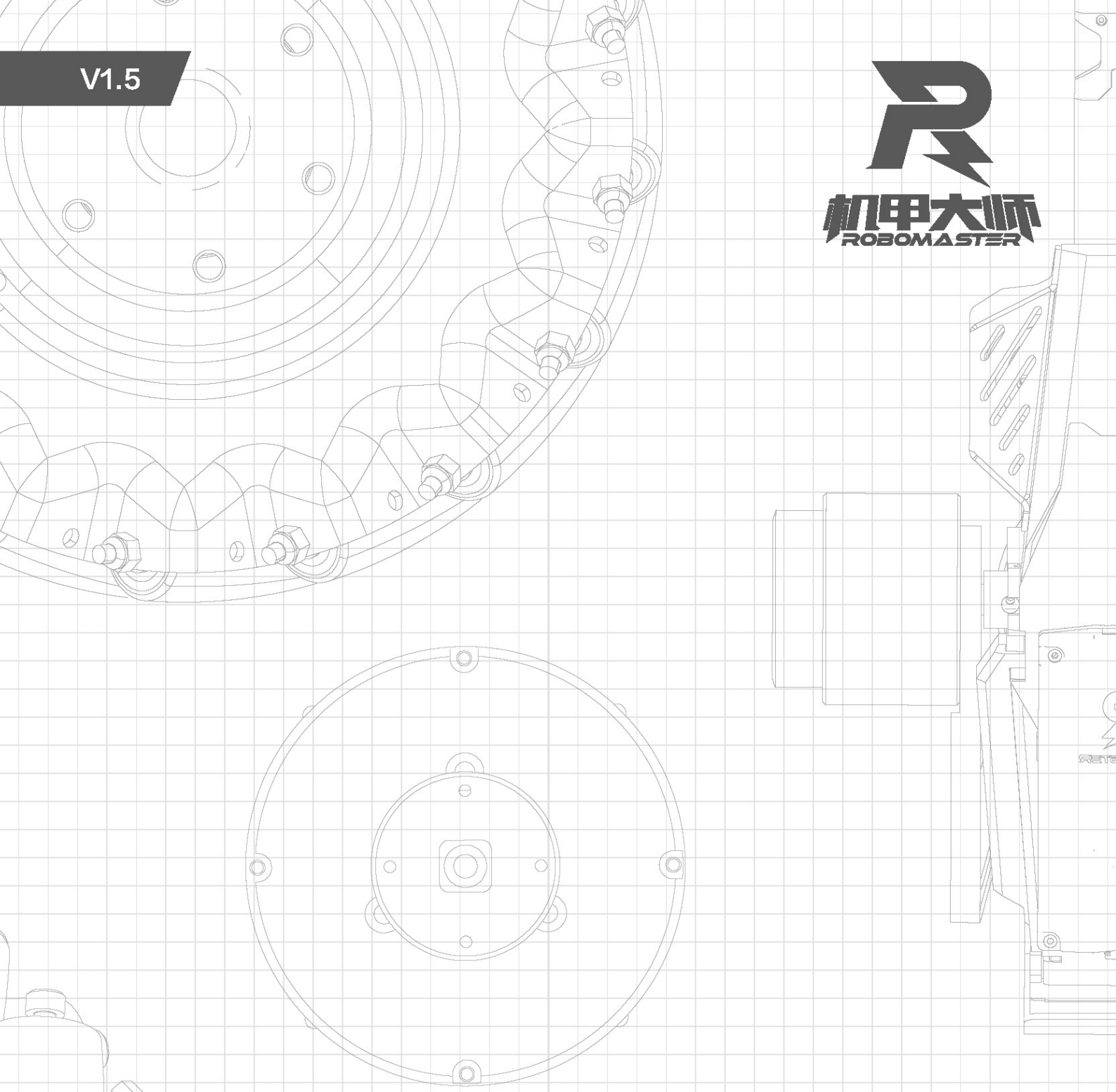
V1.5



Referee System Serial Port Protocol Appendix

Prepared by RoboMaster Organizing Committee

Updated in July, 2023

Change Log

Date	Version	Changes
July 17, 2023	V1.5	1. Added to the descriptions of the transmission/trigger conditions and sender/receiver of all command words. 2. Added to the descriptions of the regular link and VTM link of the Referee System. 3. Revised command codes 0x0101, 0x0204, 0x0205, 0x0208, 0x0209. 4. Added command codes 0x020B, 0X020C, 0x0307, 0x0306. 5. Deleted command codes 0x0004, 0x0005, 0x0103. 6. Optimized some descriptions.
August 5, 2022	V1.4	Revised the transmission frequencies and triggering methods for command codes 0x000, 0x0003, 0x0101, 0x0105, 0x0204, 0x0208, and 0x0209.
December 31, 2021	V1.3	Revised the distribution of the Buff/Debuff Zones and lurking mode in the RoboMaster AI Challenge. 0x0005.
April 30, 2021	V1.2	Fixed some errors.
April 19, 2021	V1.1	Fixed some errors.
February 30, 2021	V1.0	First Release

TABLE OF CONTENTS

1. Serial Port Protocol Format

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.

Table 1-1 Communication protocol format

frame_header	cmd_id	Data	frame_tail
5-byte	2-byte	n-byte	2-byte, CRC16, full packet check

Table 1-2 Frame header format

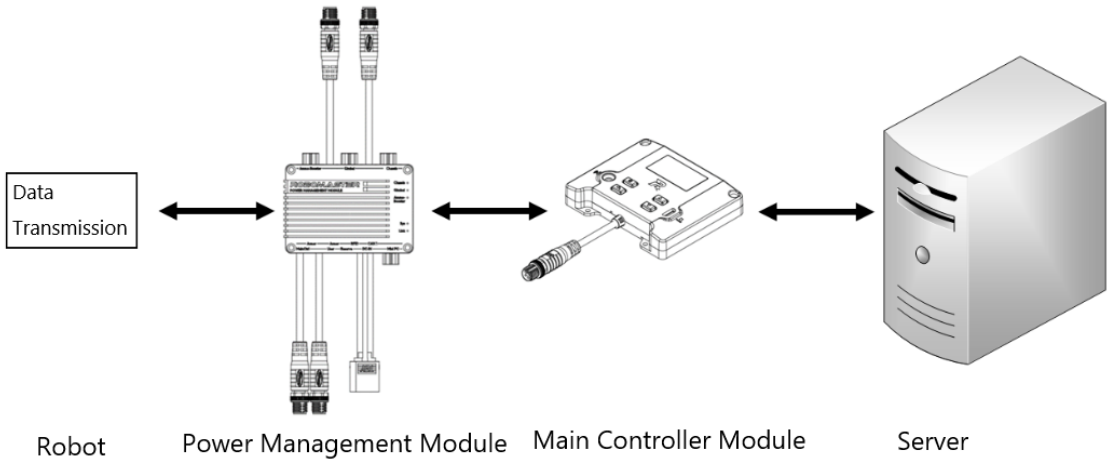
SOF	data_length	seq	CRC8
1-byte	2-byte	1-byte	1-byte

Table 1-3 Frame header definitions

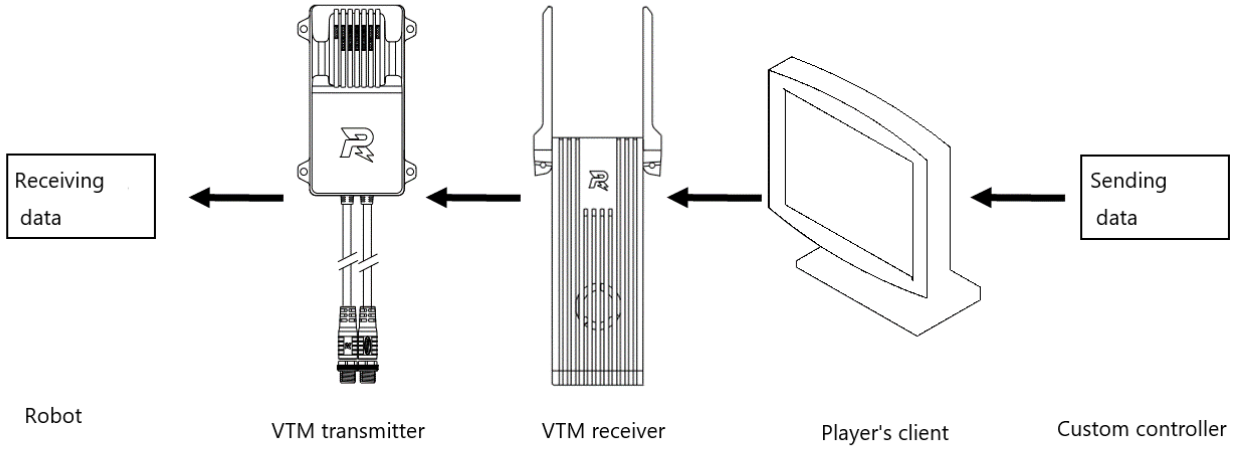
Domain	Offset position	Size (byte)	Detailed description
SOF	0	1	Data frame start byte; the fixed value is 0×A5
data_length	1	2	Length of data in the data frame
seq	3	1	Sequence number
CRC8	4	1	Frame header CRC8

There are two types of serial port data links for the Referee System: regular links and VTM Links.

- For regular links, data is relayed via the Referee System's server and Main Controller Module, where it is transmitted and received by the user serial port of the Power Management Module, as shown below:



- VTM Links involve relaying data through the Referee System's player client and VTM, where it is transmitted and received by the serial port of the VTM (Transmitter), as shown below:



Latency may occur for data links of the Referee System, which can result in data measurement error. The data is for your reference only.

2. Command Code IDs

Table2-1 Command code ID overview

Command code	Data segment length	Function description/transmission trigger conditions	Sender/receiver
0x0001	11	Competition status data; transmitted at a fixed frequency of 3Hz	Server→All robots
0x0002	1	Competition result data; transmitted at end of the competition	Server→All robots
0x0003	32	Robot health data; transmitted at a fixed frequency of 3Hz	Server→All robots
0x0101	4	Site event data; transmitted at a fixed frequency of 3Hz	Server→All own team's robots
0x0102	4	Official Projectile Supplier action identifier data; transmitted when the Official Projectile Supplier releases projectiles	Server→All own team's robots
0x0104	2	Referee warning data; transmitted when one's team is issued a penalty/forfeiture	Server→All robots of the penalized team
0x0105	1	Dart launching time data; transmitted at a fixed frequency of 3Hz	Server→All own team's robots
0x0201	27	Robot performance system data; transmitted at a fixed frequency of 10Hz	Main Controller Module→Corresponding robot
0x0202	16.	Real-time power heat data; transmitted at a fixed frequency of 50Hz	Main Controller Module→Corresponding robot
0x0203	16.	Robot position data; transmitted at a fixed frequency of 10Hz	Main Controller Module→Corresponding robot

Command code	Data segment length	Function description/transmission trigger conditions	Sender/receiver
0x0204	1	Robot buff data; transmitted at a fixed frequency of 3Hz	Server→Corresponding robot
0x0205	1	Air support time data; transmitted at a fixed frequency of 10Hz	Server→Own team's Aerial Robots
0x0206	1	Damage status data; transmitted after the damage occurs	Main Controller Module→Corresponding robot
0x0207	7	Real-time launching data; transmitted after a projectile is launched	Main Controller Module→Corresponding robot
0x0208	6	Projectile Allowance; transmitted at a fixed frequency of 10Hz	Server→Own team's Hero, Standard, Sentry, and Aerial Robots
0x0209	4	Robot RFID status; transmitted at a fixed frequency of 3Hz	Server→Own team's robots with an RFID module
0x020A	6	Dart player client command data; transmitted at a fixed frequency of 10Hz after the dart gate is online	Server→Own team's dart robots
0x020B	40	Ground Robot position data; transmitted at a fixed frequency of 1Hz	Server→Own team's Sentry Robots
0x020C	6	Radar marked progress data; transmitted at a fixed frequency of 1Hz	Server→Own team's radar robots
0x0301	128	Robot interaction data; transmitted at a maximum frequency of 10 Hz when triggered by the sender	-

Command code	Data segment length	Function description/transmission trigger conditions	Sender/receiver
0x0302	30	Custom Controller and robot interaction data; transmitted at a maximum frequency of 30 Hz when triggered by the sender	Custom Controller→Robots with VTM links to player clients
0x0303	15	Player client's small map interaction data; transmitted when triggered by the player client	Tap the player client→Server→Sender's own chosen robots
0x0304	12	Keyboard, mouse, and remote control data; transmitted at a fixed frequency of 30Hz	Client→Robots with VTM links to player clients
0x0305	10	Player client's small map receiving radar data; transmitted at a maximum frequency of 10Hz	Radar→Server→All own team's player clients
0x0306	8	Custom Controller and robot interaction data; transmitted at a maximum frequency of 30 Hz when triggered by the sender	Custom Controller→Player client
0x0307	103	Player client's small map receiving sentry data; transmitted at a maximum frequency of 1Hz	Sentry→ Own team's Aerial Gimbal Operator player client

Table 2-2 0x0001

Byte offset	Size	Descriptions
0	1	0-3 bit: Competition type • 1: RoboMaster University Championship • 2: RoboMaster University Technical Challenge • 3: ICRA RoboMaster University AI Challenge • 4: RoboMaster University League 3V3 Match • 5: RoboMaster University League Standard Match 4-7 bit: Current stage

Byte offset	Size	Descriptions
		• 0: Pre-competition stage • 1: Setup period • 2: Initialization stage • 3: 5-second countdown • 4: Competition ongoing • 5: Calculating competition results
1	2	Remaining time in the current round; unit: second
3	8	UNIX time, effective after the robot is correctly linked to the Referee System's NTP server

```
typedef _packed struct
{
   uint8_t game_type : 4;
   uint8_t game_progress : 4;
   uint16_t stage_remain_time;
   uint64_t SyncTimeStamp;
}game_status_t;
```

Table 2-3 0x0002

Byte offset	Size	Descriptions
0	1	• 0: Draw • 1: Red Team wins • 2: Blue Team wins

```
typedef _packed struct
{
    uint8_t winner;
}game_result_t;
```

Table 2-4 0x0003

Byte offset	Size	Descriptions
0	2	RED 1 Hero HP. If the robot has not entered the stage or has been ejected, the HP is zero.
2	2	RED 2 Engineer HP

Byte offset	Size	Descriptions
4	2	RED 3 Standard HP
6	2	RED 4 Standard HP
8	2	RED 5 Standard HP
10	2	RED 7 Sentry HP
12	2	Red Outpost HP
14	2	Red Base HP
16.	2	Blue 1 Hero HP
18	2	Blue 2 Engineer HP
20	2	Blue 3 Standard HP
22	2	Blue 4 Standard HP
24	2	Blue 5 Standard HP
26	2	Blue 7 Sentry HP
28	2	Blue Outpost HP
30	2	Blue 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;
}game_robot_HP_t;
```

Table 2-5 0x0101

Byte offset	Size	Descriptions
0	8	0: Non-occupied/non-activated 1: Occupied/activated bit 0-2: ● bit 0: Occupation status of the Restoration Zone in front of own Official Projectile Supplier; “1” means it is occupied. ● bit 1: Occupation status of the Restoration Zone to the left of own Official Projectile Supplier (facing the Restoration Zone); “1” means it is occupied. ● bit 2: Occupation status of the Restoration Zone to the right of own Official Projectile Supplier (facing the Restoration Zone); “1” means it is occupied. bits 3-5: Status of own Power Rune: ● bit 3: Occupation status of own Power Rune Activation Point; “1” means it is occupied. ● bit 4: Activation status of own Small Power Rune; “1” means it is activated. ● bit 5: Activation status of own Large Power Rune; “1” means it is activated. bit 6: Occupation status of own Ring-Shaped Elevated Ground; “1” means it is occupied. bit 7: Occupation status of own R3 Trapezoid-Shaped Elevated Ground; “1” means it is occupied. bit 8: Occupation status of own R4 Trapezoidal Elevated Ground; “1” means it is occupied. bit9-16: Value of own Base’s Virtual Shield (0-250) bit17-27: HP of own Outpost (0-1500) bit28: Whether the Sentry is in own Patrol Zone bit29-31: Reserved

```
typedef _packed struct
{
  uint32_t event_data;
}event_data_t;
```


Table 2-6 0x0102

Byte offset	Size	Descriptions
0	1	Official Projectile Supplier Outlet ID: 1: Official Projectile Supplier Outlet to the left of own side (facing the Official Projectile Supplier) 2: Official Projectile Supplier Outlet to the right of own side (facing the Official Projectile Supplier)
1	1	Reloading robot ID: ● 0: No robot is reloading projectiles ● 1: Red Hero Robot is reloading projectiles ● 3/4/5: Red Standard Robot is reloading projectiles ● 101: Blue Hero Robot is reloading projectiles ● 103/104/105: Blue Standard Robot is reloading projectiles
2	1	Status of the projectile outlet: ● 0: Closed ● 1: Preparing projectiles ● 2: Releasing projectiles
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 2-7 0x0104

Byte offset	Size	Descriptions
0	1	Penalty level: ● 1: Yellow Card ● 2: Red Card ● 3: Forfeiture
1	1	● Offending robot ID (e.g., Red 1 Robot's ID is 1, Blue 1 Robot's ID is 101) ● In the case of a forfeiture or where both teams have been issued a Yellow Card, the value is 0.

```
typedef _packed struct
{
  uint8_t level;
  uint8_t offending_robot_id;
}referee_warning_t;
```

Table 2-8 0x0105

Byte offset	Size	Descriptions
0	1	Own team's remaining time for dart launching; unit: second

```
typedef _packed struct
{
  uint8_t dart_remaining_time;
}dart_remaining_time_t;
```

Table 2-9 0x0201

Byte offset	Size	Descriptions
0	1	Current robot ID
1	1	Robot level: ● 1: Level 1 ● 2: Level 2 ● 3: Level 3
2	2	Robot's current HP
4	2	Robot maximum HP

Byte offset	Size	Descriptions
6	2	Robot 1 17mm barrel cooling value per second
8	2	Robot 1 17mm barrel heat limit
10	2	Robot 1 17mm Launching Mechanism's Initial Launch Speed Limit (unit: 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 Launching Mechanism's Initial Launch Speed Limit (unit: m/s)
18	2	Robot 1 42mm barrel cooling value per second
20	2	Robot 42mm barrel heat limit
22	2	Robot 1 42mm Launching Mechanism's Initial Launch Speed Limit (unit: m/s)
24	2	Robot Chassis Power Consumption Limit
26	1	Power Management Module output status: ● 0 bit: Output from gimbal port: "0" denotes zero output, "1" denotes 24V output. ● 1 bit: Output from chassis port: "0" denotes zero output, "1" denotes 24V output. ● 2 bit: Output from shooter port: "0" denotes zero output, "1" denotes 24V output.

```
typedef _packed struct
{
  uint8_t robot_id;
  uint8_t robot_level;
  uint16_t current_HP;
  uint16_t maximum_HP;
  uint16_t shooter_id1_17mm_barrel_cooling_value;
  uint16_t shooter_id1_17mm_barrel_heat_limit;
  uint16_t shooter_id1_17mm_initial_launching_speed_limit;
  uint16_t shooter_id2_17mm_barrel_cooling_valuecooling_rate;
  uint16_t shooter_id2_17mm_barrel_heatcooling_limit;
```



```
  uint16_t shooter_id2_17mm initial_launching_speed_limit;
  uint16_t shooter_id1_42mm_ barrel_cooling_ value;
  uint16_t shooter_id1_42mm_barrel_heat_cooling_limit;
  uint16_t shooter_id1_42mm_initial_launching_speed_limit;
  uint16_t chassis_power_limit;
  uint8_t power_management_gimbal_output : 1;
  uint8_t power_management_chassis_output : 1;
  uint8_t power_management_shooter_output : 1;
}robot_status_t;
```

Table 2-10 0x0202

Byte offset	Size	Descriptions
0	2	Power Management Module chassis port output voltage (unit: mV)
2	2	Power Management Module chassis port output current (unit: mA)
4	4	Chassis Power (unit: W)
8	2	Buffer Energy (unit: J)
10	2	Barrel heat of the 1st 17mm Launching Mechanism
12	2	Barrel heat of the 2nd Launching Mechanism
14	2	Barrel heat of the 42mm Launching Mechanism

```
typedef _packed struct
{
  uint16_t chassis_voltage;
  uint16_t chassis_current;
  float chassis_power;
  uint16_t buffer_energy;
  uint16_t shooter_17mm_1_barrel_heat;
  uint16_t shooter_17mm_2_barrel_heat;
  uint16_t shooter_42mm_barrel_heat;
}power_heat_data_t;
```

Table 2-11 0x0203

Byte offset	Size	Descriptions
0	4	The x-coordinate of this robot’s position; unit: m
4	4	The y-coordinate of this robot’s position; unit: m
8	4	The z-coordinate of this robot’s position; unit: m

Byte offset	Size	Descriptions
12	4	Direction of this robot's Speed Monitor Module; unit: degree. True north is 0 degrees.

```
typedef _packed struct
{
   float x;
   float y;
   float z;
   float angle;
}robot_pos_t;
```

Table 2-12 0x0204

Byte offset	Size	Descriptions
0	1	Robot's HP Recovery Buff (in percentage; a value of 10 means the maximum HP recovery per second is 10%)
1	1	Robot barrel cooling rate (in absolute value; a value of 5 means a cooling rate of 5 times)
2	1	Robot defense buff (in percentage; a value of 50 means a defense buff of 50%)
3	2	Robot attack buff (in percentage; a value of 50 means an attack buff of 50%)

```
typedef _packed struct
{
   uint8_t recovery_buff;
   uint8_t cooling_buff;
   uint8_t defence_buff;
   uint16_t attack_buff;
}buff_t;
```

Table 2-13 0x0205

Byte offset	Size	Descriptions
0	1	Aerial robot status (0 means it is cooling, 1 means cooling is completed, and 2 means air support is ongoing)
1	1	The remaining time of this status (unit is "s", rounded down to the nearest integer, and when the remaining cooling time is 1.9s, the value is 1) If the cooling time is 0 and air support is not called, the value is 0.

```
typedef _packed struct
{
```



```
   uint8_t airforce_status;
   uint8_t time_remain;
}air_support_data_t;
```

Table 2-14 0x0206

Byte offset	Size	Descriptions
0	1	bit0-3: When HP deduction is due to an Armor Module or Speed Monitor Module, the 4-bit value is the ID of the Armor Module or Speed Monitor Module; if it is due to other reasons, the value is 0. bit4-7: type of HP changes ● 0 HP deduction due to Armor being hit by projectiles ● 1 HP deduction due to the Critical Module Going Offline Referee System Modules going offline ● 2 HP deduction due to exceeding the Initial Launching Speed Limit ● 3 HP deduction due to exceeding Barrel Heat Limit ● 4 HP deduction due to exceeding Chassis Power Consumption Limit ● 5 HP deduction due to the Armor Module suffering a collision

```
typedef _packed struct
{
   uint8_t armor_id : 4;
   uint8_t HP_deduction_reason : 4;
}hurt_data_t;
```

Table 2-15 0x0207

Byte offset	Size	Descriptions
0	1	Projectile types: ● 1: 17mm projectile ● 2: 42mm projectile
1	1	Launching mechanism ID: ● 1: First 17mm Launching Mechanism ● 2: Second 17mm Launching Mechanism ● 3: 42mm launching mechanism

Byte offset	Size	Descriptions
2	1	Projectile launch speed (unit: Hz)
3	4	Projectile initial speed (unit: m/s)

```
typedef _packed struct
{
   uint8_t bullet_type;
   uint8_t shooter_number;
   uint8_t launching_frequency;
   float initial_speed;
}shoot_data_t;
```

Table 2-16 0x0208

Byte offset	Size	Descriptions
0	2	17mm projectile allowance
2	2	42mm projectile allowance
4	2	Number of remaining coins

```
typedef _packed struct
{
   uint16_t projectile_allowance_17mm;
   uint16_t projectile_allowance_42mm;
   uint16_t remaining_gold_coin;
}projectile_allowance_t;
```

Table 2-17 0x0209

Byte offset	Size	Descriptions
0	4	Meaning of 1/0 bit value: Whether the buff point's RFID is detected ● bit0: Own Base Buff Point ● bit1: Own Elevated Ground Buff Point ● bit2: Opponent's Elevated Ground Buff Point ● bit3: Own R3/B3 Trapezoid-Shaped Elevated Ground ● bit4: Opponent's R3/B3 Trapezoid-Shaped Elevated Ground ● bit5: Own R4/B4 Trapezoid-Shaped Elevated Ground ● bit6: Opponent's R4/B4 Trapezoid-Shaped Elevated Ground ● bit7: Own Power Rune Activation Point

Byte offset	Size	Descriptions
		● bit8: Own Launch Ramp Buff Point (in front of the Launch Ramp near own side) ● bit9: Own Launch Ramp Buff Point (behind the Launch Ramp near own side) ● bit10: Opponent's Launch Ramp Buff Point (in front of the Launch Ramp near the other side) ● bit11: Opponent's Launch Ramp Buff Point (behind the Launch Ramp near the other side) ● bit12: Own Outpost Buff Point ● bit13: Own Restoration Zone (deemed activated if any one is detected) ● bit14: Own Sentry Patrol Zones ● bit15: Opponent's Sentry Patrol Zones ● bit16: Own Large Resource Island Buff Point ● bit17: Opponent's Large Resource Island Buff Point ● bit18: Own Controlled Zones ● bit19: Opponent's Controlled Zones ● bit20-31: Reserved Note: The RFID of the Base Buff Point, Elevated Ground Buff Point, Outpost Buff Point, Launch Ramp Buff Point, Resource Island Buff Point, Restoration Zone, Controlled Zones, and Sentry Patrolled Zones are only effective within the competition. If an RFID is detected outside the competition, the corresponding value remains 0.

```
typedef _packed struct
{
  uint32_t rfid_status;
}rfid_status_t;
```

Table 2-18 0x020A

Byte offset	Size	Descriptions
0	1	Current status of the Dart Launching Station: ● 1: Closed

Byte offset	Size	Descriptions
		● 2: Opening or closing ● 0: Opened
1	1	Dart target: (Outpost as default) ● 0: Outpost ● 1: Base
2	2	Time remaining in the competition when switching targets; unit: s; the value is defaulted to 0 if no target switching is involved.
4	2	Time remaining in the competition when the Operator confirms the launch command for the last time; unit: s; the initial value is 0.

```
typedef _packed struct
{
  uint8_t dart_launch_opening_status;
  uint8_t dart_attack_target;
  uint16_t target_change_time;
  uint16_t latest_launch_cmd_time;
}dart_client_cmd_t;
```

Table 2-19 0x020B

The intersection of the site perimeter wall near the Red Team's Official Projectile Supplier is the origin; the orientation of the site's longer edge facing the Blue Team is the positive x-axis direction, while the orientation of the site's shorter edge facing the Red Team's Landing Pad is the positive y-axis direction.

Byte offset	Size	Descriptions
0	4	The x-axis coordinate of own Hero Robot; unit: m
4	4	The y-axis coordinate of own Hero Robot; unit: m
8	4	The x-axis coordinate of own Engineer Robot; unit: m
12	4	The y-axis coordinate of own Engineer Robot; unit: m
16.	4	The x-axis coordinate of own Standard Robot No. 3; unit: m

Byte offset	Size	Descriptions
20	4	The y-axis coordinate of own Standard Robot No. 3; unit: m
24	4	The x-axis coordinate of own Standard Robot No. 4; unit: m
28	4	The y-axis coordinate of own Standard Robot No. 4; unit: m
32	4	The x-axis coordinate of own Standard Robot No. 5; unit: m
36	4	The y-axis coordinate of own Standard Robot No. 5; unit: m

```
typedef _packed struct
{
  float hero_x;
  float hero_y;
  float engineer_x;
  float engineer_y;
  float standard_3_x;
  float standard_3_y;
  float standard_4_x;
  float standard_4_y;
  float standard_5_x;
  float standard_5_y;
}ground_robot_position_t;
```

Table 2-20 0x020C

Byte offset	Size	Descriptions
0	1	Marked progress of opponent's Hero Robot: 0-120
1	1	Marked progress of opponent's Engineer Robot: 0-120
2	1	Marked progress of opponent's Standard Robot No. 3: 0-120
3	1	Marked progress of opponent's Standard Robot No. 4: 0-120
4	1	Marked progress of opponent's Standard Robot No. 5: 0-120
5	1	Marked progress of opponent's Sentry Robot: 0-120

```
typedef _packed struct
{
  uint8_t mark_hero_progress;
  uint8_t mark_engineer_progress;
  uint8_t mark_standard_3_progress;
  uint8_t mark_standard_4_progress;
  uint8_t mark_standard_5_progress;
```



```
  uint8_t mark_sentry_progress;
}radar_mark_data_t;
```

The robot interaction data is transmitted through regular links. Its data segments include a unified data segment header structure. A data segment header structure consists of the sub-content ID, sender and receiver ID, and content data segment. The total length of a student interaction data packet must not exceed 128 characters. After deducting the 9 characters for frame_header, cmd_id, and frame_tail, and the 6 characters for the data header structure, the maximum length of a student interaction data segment is 113 characters.

Every 1000 ms, the maximum data length a Hero, Standard, Engineer, Aerial Robot or dart can receive is 3720 characters; while for radars and Sentry Robots, the limit is 5120 characters.

Table 2-21 0x0301

Byte offset	Size	Descriptions	Notes
0	2	Sub-content ID	Must be an open sub-content ID
2	2	Sender ID	Must match with own ID. See the appendix for the IDs.
4	2	Receiver ID	● Limited to communication within own team ● Must be an inter-robot communication receiver permitted by the rules ● If the receiver is a player client, the data can only be transmitted to the player client corresponding to the sender. ● For the IDs please see the appendix.
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;
  uint8_t user_data[x];
}robot_interaction_data_t;
```

Sub-content ID	Content data segment length	Function Description
0x0200~0x02FF	x≤113	Inter-robot communication
0x0100	2	Player client deletes graphic layers.

Sub-content ID	Content data segment length	Function Description
0x0101	15	Player client draws one graphic.
0x0102	30	Player client draws two graphics.
0x0103	75	Player client draws five graphics.
0x0104	105	Player client draws seven graphics.
0x0110	45	Player 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.

Table 2-22 Sub-content ID: 0x0100

Byte offset	Size	Descriptions	Notes
0	1	Delete operation	● 0: No operation ● 1: Delete graphic layers ● 2: Delete all
1	1	Number of graphic layers	Number of graphic layers: 0~9

```
typedef _packed struct
{
  uint8_t delete_type;
  uint8_t layer;
}interaction_layer_delete_t;
```

Table 2-23 Sub-content ID: 0x0101

Byte offset	Size	Descriptions	Notes
0	3	Graphic name	Used as an index in graphic layer deletion, revision and other operations.
3	4	Graphic configuration	bit 0-2: graphic operation: ● 0: No operation ● 1: Add

Byte offset	Size	Descriptions	Notes
			● 2: Revise ● 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 bit 6-9: number of graphic layers (0-9) bits 10-13: color: ● 0: Red/Blue (Own team's color) ● 1: Yellow ● 2: Green ● 3: Orange ● 4: Purplish red ● 5: Pink ● 6: Cyan ● 7: Black ● 8: White bits 14-31: The meaning differs according to the drawn graphics, as shown below.
7	4	Graphic configuration 2	bit 0-9: Line width; recommended ratio between font size and line width is 10: 1

Byte offset	Size	Descriptions	Notes
			bits 10-20: Start point/origin's x-coordinate bits 21-31: Start point/origin's y-coordinate
11	4	Graphic configuration 3	The meaning differs according to the drawn graphics, as shown below.

```
typedef _packed struct
{
  uint8_t figure_name[3];
  uint32_t operate_tpye:3;
  uint32_t figure_tpye:3;
  uint32_t layer:4;
  uint32_t color:4;
  uint32_t details_a:9;
  uint32_t details_b:9;
  uint32_t width:10;
  uint32_t start_x:11;
  uint32_t start_y:11;
  uint32_t details_c:10;
  uint32_t details_d:11;
  uint32_t details_e:11;
}interaction_figure_t;
```

Graphic detail parameter description

Type	details_a	details_b	details_c	details_d	details_e
Straight line	-	-	-	Coordinate x of end point	Coordinate y of end point
Rectangle	-	-	-	Coordinate x of diagonal vertex	Coordinate y of diagonal vertex
Circle	-	-	Radius	-	-
Ellipse	-	-	-	Length of x axis	Length of y axis

Type	details_a	details_b	details_c	details_d	details_e
Arc	Start angle	End angle	-	Length of x axis	Length of y axis
Floating number	Font size	No effect	Divide the value by 1000 to obtain the actual displayed value		
Integer	Font size	-	32-bit integer, int32_t		
Character	Font size	Character length	-	-	-

Angle value means: 0° points at 12 o'clock, drawn in the clockwise direction;

Screen position: (0, 0) is at the lower left corner of the screen, while (1920, 1080) is the upper right corner;

Floating number: All integers are 32-bit; for the floating number, the actual displayed value is the entered value/1000, e.g., if the value entered corresponding to details_c, details_d, and details_e is 1234, the actual displayed value at the player client will be 1.234.

The graphics may still appear even if the transmitted value exceeds the restrictions of its corresponding data type, but the display quality may be compromised.

Table 2-24 Sub-content ID: 0x0102

Byte offset	Size	Descriptions	Notes
0	15	Graphic 1	Same as the 0x0101 data segment.
15	15	Graphic 2	Same as the 0x0101 data segment.

```
typedef _packed struct
{
  interaction_figure_t interaction_figure[2];
}interaction_figure_2_t;
```


Table 2-25 Sub-content ID: 0x0103

Byte offset	Size	Descriptions	Notes
0	15	Graphic 1	Same as the 0x0101 data segment.
15	15	Graphic 2	Same as the 0x0101 data segment.
30	15	Graphic 3	Same as the 0x0101 data segment.
45	15	Graphic 4	Same as the 0x0101 data segment.
60	15	Graphic 5	Same as the 0x0101 data segment.

```
typedef _packed struct
{
  interaction_figure_t interaction_figure[5];
}interaction_figure_3_t;
```

Table 2-26 Sub-content ID: 0x0104

Byte offset	Size	Descriptions	Notes
0	15	Graphic 1	Same as the 0x0101 data segment.
15	15	Graphic 2	Same as the 0x0101 data segment.
30	15	Graphic 3	Same as the 0x0101 data segment.
45	15	Graphic 4	Same as the 0x0101 data segment.
60	15	Graphic 5	Same as the 0x0101 data segment.
75	15	Graphic 6	Same as the 0x0101 data segment.
90	15	Graphic 7	Same as the 0x0101 data segment.

```
typedef _packed struct
{
  interaction_figure_t interaction_figure[7];
}interaction_figure_4_t;
```


Table 2-27 Player client drawing characters 0x0301

Byte offset	Size	Descriptions	Notes
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. Data can be sent to only a player 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;
```

3. Small Map Interaction Data

3.1 Data Transmission by Player Clients

An Aerial Gimbal Operator can send fixed data to robots through a player client's small map.

The command code is 0x0303; transmitted when triggered, at an interval of at least 3 seconds.

Transmission method 1: ① Tap own robot's avatar; ② (Optional) Press a keyboard key or tap an opponent robot's avatar; ③ Tap any location on the small map. Through this method, map coordinate data is sent to robots selected by your own team. If the opponent's robot avatar is tapped, the target robot ID will replace the coordinate data.

Transmission method 2: ①(Optional) Press a keyboard key or tap an opponent's robot avatar; ②Tap any point on the small map. Through this method, map coordinate data is sent to all robots of your own team. If the opponent's robot avatar is tapped, the target robot ID will replace the coordinate data.

Table 3-1 Command code ID: 0x0303

Byte offset	Size	Descriptions	Notes
0	4	Target position's x-axis coordinate, unit is m	When the target robot ID is sent, the value is 0.

Byte offset	Size	Descriptions	Notes
4	4	Target position's y-axis coordinate, unit is m	When the target robot ID is sent, the value is 0.
8	4	Target position's z-axis coordinate, unit is m	Currently the value remains 0
12	1	The generic key value of the key pressed by the Aerial Gimbal Operator	0 if no key is pressed
13	2	Target robot ID	When sending coordinate data, the value is 0

```
typedef _packed struct
{
  float target_position_x;
  float target_position_y;
  float target_position_z;
  uint8_t commd_keyboard;
  uint16_t target_robot_id;
}map_command_t;
```

3.2 Receiving of Data by Player Clients

A player client's small map can receive robot data.

Through regular links, a radar can send an opponent robot's coordinate data to all of its own team's player clients. The position will be displayed on the small maps of all its own team's player clients.

Table 3-2 Command code ID: 0x0305

Byte offset	Size	Descriptions	Notes
0	2	Target robot ID	
2	4	Target's x-axis coordinate; unit: m	Not displayed when x and y exceed the boundaries.
6	4	Target's y-axis coordinate; unit: m	Not displayed when x and y exceed the boundaries.

```
typedef _packed struct
{
  uint16_t target_robot_id;
```



```
  float target_position_x;
  float target_position_y;
}map_robot_data_t;
```

Through regular links, a Sentry Robot can send route coordinate data to its own team's Aerial Robot. The route will be displayed on the small map.

Table 3-3 Command code ID: 0x0307

Byte offset	Size	Descriptions	Notes
0	1	1: Go to the target point to attack; 2: Go to the target point to defend; 3: Move to the target point	-
1	2	Route starting point's x-axis coordinate, unit: dm	The lower left corner of the small map is the origin. The horizontal right is the positive x-axis direction, and vertical upward is the positive y-axis direction. The displayed position will zoom in and out according to the site's and small map's dimensions. Positions exceeding the boundaries will be displayed on the boundaries.
3	2	Route starting point's y-axis coordinate; unit: dm	
5	49	Route starting point's x-axis increase number group; unit: dm	The increase is calculated relative to the previous point, and consists of 49 new points in total. The points are formed corresponding to the increase on the x and y-axis.
54	49	Route starting point's y-axis increase number group; unit: dm	The increase is calculated relative to the previous point, and consists of 49 new points in total. The points are formed corresponding to the increase on the x and y-axis.

```
typedef _packed struct
{
  uint8_t intention;
  uint16_t start_position_x;
  uint16_t start_position_y;
  int8_t delta_x[49];
  int8_t delta_y[49];
```



```
}map_sentry_data_t;
```

4. VTM Link Data Descriptions

4.1 Custom Controller and Robot Interaction Data Descriptions

An Operator may use a Custom Controller to send data to corresponding robots through VTM Links.

Table 4-1 Command code ID: 0x0302

Byte offset	Size	Descriptions
0	30	Custom data

```
typedef _packed struct
{
  uint8_t data[x];
}custom_robot_data_t;
```

4.2 Mouse Remote Control Data

Transmits mouse remote control data sent through a remote control, synchronously to the corresponding robots through VTM Links.

Table 4-2 Command code ID: 0x0304

Byte offset	Size	Descriptions
0	2	Mouse x-axis moving speed; a negative value indicates a left movement
2	2	Mouse y-axis moving speed; a negative value indicates a downward movement
4	2	Mouse scroll wheel's moving speed; a negative value indicates a backward movement
6	1	Whether the mouse's left button is pressed: 0 denotes it is not pressed; 1 denotes it is pressed.
7	1	Whether the mouse's right button is pressed: 0 denotes it is not pressed; 1 denotes it is pressed.
8	2	Keyboard key information. Each bit corresponds to a key. 0 indicates it is not pressed; 1 indicates it is pressed: ● bit 0: W key

Byte offset	Size	Descriptions
		• bit 1: S key • bit 2: Button A • bit 3: D key • bit 4: Shift key • bit 5: Ctrol key • bit 6: Q key • bit 7: E key • bit 8: R key • bit 9: F key • bit 10: G key • bit 11: Z key • bit 12: X key • bit 13: C key • bit 14: V key • bit 15: Button B
10	2	Characters maintained

```
typedef _packed 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;
}remote_control_t;
```

5. Non-Link Data Description

An Operator may use a Custom Controller to operate a player client by simulating a keyboard and mouse.

Table 5-1 Command code ID: 0x0306

Byte offset	Size	Descriptions	Notes
0	2	Keyboard key value: ● bits 0-7: Key 1 value ● bits 8-15: Key 2 value	● Only responds to keys made available by the player client ● Uses a generic key value and supports 2 non-conflict keys. Any change in key sequence will not alter the effect of the pressed keys. In the absence of any new key information, the pressed status of the last data frame will be maintained.
2	2	● bit 0-11: Mouse's x-axis pixel position ● bits 12-15: Mouse's left-click status	● Absolute pixel value is used for position information (the resolution used by player clients in the competition is 1920x1080, the coordinates for the screen's upper left corner is (0,0)) ● When the mouse's pressed status is 1, it means the button is pressed; any other value indicates the button is not pressed. This information is responded to only after the mouse icon appears. In the absence of any new mouse information, the player client maintains the mouse information of the last data frame. After the mouse icon disappears, the data is no longer retained.
4	2	● bit 0-11: Mouse y-axis pixel position ● bits 12-15: Mouse right-click status	
6	2	Reserved bits	-

To move and click the mouse once, you need to first send the data frame for the specified position when the mouse is not pressed, then send the data frame for when the mouse is pressed at the same position. Finally, send the data frame for when the mouse is pressed at the same position.

typedef _packed struct


```
{
   uint16_t key_value;
   uint16_t x_position:12;
   uint16_t mouse_left:4;
   uint16_t y_position:12;
   uint16_t mouse_right:4;
   uint16_t reserved;
}custom_client_data_t;
```


Appendix 1 CRC Verification Code Examples

```
//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);
```

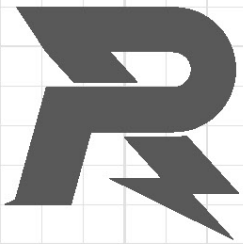

Appendix 2 ID Descriptions

The Robot IDs are shown below:

- 1: Red Team's Hero Robot
- 2: Red Team's Engineer Robot
- 3/4/5: Red Team's Standard Robot (corresponding to Robot ID3-5)
- 6: Red Team's Aerial Robot
- 7: Red Team's Sentry Robot
- 8: Red Team's Dart
- 9: Red Team's Radar
- 10: Red Team's Outpost
- 11: Red Team's Base
- 101: Blue Team's Hero Robot
- 102: Blue Team's Engineer Robot
- 103/104/105: Blue Team's Standard Robot (corresponding to Robot ID3-5)
- 106: Blue Team's Aerial Robot
- 107: Blue Team's Sentry Robot
- 108: Red Team's Radar
- 109: Blue Team's Radar
- 110: Blue Team's Outpost
- 111: Blue Team's Base

The player client IDs are shown below:

- 0x0101: Red Team's Hero Robot player client
- 0x0102: Red Team's Engineer Robot player client
- 0x0103/0x0104/0x0105: Red Team's Standard Robot player client (corresponding to Robot ID3-5)
- 0x0106: Red Team's Aerial Robot player client
- 0x0165: Blue Team's Hero Robot player client
- 0x0166: Blue Team's Engineer Robot player client
- 0x0167/0x0168/0x0169: Blue Team's Standard Robot player client (corresponding to Robot ID3-5)



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: T2, 22F, DJI Sky City, No. 55 Xianyuan Road, Nanshan District, Shenzhen, China