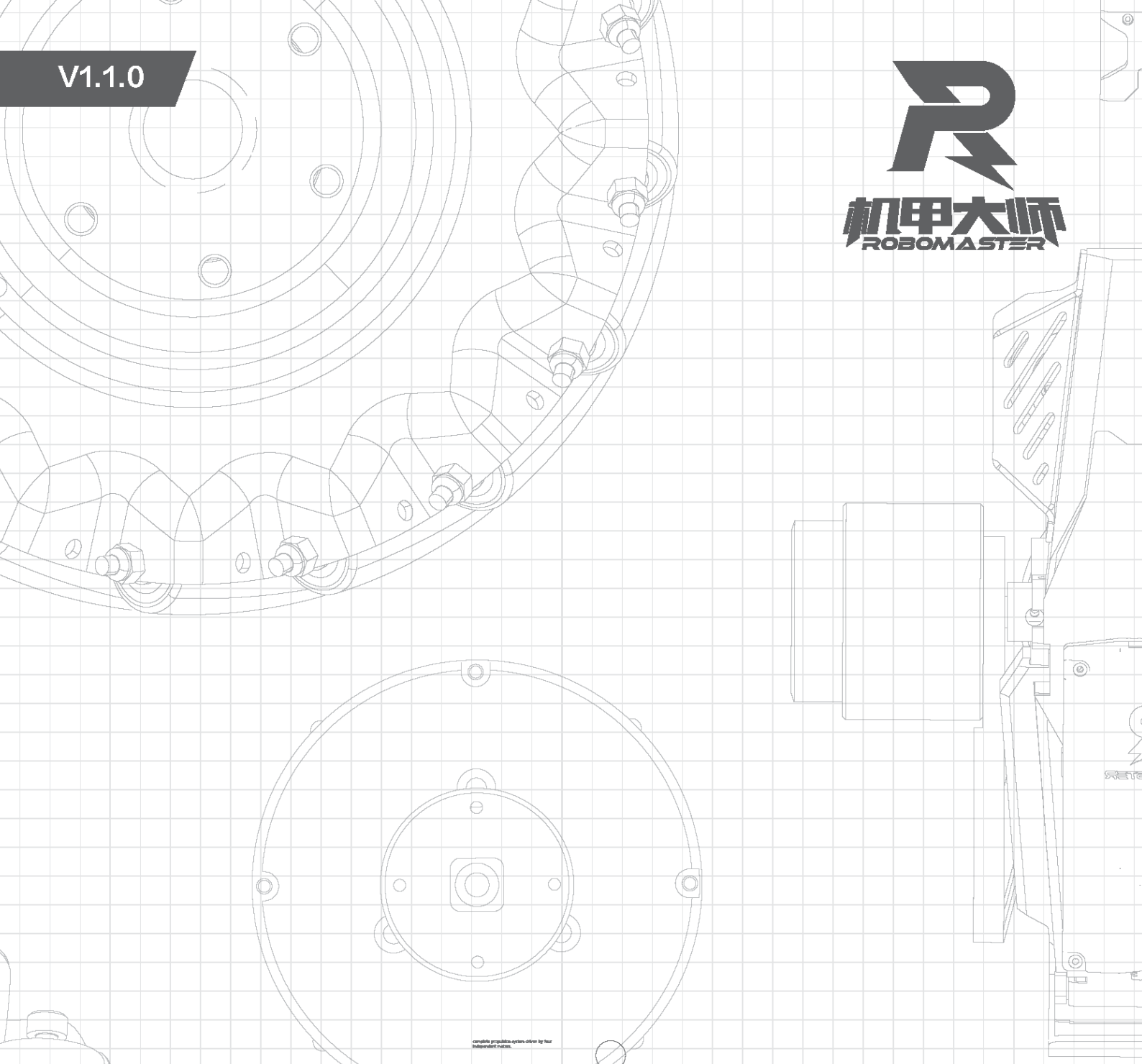
V1.1.0



第二十五届全国大学生机器人大赛
ROBOMASTER 2026
机甲大师高校系列赛
通信协议

RoboMaster 组委会 编制

2025 年 12 月 发布

修改日志

日期	版本	修订记录
2025.12.18	V1.1.0	补充雷达无线链路相关说明
2025.11.27	V1.0.0	首次发布

前言

本通信协议在 RMUC 与 RMUL 两项赛事中的适用范围如下:

- 对于在 RMUC 与 RMUL 之间共用的兵种、机制或场地道具等条目, 本协议内容对两项赛事均适用;
- 对于仅适用于某一赛项的兵种、机制或场地道具等条目，默认不适用于另一赛项。

目录

1. 串口协议

1.1 串口协议格式

通信方式为串口，配置为：常规链路的波特率为 115200，图传链路的波特率为 921600，8 位数据位，1 位停止位，无硬件流控，无校验位。

表 1-1 通信协议格式

frame_header	cmd_id	data	frame_tail
5-byte	2-byte	n-byte	2-byte，CRC16，整包校验

表 1-2 frame_header 格式

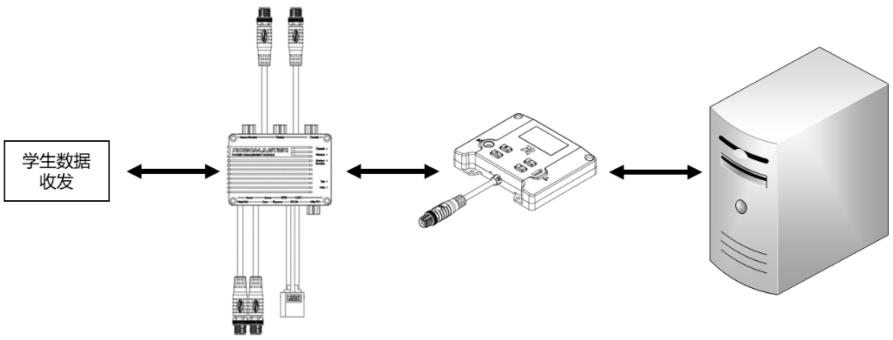
SOF	data_length	seq	CRC8
1-byte	2-byte	1-byte	1-byte

表 1-3 帧头详细定义

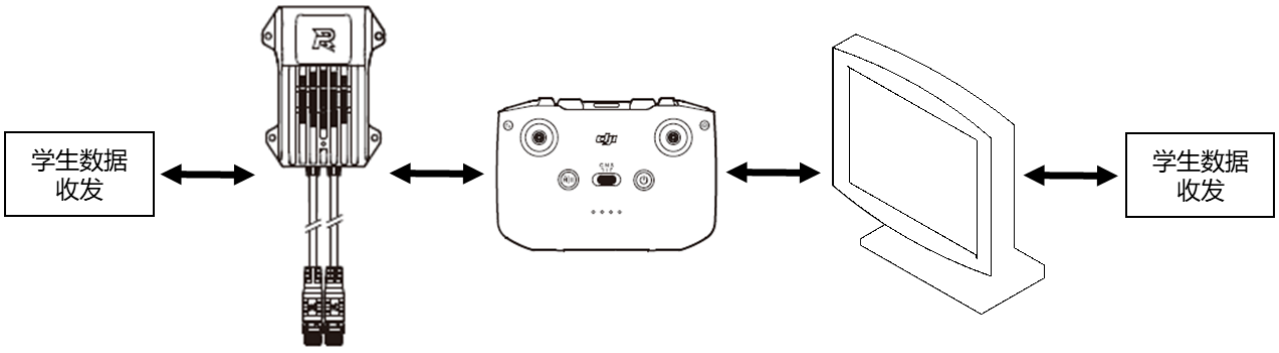
域	偏移位置	大小（字节）	详细描述
SOF	0	1	数据帧起始字节，固定值为 0xA5
data_length	1	2	数据帧中 data 的长度
seq	3	1	包序号
CRC8	4	1	帧头 CRC8 校验

裁判系统串口数据链路有三种：常规链路、图传链路、雷达无线链路。

- 常规链路由裁判系统服务器和主控模块进行数据转发，从电源管理模块的 User 串口收发数据，示意图如下：



- 图传链路由裁判系统选手端和图传模块进行数据转发，从图传模块（发送端）的串口接收数据，示意图如下：



- 雷达无线链路由裁判系统信号发射源进行数据发送，从雷达接收电磁波并解析信息。

- 正常工作状态下，裁判系统数据延迟约为 130ms，丢包率小于 1%；
- 在赛场网络环境较恶劣时，裁判系统数据延迟约为 200ms，丢包率约为 3%；
- 测量数据可能存在误差，数据仅供参考。

1.2 命令码 ID 和常规链路数据说明

表 1-4 命令码 ID 一览

命令码	数据段长度	说明	发送方/接收方	所属数据链路
0x0001	11	比赛状态数据，固定以 1Hz 频率发送	服务器→全体机器人	常规链路
0x0002	1	比赛结果数据，比赛结束触发发送	服务器→全体机器人	常规链路
0x0003	16	机器人血量数据，固定以 3Hz 频率发送	服务器→全体机器人	常规链路
0x0101	4	场地事件数据，固定以 1Hz 频率发送	服务器→己方全体机器人	常规链路
0x0104	3	裁判警告数据，己方判罚/判负时触发发送，其余时间以 1Hz 频率发送	服务器→被判罚方全体机器人	常规链路
0x0105	3	飞镖发射相关数据，固定以 1Hz 频率发送	服务器→己方全体机器人	常规链路
0x0201	13	机器人性能体系数据，固定以 10Hz 频率发送	主控模块→对应机器人	常规链路
0x0202	14	实时底盘缓冲能量和射击热量数据，固定以 10Hz 频率发送	主控模块→对应机器人	常规链路
0x0203	16	机器人位置数据，固定以 1Hz 频率发送	主控模块→对应机器人	常规链路
0x0204	8	机器人增益和底盘能量数据，固定以 3Hz 频率发送	服务器→对应机器人	常规链路
0x0206	1	伤害状态数据，伤害发生后发送	主控模块→对应机器人	常规链路

命令码	数据段长度	说明	发送方/接收方	所属数据链路
0x0207	7	实时射击数据，弹丸发射后发送	主控模块→对应机器人	常规链路
0x0208	6	允许发弹量，固定以 10Hz 频率发送	服务器→己方英雄、步兵、哨兵、空中机器人	常规链路
0x0209	5	机器人 RFID 模块状态，固定以 3Hz 频率发送	服务器→己方装有 RFID 模块的机器人	常规链路
0x020A	6	飞镖选手端指令数据，固定以 3Hz 频率发送	服务器→己方飞镖机器人	常规链路
0x020B	40	地面机器人位置数据，固定以 1Hz 频率发送	服务器→己方哨兵机器人	常规链路
0x020C	2	雷达标记进度数据，固定以 1Hz 频率发送	服务器→己方雷达机器人	常规链路
0x020D	6	哨兵自主决策信息同步，固定以 1Hz 频率发送	服务器→己方哨兵机器人	常规链路
0x020E	1	雷达自主决策信息同步，固定以 1Hz 频率发送	服务器→己方雷达机器人	常规链路
0x0301	127	机器人交互数据，发送方触发发送，频率上限为 30Hz	-	常规链路
0x0302	30	自定义控制器与机器人交互数据，发送方触发发送，频率上限为 30Hz	自定义控制器→选手端图传连接的机器人	图传链路
0x0303	15	选手端小地图交互数据，选手端触发发送	选手端点击→服务器→发送方选择的己方机器人	常规链路
0x0304	12	键鼠遥控数据，固定 30Hz 频率发送	选手端→选手端图传连接的机器人	图传链路

命令码	数据段长度	说明	发送方/接收方	所属数据链路
0x0305	24	选手端小地图接收雷达数据，频率上限为 5Hz	雷达→服务器→己方所有选手端	常规链路
0x0306	8	自定义控制器与选手端交互数据，发送方触发发送，频率上限为 30Hz	自定义控制器→选手端	-
0x0307	103	选手端小地图接收路径数据，频率上限为 1Hz	哨兵/半自动控制机器人→对应操作手选手端	常规链路
0x0308	34	选手端小地图接收机器人数据，频率上限为 3Hz	己方机器人→己方选手端	常规链路
0x0309	30	自定义控制器接收机器人数据，频率上限为 10Hz	己方机器人→对应操作手选手端连接的自定义控制器	图传链路
0x0310	150	机器人发送给自定义客户端的数据，频率上限为 50Hz	己方机器人→图传链路→对应操作手选手端连接的自定义客户端	图传链路
0x0F01	• 发送 1 • 接收 1	设置图传出图信道，频率上限为 1 z	• 发送：机器人→图传发送端 • 接收：图传发送端→机器人	图传链路
0x0F02	• 发送 0 • 接收 1	查询当前出图信道，频率上限为 2Hz	• 发送：机器人→图传发送端 • 接收：图传发送端→机器人	图传链路
0x0A01	24	对方机器人的位置坐标，频率上限为 10Hz	信号发射源→雷达	雷达无线链路
0x0A02	12	对方机器人的血量信息，频率上限为 10Hz	信号发射源→雷达	雷达无线链路

命令码	数据段长度	说明	发送方/接收方	所属数据链路
0x0A03	10	对方机器人的剩余发弹量信息，频率上限为 10Hz	信号发射源→雷达	雷达无线链路
0x0A04	8	对方队伍的宏观状态信息，频率上限为 10Hz	信号发射源→雷达	雷达无线链路
0x0A05	36	对方各机器人当前增益效果，频率上限为 10Hz	信号发射源→雷达	雷达无线链路
0x0A06	6	对方干扰波密钥，频率上限为 10Hz	信号发射源→雷达	雷达无线链路

表 1-5 0x0001

字节偏移量	大小	说明
0	1	bit 0-3：比赛类型 • 1：RoboMaster 机甲大师超级对抗赛 • 2：RoboMaster 机甲大师高校单项赛 • 3：ICRA RoboMaster 高校人工智能挑战赛 • 4：RoboMaster 机甲大师高校联盟赛 3V3 对抗 • 5：RoboMaster 机甲大师高校联盟赛步兵对抗 bit 4-7：当前比赛阶段 • 0：未开始比赛 • 1：准备阶段 • 2：十五秒裁判系统自检阶段 • 3：五秒倒计时 • 4：比赛中 • 5：比赛结算中
1	2	当前阶段剩余时间，单位：秒

字节偏移量	大小	说明
3	8	UNIX 时间，当机器人正确连接到裁判系统的 NTP 服务器后生效

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

表 1-6 0x0002

字节偏移量	大小	说明
0	1	• 0：平局 • 1：红方胜利 • 2：蓝方胜利

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

表 1-7 0x0003

字节偏移量	大小	说明
0	2	己方 1 号英雄机器人血量，若该机器人未上场或者被罚下，则血量为 0，下文同理
2	2	己方 2 号工程机器人血量
4	2	己方 3 号步兵机器人血量
6	2	己方 4 号步兵机器人血量
8	2	保留位
10	2	己方 7 号哨兵机器人血量

字节偏移量	大小	说明
12	2	己方前哨站血量
14	2	己方基地血量

```
typedef _packed struct
{
uint16_t ally_1_robot_HP;
uint16_t ally_2_robot_HP;
uint16_t ally_3_robot_HP;
uint16_t ally_4_robot_HP;
uint16_t reserved;
uint16_t ally_7_robot_HP;
uint16_t ally_outpost_HP;
uint16_t ally_base_HP;
} game_robot_HP_t;
```

表 1-8 0x0101

字节偏移量	大小	说明
0	4	0：未占领/未激活 1：已占领/已激活 ● bit 0-2: ➢ bit 0：己方与资源区区不重叠的补给区占领状态，1 为已占领 ➢ bit 1：己方与资源区重叠的补给区占领状态，1 为已占领 ➢ bit 2：己方补给区的占领状态，1 为已占领（仅 RMUL 适用） ● bit 3-6：己方能量机关状态 ➢ bit 3-4：己方小能量机关的激活状态，0 为未激活，1 为已激活，2 为正在激活 ➢ bit 5-6：己方大能量机关的激活状态，0 为未激活，1 为已激活，2 为正在激活

字节偏移量	大小	说明
		• bit 7-8：己方中央高地的占领状态，1 为被己方占领，2 为被对方占领 • bit 9-10：己方梯形高地的占领状态，1 为已占领 • bit 11-19：对方飞镖最后一次击中己方前哨站或基地的时间（0-420，开局默认为 0） • bit 20-22：对方飞镖最后一次击中己方前哨站或基地的具体目标，开局默认为 0，1 为击中前哨站，2 为击中基地固定目标，3 为击中基地随机固定目标，4 为击中基地随机移动目标，5 为击中基地末端移动目标 • bit 23-24：中心增益点的占领状态，0 为未被占领，1 为被己方占领，2 为被对方占领，3 为被双方占领。（仅 RMUL 适用） • bit 25-26：己方堡垒增益点的占领状态，0 为未被占领，1 为被己方占领，2 为被对方占领，3 为被双方占领 • bit 27-28：己方前哨站增益点的占领状态，0 为未被占领，1 为被己方占领，2 为被对方占领 • bit 29：己方基地增益点的占领状态，1 为已占领 • bit 30-31：保留位

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

表 1-9 0x0104

字节偏移量	大小	说明
0	1	己方最后一次受到判罚的等级： • 1：双方黄牌 • 2：黄牌 • 3：红牌

字节偏移量	大小	说明
		● 4：判负
1	1	● 己方最后一次受到判罚的违规机器人 ID。（如红 1 机器人 ID 为 1，蓝 1 机器人 ID 为 101） ● 判负和双方黄牌时，该值为 0
2	1	己方最后一次受到判罚的违规机器人对应判罚等级的违规次数。（开局默认为 0。）

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

表 1-10 0x0105

字节偏移量	大小	说明
0	1	己方飞镖发射剩余时间，单位：秒
1	2	● bit 0-2: 最近一次己方飞镖击中的目标，开局默认为 0，1 为击中前哨站，2 为击中基地固定目标，3 为击中基地随机固定目标，4 为击中基地随机移动目标，5 为击中基地末端移动目标 ● bit 3-5: 对方最近被击中的目标累计被击中计次数，开局默认为 0，至多为 4 ● bit 6-7: 飞镖此时选定的击打目标，开局默认或未选定/选定前哨站时为 0，选中基地固定目标为 1，选中基地随机固定目标为 2，选中基地随机移动目标为 3，选中基地末端移动目标为 4 ● bit 8-15：保留

字节偏移量	大小	说明

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

表 1-11 0x0201

字节偏移量	大小	说明
0	1	本机器人 ID
1	1	机器人等级
2	2	机器人当前血量
4	2	机器人血量上限
6	2	机器人射击热量每秒冷却值
8	2	机器人射击热量上限
10	2	机器人底盘功率上限
12	1	电源管理模块的输出情况： • bit 0：gimbal 口输出，0 为无输出，1 为 24V 输出 • bit 1：chassis 口输出，0 为无输出，1 为 24V 输出 • bit 2：shooter 口输出，0 为无输出，1 为 24V 输出

```
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;
```

表 1-12 0x0202

字节偏移量	大小	说明
0	2	保留位
2	2	保留位
4	4	保留位
8	2	缓冲能量（单位：J）
10	2	第 1 个 17mm 发射机构的射击热量
12	2	42mm 发射机构的射击热量

```
typedef _packed struct
{
  uint16_t reserved;
  uint16_t reserved;
  float reserved;
  uint16_t buffer_energy;
  uint16_t shooter_17mm_1_barrel_heat;
  uint16_t shooter_42mm_barrel_heat;
}power_heat_data_t;
```

表 1-13 0x0203

字节偏移量	大小	说明
0	4	本机器人位置 x 坐标，单位：m
4	4	本机器人位置 y 坐标，单位：m

字节偏移量	大小	说明
8	4	本机器人测速模块的朝向，单位：度。正北为 0 度

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

表 1-14 0x0204

字节偏移量	大小	说明
0	1	机器人回血增益（百分比，值为 10 表示每秒恢复血量上限的 10%）
1	2	机器人射击热量冷却增益具体值（直接值，值为 x 表示热量冷却增加 x/s）
3	1	机器人防御增益（百分比，值为 50 表示 50%防御增益）
4	1	机器人负防御增益（百分比，值为 30 表示-30%防御增益）
5	2	机器人攻击增益（百分比，值为 50 表示 50%攻击增益）
6	1	bit 0-6：机器人剩余能量值反馈，以 16 进制标识机器人剩余能量值比例，仅在机器人剩余能量小于 50%时反馈，其余默认反馈 0x80。机器人初始能量视为 100% • bit 0：在剩余能量≥125%时为 1，其余情况为 0 • bit 1：在剩余能量≥100%时为 1，其余情况为 0 • bit 2：在剩余能量≥50%时为 1，其余情况为 0 • bit 3：在剩余能量≥30%时为 1，其余情况为 0 • bit 4：在剩余能量≥15%时为 1，其余情况为 0 • bit 5：在剩余能量≥5%时为 1，其余情况为 0 • bit 6：在剩余能量≥1%时为 1，其余情况为 0


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

表 1-15 0x0206

字节偏移量	大小	说明
0	1	bit 0-3：当扣血原因为装甲模块被弹丸攻击、受撞击或离线时，该 4 bit 组成的数值为装甲模块或测速模块的 ID 编号；当其他原因导致扣血时，该数值为 0 bit 4-7：血量变化类型 ● 0：装甲模块被弹丸攻击导致扣血 ● 1：装甲模块或超级电容管理模块离线导致扣血 ● 5：装甲模块受到撞击导致扣血

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

0x0206 的受伤害情况为机器人裁判系统本地判定，即时发送，但实际是否受到对应伤害受规则条例影响，请以服务器最终判定为准。

表 1-16 0x0207

字节偏移量	大小	说明
0	1	弹丸类型：

字节偏移量	大小	说明
		• bit 1：17mm 弹丸 • bit 2：42mm 弹丸
1	1	发射机构 ID： • 1： 17mm 发射机构 • 2：保留位 • 3：42mm 发射机构
2	1	弹丸射速（单位：Hz）
3	4	弹丸初速度（单位：m/s）

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

表 1-17 0x0208

字节偏移量	大小	说明
0	2	机器人自身拥有的 17mm 弹丸允许发弹量
2	2	42mm 弹丸允许发弹量
4	2	剩余金币数量
6	2	堡垒增益点提供的储备 17mm 弹丸允许发弹量； 该值与机器人是否实际占领堡垒无关

```
typedef _packed struct
{
  uint16_t projectile_allowance_17mm;
```



```
  uint16_t projectile_allowance_42mm;
  uint16_t remaining_gold_coin;
  uint16_t projectile_allowance_fortress;
}projectile_allowance_t;
```

表 1-18 0x0209

字节偏移量	大小	说明
0	4	bit 位值为 1/0 的含义：是否已检测到该增益点 RFID 卡 ● bit 0：己方基地增益点 ● bit 1：己方中央高地增益点 ● bit 2：对方中央高地增益点 ● bit 3：己方梯形高地增益点 ● bit 4：对方梯形高地增益点 ● bit 5：己方地形跨越增益点（飞坡）（靠近己方一侧飞坡前） ● bit 6：己方地形跨越增益点（飞坡）（靠近己方一侧飞坡后） ● bit 7：对方地形跨越增益点（飞坡）（靠近对方一侧飞坡前） ● bit 8：对方地形跨越增益点（飞坡）（靠近对方一侧飞坡后） ● bit 9：己方地形跨越增益点（中央高地下方） ● bit 10：己方地形跨越增益点（中央高地上方） ● bit 11：对方地形跨越增益点（中央高地下方） ● bit 12：对方地形跨越增益点（中央高地上方） ● bit 13：己方地形跨越增益点（公路下方） ● bit 14：己方地形跨越增益点（公路上方） ● bit15：对方地形跨越增益点（公路下方） ● bit16：对方地形跨越增益点（公路上方） ● bit 17：己方堡垒增益点 ● bit 18：己方前哨站增益点 ● bit 19：己方与资源区不重叠的补给区/RMUL 补给区

字节偏移量	大小	说明
		• bit 20：己方与资源区重叠的补给区 • bit 21：己方装配增益点 • bit 22：对方装配增益点 • bit 23：中心增益点（仅 RMUL 适用） • bit 24：对方堡垒增益点 • bit 25：对方前哨站增益点 • bit 26：己方地形跨越增益点（隧道）（靠近己方一侧公路区下方） • bit 27：己方地形跨越增益点（隧道）（靠近己方一侧公路区上方） • bit 28：己方地形跨越增益点（隧道）（靠近己方梯形高地较低处） • bit 29：己方地形跨越增益点（隧道）（靠近己方梯形高地较高处） • bit 30：对方地形跨越增益点（隧道）（靠近对方一侧公路区下方） • bit 31：对方地形跨越增益点（隧道）（靠近对方一侧公路区上方） 注：所有 RFID 卡仅在赛内生效。在赛外，即使检测到对应的 RFID 卡，对应值也为 0。
4	1	• bit 0：对方地形跨越增益点（隧道）（靠近对方梯形高地较低处） • bit 1：对方地形跨越增益点（隧道）（靠近对方梯形高地较高处）

```
typedef _packed struct
{
  uint32_t rfid_status;
  uint8_t rfid_status_2;
}rfid_status_t;
```

表 1-19 0x020A

字节偏移量	大小	说明
0	1	当前飞镖发射站的状态： • 1：关闭 • 2：正在开启或者关闭中

字节偏移量	大小	说明
		● 0：已经开启
1	1	保留位
2	2	切换击打目标时的比赛剩余时间，单位：秒，无/未切换动作，默认为 0。
4	2	最后一次操作手确定发射指令时的比赛剩余时间，单位：秒，初始值为 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;
```

表 1-20 0x020B

字节偏移量	大小	说明
0	4	己方英雄机器人位置 x 轴坐标，单位：m
4	4	己方英雄机器人位置 y 轴坐标，单位：m
8	4	己方工程机器人位置 x 轴坐标，单位：m
12	4	己方工程机器人位置 y 轴坐标，单位：m
16	4	己方 3 号步兵机器人位置 x 轴坐标，单位：m
20	4	己方 3 号步兵机器人位置 y 轴坐标，单位：m
24	4	己方 4 号步兵机器人位置 x 轴坐标，单位：m
28	4	己方 4 号步兵机器人位置 y 轴坐标，单位：m
32	4	保留位

字节偏移量	大小	说明
36	4	保留位

场地围挡在红方补给站附近的交点为坐标原点，沿场地长边向蓝方为 X 轴正方向，沿场地短边向红方停机坪为 Y 轴正方向。

```
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 reserved;
  float reserved;
}ground_robot_position_t;
```

表 1-21 0x020C

字节偏移量	大小	说明	备注
0	2	• bit 0：对方 1 号英雄机器人易伤情况 • bit 1：对方 2 号工程机器人易伤情况 • bit 2：对方 3 号步兵机器人易伤情况 • bit 3：对方 4 号步兵机器人易伤情况 • bit 4：对方哨兵机器人易伤情况 • bit 5：己方 1 号英雄机器人特殊标识情况	• 对方机器人：在对应机器人被标记进度≥100 时发送 1，被标记进度<100 时发送 0。 • 己方机器人：在对应机器人被标记进度≥50 时发送 1，被标记进度<50 时发送 0。

字节偏移量	大小	说明	备注
		• bit 6：己方 2 号工程机器人特殊标识情况 • bit 7：己方 3 号步兵机器人特殊标识情况 • bit 8：己方 4 号步兵机器人特殊标识情况 • bit 9：己方哨兵机器人特殊标识情况 • bit 10-15：保留位	

```
typedef _packed struct
{
  uint16_t mark_ progress;
}radar_mark_data_t;
```

表 1-22 0x020D

字节偏移量	大小	说明
0	4	• bit 0-10：除远程兑换外，哨兵机器人成功兑换的允许发弹量，开局为 0，在哨兵机器人成功兑换一定允许发弹量后，该值将变为哨兵机器人成功兑换的允许发弹量值 • bit 11-14：哨兵机器人成功远程兑换允许发弹量的次数，开局为 0，在哨兵机器人成功远程兑换允许发弹量后，该值将变为哨兵机器人成功远程兑换允许发弹量的次数 • bit 15-18：哨兵机器人成功远程兑换血量的次数，开局为 0，在哨兵机器人成功远程兑换血量后，该值将变为哨兵机器人成功远程兑换血量的次数 • bit 19：哨兵机器人当前是否可以确认免费复活，可以确认免费复活时值为 1，否则为 0 • bit 20：哨兵机器人当前是否可以兑换立即复活，可以兑换立即复活时值为 1，否则为 0 • bit 21-30：哨兵机器人当前若兑换立即复活需要花费的金币数。

字节偏移量	大小	说明
		● bit 31：保留
4	2	● bit 12-13：哨兵当前姿态，1 为进攻姿态，2 为防御姿态，3 为移动姿态 ● bit 14：己方能量机关是否能够进入正在激活状态，1 为当前可激活 ● bit 15：保留位

```
typedef _packed struct
{
  uint32_t sentry_info;
  uint16_t sentry_info_2;
} sentry_info_t;
```

表 1-23 0x020E

字节偏移量	大小	说明
0	1	● bit 0-1：雷达是否拥有触发双倍易伤的机会，开局为 0，数值为雷达拥有触发双倍易伤的机会，至多为 2 ● bit 2：对方是否正在被触发双倍易伤 ➢ 0：对方未被触发双倍易伤 ➢ 1：对方正在被触发双倍易伤 ● bit 3-4：己方加密等级（即对方干扰波难度等级），开局为 1，最高为 3 ● bit 5：当前是否可以修改密钥，1 为可修改 ● bit 6-7：保留位

```
typedef _packed struct
{
  uint8_t radar_info;
} radar_info_t;
```


机器人交互数据通过常规链路发送，其数据段包含一个统一的数据段头结构。数据段头结构包括内容 ID、发送者和接收者的 ID、内容数据段。机器人交互数据包的总长不超过 127 个字节，减去 frame_header、cmd_id 和 frame_tail 的 9 个字节以及数据段头结构的 6 个字节，故机器人交互数据的内容数据段最大为 112 个字节。

每 1000 毫秒，英雄、工程、步兵、空中机器人、飞镖能够接收数据的上限为 3720 字节，雷达和哨兵机器人能够接收数据的上限为 5120 字节。

由于存在多个内容 ID，但整个 cmd_id 上行频率最大为 30Hz，请合理安排带宽。

表 1-24 0x0301

字节偏移量	大小	说明	备注
0	2	子内容 ID	需为开放的子内容 ID
2	2	发送者 ID	需与自身 ID 匹配，ID 编号详见附录
4	2	接收者 ID	• 仅限己方通信 • 需为规则允许的多机通讯接收者 • 若接收者为选手端，则仅可发送至发送者对应的选手端 • ID 编号详见附录
6	x	内容数据段	x 最大为 112

```
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;
```

子内容 ID	内容数据段长度	功能说明
0x0200~0x02FF	x≤112	机器人之间通信
0x0100	2	选手端删除图层

子内容 ID	内容数据段长度	功能说明
0x0101	15	选手端绘制一个图形
0x0102	30	选手端绘制两个图形
0x0103	75	选手端绘制五个图形
0x0104	105	选手端绘制七个图形
0x0110	45	选手端绘制字符图形
0x0120	4	哨兵自主决策指令
0x0121	1	雷达自主决策指令

表 1-25 子内容 ID：0x0100

字节偏移量	大小	说明	备注
0	1	删除操作	● 0：空操作 ● 1：删除图层 ● 2：删除所有
1	1	图层数	图层数：0~9

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

表 1-26 子内容 ID：0x0101

字节偏移量	大小	说明	备注
0	3	图形名	在图形删除、修改等操作中，作为索引
3	4	图形配置 1	bit 0-2：图形操作

字节偏移量	大小	说明	备注
			● 0：空操作 ● 1：增加 ● 2：修改 ● 3：删除 bit 3-5：图形类型 ● 0：直线 ● 1：矩形 ● 2：正圆 ● 3：椭圆 ● 4：圆弧 ● 5：浮点数 ● 6：整型数 ● 7：字符 bit 6-9：图层数（0~9） bit 10-13：颜色 ● 0：红/蓝（己方颜色） ● 1：黄色 ● 2：绿色 ● 3：橙色 ● 4：紫红色 ● 5：粉色 ● 6：青色 ● 7：黑色 ● 8：白色

字节偏移量	大小	说明	备注
			bit 14-31：根据绘制的图形不同，含义不同，详见“表 1-27 图形细节参数说明”
7	4	图形配置 2	bit 0-9：线宽，建议字体大小与线宽比例为 10：1 bit 10-20：起点/圆心 x 坐标 bit 21-31：起点/圆心 y 坐标
11	4	图形配置 3	根据绘制的图形不同，含义不同，详见“表 1-27 图形细节参数说明”

```
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;
```

表 1-27 图形细节参数说明

类型	details_a	details_b	details_c	details_d	details_e
直线	-	-	-	终点 x 坐标	终点 y 坐标
矩形	-	-	-	对角顶点 x 坐标	对角顶点 y 坐标
正圆	-	-	半径	-	-
椭圆	-	-	-	x 半轴长度	y 半轴长度
圆弧	起始角度	终止角度	-	x 半轴长度	y 半轴长度
浮点数	字体大小	无作用	该值除以 1000 即实际显示值		
整型数	字体大小	-	32 位整型数，int32_t		
字符	字体大小	字符长度	-	-	-

- 角度值含义为：0°指 12 点钟方向，顺时针绘制；
- 屏幕位置：（0,0）为屏幕左下角（1920，1080）为屏幕右上角；
- 浮点数：整型数均为 32 位，对于浮点数，实际显示的值为输入的值/1000，如在 details_c、details_d、details_e 对应的字节输入 1234，选手端实际显示的值将为 1.234。
- 即使发送的数值超过对应数据类型的限制，图形仍有可能显示，但此时不保证显示的效果。

表 1-28 子内容 ID：0x0102

字节偏移量	大小	说明	备注
0	15	图形 1	与 0x0101 的数据段相同
15	15	图形 2	与 0x0101 的数据段相同

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

表 1-29 子内容 ID：0x0103

字节偏移量	大小	说明	备注
0	15	图形 1	与 0x0101 的数据段相同
15	15	图形 2	与 0x0101 的数据段相同
30	15	图形 3	与 0x0101 的数据段相同
45	15	图形 4	与 0x0101 的数据段相同
60	15	图形 5	与 0x0101 的数据段相同

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

表 1-30 子内容 ID：0x0104

字节偏移量	大小	说明	备注
0	15	图形 1	与 0x0101 的数据段相同
15	15	图形 2	与 0x0101 的数据段相同
30	15	图形 3	与 0x0101 的数据段相同

字节偏移量	大小	说明	备注
45	15	图形 4	与 0x0101 的数据段相同
60	15	图形 5	与 0x0101 的数据段相同
75	15	图形 6	与 0x0101 的数据段相同
90	15	图形 7	与 0x0101 的数据段相同

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

表 1-31 子内容 ID：0x0110

字节偏移量	大小	说明	备注
0	2	数据的内容 ID	0x0110
2	2	发送者的 ID	需要校验发送者的 ID 正确性
4	2	接收者的 ID	需要校验接收者的 ID 正确性，仅支持发送机器人对应的选手端
6	15	字符配置	详见图形数据介绍
21	30	字符	-

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


表 1-32 哨兵自主决策指令：0x0120

字节偏移量	大小	说明	备注
0	4	哨兵自主决策相关指令	● bit 0：哨兵机器人是否确认复活 ➢ 0 表示哨兵机器人确认不复活，即使此时哨兵的复活读条已经完成 ➢ 1 表示哨兵机器人确认复活，若复活读条完成将立即复活 ● bit 1：哨兵机器人是否确认兑换立即复活 ➢ 0 表示哨兵机器人确认不兑换立即复活； ➢ 1 表示哨兵机器人确认兑换立即复活，若此时哨兵机器人符合兑换立即复活的规则要求，则会立即消耗金币兑换立即复活 ● bit 2-12：哨兵将要兑换的发弹量值，开局为 0，修改此值后，哨兵在补血点即可兑换允许发弹量 此值的变化需要单调递增，否则视为不合法。 **示例：此值开局仅能为 0，此后哨兵可将其从 0 修改至 X，则消耗 X 金币成功兑换 X 允许发弹量。此后哨兵可将其从 X 修改至 X+Y，以此类推。** ● bit 13-16：哨兵远程兑换发弹量的请求次数，开局为 0，修改此值即可请求远程兑换发弹量 此值的变化需要单调递增且每次仅能增加 1，否则视为不合法。 **示例：此值开局仅能为 0，此后哨兵可将其从 0 修改至 1，则消耗金币远程兑换允许发弹量。此后哨兵可将其从 1 修改至 2，以此类推。** ● bit 17-20：哨兵远程兑换血量的请求次数，开局为 0，修改此值即可请求远程兑换血量

字节偏移量	大小	说明	备注
			此值的变化需要单调递增且每次仅能增加 1，否则视为不合法。 **示例：此值开局仅能为 0，此后哨兵可将其从 0 修改至 1，则消耗金币远程兑换血量。此后哨兵可将其从 1 修改至 2，以此类推。** 在哨兵发送该子命令时，服务器将按照从相对低位到相对高位的原则依次处理这些指令，直至全部成功或不能处理为止。 **示例：若队伍金币数为 0，此时哨兵战亡，“是否确认复活”的值为 1，“是否确认兑换立即复活”的值为 1，“确认兑换的允许发弹量值”为 100。（假定之前哨兵未兑换过允许发弹量）由于此时队伍金币数不足以使哨兵兑换立即复活，则服务器将会忽视后续指令，等待哨兵发送的下一组指令。** ● bit 21-22：哨兵修改当前姿态指令，1 为进攻姿态，2 为防御姿态，3 为移动姿态，默认为 3；修改此值即可改变哨兵姿态。 ● bit 23：哨兵机器人是否确认使能量机关进入正在激活状态，1 为确认。默认为 0。 ● bit 24-31：保留位。

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

表 1-33 雷达自主决策指令：0x0121

字节偏移量	大小	说明	备注
0	1	雷达是否确认触发双倍易伤	开局为 0，修改此值即可请求触发双倍易伤，若此时雷达拥有触发双倍易伤的机会，则可触发。 此值的变化需要单调递增且每次仅能增加 1，否则视为不合法。

字节偏移量	大小	说明	备注
			示例：此值开局仅能为 0，此后雷达可将其从 0 修改至 1，若雷达拥有触发双倍易伤的机会，则触发双倍易伤。此后雷达可将其从 1 修改至 2，以此类推。 若雷达请求双倍易伤时，双倍易伤正在生效，则第二次双倍易伤将在第一次双倍易伤结束后生效。
1	7	密钥更新或验证指令	每个字节均为 ASCII 码编码的字母或数字。开局为随机值。byte1 为指令类型，byte2-7 为密钥值。 当 byte1 值为 1 时，修改此值即可更新己方加密密钥；当 byte1 值为 2 时，修改此值即可将雷达破解的对方密钥传输给服务器以验证是否正确破解。 注意： ● 仅开局和每次对方破解成功使得加密等级（己方干扰波难度）提高时可以修改密钥，其余时间修改无效。 ● 当 byte1 值为 2 时，每次更新验证密钥后的 10 秒内，再次更新无效。


```
typedef _packed struct
{
  uint8_t radar_cmd;
  uint8_t password_cmd;
  uint8_t password_1;
  uint8_t password_2;
  uint8_t password_3;
  uint8_t password_4;
  uint8_t password_5;
  uint8_t password_6;
} radar_cmd_t;
```

表 1-34 设置图传出图信道：0x0F01

字节偏移量	大小	说明	备注
1	接收 1	设置出图信道并接收设置反馈	设置值 1-6：设置信道为 1~6； 反馈值 0：设置信道成功； 反馈值 1：图传启动中，无法设置信道； 反馈值 2：设置信道有误，无法设置。

表 1-35 设置图传出图信道：0x0F02

字节偏移量	大小	说明	备注
0/1	发送 0	查询出图信道。	数据段无需信息，仅发送该命令码即可查询图传出图信道。 反馈值 0：未设置信道； 反馈值 1~6：当前运行的信道。

1.3 小地图交互数据

1.3.1 选手端下发数据

- 云台手可通过选手端大地图向机器人发送固定数据。

命令码为 0x0303，触发时发送，两次发送间隔不得低于 0.5 秒。

发送方式一：

① 点击己方机器人头像；

②（可选）按下一个键盘按键或点击对方机器人头像；

③点击小地图任意位置。该方式向己方选定的机器人发送地图坐标数据，若点击对方机器人头像，则以目标机器人 ID 代替坐标数据。

发送方式二：

①（可选）按下一个键盘按键或点击对方机器人头像；

②点击小地图任意位置。该方式向己方所有机器人发送地图坐标数据，若点击对方机器人头像，则以目标机器人 ID 代替坐标数据。

- 半自动控制方式的机器人对应的操作手可通过选手端大地图向机器人发送固定数据。

命令码为 0x0303，触发时发送，两次发送间隔不得低于 3 秒。

发送方式：

①（可选）按下一个键盘按键或点击对方机器人头像；

②点击小地图任意位置。该方式向操作手对应的机器人发送地图坐标数据，若点击对方机器人头像，则以目标机器人 ID 代替坐标数据。

一台半自动控制方式的机器人既可以接收云台手发送的信息，也可以接收对应操作手的信息。两种信息的来源将在下表中“信息来源”中进行区别。

为降低机器人串口接收设备的偶发不稳定性对通信的影响，0x0303 协议的发送机制有所特殊处理，具体如下：选手端触发 1 次发送后，服务器将以 100ms 的间隔向机器人额外发送 4 次，共 5 次。此后，直到下一次选手端触发发送前，服务器都将以 1Hz 的频率持续定频发送最近一次的包。触发时的连续发送和 1Hz 定频发送计时相互独立。队伍需关注多次收到重复协议内容的处理方式。

表 1-36 命令码 ID：0x0303

字节偏移量	大小	说明	备注
0	4	目标位置 x 轴坐标，单位 m	当发送目标机器人 ID 时，该值为 0
4	4	目标位置 y 轴坐标，单位 m	当发送目标机器人 ID 时，该值为 0
8	1	云台手按下的键盘按键通用键值	无按键按下，则为 0
9	1	对方机器人 ID	当发送坐标数据时，该值为 0
10	2	信息来源 ID	信息来源的 ID，ID 对应关系详见附录

```
typedef _packed struct
{
  float target_position_x;
  float target_position_y;
  uint8_t cmd_keyboard;
  uint8_t target_robot_id;
  uint16_t cmd_source;
}map_command_t;
```

1.3.2 选手端接收数据

选手端小地图可接收机器人数据。

雷达可通过常规链路向己方所有选手端发送对方机器人的坐标数据，该位置会在己方选手端小地图显示。

表 1-37 命令码 ID：0x0305

字节偏移量	大小	说明	备注
0	2	英雄机器人 x 位置坐标，单位：cm	当 x、y 超出边界时显示在对应边缘处，当 x、y 均为 0 时，视为未发送此机器人坐标。
2	2	英雄机器人 y 位置坐标，单位：cm	
4	2	工程机器人 x 位置坐标，单位：cm	

字节偏移量	大小	说明	备注
6	2	工程机器人 y 位置坐标，单位：cm	
8	2	3 号步兵机器人 x 位置坐标，单位：cm	
10	2	3 号步兵机器人 y 位置坐标，单位：cm	
12	2	4 号步兵机器人 x 位置坐标，单位：cm	
14	2	4 号步兵机器人 y 位置坐标，单位：cm	
16	2	5 号步兵机器人 x 位置坐标，单位：cm	
18	2	5 号步兵机器人 y 位置坐标，单位：cm	
20	2	哨兵机器人 x 位置坐标，单位：cm	
22	2	哨兵机器人 y 位置坐标，单位：cm	

```
typedef _packed struct
{
  uint16_t hero_position_x;
  uint16_t hero_position_y;
  uint16_t engineer_position_x;
  uint16_t engineer_position_y;
  uint16_t infantry_3_position_x;
  uint16_t infantry_3_position_y;
  uint16_t infantry_4_position_x;
  uint16_t infantry_4_position_y;
  uint16_t infantry_5_position_x;
  uint16_t infantry_5_position_y;
  uint16_t sentry_position_x;
  uint16_t sentry_position_y;
} map_robot_data_t;
```


哨兵机器人或半自动控制方式的机器人可通过常规链路向对应的操作手选手端发送路径坐标数据，该路径会在小地图上显示。

表 1-38 命令码 ID：0x0307

字节偏移量	大小	说明	备注
0	1	1：到目标点攻击 2：到目标点防守 3：移动到目标点	-
1	2	路径起点 x 轴坐标，单位：dm	小地图左下角为坐标原点，水平向右为 X 轴正方向，竖直向上为 Y 轴正方向。显示位置将按照场地尺寸与小地图尺寸等比缩放，超出边界的位置将在边界处显示
3	2	路径起点 y 轴坐标，单位：dm	
5	49	路径点 x 轴增量数组，单位：dm	增量相较于上一个点位进行计算，共 49 个新点位，X 与 Y 轴增量对应组成点位
54	49	路径点 y 轴增量数组，单位：dm	
103	2	发送者 ID	需与自身 ID 匹配，ID 编号详见附录

```
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;
```

己方机器人可通过常规链路向己方任意选手端发送自定义的消息，该消息会在己方选手端特定位置显示。

表 1-39 命令码 ID：0x0308

字节偏移量	大小	说明	备注
0	2	发送者的 ID	需要校验发送者的 ID 正确性
2	2	接收者的 ID	需要校验接收者的 ID 正确性，仅支持发送己方选手端
4	30	字符	以 utf-16 格式编码发送，支持显示中文。编码发送时请注意数据的大小端问题

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


1.4 图传链路数据说明

1.4.1 自定义控制器与机器人交互数据说明

操作手可使用自定义控制器通过图传链路向对应的机器人发送数据。

表 1-40 命令码 ID：0x0302

字节偏移量	大小	说明
0	30	自定义数据

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

机器人可通过图传链路向对应的操作手选手端连接的自定义控制器发送数据（RMUL 暂不适用）。

表 1-41 命令码 ID：0x0309

字节偏移量	大小	说明
0	30	自定义数据

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

表 1-42 命令码 ID：0x0310

字节偏移量	大小	说明
0	150	自定义数据

```
typedef _packed struct
{
    uint8_t data[x];
}robot_custom_ data_2_t;
```


1.4.2 键鼠遥控数据

通过遥控器发送的键鼠遥控数据将同步通过图传链路发送给对应机器人。

表 1-43 命令码 ID：0x0304

字节偏移量	大小	说明
0	2	鼠标 x 轴移动速度，负值标识向左移动
2	2	鼠标 y 轴移动速度，负值标识向下移动
4	2	鼠标滚轮移动速度，负值标识向后滚动
6	1	鼠标左键是否按下：0 为未按下；1 为按下
7	1	鼠标右键是否按下：0 为未按下，1 为按下
8	2	键盘按键信息，每个 bit 对应一个按键，0 为未按下，1 为按下： • bit 0：W 键 • bit 1：S 键 • bit 2：A 键 • bit 3：D 键 • bit 4：Shift 键 • bit 5：Ctrl 键 • bit 6：Q 键 • bit 7：E 键 • bit 8：R 键 • bit 9：F 键 • bit 10：G 键 • bit 11：Z 键 • bit 12：X 键

字节偏移量	大小	说明
		• bit 13：C 键 • bit 14：V 键 • bit 15：B 键
10	2	保留位

```
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;
```


1.5 非链路数据说明

操作手可使用自定义控制器模拟键鼠操作选手端。

表 1-44 命令码 ID：0x0306

字节偏移量	大小	说明	备注
0	2	键盘键值： ● bit 0-7：按键 1 键值 ● bit 8-15：按键 2 键值	● 仅响应选手端开放的按键 ● 使用通用键值，支持 2 键无冲，键值顺序变更不会改变按下状态，若无新的按键信息，将保持上一帧数据的按下状态
2	2	● bit 0-11：鼠标 X 轴像素位置 ● bit 12-15：鼠标左键状态	● 位置信息使用绝对像素点值（赛事客户端使用的分辨率为 1920×1080，屏幕左上角为（0，0）） ● 鼠标按键状态 1 为按下，其他值为未按下，仅在出现鼠标图标后响应该信息，若无新的鼠标信息，选手端将保持上一帧数据的鼠标信息，当鼠标图标消失后该数据不再保持
4	2	● bit 0-11：鼠标 Y 轴像素位置 ● bit 12-15：鼠标右键状态	
6	2	保留位	-

一次鼠标移动点击需要先发送鼠标未按下及指定位置的数据帧，再发送保持该位置时按下鼠标的数据帧，最后发送保持该位置时鼠标未按下的数据帧

```
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;
```

1.6 雷达无线链路数据说明

表 1-45 命令码 ID：0x0A01

字节偏移量	大小	说明
0	2	对方英雄机器人位置 x 轴坐标，单位：cm
2	2	对方英雄机器人位置 y 轴坐标，单位：cm
4	2	对方工程机器人位置 x 轴坐标，单位：cm
6	2	对方工程机器人位置 y 轴坐标，单位：cm
8	2	对方 3 号步兵机器人位置 x 轴坐标，单位：cm
10	2	对方 3 号步兵机器人位置 y 轴坐标，单位：cm
12	2	对方 4 号步兵机器人位置 x 轴坐标，单位：cm
14	2	对方 4 号步兵机器人位置 y 轴坐标，单位：cm
16	2	对方空中机器人位置 x 轴坐标，单位：cm
18	2	对方空中机器人位置 y 轴坐标，单位：cm
20	2	对方哨兵机器人位置 x 轴坐标，单位：cm
22	2	对方哨兵机器人位置 y 轴坐标，单位：cm

表 1-46 命令码 ID：0x0A02

字节偏移量	大小	说明
0	2	对方 1 号英雄机器人血量，若该机器人未上场或者被罚下，则血量为 0，下文同理
2	2	对方 2 号工程机器人血量
4	2	对方 3 号步兵机器人血量

字节偏移量	大小	说明
6	2	对方 4 号步兵机器人血量
8	2	保留位
10	2	对方 7 号哨兵机器人血量

表 1-47 命令码 ID：0x0A03

字节偏移量	大小	说明
0	2	对方 1 号英雄机器人允许发弹量
2	2	对方 3 号步兵机器人允许发弹量（含堡垒提供的储备允许发弹量，下同）
4	2	对方 4 号步兵机器人允许发弹量
6	2	对方 6 号空中机器人允许发弹量
8	2	对方 7 号哨兵机器人允许发弹量

表 1-48 命令码 ID：0x0A04

字节偏移量	大小	说明
0	2	对方剩余金币数
2	2	对方累计总金币数
4	4	• bit 0：对方补给区占领状态 • bit 1-2：对方中央高地的占领状态，1 为被对方占领，2 为被己方占领 • bit 3：对方梯形高地的占领状态，1 为已占领 • bit 4-5：对方堡垒增益点的占领状态，0 为未被占领，1 为被对方占领，2 为被己方占领，3 为被双方占领 • bit 6-7：对方前哨站增益点的占领状态，0 为未被占领，1 为被对方占领，2 为被己方占领 • bit 8：对方基地增益点的占领状态，1 为已占领

字节偏移量	大小	说明
		• bit 9：靠近对方一侧飞坡前地形跨越增益点（隧道）中心处场地交互模块卡的状态，1 为被对方占领 • bit 10：靠近对方一侧飞坡后地形跨越增益点（隧道）中心处场地交互模块卡的状态，1 为被对方占领 • bit 11：靠近己方一侧飞坡前地形跨越增益点（隧道）中心处场地交互模块卡的状态，1 为被对方占领 • bit 12：靠近己方一侧飞坡后地形跨越增益点（隧道）中心处场地交互模块卡的状态，1 为被对方占领 • bit 13：对方地形跨越增益点（高地）上部场地交互模块卡的状态，1 为被对方占领 • bit 14：对方地形跨越增益点（飞坡）上部场地交互模块卡的状态，1 为被对方占领 • bit 15：对方地形跨越增益点（公路）上部场地交互模块卡的状态，1 为被对方占领

表 1-49 命令码 ID：0x0A05

字节偏移量	大小	说明
0	1	对方英雄机器人回血增益（百分比，值为 10 表示每秒恢复血量上限的 10%，下同）
1	2	对方英雄机器人射击热量冷却增益具体值（直接值，值为 x 表示热量冷却增加 x/s，下同）
3	1	对方英雄机器人防御增益（百分比，值为 50 表示 50%防御增益，下同）
4	1	对方英雄机器人负防御增益（百分比，值为 30 表示-30%防御增益，下同）
5	2	对方英雄机器人攻击增益（百分比，值为 50 表示 50%攻击增益，下同）
7	1	对方工程机器人回血增益
8	2	对方工程机器人射击热量冷却增益具体值
10	1	对方工程机器人防御增益
11	1	对方工程机器人负防御增益
12	2	对方工程机器人攻击增益

字节偏移量	大小	说明
14	1	对方 3 号步兵机器人回血增益
15	2	对方 3 号步兵机器人射击热量冷却增益具体值
17	1	对方 3 号步兵机器人防御增益
18	1	对方 3 号步兵机器人负防御增益
19	2	对方 3 号步兵机器人攻击增益
21	1	对方 4 号步兵机器人回血增益
22	2	对方 4 号步兵机器人射击热量冷却增益具体值
24	1	对方 4 号步兵机器人防御增益
25	1	对方 4 号步兵机器人负防御增益
26	2	对方 4 号步兵机器人攻击增益
28	1	对方哨兵机器人回血增益
29	2	对方哨兵机器人射击热量冷却增益具体值
31	1	对方哨兵机器人防御增益
32	1	对方哨兵机器人负防御增益
33	2	对方哨兵机器人攻击增益
35	1	对方哨兵机器人当前姿态

表 1-50 命令码 ID：0x0A06

字节偏移量	大小	说明
0	6	每个字节均为 ASCII 码编码的字母或数字。

2. 自定义客户端协议

自定义客户端的数据流格式为 Protobuf v3 （Protocol Buffers），采用 MQTT 作为发布/订阅消息传输协议。MQTT 直接传输 Protobuf 序列化的二进制流。topic 为对应的指令名。服务器 IP 地址为固定 IP：192.168.12.1，端口：3333。自定义客户端 IP 地址为 DHCP 自动分配。

通信流程如下：

1. 结合文档中的 Protobuf 消息格式编写 proto 文件，用 protoc 生成不同语言的类代码
2. 在发布者端，将对象序列化成二进制
3. 通过 MQTT publish 把二进制发送到对应 Topic
4. 在订阅者端，接收消息并用 Protobuf 反序列化

示例代码，详见“附录三：自定义客户端示例通信代码”。

此外，自定义客户端可通过 udp 监听 3334 端口获取图传码流数据。编码格式为 hevc，每个 udp 包的码流数据的前 8 个字节固定为：

帧编号（递增）：2 byte

当前帧内分片序号：2 byte

当前帧总字节数：4 byte

2.1 指令概览

表 2-1 指令概览

指令（message）名	指令用途	发送方/接收方	最高 qos 等级	频率
RemoteControl	传输鼠标键盘输入和自定义数据	自定义客户端→服务器	1	75Hz
GameStatus	同步比赛全局状态信息	服务器→自定义客户端	1	5Hz
GlobalUnitStatus	同步基地、前哨站和所有机器人状态	服务器→自定义客户端	1	1Hz
GlobalLogisticsStatus	同步全局后勤信息	服务器→自定义客户端	1	1Hz
GlobalSpecialMechanism	同步正在生效的全局特殊机制	服务器→自定义客户端	1	1Hz

Event	全局事件通知	服务器→自定义客户端	1	触发式发送
RobotInjuryStat	机器人一次存活期间累计受伤统计	服务器→自定义客户端	1	1Hz
RobotRespawnStatus	机器人复活状态同步	服务器→自定义客户端	1	1Hz
RobotStaticStatus	机器人固定属性和配置	服务器→自定义客户端	1	1Hz
RobotDynamicStatus	机器人实时数据	服务器→自定义客户端	1	10Hz
RobotModuleStatus	机器人各模块运行状态	服务器→自定义客户端	1	1Hz
RobotPosition	机器人空间坐标和朝向	服务器→自定义客户端	1	1Hz
Buff	Buff 效果信息	服务器→自定义客户端	1	获得增益时触发发送，此后 1Hz 定频发送直到失去增益
PenaltyInfo	判罚信息同步	服务器→自定义客户端	1	触发式发送
RobotPathPlanInfo	哨兵轨迹规划信息	服务器→自定义客户端	1	1Hz
MapClickInfoNotify	云台手地图点击标记	自定义客户端→服务器	1	触发式发送
RaderInfoToClient	雷达发送的机器人位置信息	服务器→自定义客户端	1	1Hz
CustomByteBlock	机器人自定义上传数据流（机器人端对应 0x0310）	机器人→图传链路→自定义客户端	1	50Hz
AssemblyCommand	工程装配指令	自定义客户端→服务器	1	1Hz
TechCoreMotionStateSync	科技核心运动状态同步	服务器→自定义客户端	1	1Hz
RobotPerformanceSelectionCommand	步兵/英雄选择性能体系	自定义客户端→服务器	1	1Hz
RobotPerformanceSelectionSync	步兵/英雄性能体系状态同步	服务器→自定义客户端	1	1Hz

HeroDeployModeEventCommand	英雄部署模式相关指令	自定义客户端→服务器	1	1Hz
DeployModeStatusSync	英雄部署模式状态同步	服务器→自定义客户端	1	1Hz
RuneActivateCommand	能量机关激活指令	自定义客户端→服务器	1	1Hz
RuneStatusSync	能量机关状态同步	服务器→自定义客户端	1	1Hz
SentinelStatusSync	哨兵姿态相关信息同步	服务器→自定义客户端	1	1Hz
DartCommand	飞镖控制指令	自定义客户端→服务器	1	1Hz
DartSelectTargetStatusSync	飞镖目标选择状态同步	服务器→自定义客户端	1	1Hz
GuardCtrlCommand	哨兵控制指令请求	自定义客户端→服务器	1	1Hz
GuardCtrlResult	哨兵控制指令结果反馈	服务器→自定义客户端	1	1Hz
AirSupportCommand	空中支援指令	自定义客户端→服务器	1	1Hz
AirSupportStatusSync	空中支援状态反馈	服务器→自定义客户端	1	1Hz

2.2 详细协议定义

2.2.1 RemoteControl

用途：传输鼠标键盘输入和自定义数据

数据编号	数据类型	数据用途
1	int32	鼠标 x 轴移动速度，负值标识向左移动
2	int32	鼠标 y 轴移动速度，负值标识向下移动
3	int32	鼠标滚轮移动速度，负值标识向后滚动
4	bool	左键是否按下（false=抬起, true=按下）
5	bool	右键是否按下（false=抬起, true=按下）

数据编号	数据类型	数据用途
6	uint32	键盘按键位掩码
7	bool	中键是否按下（false=抬起, true=按下）
8	bytes	最大 30 字节的自定义数据

```
message RemoteControl {
  int32 mouse_x = 1;
  int32 mouse_y = 2;
  int32 mouse_z = 3;
  bool left_button_down = 4;
  bool right_button_down = 5;
  uint32 keyboard_value = 6;
  bool mid_button_down = 7;
  bytes data = 8;
}
```

2.2.2 GameStatus

用途：同步比赛全局状态信息

数据编号	数据类型	数据用途
1	uint32	当前局号（从 1 开始）
2	uint32	总局数
3	uint32	红方得分
4	uint32	蓝方得分
5	uint32	当前阶段
6	int32	当前阶段剩余时间（秒）
7	int32	当前阶段已过时间（秒）

数据编号	数据类型	数据用途
8	bool	是否暂停

current_stage 枚举值：

- 0：未开始比赛
- 1：准备阶段
- 2：十五秒裁判系统自检阶段
- 3：五秒倒计时
- 4：比赛中
- 5：比赛结算中

```
message GameStatus {
  uint32 current_round = 1;
  uint32 total_rounds = 2;
  uint32 red_score = 3;
  uint32 blue_score = 4;
  uint32 current_stage = 5;
  int32 stage_countdown_sec = 6;
  int32 stage_elapsed_sec = 7;
  bool is_paused = 8;
}
```

2.2.3 GlobalUnitStatus

用途：同步基地、前哨站和所有机器人状态

数据编号	数据类型	数据用途
1	uint32	基地当前血量（0=已摧毁）
2	uint32	基地状态
3	uint32	基地当前护盾值（0=无护盾）

数据编号	数据类型	数据用途
4	uint32	前哨站当前血量（0=已摧毁）
5	uint32	前哨站状态
6	repeated uint32	所有机器人血量（先己方后对方）
7	repeated int32	己方机器人剩余累计发弹量
8	uint32	己方累计总伤害
9	uint32	对方累计总伤害

base_status 枚举值：

- 0：无敌
- 1：解除无敌，护甲未展开
- 2：解除无敌，护甲展开

outpost_status 枚举值：

- 0：无敌
- 1：存活，解除无敌，中部装甲旋转
- 2：存活，解除无敌，中部装甲停转
- 3：被击毁，不可重建
- 4：被击毁，可重建

```
message GlobalUnitStatus {
  uint32 base_health = 1;
  uint32 base_status = 2;
  uint32 base_shield = 3;
  uint32 outpost_health = 4;
  uint32 outpost_status = 5;
  repeated uint32 robot_health = 6;
  repeated int32 robot_bullets = 7;
  uint32 total_damage_red = 8;
```



```
  uint32 total_damage_blue = 9;
}
```

2.2.4 GlobalLogisticsStatus

用途：同步全局后勤信息

数据编号	数据类型	数据用途
1	uint32	己方当前经济
2	uint64	己方累计总经济
3	uint32	己方科技等级（工程机器人曾装配最高难度）
4	uint32	己方加密等级（雷达解析信息波机制）

```
message GlobalLogisticsStatus {
  uint32 remaining_economy = 1;
  uint64 total_economy_obtained = 2;
  uint32 tech_level = 3;
  uint32 encryption_level = 4;
}
```

2.2.5 GlobalSpecialMechanism

用途：同步正在生效的全局特殊机制

数据编号	数据类型	数据用途
1	repeated uint32	正在生效的机制 ID 列表
2	repeated int32	对应的时间参数（秒）

mechanism_id 枚举值：

- 1：己方堡垒被对方占领计时
- 2：对方堡垒被己方占领计时

```
message GlobalSpecialMechanism {
  repeated uint32 mechanism_id = 1;
  repeated int32 mechanism_time_sec = 2;
}
```

2.2.6 Event

用途：全局事件通知

数据编号	数据类型	数据用途
1	int32	事件编号
2	string	事件参数（含义随 event_id 变化）

event_id 枚举值：

- 1：击杀事件（参数为击杀者 id+被击毁机器人 id 的拼接）
- 2：基地、前哨站被摧毁事件（参数值为被击毁的目标 id，如蓝方前哨站为 111）
- 3：能量机关可激活次数变化（参数值为变化后可激活次数）
- 4：能量单元当前可进入正在激活状态（无参数值）
- 5：当前能量机关被成功激活的灯臂数量、平均环数（参数值为激活成功臂数+平均环数的
- 6：能量机关被激活（含激活类型）
- 7：己方英雄进入部署模式（无参数值）
- 8：己方英雄造成狙击伤害（参数值为累计造成狙击伤害数量）
- 9：对方英雄造成狙击伤害（参数值为累计造成狙击伤害数量）
- 10：己方呼叫空中支援（无参数值）
- 11：己方空中支援被打断（参数值为对方还剩余的可打断次数）
- 12：对方呼叫空中支援（无参数值）
- 13：对方空中支援被打断（参数值为己方还剩余的可打断次数）

- 14：飞镖命中（参数值为命中目标，1 为击中前哨站，2 为击中基地固定目标，3 为击中基地随机固定目标，4 为击中基地随机移动目标，5 为击中基地末端移动目标）
- 15：双方飞镖闸门开启（参数值 1 为己方开启，2 为对方开启）
- 16：己方基地遭到攻击（无参数值，每次触发存在 5s 内置冷却）
- 17：双方前哨站停转（参数值 1 为己方前哨，2 为对方前哨）
- 18：双方基地护甲展开（参数值 1 为己方基地，2 为对方基地）

```
message Event {
  int32 event_id = 1;
  string param = 2;
}
```

2.2.7 RobotInjuryStat

用途：机器人一次存活期间累计受伤统计

数据编号	数据类型	数据用途
1	uint32	该次存活累计受伤总计
2	uint32	撞击伤害
3	uint32	17mm 弹丸伤害
4	uint32	42mm 弹丸伤害
5	uint32	飞镖溅射伤害
6	uint32	模块离线扣血
7	uint32	WiFi 离线扣血
8	uint32	判罚扣血
9	uint32	服务器强制使其战亡扣血
10	uint32	击杀者 ID（若未检测到击杀者，值为 0）

```
message RobotInjuryStat {
```



```
  uint32 total_damage = 1;
  uint32 collision_damage = 2;
  uint32 small_projectile_damage = 3;
  uint32 large_projectile_damage = 4;
  uint32 dart_splash_damage = 5;
  uint32 module_offline_damage = 6;
  uint32 wifi_offline_damage = 7;
  uint32 penalty_damage = 8;
  uint32 server_kill_damage = 9;
  uint32 killer_id = 10;
}
```

2.2.8 RobotRespawnStatus

用途：机器人复活状态同步

数据编号	数据类型	数据用途
1	bool	是否处于待复活状态
2	uint32	复活所需总读条
3	uint32	当前复活读条进度
4	bool	是否可以免费复活
5	uint32	花费金币复活所需金币数
6	bool	是否允许花费金币复活

```
message RobotRespawnStatus {
  bool is_pending_respawn = 1;
  uint32 total_respawn_progress = 2;
  uint32 current_respawn_progress = 3;
  bool can_free_respawn = 4;
  uint32 gold_cost_for_respawn = 5;
```



```
  bool can_pay_for_respawn = 6;
}
```

2.2.9 RobotStaticStatus

用途：机器人固定属性和配置

数据编号	数据类型	数据用途
1	uint32	连接状态（0=未连接, 1=已连接）
2	uint32	上场状态（0=已上场, 1=未上场）
3	uint32	存活状态（0=未知, 1=存活, 2=战亡）
4	uint32	机器人编号
5	uint32	机器人类型
6	uint32	性能体系-发射机构
7	uint32	性能体系-底盘
8	uint32	当前等级
9	uint32	最大血量
10	uint32	最大热量
11	float	热量冷却速率（每秒）
12	uint32	最大功率
13	uint32	最大缓冲能量
14	uint32	最大底盘能量

performance_system_shooter 枚举值：

- 1：冷却优先
- 2：爆发优先
- 3：英雄近战优先
- 4：英雄远程优先

performance_system_chassis 枚举值：

- 1：血量优先
- 2：功率优先
- 3：英雄近战优先
- 4：英雄远程优先

```
message RobotStaticStatus {
  uint32 connection_state = 1;
  uint32 field_state = 2;
  uint32 alive_state = 3;
  uint32 robot_id = 4;
  uint32 robot_type = 5;
  uint32 performance_system_shooter = 6;
  uint32 performance_system_chassis = 7;
  uint32 level = 8;
  uint32 max_health = 9;
  uint32 max_heat = 10;
  float heat_cooldown_rate = 11;
  uint32 max_power = 12;
  uint32 max_buffer_energy = 13;
  uint32 max_chassis_energy = 14;
}
```

2.2.10 RobotDynamicStatus

用途：机器人实时数据

数据编号	数据类型	数据用途
1	uint32	当前血量

数据编号	数据类型	数据用途
2	float	当前热量
3	float	上一次弹丸射速
4	uint32	当前剩余底盘能量
5	uint32	当前缓冲能量
6	uint32	当前经验值
7	uint32	升级所需经验
8	uint32	累计已发弹量
9	uint32	剩余允许发弹量
10	bool	是否处于脱战状态
11	uint32	脱战状态倒计时
12	bool	是否可以远程补血
13	bool	是否可以远程补弹

```
message RobotDynamicStatus {
  uint32 current_health = 1;
  float current_heat = 2;
  float last_projectile_fire_rate = 3;
  uint32 current_chassis_energy = 4;
  uint32 current_buffer_energy = 5;
  uint32 current_experience = 6;
  uint32 experience_for_upgrade = 7;
  uint32 total_projectiles_fired = 8;
  uint32 remaining_ammo = 9;
  bool is_out_of_combat = 10;
  uint32 out_of_combat_countdown = 11;
  bool can_remote_heal = 12;
```



```
  bool can_remote_ammo = 13;
}
```

2.2.11 RobotModuleStatus

用途：机器人各模块运行状态

数据编号	数据类型	数据用途
1	uint32	电源管理模块状态（0=离线，1=在线）
2	uint32	RFID 模块状态（0=离线，1=在线）
3	uint32	灯条模块状态（0=离线，1=在线）
4	uint32	17mm 发射机构状态（0=离线，1=在线）
5	uint32	42mm 发射机构状态（0=离线，1=在线）
6	uint32	定位模块状态（0=离线，1=在线）
7	uint32	装甲模块状态（0=离线，1=在线）
8	uint32	图传模块状态（0=离线，1=在线）
9	uint32	电容模块状态（0=离线，1=在线）
10	uint32	主控状态（0=离线，1=在线）

```
message RobotModuleStatus {
  uint32 power_manager = 1;
  uint32 rfid = 2;
  uint32 light_strip = 3;
  uint32 small_shooter = 4;
  uint32 big_shooter = 5;
  uint32 uwb = 6;
  uint32 armor = 7;
  uint32 video_transmission = 8;
```

```
    uint32 capacitor = 9;
    uint32 main_controller = 10;
}
```

2.2.12 RobotPosition

用途：机器人空间坐标和朝向

数据编号	数据类型	数据用途
1	float	世界坐标 X 轴
2	float	世界坐标 Y 轴
3	float	世界坐标 Z 轴
4	float	本机器人测速模块的朝向，单位：度，正北为 0 度

```
message RobotPosition {
    float x = 1;
    float y = 2;
    float z = 3;
    float yaw = 4;
}
```

2.2.13 Buff

用途：Buff 效果信息

数据编号	数据类型	数据用途
1	uint32	机器人 ID
2	uint32	Buff 类型
3	int32	Buff 增益值

数据编号	数据类型	数据用途
4	uint32	Buff 最大剩余时间
5	uint32	Buff 剩余时间
6	string	额外文字参数

buff 类型枚举值：

- 1：攻击增益
- 2：防御增益
- 3：射击热量冷却增益
- 4：底盘功率增益
- 5：回血增益
- 6：可兑换允许发弹量
- 7：地形跨越增益（预备）（指诸如检测到飞坡前半部分的场地交互模块，但未获取实际地形跨越增益（飞坡）的情况）

```
message Buff {
  uint32 robot_id = 1;
  uint32 buff_type = 2;
  int32 buff_level = 3;
  uint32 buff_max_time = 4;
  uint32 buff_left_time = 5;
  string msg_params = 6;
}
```

2.2.14 PenaltyInfo

用途：判罚信息同步

数据编号	数据类型	数据用途
1	uint32	当前受罚类型

数据编号	数据类型	数据用途
2	uint32	当前受罚效果时长（秒）
3	uint32	当前判罚数量

penalty_type 枚举值：

- 1：黄牌
- 2：双方黄牌
- 3：红牌
- 4：超功率
- 5：超热量
- 6：超射速

```
message PenaltyInfo {
  uint32 penalty_type = 1;
  uint32 penalty_effect_sec = 2;
  uint32 total_penalty_num = 3;
}
```

2.2.15 RobotPathPlanInfo

用途：哨兵轨迹规划信息

数据编号	数据类型	数据用途
1	uint32	哨兵意图（1=攻击, 2=防守, 3=移动）
2	uint32	起始点 X 坐标（分米）
3	uint32	起始点 Y 坐标（分米）
4	repeated int32	相对起始点 X 增量数组（-128~+127，长度 49）
5	repeated int32	相对起始点 Y 增量数组（-128~+127，长度 49）
6	uint32	发送者 ID


```
message RobotPathPlanInfo {
  uint32 intention = 1;
  uint32 start_pos_x = 2;
  uint32 start_pos_y = 3;
  repeated int32 offset_x = 4 [packed = true];
  repeated int32 offset_y = 5 [packed = true];
  uint32 sender_id = 6;
}
```

2.2.16 MapClickInfoNotify

用途：云台手地图点击标记

数据编号	数据类型	数据用途
1	uint32	发送范围（0=指定客户端, 1=除哨兵, 2=包含哨兵）
2	bytes	目标机器人 ID 列表（固定 7 字节，如发送给己方英雄和工程机器人，红方需要填入 1，2；蓝方需要填入 101，102，后续填 0）
3	uint32	标记类型（1=攻击, 2=防御, 3=警戒, 4=自定义）
4	uint32	标定的对方 ID
5	uint32	自定义图标 ASCII 码
6	uint32	标记模式（1=地图, 2=对方机器人）
7	uint32	屏幕坐标 X（像素）
8	uint32	屏幕坐标 Y（像素）
9	float	地图坐标 X
10	float	地图坐标 Y

```
message MapClickInfoNotify {
  uint32 is_send_all = 1;
  bytes robot_id = 2;
  uint32 mode = 3;
  uint32 enemy_id = 4;
  uint32 ascii = 5;
  uint32 type = 6;
  uint32 screen_x = 7;
  uint32 screen_y = 8;
  float map_x = 9;
  float map_y = 10;
}
```

2.2.17 RaderInfoToClient

用途：雷达发送的机器人位置信息

数据编号	数据类型	数据用途
1	uint32	目标机器人 ID
2	float	目标位置 X（米）
3	float	目标位置 Y（米）
4	float	朝向角度
5	uint32	是否特殊标识（0=否, 1=是）

```
message RaderInfoToClient {
```



```
  uint32 target_robot_id = 1;
  float target_pos_x = 2;
  float target_pos_y = 3;
  float torward_angle = 4;
  uint32 is_high_light = 5;
}
```

2.2.18 CustomByteBlock

用途：自定义数据流

数据编号	数据类型	数据用途
1	bytes	最大为 1.2 kbit 的自定义数据包

```
message CustomByteBlock {
  bytes data = 1;
}
```

2.2.19 AssemblyCommand

用途：工程装配指令

数据编号	数据类型	数据用途
1	uint32	装配操作类型（1=确认装配, 2=取消装配）
2	uint32	选中的装配难度

```
message AssemblyCommand {
  uint32 operation = 1;
  uint32 difficulty = 2;
}
```


2.2.20 TechCoreMotionStateSync

用途：科技核心运动状态同步

数据编号	数据类型	数据用途
1	uint32	当前可选择的最高装配难度等级
2	uint32	科技核心状态

status 枚举值：

- 1：未进入装配状态
- 2：已选择装配难度，科技核心移动中
- 3：科技核心移动完成，可进行首个装配步骤
- 4：上一个装配步骤已完成，可进行下一个装配步骤
- 5：装配步骤已全部完成
- 6：已确认装配，科技核心移动中

```
message TechCoreMotionStateSync {
  uint32 maximum_difficulty_level = 1;
  uint32 status = 2;
}
```

2.2.21 RobotPerformanceSelectionCommand

用途：步兵/英雄选择性能体系

数据编号	数据类型	数据用途
1	uint32	发射机构性能体系
2	uint32	底盘性能体系

```
message RobotPerformanceSelectionCommand {
  uint32 shooter = 1;
```



```
  uint32 chassis = 2;
}
```

2.2.22 RobotPerformanceSelectionSync

用途：步兵/英雄性能体系状态同步

数据编号	数据类型	数据用途
1	uint32	发射机构性能体系
2	uint32	底盘性能体系

```
message RobotPerformanceSelectionSync {
  uint32 shooter = 1;
  uint32 chassis = 2;
}
```

2.2.23 HeroDeployModeEventCommand

用途：英雄部署模式指令

数据编号	数据类型	数据用途
1	uint32	模式（0=退出, 1=进入）

```
message HeroDeployModeEventCommand {
  uint32 mode = 1;
}
```

2.2.24 DeployModeStatusSync

用途：英雄部署模式状态同步

数据编号	数据类型	数据用途
1	uint32	当前部署模式状态（0 为未部署，1 为已部署）

```
message DeployModeStatusSync {
  uint32 status = 1;
}
```

2.2.25 RuneActivateCommand

用途：能量机关激活指令

数据编号	数据类型	数据用途
1	uint32	激活（1=开启）

```
message RuneActivateCommand {
  uint32 activate = 1;
}
```

2.2.26 RuneStatusSync

用途：能量机关状态同步

数据编号	数据类型	数据用途
1	uint32	当前能量机关状态枚举
2	uint32	当前已激活的灯臂数量
3	uint32	总环数

rune_status 枚举值：

- 1：未激活
- 2：正在激活
- 3：已激活

```
message RuneStatusSync {
  uint32 rune_status = 1;
  uint32 activated_arms = 2;
  uint32 average_rings = 3;
}
```

2.2.27 SentinelStatusSync

用途：哨兵姿态和弱化状态

数据编号	数据类型	数据用途
1	uint32	姿态 ID（1 为进攻姿态，2 为防御姿态，3 为移动姿态）
2	bool	是否弱化

```
message SentinelStatusSync {
  uint32 posture_id = 1;
  bool is_weakened = 2;
}
```

2.2.28 DartCommand

用途：飞镖控制指令

数据编号	数据类型	数据用途
1	uint32	目标 ID（1 为前哨站，2 为基地固定目标，3 为基地随机固定目标，4 为基地随机移动目标，5 为基地末端移动目标）
2	bool	闸门开关

```
message DartCommand {
  uint32 target_id = 1;
  bool open = 2;
}
```

2.2.29 DartSelectTargetStatusSync

用途：飞镖目标选择状态同步

数据编号	数据类型	数据用途
1	uint32	目标 ID
2	bool	闸门状态

```
message DartSelectTargetStatusSync {
  uint32 target_id = 1;
  bool open = 2;
}
```

2.2.30 GuardCtrlCommand

用途：哨兵控制指令请求

数据编号	数据类型	数据用途
1	uint32	指令编号（0=无效）

指令编号枚举：

- 1：补血点补弹
- 2：补给站实体补弹
- 3：远程补弹
- 4：远程回血
- 5：确认复活
- 6：确认花费金币复活
- 7：地图标点
- 8：切换为进攻姿态
- 9：切换为防御姿态
- 10：切换为移动姿态

```
message GuardCtrlCommand {
  uint32 command_id = 1;
}
```

2.2.31 GuardCtrlResult

用途：哨兵控制指令结果反馈

数据编号	数据类型	数据用途
1	uint32	对应的指令编号
2	uint32	执行结果码

result_code 枚举值：

- 0：成功
- 其他：失败

```
message GuardCtrlResult {
  uint32 command_id = 1;
  uint32 result_code = 2;
}
```

2.2.32 AirSupportCommand

用途：空中支援指令

数据编号	数据类型	数据用途
1	uint32	指令类型

command_id 枚举值：

- 1：免费呼叫空中支援
- 2：花费金币呼叫空中支援（仍优先使用免费时长）
- 3：中断空中支援

```
message AirSupportCommand {
  uint32 command_id = 1;
}
```

2.2.33 AirSupportStatusSync

用途：空中支援状态反馈

数据编号	数据类型	数据用途
1	uint32	空中支援状态

数据编号	数据类型	数据用途
2	uint32	免费空中支援剩余时间
3	uint32	付费空中支援已花费金币

airsupport_status 枚举值:

- 0：未进行空中支援
- 1：正在空中支援
- 2：空中支援被对方锁定而无法开启

```
message AirSupportStatusSync {
  uint32 airsupport_status = 1;
  uint32 left_time = 2;
  uint32 cost_coins = 3;
}
```


附录一：CRC 校验代码示例

```
//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);
```

附录二：ID 编号说明

机器人 ID 编号如下所示：

- 1：红方英雄机器人
- 2：红方工程机器人
- 3/4/5：红方步兵机器人（与机器人 ID 3~5 对应）
- 6：红方空中机器人
- 7：红方哨兵机器人
- 8：红方飞镖
- 9：红方雷达
- 10：红方前哨站
- 11：红方基地
- 101：蓝方英雄机器人
- 102：蓝方工程机器人
- 103/104/105：蓝方步兵机器人（与机器人 ID 3~5 对应）
- 106：蓝方空中机器人
- 107：蓝方哨兵机器人
- 108：蓝方飞镖
- 109：蓝方雷达
- 110：蓝方前哨站
- 111：蓝方基地

选手端 ID 如下所示：

- 0x0101：红方英雄机器人选手端
- 0x0102：红方工程机器人选手端
- 0x0103/0x0104/0x0105：红方步兵机器人选手端（与机器人 ID 3~5 对应）
- 0x0106：红方空中机器人选手端
- 0x016A：蓝方空中机器人选手端
- 0x0165：蓝方英雄机器人选手端

- 0x0166：蓝方工程机器人选手端
- 0x0167/0x0168/0x0169：蓝方步兵机器人选手端（与机器人 ID 3~5 对应）
- 0x8080：裁判系统服务器（用于哨兵和雷达自主决策指令）

附录三：自定义客户端示例通信代码

```
using MQTTnet;
using MQTTnet.Client;
using MQTTnet.Diagnostics;
using MQTTnet.Protocol;
using MQTTnet.Server;
using System;
using System.Collections;
using System.Collections.Generic;
using System.Text;
using System.Threading.Tasks;
using UnityEngine;

/// <summary>
/// MQTT 客户端
/// </summary>
public class MyMqttClient
{
    private IMqttClient m_MqttClient = null;
    private string m_ClientID;

    /// <summary>
    /// 创建客户端并连接服务器
    /// </summary>
    /// <param name="clientID"></param>
    /// <param name="ip"></param>
    /// <param name="port"></param>
    public MyMqttClient(string clientID, string ip = "127.0.0.1", int port = 3333)
    {
        m_ClientID = clientID;
        //客户端选项生成器
        var options = new MqttClientOptionsBuilder()
        .WithClientId(m_ClientID)
        .WithTcpServer(ip, port)
```



```
        .Build();
        //创建客户端
        m_MqttClient = new MqttFactory().CreateMqttClient();
        //监测客户端 连接/断开连接 完成
        m_MqttClient.ConnectedAsync += ClientConnected;
        m_MqttClient.DisconnectedAsync += ClientDisConnected;
        //客户端接收到消息
        m_MqttClient.ApplicationMessageReceivedAsync += ReceiveMsg;
        //连接服务器
        m_MqttClient.ConnectAsync(options);
    }

    /// <summary>
    /// 接收到消息
    /// </summary>
    /// <param name="args"></param>
    /// <returns></returns>
    private Task ReceiveMsg(MqttApplicationMessageReceivedEventArgs args)
    {
                        Debuger.Log(DebugLogEnum.CustomClient, $"recv topic:
{args.ApplicationMessage.Topic}");
        //消息 post 出现，以免出现消息堵塞
        MqttSubProtoEvent ev = new MqttSubProtoEvent();
        ev.topic = args.ApplicationMessage.Topic;
        ev.data = args.ApplicationMessage.Payload;
        ThreadPool.CustomClientLoop.PostEvent(ev);
        return Task.CompletedTask;
    }

    /// <summary>
    /// 断开连接完成
    /// </summary>
    /// <param name="args"></param>
    /// <returns></returns>
    private Task ClientDisConnected(MqttClientDisconnectedEventArgs args)
```



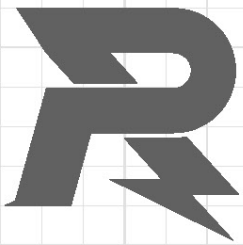
```
    {
        return Task.CompletedTask;
    }

    /// <summary>
    /// 连接完成
    /// </summary>
    /// <param name="args"></param>
    /// <returns></returns>ReceiveMsg
    private Task ClientConnected(MqttClientConnectedEventArgs args)
    {
       Subscribe("RemoteControl");
        Subscribe("CustomRobotData");
        return Task.CompletedTask;
    }

    /// <summary>
    /// 发布消息
    /// </summary>
     public void PublishMsg(string topic, string message, MqttQualityOfServiceLevel level = MqttQualityOfServiceLevel.ExactlyOnce, bool isRetain = false)
    {
        long unixUs = DateTime.UtcNow.Ticks / 10; // 微秒级
        m_MqttClient.PublishStringAsync(topic, message, level, isRetain);
    }
    /// <summary>
    /// 发布消息
    /// </summary>
    public void PublishBytesMsg(string topic, byte[] message, MqttQualityOfServiceLevel level = MqttQualityOfServiceLevel.ExactlyOnce, bool isRetain = false)
    {
        long unixUs = DateTime.UtcNow.Ticks / 10; // 微秒级
        m_MqttClient.PublishBinaryAsync(topic, message, level, isRetain);
    }
```



```
    /// <summary>
    /// 订阅主题
    /// </summary>
    public void Subscribe(string topic)
    {
        m_MqttClient.SubscribeAsync(new MqttTopicFilterBuilder().WithTopic(topic).Build());
    }
}
```



邮箱：robomaster@dji.com
论坛：https://bbs.robomaster.com
官网：https://www.robomaster.com
电话：+86 0755-84357613（周一至周五10:30-19:30）
地址：广东省深圳市南山区西丽街道仙茶路与兴科路交叉口大疆天空之城T2 22F