99, 0xc7, 0x25, 0x7b, 0x3a, 0x64, 0x86, 0xd8, 0x5b, 0x05, 0xe7,
0xb9,
0x8c, 0xd2, 0x30, 0x6e, 0xed, 0xb3, 0x51, 0x0f, 0x4e, 0x10, 0xf2, 0xac, 0x2f, 0x71, 0x93, 0xcd, 0x11, 0x4f, 0xad,
0xf3, 0x70, 0x2e, 0xcc, 0x92, 0xd3, 0x8d, 0x6f, 0x31, 0xb2, 0xec, 0x0e, 0x50, 0xaf, 0xf1, 0x13, 0x4d, 0xce, 0x90,
0x72, 0x2c, 0x6d, 0x33, 0xd1, 0x8f, 0x0c, 0x52, 0xb0, 0xee, 0x32, 0x6c, 0x8e, 0xd0, 0x53, 0x0d, 0xef, 0xb1, 0xf0,
0xae, 0x4c, 0x12, 0x91, 0xcf, 0x2d, 0x73, 0xca, 0x94, 0x76, 0x28, 0xab, 0xf5, 0x17, 0x49, 0x08, 0x56, 0xb4, 0xea,
```



```
0x69, 0x37, 0xd5, 0x8b, 0x57, 0x09, 0xeb, 0xb5, 0x36, 0x68, 0x8a, 0xd4, 0x95, 0xcb, 0x29, 0x77, 0xf4, 0xaa,
0x48, 0x16, 0xe9, 0xb7, 0x55, 0x0b, 0x88, 0xd6, 0x34, 0x6a, 0x2b, 0x75, 0x97, 0xc9, 0x4a, 0x14, 0xf6, 0xa8,
0x74, 0x2a, 0xc8, 0x96, 0x15, 0x4b, 0xa9, 0xf7, 0xb6, 0xe8, 0x0a, 0x54, 0xd7, 0x89, 0x6b, 0x35,
};
unsigned char Get_CRC8_Check_Sum(unsigned char *pchMessage,unsigned int dwLength,unsigned char
ucCRC8)
{
unsigned char ucIndex;
while (dwLength--)
{
ucIndex = ucCRC8^(*pchMessage++);
ucCRC8 = CRC8_TAB[ucIndex];
}
return(ucCRC8);
}
/*
** Descriptions: CRC8 Verify function
** Input: Data to Verify,Stream length = Data + checksum
** Output: True or False (CRC Verify Result)
*/
unsigned int Verify_CRC8_Check_Sum(unsigned char *pchMessage, unsigned int dwLength)
{
unsigned char ucExpected = 0;
if ((pchMessage == 0) || (dwLength <= 2)) return 0;
ucExpected = Get_CRC8_Check_Sum (pchMessage, dwLength-1, CRC8_INIT);
return ( ucExpected == pchMessage[dwLength-1] );
}
/*
** Descriptions: append CRC8 to the end of data
** Input: Data to CRC and append,Stream length = Data + checksum
** Output: True or False (CRC Verify Result)
*/
void Append_CRC8_Check_Sum(unsigned char *pchMessage, unsigned int dwLength)
{
unsigned char ucCRC = 0;
if ((pchMessage == 0) || (dwLength <= 2)) return;
ucCRC = Get_CRC8_Check_Sum ( (unsigned char *)pchMessage, dwLength-1, CRC8_INIT);
pchMessage[dwLength-1] = ucCRC;
}

uint16_t CRC_INIT = 0xffff;
const uint16_t wCRC_Table[256] =
```



```
{
0x0000, 0x1189, 0x2312, 0x329b, 0x4624, 0x57ad, 0x6536, 0x74bf,
0x8c48, 0x9dc1, 0xaf5a, 0xbed3, 0xca6c, 0xdbe5, 0xe97e, 0xf8f7,
0x1081, 0x0108, 0x3393, 0x221a, 0x56a5, 0x472c, 0x75b7, 0x643e,
0x9cc9, 0x8d40, 0xbfdb, 0xae52, 0xdaed, 0xcb64, 0xf9ff, 0xe876,
0x2102, 0x308b, 0x0210, 0x1399, 0x6726, 0x76af, 0x4434, 0x55bd,
0xad4a, 0xbcc3, 0x8e58, 0x9fd1, 0xeb6e, 0xfae7, 0xc87c, 0xd9f5,
0x3183, 0x200a, 0x1291, 0x0318, 0x77a7, 0x662e, 0x54b5, 0x453c,
0xbdcb, 0xac42, 0x9ed9, 0x8f50, 0xfbef, 0xea66, 0xd8fd, 0xc974,
0x4204, 0x538d, 0x6116, 0x709f, 0x0420, 0x15a9, 0x2732, 0x36bb,
0xce4c, 0xdfc5, 0xed5e, 0xfcd7, 0x8868, 0x99e1, 0xab7a, 0xbaf3,
0x5285, 0x430c, 0x7197, 0x601e, 0x14a1, 0x0528, 0x37b3, 0x263a,
0xdecd, 0xcf44, 0xfddf, 0xec56, 0x98e9, 0x8960, 0xbbfb, 0xaa72,
0x6306, 0x728f, 0x4014, 0x519d, 0x2522, 0x34ab, 0x0630, 0x17b9,
0xef4e, 0xfec7, 0xcc5c, 0xddd5, 0xa96a, 0xb8e3, 0x8a78, 0x9bf1,
0x7387, 0x620e, 0x5095, 0x411c, 0x35a3, 0x242a, 0x16b1, 0x0738,
0xffcf, 0xee46, 0xdcdd, 0xcd54, 0xb9eb, 0xa862, 0x9af9, 0x8b70,
0x8408, 0x9581, 0xa71a, 0xb693, 0xc22c, 0xd3a5, 0xe13e, 0xf0b7,
0x0840, 0x19c9, 0x2b52, 0x3adb, 0x4e64, 0x5fed, 0x6d76, 0x7cff,
0x9489, 0x8500, 0xb79b, 0xa612, 0xd2ad, 0xc324, 0xf1bf, 0xe036,
0x18c1, 0x0948, 0x3bd3, 0x2a5a, 0x5ee5, 0x4f6c, 0x7df7, 0x6c7e,
0xa50a, 0xb483, 0x8618, 0x9791, 0xe32e, 0xf2a7, 0xc03c, 0xd1b5,
0x2942, 0x38cb, 0x0a50, 0x1bd9, 0x6f66, 0x7eef, 0x4c74, 0x5dfd,
0xb58b, 0xa402, 0x9699, 0x8710, 0xf3af, 0xe226, 0xd0bd, 0xc134,
0x39c3, 0x284a, 0x1ad1, 0x0b58, 0x7fe7, 0x6e6e, 0x5cf5, 0x4d7c,
0xc60c, 0xd785, 0xe51e, 0xf497, 0x8028, 0x91a1, 0xa33a, 0xb2b3,
0x4a44, 0x5bcd, 0x6956, 0x78df, 0x0c60, 0x1de9, 0x2f72, 0x3efb,
0xd68d, 0xc704, 0xf59f, 0xe416, 0x90a9, 0x8120, 0xb3bb, 0xa232,
0x5ac5, 0x4b4c, 0x79d7, 0x685e, 0x1ce1, 0x0d68, 0x3ff3, 0x2e7a,
0xe70e, 0xf687, 0xc41c, 0xd595, 0xa12a, 0xb0a3, 0x8238, 0x93b1,
0x6b46, 0x7acf, 0x4854, 0x59dd, 0x2d62, 0x3ceb, 0x0e70, 0x1ff9,
0xf78f, 0xe606, 0xd49d, 0xc514, 0xb1ab, 0xa022, 0x92b9, 0x8330,
0x7bc7, 0x6a4e, 0x58d5, 0x495c, 0x3de3, 0x2c6a, 0x1ef1, 0x0f78
};
/*
** Descriptions: CRC16 checksum function
** Input: Data to check,Stream length, initialized checksum
** Output: CRC checksum
*/
uint16_t Get_CRC16_Check_Sum(uint8_t *pchMessage,uint32_t dwLength,uint16_t wCRC)
{
```



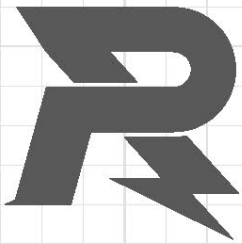
```
Uint8_t chData;
if (pchMessage == NULL)
{
return 0xFFFF;
}
while(dwLength--)
{
chData = *pchMessage++;
(wCRC) = ((uint16_t)(wCRC) >> 8) ^ wCRC_Table[((uint16_t)(wCRC) ^ (uint16_t)(chData)) & 0x00ff];
}
return wCRC;
}

/*
** Descriptions: CRC16 Verify function
** Input: Data to Verify,Stream length = Data + checksum
** Output: True or False (CRC Verify Result)
*/
uint32_t Verify_CRC16_Check_Sum(uint8_t *pchMessage, uint32_t dwLength)
{
uint16_t wExpected = 0;
if ((pchMessage == NULL) || (dwLength <= 2))
{
return __FALSE;
}
wExpected = Get_CRC16_Check_Sum ( pchMessage, dwLength - 2, CRC_INIT);
return ((wExpected & 0xff) == pchMessage[dwLength - 2] && ((wExpected >> 8) & 0xff) ==
pchMessage[dwLength - 1]);
}

/*
** Descriptions: append CRC16 to the end of data
** Input: Data to CRC and append,Stream length = Data + checksum
** Output: True or False (CRC Verify Result)
*/
void Append_CRC16_Check_Sum(uint8_t * pchMessage,uint32_t dwLength)
{
uint16_t wCRC = 0;
if ((pchMessage == NULL) || (dwLength <= 2))
{
return;
}
```



```
wCRC = Get_CRC16_Check_Sum ( (U8 *)pchMessage, dwLength-2, CRC_INIT );
pchMessage[dwLength-2] = (U8)(wCRC & 0x00ff);
pchMessage[dwLength-1] = (U8)((wCRC >> 8)& 0x00ff);
```

E-mail: robomaster@dji.com
Forum: bbs.robomaster.com
Website: www.robomaster.com
Tel: +86 (0)755 36383255 (GTC+8, 10:30AM-7:30PM, Monday to Friday)
Address: Room 202, Floor 2, Integrated Circuit Design & Application Industrial Park, No. 1089, Chaguang Road, Xili County, Nanshan District, Shenzhen City, Guangdong Province, China