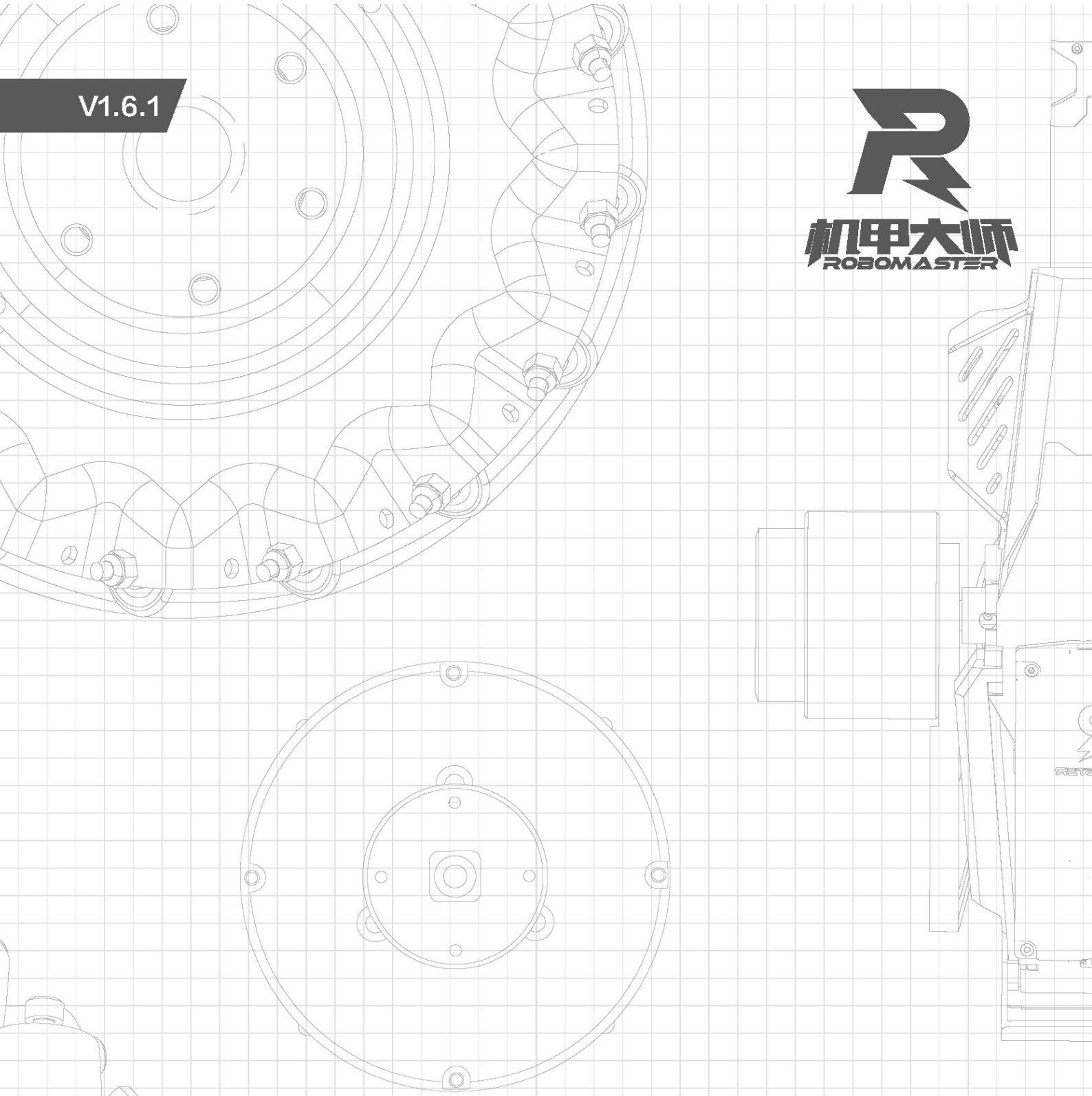
V1.6.1



Referee System Serial Port Protocol Appendix

Prepared by RoboMaster Organizing Committee

Updated in January, 2024

Change Log

Date	Version	Changes
January 26, 2024	V1.6.1	1. Revised command codes 0x0101, 0x0201, 0x0203, 0x0209, 0x020A, 0x0303. 2. Resolved known issues such as inconsistent data segment lengths and structure definition errors. 3. Optimized some descriptions.
November 22, 2023	V1.6	1. Revised the transmission frequencies for command codes 0x0001, 0x0101, 0x0105, 0x0203, 0x0205, 0x020A 2. Revised command codes 0x0101, 0x0102, 0x0104, 0x0105, 0x0201, 0x0203, 0x0209, 0x0301, 0x0303, 0x0307. 3. Added command codes 0x020D, 0x020E, 0x0308. 4. Optimized some descriptions.
July 7, 2023	V1.5	1. Added descriptions about the transmission/trigger conditions and sender/receiver of all command words. 2. Added descriptions about the regular link and VTM link of the Referee System. 3. Revised command codes 0x0101, 0x0204, 0x0205, 0x0208, and 0x0209. 4. Added command codes 0x020B, 0X020C, 0x0307, and 0x0306. 5. Deleted command codes 0x0004, 0x0005, and 0x0103. 6. Optimized some descriptions.
August 5, 2022	V1.4	Revised the transmission frequencies and trigger methods for command codes 0x0001, 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.

Date	Version	Changes
April 19, 2021	V1.1	Fixed some errors.
February 3, 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 of 115200, an 8-bit data bit, and a 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

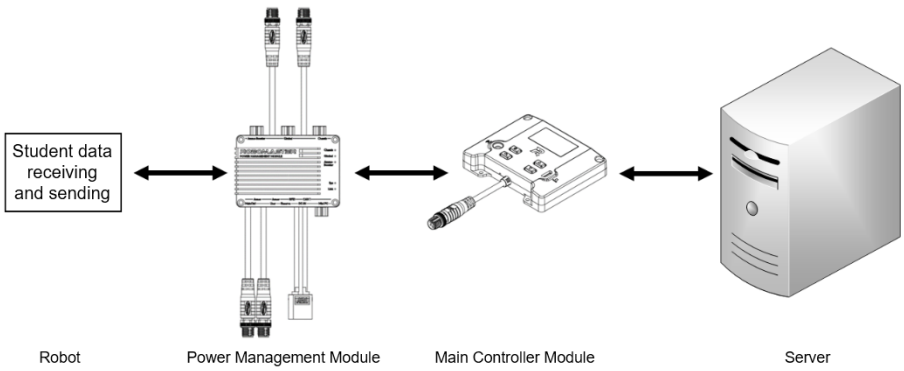
SOF	data_length	seq	CRC8
1-byte	2-byte	1-byte	1-byte

Table 1-3 Frame header definitions

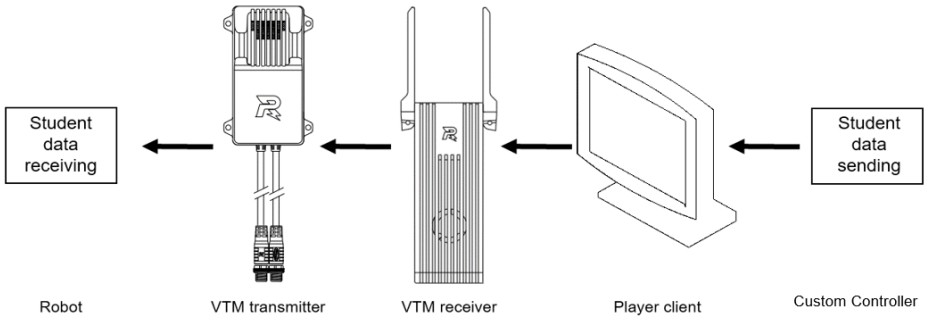
Domain	Offset Position	Size (Byte)	Description
SOF	0	1	Data frame start byte; the fixed value is 0×A5
data_length	1	2	Length of data in the data frame
seq	3	1	Sequence number
CRC8	4	1	Frame header CRC8

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:



- In the normal operating state, the delay of data transmission through the Referee System is approximately 130 ms, with a packet loss rate of less than 1%.
- In a Competition Area with poor network conditions, the delay of data transmission through the Referee System is approximately 200 ms, with a packet loss rate of approximately 3%.
- There may be errors in the measurement data, and the data is for your reference only.

2. Command Code IDs and Regular Link Data Description

Table 2-1 Overview of command code IDs

Command Code	Data Segment Length	Description	Sender/Receiver	Data Link
0x0001	11	Competition status data, transmitted at a fixed frequency of 1 Hz.	Server→All robots	Regular link
0x0002	1	Competition result data, transmitted upon completion of the competition.	Server→All robots	Regular link
0x0003	32	Robot health data, transmitted at a fixed frequency of 3 Hz.	Server→All robots	Regular link
0x0101	4	Site event data, transmitted at a fixed frequency of 1 Hz.	Server→All robots of the own side	Regular link
0x0102	4	Action identifier data of the Official Projectile Supplier, transmitted when the Official Projectile Supplier releases projectiles.	Server→All robots of the own side	Regular link
0x0104	3	Referee warning data, transmitted when one's team is issued a penalty/forfeiture and at a fixed frequency of 1 Hz in other cases.	Server→All robots of the penalized team	Regular link
0x0105	3	Dart launching data, transmitted at a fixed frequency of 1 Hz.	Server→All robots of the own side	Regular link
0x0201	13	Robot performance system data, transmitted at a fixed frequency of 10 Hz.	Main Controller Module→Corresponding robot	Regular link

Command Code	Data Segment Length	Description	Sender/Receiver	Data Link
0x0202	16	Real-time chassis power and barrel heat data, transmitted at a fixed frequency of 50 Hz.	Main Controller Module→Corresponding robot	Regular link
0x0203	16	Robot position data, transmitted at a fixed frequency of 1 Hz.	Main Controller Module→Corresponding robot	Regular link
0x0204	6	Robot buff data, transmitted at a fixed frequency of 3 Hz.	Server→Corresponding robot	Regular link
0x0205	2	Air support time data, transmitted at a fixed frequency of 1 Hz.	Server→Own side's Aerial Robots	Regular link
0x0206	1	Damage status data, transmitted after the damage occurs.	Main Controller Module→Corresponding robot	Regular link
0x0207	7	Real-time launching data, transmitted after a projectile is launched.	Main Controller Module→Corresponding robot	Regular link
0x0208	6	Projectile allowance, transmitted at a fixed frequency of 10 Hz.	Server→Own side's Hero, Standard, Sentry, and Aerial Robots	Regular link
0x0209	4	Robot RFID module status, transmitted at a fixed frequency of 3 Hz.	Server→Own side's robots with an RFID module	Regular link
0x020A	6	Dart player's client command data, transmitted at a fixed frequency of 3 Hz.	Server→Own side's Dart Robots	Regular link

Command Code	Data Segment Length	Description	Sender/Receiver	Data Link
0x020B	40	Ground Robot position data, transmitted at a fixed frequency of 1 Hz.	Server→Own side's Sentry Robots	Regular link
0x020C	6	Radar-marked progress data, transmitted at a fixed frequency of 1 Hz.	Server→Own side's Radar Robots	Regular link
0x020D	4	Decision-making data of Sentry Robot, transmitted at a fixed frequency of 1 Hz.	Server→Own side's Sentry Robots	Regular link
0x020E	1	Decision-making data of Radar, transmitted at a fixed frequency of 1 Hz.	Server→Own side's Radar Robots	Regular link
0x0301	128	Robot interaction data, transmitted at a maximum frequency of 10 Hz when triggered by the sender.	-	Regular link
0x0302	30	Data about the interaction between the Custom Controller and robots, transmitted at a maximum frequency of 30 Hz when triggered by the sender.	Custom Controller→Robots with VTM links to player clients	VTM link
0x0303	15	Player client's small map interaction data, transmitted when triggered by the player client.	Tap the player client→Server→Own side's robots selected by the sender	Regular link
0x0304	12	Keyboard, mouse, and remote control data, transmitted at a fixed frequency of 30 Hz.	Player client→Robots with VTM links to player clients	VTM link

Command Code	Data Segment Length	Description	Sender/Receiver	Data Link
0x0305	10	Radar data received by player clients' Small Maps, transmitted at a maximum frequency of 10 Hz.	Radar→Server→All player clients of the own side	Regular link
0x0306	8	Data about the interaction between the Custom Controller and player clients, transmitted at a maximum frequency of 30 Hz when triggered by the sender.	Custom Controller→Player client	-
0x0307	103	Sentry data received by player clients' Small Maps, transmitted at a maximum frequency of 1 Hz.	Sentry/Semiautomatic Control Robot→Player client of the corresponding operator	Regular link
0x0308	34	Robot data received by player clients' Small Map, transmitted at a maximum frequency of 3 Hz.	Own side's robots→Own side's player clients	Regular link

Table 2-2 0x0001

Byte Offset	Size	Description
0	1	Bits 0-3: competition type • 1: RoboMaster University Championship (RMUC) • 2: RoboMaster University Technical Challenge (RMUT) • 3: ICRA RoboMaster University AI Challenge (RMUA) • 4: RoboMaster University League (RMUL) 3V3 Match • 5: RoboMaster University League (RMUL) Standard Match Bits 4-7: current stage • 0: pre-competition stage • 1: preparation stage

Byte Offset	Size	Description
		• 2: 15-second Referee System initialization period • 3: 5-second countdown • 4: in competition • 5: competition result calculation
1	2	Remaining time in the current round; unit: second.
3	8	UNIX time, effective after the robot is correctly connected 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	Description
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	Description
0	2	HP of Red 1 Hero Robot. If the robot has not entered the stage or has been ejected, the HP is zero.
2	2	HP of Red 2 Engineer Robot.
4	2	HP of Red 3 Standard Robot.
6	2	HP of Red 4 Standard Robot.

Byte Offset	Size	Description
8	2	HP of Red 5 Standard Robot.
10	2	HP of Red 7 Sentry Robot.
12	2	HP of Red Outpost.
14	2	HP of Red Base.
16	2	HP of Blue 1 Hero Robot.
18	2	HP of Blue 2 Engineer Robot.
20	2	HP of Blue 3 Standard Robot.
22	2	HP of Blue 4 Standard Robot.
24	2	HP of Blue 5 Standard Robot.
26	2	HP of Blue 7 Sentry Robot.
28	2	HP of Blue Outpost.
30	2	HP of Blue Base.

```
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_t 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	Description
0	4	0: non-occupied/non-activated 1: occupied/activated Bits 0-2: ● Bit 0: occupation status of the Restoration Zone in front of the own side's Official Projectile Supplier; a value of 1 indicates that it is occupied. ● Bit 1: occupation status of the Restoration Zone inside the own side's Official Projectile Supplier; a value of 1 indicates that it is occupied. ● Bit 2: occupation status of the own side's Supplier Zone; a value of 1 indicates that it is occupied, which applies only to RMUL. Bits 3-5: status of the own side's Power Rune ● Bit 3: occupation status of the own side's Power Rune Activation Point; a value of 1 indicates that it is occupied. ● Bit 4: activation status of the own side's Small Power Rune; a value of 1 indicates that it is activated. ● Bit 5: activation status of the own side's Large Power Rune; a value of 1 indicates that it is activated. Bits 6-11: occupation status of the own side's Elevated Ground ● Bits 6-7: occupation status of the own side's Ring-Shaped Elevated Ground; a value of 1 indicates that it is occupied by the own side; a value of 2 indicates that it is occupied by the opponent. ● Bits 8-9: occupation status of the own side's Trapezoid-Shaped Elevated Ground; a value of 1 indicates that it is occupied by the own side; a value of 2 indicates that it is occupied by the opponent. ● Bits 10-11: occupation status of the own side's Trapezoid-Shaped Elevated Ground; a value of 1 indicates that it is occupied by the own side; a value of 2 indicates that it is occupied by the opponent. Bits 12-18: percentage of the remaining value of the own Base's Virtual Shield (rounded to the nearest integer)

Byte Offset	Size	Description
		Bits 19-27: time when the own side's Outpost or Base was last hit by darts (valid values: 0 to 420; default value at the start of a round: 0) Bits 28-29: specific target that was hit when the own side's Outpost or Base was last hit by darts. The default value at the start of a round is 0, a value of 1 indicates that the dart hits the Outpost, a value of 2 indicates that the dart hits a fixed target in the Base, and a value of 3 indicates that the dart hits a random target in the Base. Bits 30-31: occupation status of the Central Buff Point; a value of 0 indicates that it is not occupied; a value of 1 indicates that it is occupied by the own side; a value of 2 indicates that it is occupied by the opponent; a value of 3 indicates that it is occupied by both sides, which applies only to RMUL

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

Table 2-6 0x0102

Byte Offset	Size	Description
0	1	Reserved bit
1	1	Reloading robot ID. ● 0: No robot is reloading projectiles. ● 1: The Hero Robot of the Red Team is reloading projectiles. ● 3/4/5: The Standard Robot of the Red Team is reloading projectiles. ● 101: The Hero Robot of the Blue Team is reloading projectiles. ● 103/104/105: The Standard Robot of the Blue Team is reloading projectiles.
2	1	Status of the projectile outlet. ● 0: closed ● 1: preparing projectiles ● 2: releasing projectiles
3	1	Number of supplied projectiles.

Byte Offset	Size	Description
		● 50: 50 projectiles ● 100: 100 projectiles ● 150: 150 projectiles ● 200: 200 projectiles

```
typedef _packed struct
{
  uint8_t reserved;
  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	Description
0	1	Level of penalty that was last received by the own side. ● 1: Both teams received a Yellow Card. ● 2: Yellow Card ● 3: Red Card ● 4: Forfeiture
1	1	● ID of the own side's offending robot that received the last penalty. (For example, Red 1 Robot's ID is 1, and 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.
2	1	Number of violations (at the corresponding penalty level) triggered by the own side's offending robot that received the last penalty. (The default value at the start of a round is 0.)

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


Table 2-8 0x0105

Byte Offset	Size	Description
0	1	Own side's remaining time for dart launching; unit: second.
1	2	Bits 0-1: Target that was last hit by a dart of the own side. The default value at the start of a round is 0, a value of 1 indicates that the dart hits the Outpost, a value of 2 indicates that the dart hits a fixed target in the Base, and a value of 3 indicates that the dart hits a random target in the Base. Bits 2-4: Total number of recent hits to a target in the opponent team. The default value at the start of a round is 0, and the maximum value is 4. Bits 5-6: Target currently selected to be hit by the dart. The default value at the start of a round is 0. When no hit target is selected or the Outpost is selected, the value is also 0. A value of 1 indicates that a fixed target in the Base is selected, and a value of 2 indicates that a random target in the Base is selected. Bits 7-15: Reserved bits

```
typedef _packed struct
{
   uint8_t dart_remaining_time;
   uint16_t dart_info;
}dart_info_t;
```

Table 2-9 0x0201

Byte Offset	Size	Description
0	1	Current robot ID.
1	1	Robot level.
2	2	Robot's current HP.
4	2	Robot's maximum HP.
6	2	Robot's barrel cooling value per second.

Byte Offset	Size	Description
8	2	Robot's barrel heat limit.
10	2	Robot's chassis power consumption limit.
12	1	Output status of the Power Management Module. ● Bit 0: output from the gimbal port. A value of 0 indicates zero output, and a value of 1 indicates 24 V output. ● Bit 1: output from the chassis port. A value of 0 indicates zero output, and a value of 1 indicates 24 V output. ● Bit 2: output from the shooter port. A value of 0 indicates zero output, and a value of 1 indicates 24 V output. ● Bits 3-7: Reserved bits

```
typedef _packed struct
{
  uint8_t robot_id;
  uint8_t robot_level;
  uint16_t current_HP;
  uint16_t maximum_HP;
  uint16_t shooter_barrel_cooling_value;
  uint16_t shooter_barrel_heat_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	Description
0	2	Output voltage of the chassis port in the Power Management Module; unit: mV.
2	2	Output current of the chassis port in the Power Management Module; 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 17mm 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	Description
0	4	The x-coordinate of the robot's position; unit: m.
4	4	The y-coordinate of the robot's position; unit: m.
8	4	Direction of the robot's Speed Monitor Module; unit: degree. True north is 0 degrees.

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

Table 2-12 0x0204

Byte Offset	Size	Description
0	1	Robot's HP recovery buff (in percentage; a value of 10 indicates that HP recovery per second is 10% of the maximum HP.)
1	1	Robot's barrel cooling rate (in absolute value; a value of 5 indicates a cooling rate of 5 times.)
2	1	Robot's defense buff (in percentage; a value of 50 indicates a defense buff of 50%.)
3	1	Robot's negative defense buff (in percentage; a value of 30 indicates a defense buff of -30%.)
4	2	Robot's attack buff (in percentage; a value of 50 indicates an attack buff of 50%.)


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

Table 2-13 0x0205

Byte Offset	Size	Description
0	1	Aerial Robot's status (a value of 0 indicates that it is cooling, a value of 1 indicates that cooling is completed, and a value of 2 indicates that air support is ongoing.)
1	1	Remaining time of the current status (unit: s; the value is rounded down to the nearest integer. For example, if the remaining cooling time is 1.9 s, the value is rounded down to 1.) If the cooling time is 0 but no air support is called for, 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	Description
0	1	Bits 0-3: If HP deduction is caused because an Armor Module is attacked by projectiles, hit in a collision, or goes offline, or a Speed Monitor Module goes offline, the 4-bit value indicates the ID of the Armor Module or Speed Monitor Module. If HP deduction is caused by other reasons, the value is 0. Bits 4-7: type of an HP change. ● 0: HP deduction is caused because an Armor Module is attacked by projectiles. ● 1: HP deduction is caused because the Critical Referee System Module goes offline. ● 2: HP deduction is caused because the Initial Launching Speed exceeds the upper limit.

Byte Offset	Size	Description
		● 3: HP deduction is caused because the Barrel Heat exceeds the upper limit. ● 4: HP deduction is caused because the Chassis Power Consumption Limit is exceeded. ● 5: HP deduction is caused because the Armor Module suffers 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	Description
0	1	Projectile type. ● 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
2	1	Projectile launching frequency; unit: Hz.
3	4	Initial projectile 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	Description
0	2	17mm projectile allowance.

Byte Offset	Size	Description
2	2	42mm projectile allowance.
4	2	Number of remaining Gold 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	Description
0	4	Meaning of bit value 0 or 1: whether the Buff Point's RFID card is detected. ● Bit 0: own side's Base Buff Point ● Bit 1: own side's Ring-Shaped Elevated Ground Buff Point ● Bit 2: opponent's Ring-Shaped Elevated Ground Buff Point ● Bit 3: own side's R3/B3 Trapezoid-Shaped Elevated Ground Buff Point ● Bit 4: opponent's R3/B3 Trapezoid-Shaped Elevated Ground Buff Point ● Bit 5: own side's R4/B4 Trapezoid-Shaped Elevated Ground Buff Point ● Bit 6: opponent's R4/B4 Trapezoid-Shaped Elevated Ground Buff Point ● Bit 7: own side's Power Rune Activation Point ● Bit 8: own side's Launch Ramp Buff Point (in front of the Launch Ramp near own side) ● Bit 9: own side's Launch Ramp Buff Point (behind the Launch Ramp near own side) ● Bit 10: opponent's Launch Ramp Buff Point (in front of the Launch Ramp near the other side) ● Bit 11: opponent's Launch Ramp Buff Point (behind the Launch Ramp near the other side) ● Bit 12: own side's Outpost Buff Point ● Bit 13: own side's Restoration Zone (deemed activated if anyone is detected)

Byte Offset	Size	Description
		● Bit 14: own side's Sentry Patrol Zones ● Bit 15: opponent's Sentry Patrol Zones ● Bit 16: own side's Large Resource Island Buff Point ● Bit 17: opponent's Large Resource Island Buff Point ● Bit 18: own side's Exchange Zone ● Bit 19: Central Buff Point (for RMUL only) ● Bits 20-31: Reserved bits Note: The RFID card of the Base Buff Point, Elevated Ground Buff Point, Launch Ramp Buff Point, Outpost Buff Point, Resource Island Buff Point, Restoration Zone, Exchange Zones, Central Buff Point (for RMUL only), and Sentry Patrol Zones are effective only during the competition. If an RFID card 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	Description
0	1	Current status of the Dart Launching Station. ● 1: closed ● 2: opening or closing ● 0: opened
1	1	Reserved bits
2	2	Remaining competition time when the attack target is changed. Unit: s. If no target change occurs, the value is 0 by default.
4	2	Remaining competition time when the Operator confirms the launch command for the last time. Unit: s. Initial value: 0.

```
typedef _packed struct
{
```



```
    uint8_t dart_launch_opening_status;
    uint8_t reserved;
    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	Description
0	4	The x-axis coordinate of the own side's Hero Robot; unit: m.
4	4	The y-axis coordinate of the own side's Hero Robot; unit: m.
8	4	The x-axis coordinate of the own side's Engineer Robot; unit: m.
12	4	The y-axis coordinate of the own side's Engineer Robot; unit: m.
16	4	The x-axis coordinate of the own side's Standard Robot No. 3; unit: m.
20	4	The y-axis coordinate of the own side's Standard Robot No. 3; unit: m.
24	4	The x-axis coordinate of the own side's Standard Robot No. 4; unit: m.
28	4	The y-axis coordinate of the own side's Standard Robot No. 4; unit: m.
32	4	The x-axis coordinate of the own side's Standard Robot No. 5; unit: m.
36	4	The y-axis coordinate of the own side's 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	Description
0	1	Marked progress of the opponent's Hero Robot. 0-120
1	1	Marked progress of the opponent's Engineer Robot. 0-120
2	1	Marked progress of the opponent's Standard Robot No. 3. 0-120
3	1	Marked progress of the opponent's Standard Robot No. 4. 0-120
4	1	Marked progress of the opponent's Standard Robot No. 5. 0-120
5	1	Marked progress of the 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;
```

Table 2-21 0x020D

Byte Offset	Size	Description
0	4	Bits 0-10: the projectile allowance successfully exchanged by a Sentry Robot, excluding the projectile exchanged through remote exchange. The value at the start of a round is 0. After the Sentry Robot successfully exchanges for a specific amount of projectile allowance, the value changes to the exchanged projectile allowance. Bits 11-14: the number of a Sentry Robot's successful remote exchanges for projectile allowance. The value at the start of a round is 0. After the Sentry Robot successfully exchanges for projectile allowance remotely, the value changes to the number of successful remote exchanges for projectile allowance. Bits 15-18: the number of a Sentry Robot's successful remote HP exchanges. The value at the start of a round is 0. After the Sentry Robot successfully exchanges for HP remotely, the value changes to the number of successful remote HP exchanges.

Byte Offset	Size	Description
		Bits 19-31: Reserved bits

```
typedef _packed struct
{
   uint32_t sentry_info;
} sentry_info_t;
```

Table 2-22 0x020E

Byte Offset	Size	Description
0	1	Bits 0-1: the number of chances for the Radar to trigger the "double vulnerability" effect. The value at the start of a round is 0. Maximum value: 2. Bit 2: whether the "double vulnerability" effect is being triggered for the opponent. ● A value of 0 indicates that the "double vulnerability" effect is not triggered for the opponent. ● A value of 1 indicates that the "double vulnerability" effect is being triggered for the opponent. Bits 3-7: Reserved bits

```
typedef _packed struct
{
   uint8_t radar_info;
} radar_info_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 content ID, sender ID, receiver ID, and content data segment. The total length of a robot interaction data packet cannot exceed 128 bytes. After the nine bytes for frame_header, cmd_id, and frame_tail, and the six bytes for the data segment header structure are deducted, the maximum length of a content data segment in a robot interaction data packet is 113 bytes.

The maximum length of data that can be received by a Hero, Engineer, Standard, or Aerial Robot or Dart every 1000 ms is 3720 bytes, while the maximum length of data that can be received by a Radar or a Sentry Robot every 1000 ms is 5120 bytes.

Table 2-23 0x0301

Byte Offset	Size	Description	Remarks
0	2	Sub-content ID	It needs to be an open sub-content ID.

Byte Offset	Size	Description	Remarks
2	2	Sender ID	It needs to be matched with one's own ID. For more information about the IDs, see the appendix.
4	2	Receiver ID.	● It is used for communication only within one's own team. ● The receiver must be an inter-robot communication receiver permitted by the rules. ● If the receiver is a player client, the data can be transmitted only to the player client corresponding to the sender. ● For more information about the IDs, 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	A player client deletes graphic layers.
0x0101	15	A player client draws one graphic.
0x0102	30	A player client draws two graphics.
0x0103	75	A player client draws five graphics.
0x0104	105	A player client draws seven graphics.

Sub-content ID	Content Data Segment Length	Function Description
0x0110	45	A player client draws a character graphic.
0x0120	4	Decision-making command of a Sentry Robot.
0x0121	1	Decision-making command of a Radar.

Make sure that the bandwidth is properly configured because there are multiple content IDs but the maximum uplink frequency of the entire cmd_id is 10 Hz.

Table 2-24 Sub-content ID: 0x0100

Byte Offset	Size	Description	Remarks
0	1	Delete operation.	● 0: No operation is performed. ● 1: Delete a graphic layer. ● 2: Delete all graphic layers.
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 2-25 Sub-content ID: 0x0101

Byte Offset	Size	Description	Remarks
0	3	Graphic name	Used as an index in graphic deletion, revision, and other operations.
3	4	Graphic configuration 1	Bits 0-2: graphic operation ● 0: No operation is performed. ● 1. Add ● 2: Modify

Byte Offset	Size	Description	Remarks
			● 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 (valid values: 0 to 9) Bits 10-13: color ● 0: Red/Blue (Own side'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 based on the drawn graphics, as described in Table 2-26 Graphic detail parameters.
7	4	Graphic configuration 2	Bits 0-9: line width. The recommended ratio between font size and line width is 10:1. Bits 10-20: start point/origin's x-coordinate.

Byte Offset	Size	Description	Remarks
			Bits 21-31: start point/origin's y-coordinate.
11	4	Graphic configuration 3	The meaning differs based on the drawn graphics, as described in Table 2-26 Graphic detail parameters.

```
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;
```

Table 2-26 Graphic detail parameters

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

Type	details_a	details_b	details_c	details_d	details_e
Arc	Start angle	End angle	-	Length of the x axis	Length of the 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	-	-	-

- Meaning of angle value: 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 at the upper right corner.

- Floating number: All integers are 32-bit. For a floating number, the actual displayed value is 1/1000 of the entered value. For example, if the value entered corresponding to details_c, details_d, and details_e is 1234, the actual value displayed on the player's client will be 1.234.
- The graphics may still appear even if the transmitted value exceeds the upper limit of the corresponding data type, but the display quality may be compromised.

Table 2-27 Sub-content ID: 0x0102

Byte Offset	Size	Description	Remarks
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 2-28 Sub-content ID: 0x0103

Byte Offset	Size	Description	Remarks
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 2-29 Sub-content ID: 0x0104

Byte Offset	Size	Description	Remarks
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 2-30 Sub-content ID: 0x0110

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. Data can be sent only to a player client corresponding to the robot.
6	15	Character configuration	For more information, see the graphic data introduction.
21	30	Character	-

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

Table 2-31 Decision-making command of a Sentry: 0x0120

Byte Offset	Size	Description	Remarks
0	4	Relevant decision-making commands of a Sentry Robot.	Bit 0: whether the Sentry Robot confirms to revive. ● A value of 0 indicates that the Sentry Robot confirms not to revive, even though the Respawn Process of the Sentry Robot is completed. ● A value of 1 indicates that the Sentry Robot confirms to revive. The Sentry Robot revives immediately after the Respawn Process is completed. Bit 1: whether the Sentry Robot confirms to exchange for Instant Respawn. ● A value of 0 indicates that the Sentry Robot confirms not to exchange for Instant Respawn.

Byte Offset	Size	Description	Remarks
			● A value of 1 indicates that the Sentry Robot confirms to exchange for Instant Respawn. In this case, if the Sentry Robot meets the requirements for Instant Respawn, Gold Coins are immediately consumed for Instant Respawn. Bits 2-12: the projectile allowance to be obtained by a Sentry Robot through exchange. The value at the start of a round is 0. After the value is changed, the Sentry Robot can exchange Gold Coins for projectile allowance in the Restoration Zone. The value must be monotonically increased. Otherwise, the value is considered invalid. **Example: The value must be set to 0 at the start of a round. Then, the Sentry Robot can change the value from 0 to X, and a number of X Gold Coins are consumed for exchanging X projectile allowance. Subsequently, the Sentry Robot can change the value from X to X+Y. The rest can be deduced by analogy.** Bits 13-16: the number of requests sent by a Sentry Robot to obtain projectile allowance through remote exchange. A value at the start of a round is 0. After the value is changed, the Sentry Robot can request to remotely exchange Gold Coins for projectile allowance. The value must be monotonically increased by 1 at a time. Otherwise, the value is considered invalid. **Example: The value must be set to 0 at the start of a round. Then, the Sentry Robot can change the value from 0 to 1, and Gold Coins are consumed for remotely exchanging for projectile allowance. Subsequently, the Sentry Robot can change the value from 1 to 2. The rest can be deduced by analogy.** Bits 17-20: the number of requests sent by a Sentry Robot to obtain HP through remote exchange. A value at the start of a round is 0. After the value is changed, the Sentry Robot can request to remotely exchange Gold Coins for HP. The value must be monotonically increased by 1 at a time. Otherwise, the value is considered invalid. **Example: The value must be set to 0 at the start of a round. Then, the Sentry Robot can change the value from 0 to 1, and Gold Coins are consumed for remotely exchanging for HP. Subsequently, the Sentry Robot can change the value from 1 to 2. The rest can be deduced by analogy.**

Byte Offset	Size	Description	Remarks
			When the Sentry Robot sends these sub-commands, the server processes the sub-commands in sequence from lower bits to higher bits until all sub-commands are processed or no processing can be performed. **Example: The number of Gold Coins in the team is 0, and the Sentry Robot is defeated at this time. The value of bit 0 is 1, the value of bit 1 is 1, and the value of bits 2 to 12 is 100 (assuming that the Sentry Robot has not exchanged for projectile allowance before). In this case, the number of Gold Coins in the team is not enough for the Sentry Robot to exchange for Instant Respawn. Therefore, the server ignores subsequent commands and waits for the next group of commands from the Sentry Robot.** Bits 21-31: Reserved bits

```
typedef _packed struct
{
  uint32_t sentry_cmd;
} sentry_cmd_t;
```

Table 2-32 Decision-making command of a Radar: 0x0121

Byte Offset	Size	Description	Remarks
0	1	Whether the Radar confirms to trigger the "double vulnerability" effect.	The value at the start of a round is 0. The value can be changed for the Radar to request triggering the "double vulnerability" effect. If the Radar has a chance to trigger the "double vulnerability" effect, the effect is triggered. The value must be monotonically increased by 1 at a time. Otherwise, the value is considered invalid. **Example: The value at the start of a round is 0. Then, the Radar can change the value from 0 to 1. If the Radar has a chance to trigger the "double vulnerability" effect, the effect is triggered. Subsequently, the Radar can change the value from 1 to 2. The rest can be deduced by analogy.** If the "double vulnerability" effect is in effect when the Radar requests to trigger the effect, the effect requested for the second time is triggered after the effect requested for the first time elapses.

```
typedef _packed struct
{
  uint8_t radar_cmd;
} radar_cmd_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 big map.

The command code is 0x0303, which is transmitted when triggered, at an interval of at least 0.5 seconds.

Transmission method 1:

① Tap the avatar of an own side's robot.

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 an opponent robot's 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 robot's avatar.

② Tap any location on the Small Map. Through this method, map coordinate data is sent to all robots of your own team. If an opponent robot's avatar is tapped, the target robot ID will replace the coordinate data.

- An operator of a robot that uses the semiautomatic control method can send fixed data to robots through a player client's big map.

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

Transmission method:

① (Optional) Press a keyboard key or tap an opponent robot's avatar.

② Tap any location on the Small Map. Through this method, map coordinate data is sent to a robot corresponding to the operator. If an opponent robot's avatar is tapped, the target robot ID will replace the coordinate data.

The robot that chooses to use the semiautomatic control method can receive information sent by the Aerial Gimbal Operator and the information sent by a corresponding operator. For the differences between the two information sources, see the description about information sources in the table below.

Table 3-1 Command code ID: 0x0303

Byte Offset	Size	Description	Remarks
0	4	The x-axis coordinate of the target position; unit: m.	When the target robot ID is sent, the value is 0.

Byte Offset	Size	Description	Remarks
4	4	The y-axis coordinate of the target position; unit: m.	When the target robot ID is sent, the value is 0.
8	1	The generic key value of the key pressed by the Aerial Gimbal Operator.	When no key is pressed, the value is 0.
9	1	The opponent's robot ID.	When coordinate data is sent, the value is 0
10	2	The information source ID.	For more information about the ID correspondence, see the appendix.

```
typedef _packed struct
{
  float target_position_x;
  float target_position_y;
  uint8_t cmd_keyboard;
  uint8_t target_robot_id;
  uint8_t cmd_source;
}map_command_t;
```

3.2 Data Receiving 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 side's player clients. The position will be displayed on the Small Maps of all player clients of its own side.

Table 3-2 Command code ID: 0x0305

Byte Offset	Size	Description	Remarks
0	2	The target robot's ID.	-
2	4	Target's x-axis coordinate; unit: m.	Not displayed when x- and y-coordinates exceed the boundaries.
6	4	Target's y-axis coordinate; unit: m.	Not displayed when x- and y-coordinates 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 or a robot that uses semiautomatic control can send route coordinate data to the player client of the corresponding operator. The route will be displayed on the Small Map.

Table 3-3 Command code ID: 0x0307

Byte Offset	Size	Description	Remarks
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	The x-axis coordinate at the route starting point, unit: dm.	The lower left corner of the Small Map is the origin. The horizontal right is the positive x-axis direction, and the 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	The y-axis coordinate at the route starting point; unit: m.	
5	49	The array of x-axis incremental values of route points; unit: dm.	The incremental values are calculated relative to the previous point, resulting in 49 new points in total. The positions of these new points are determined based on the x- and y-axis incremental values.
54	49	The array of y-axis incremental values of route points; unit: dm.	
103	2	Sender ID	It needs to be matched with one's own ID. For more information about the IDs, see the appendix.

```
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];
  uint16_t sender_id;
}map_data_t;
```


A robot can send custom messages to any player client of its own side through a regular link. The message is displayed at a specific position on the player client.

Table 3-4 Command code ID: 0x0308

Byte Offset	Size	Description	Remarks
0	2	Sender ID	The sender ID needs to be verified.
2	2	Receiver ID	The receiver ID needs to be verified. Data can be sent only to a player client of the own side.
4	30	Character	Characters are sent in the UTF-16 encoding format and can be displayed in Chinese. Pay attention to the endianness of data when the data is encoded for transmission.

```
typedef _packed struct
{
uint16_t sender_id;
uint16_t receiver_id;
uint8_t user_data[30];
} custom_info_t;
```


4. VTM Link Data Descriptions

4.1 Custom Controller and Robot Interaction Data Descriptions

An operator can use a Custom Controller to send data to corresponding robots through VTM links.

Table 4-1 Command code ID: 0x0302

Byte Offset	Size	Description
0	30	Custom data

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

4.2 Keyboard-Mouse Remote Control Data

The keyboard and mouse remote control data sent via the remote controller is synchronized to the corresponding robot through VTM links.

Table 4-2 Command code ID: 0x0304

Byte Offset	Size	Description
0	2	x-axis moving speed of the mouse. A negative value indicates a left movement.
2	2	y-axis moving speed of the mouse. A negative value indicates a downward movement.
4	2	Scroll wheel's moving speed of the mouse. A negative value indicates a backward movement.
6	1	Whether the mouse's left button is pressed. A value of 0 indicates that it is not pressed, and a value of 1 indicates that it is pressed.
7	1	Whether the mouse's right button is pressed. A value of 0 indicates that it is not pressed, and a value of 1 indicates that it is pressed.
8	2	The keyboard key information. Each bit corresponds to a key. A value of 0 indicates that it is not pressed, and a value of 1 indicates that it is pressed: ● Bit 0: W key

Byte Offset	Size	Description
		• Bit 1: S key • Bit 2: A key • Bit 3: D key • Bit 4: Shift key • Bit 5: Ctrl 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: B key
10	2	Reserved bits

```
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 can use a Custom Controller to operate a player client by simulating a keyboard and a mouse.

Table 5-1 Command code ID: 0x0306

Byte Offset	Size	Description	Remarks
0	2	Keyboard key value. ● Bits 0-7: value of Key 1 ● Bits 8-15: value of Key 2	● Only the keys made available by the player client are responded to. ● A generic key value is used, and two keys can be simultaneously pressed with no conflict. Any change in key sequence does 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	● Bits 0-11: mouse's x-axis pixel position ● Bits 12-15: mouse's left-click status	● The position information is represented by absolute pixel values (the resolution used by player clients in the competition is 1920 x 1080, and 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	● Bits 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 obtained when the mouse is pressed at the same position, and finally send the data frame obtained 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 Code Samples

```
//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

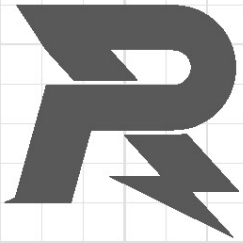
The robot IDs are as follows:

- 1: Red Team's Hero Robot
- 2: Red Team's Engineer Robot
- 3/4/5: Red Team's Standard Robot (corresponding to robot IDs 3-5)
- 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 IDs 3-5)
- 106: Blue Team's Aerial Robot
- 107: Blue Team's Sentry Robot
- 108: Red Team's Dart
- 109: Blue Team's Radar
- 110: Blue Team's Outpost
- 111: Blue Team's Base

The player client IDs are as follows:

- 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 IDs 3-5)
- 0x0106: Red Team's Aerial Robot player client
- 0x016A: Blue 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 IDs 3-5)
- 0x8080: Referee System's server (for sending decision-making commands to Sentry Robots and Radar)



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