V1.6.1

裁判系统串口协议附录

RoboMaster 组委会 编制

2024 年 1 月 发布

修改日志

日期	版本	修订记录
2024.01.22	V1.6.1	1. 修订命令码 0x0101、0x0201、0x0203、0x0209、0x020A、0x0303 2. 修订已知的数据段长度不一致或结构体定义错误 3. 完善部分描述
2023.11.22	V1.6	1. 修改了命令字 0x0001、0x0101、0x0105、0x0203、0x0205、0x020A 的发送频率 2. 修订命令码 0x0101、0x0102、0x0104、0x0105、0x0201、0x0203、0x0209、0x0301、0x0303、0x0307 3. 新增命令码 0x020D、0x020E、0x0308 4. 完善部分描述
2023.07.07	V1.5	1. 补充全部命令字的发送/触发条件和发送方/接收方的说明 2. 补充裁判系统常规链路和图传链路的说明 3. 修订命令码 0x0101、0x0204、0x0205、0x0208、0x0209 4. 新增命令码 0x020B、0X020C、0x0307、0x0306 5. 删除命令码 0x0004、0x0005、0x0103 6. 完善部分描述
2022.08.05	V1.4	修订命令码 0x0001、0x0003、0x0101、0x0105、0x0204、0x0208、0x0209 的发送频率和触发方式
2021.12.31	V1.3	修订人工智能挑战赛加成\惩罚区分布与潜伏模式状态：0x0005
2021.04.30	V1.2	修订部分错误
2021.04.19	V1.1	修订部分错误
2021.02.03	V1.0	首次发布

目录

1. 串口协议格式

通信方式为串口，配置为：波特率 115200，8 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

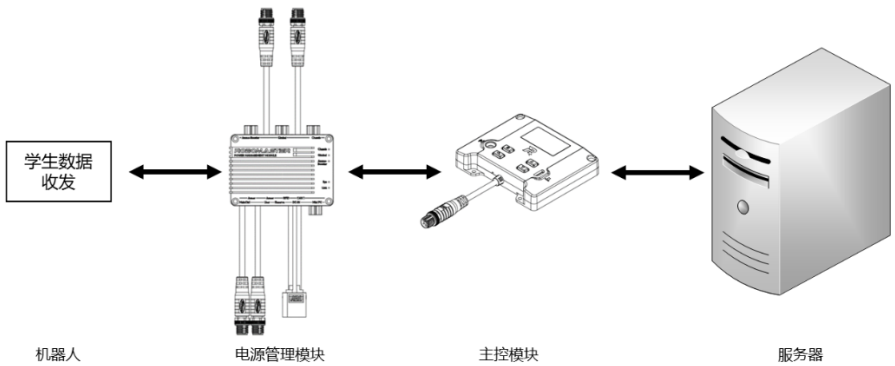
SOF	data_length	seq	CRC8
1-byte	2-byte	1-byte	1-byte

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

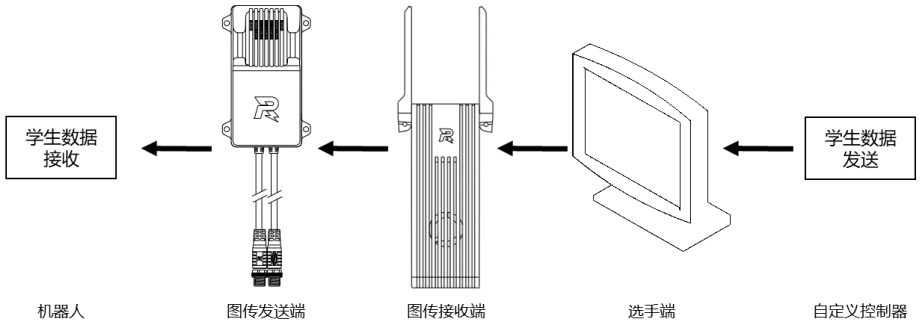
裁判系统串口数据链路有两种：常规链路、图传链路。

- 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：



- 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。

2. 命令码 ID 和常规链路数据说明

表 2-1 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	32	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
0x0102	4	补给站动作标识数据，补给站弹丸释放时触发发送	服务器→己方全体机器人	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
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	16	实时底盘功率和枪口热量数据，固定以 50Hz 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	6	机器人增益数据，固定以 3Hz 频率发送	服务器→对应机器人	常规链路
0x0205	2	空中支援时间数据，固定以 1Hz 频率发送	服务器→己方空中机器人	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
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	4	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

命令码	数据段长度	说明	发送方/接收方	所属数据链路
0x020C	6	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	4	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	128	机器人交互数据，发送方触发发送，频率上限为 10Hz	-	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
0x0305	10	选手端小地图接收雷达数据，频率上限为 10Hz	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	选手端小地图接收哨兵数据，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

表 2-2 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
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 2-3 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 2-4 0x0003

字节偏移量	大小	说明
0	2	红 1 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
2	2	红 2 工程机器人血量
4	2	红 3 步兵机器人血量
6	2	红 4 步兵机器人血量
8	2	红 5 步兵机器人血量
10	2	红 7 哨兵机器人血量
12	2	红方前哨站血量
14	2	红方基地血量
16	2	蓝 1 英雄机器人血量
18	2	蓝 2 工程机器人血量
20	2	蓝 3 步兵机器人血量
22	2	蓝 4 步兵机器人血量
24	2	蓝 5 步兵机器人血量
26	2	蓝 7 哨兵机器人血量
28	2	蓝方前哨站血量
30	2	蓝方基地血量

```
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;
```

表 2-5 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：己方补给站前补血点的占领状态，1 为已占领 ● bit 1：己方补给站内部补血点的占领状态，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 3-5：己方能量机关状态 ● bit 3：己方能量机关激活点的占领状态，1 为已占领 ● bit 4：己方小能量机关的激活状态，1 为已激活 ● bit 5：己方大能量机关的激活状态，1 为已激活 bit 6-11：己方高地占领状态 ● bit 6-7：己方环形高地的占领状态，1 为被己方占领，2 为被对方占领 ● bit 8-9：己方梯形高地的占领状态，1 为被己方占领，2 为被对方占领 ● bit 10-11：己方梯形高地的占领状态，1 为被己方占领，2 为被对方占领 bit 12-18：己方基地虚拟护盾的剩余值百分比（四舍五入，保留整数） bit 19-27：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）

字节偏移量	大小	说明
		bit 28-29：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 为击中基地随机目标 bit 30-31：中心增益点的占领情况，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）

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

表 2-6 0x0102

字节偏移量	大小	说明
0	1	保留
1	1	补弹机器人 ID： ● 0：当前无机器人补弹 ● 1：红方英雄机器人补弹 ● 3/4/5：红方步兵机器人补弹 ● 101：蓝方英雄机器人补弹 ● 103/104/105：蓝方步兵机器人补弹
2	1	出弹口开闭状态： ● 0：关闭 ● 1：弹丸准备中 ● 2：弹丸释放
3	1	补弹数量： ● 50：50 颗弹丸 ● 100：100 颗弹丸 ● 150：150 颗弹丸 ● 200：200 颗弹丸

```
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;
```

表 2-7 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 ● 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 2-8 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 0-1： 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 为击中基地随机目标 bit 2-4： 对方最近被击中的目标累计被击中计数，开局默认为 0，至多为 4 bit 5-6：

字节偏移量	大小	说明
		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，选中基地随机目标为 2 bit 7-15：保留

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

表 2-9 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	机器人枪口热量每秒冷却值
8	2	机器人枪口热量上限
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 ● bit 3-7：保留

```
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 2-10 0x0202

字节偏移量	大小	说明
0	2	电源管理模块的 chassis 口输出电压（单位：mV）
2	2	电源管理模块的 chassis 口输出电流（单位：mA）
4	4	底盘功率（单位：W）
8	2	缓冲能量（单位：J）
10	2	第 1 个 17mm 发射机构的枪口热量
12	2	第 2 个 17mm 发射机构的枪口热量
14	2	42mm 发射机构的枪口热量

```
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;
```

表 2-11 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
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 2-12 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	1	机器人枪口冷却倍率（直接值，值为 5 表示 5 倍冷却）
2	1	机器人防御增益（百分比，值为 50 表示 50%防御增益）
3	1	机器人负防御增益（百分比，值为 30 表示-30%防御增益）
4	2	机器人攻击增益（百分比，值为 50 表示 50%攻击增益）

```
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;
```

表 2-13 0x0205

字节偏移量	大小	说明
0	1	空中机器人状态（0 为正在冷却，1 为冷却完毕，2 为正在空中支援）
1	1	此状态的剩余时间（单位为：秒，向下取整，即冷却时间剩余 1.9 秒时，此值为 1） 若冷却时间为 0，但未呼叫空中支援，则该值为 0

```
typedef _packed struct
{
   uint8_t airforce_status;
   uint8_t time_remain;
}air_support_data_t;
```

表 2-14 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、离线或测速模块离线时，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

字节偏移量	大小	说明
		● 0：装甲模块被弹丸攻击导致扣血 ● 1：裁判系统重要模块离线导致扣血 ● 2：射击初速度超限导致扣血 ● 3：枪口热量超限导致扣血 ● 4：底盘功率超限导致扣血 ● 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;
```

表 2-15 0x0207

字节偏移量	大小	说明
0	1	弹丸类型： ● 1：17mm 弹丸 ● 2：42mm 弹丸
1	1	发射机构 ID： ● 1：第 1 个 17mm 发射机构 ● 2：第 2 个 17mm 发射机构 ● 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 2-16 0x0208

字节偏移量	大小	说明
0	2	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

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

表 2-17 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：己方环形高地增益点 ● bit 2：对方环形高地增益点 ● bit 3：己方 R3/B3 梯形高地增益点 ● bit 4：对方 R3/B3 梯形高地增益点 ● bit 5：己方 R4/B4 梯形高地增益点 ● bit 6：对方 R4/B4 梯形高地增益点 ● bit 7：己方能量机关激活点 ● bit 8：己方飞坡增益点（靠近己方一侧飞坡前） ● bit 9：己方飞坡增益点（靠近己方一侧飞坡后） ● bit 10：对方飞坡增益点（靠近对方一侧飞坡前） ● bit 11：对方飞坡增益点（靠近对方一侧飞坡后） ● bit 12：己方前哨站增益点 ● bit 13：己方补血点（检测到任一均视为激活） ● bit 14：己方哨兵巡逻区

字节偏移量	大小	说明
		● bit 15：对方哨兵巡逻区 ● bit 16：己方大资源岛增益点 ● bit 17：对方大资源岛增益点 ● bit 18：己方兑换区 ● bit 19：中心增益点（仅 RMUL 适用） ● bit 20-31：保留 注：基地增益点、高地增益点、飞坡增益点、前哨站增益点、资源岛增益点、补血点、兑换区、中心增益点（仅适用于 RMUL）和哨兵巡逻区的 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。

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

表 2-18 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 ● 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 2-19 0x020B

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。

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	己方 5 号步兵机器人位置 x 轴坐标，单位：m
36	4	己方 5 号步兵机器人位置 y 轴坐标，单位：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;
```


表 2-20 0x020C

字节偏移量	大小	说明
0	1	对方英雄机器人被标记进度：0-120
1	1	对方工程机器人被标记进度：0-120
2	1	对方 3 号步兵机器人被标记进度：0-120
3	1	对方 4 号步兵机器人被标记进度：0-120
4	1	对方 5 号步兵机器人被标记进度：0-120
5	1	对方哨兵机器人被标记进度：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;
```

表 2-21 0x020D

字节偏移量	大小	说明
0	4	bit 0-10：除远程兑换外，哨兵成功兑换的发弹量，开局为 0，在哨兵成功兑换一定发弹量后，该值将变为哨兵成功兑换的发弹量值。 bit 11-14：哨兵成功远程兑换发弹量的次数，开局为 0，在哨兵成功远程兑换发弹量后，该值将变为哨兵成功远程兑换发弹量的次数。 bit 15-18：哨兵成功远程兑换血量的次数，开局为 0，在哨兵成功远程兑换血量后，该值将变为哨兵成功远程兑换血量的次数。 bit 19-31：保留

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


表 2-22 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 3-7：保留

```
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 128 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 113 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。

表 2-23 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 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;
```

子内容 ID	内容数据段长度	功能说明
0x0200~0x02FF	x≤113	机器人之间通信
0x0100	2	选手端删除图层
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

由于存在多个内容 ID，但整个 cmd_id 上行频率最大为 10Hz，请合理安排带宽。

表 2-24 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 2-25 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 ● 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

字节偏移量	大小	说明	备注
			● 7：黑色 ● 8：白色 bit 14-31：根据绘制的图形不同，含义不同，详见“表 2-26 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 2-26 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 2-26 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

类型	details_a	details_b	details_c	details_d	details_e
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 2-27 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 2-28 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 2-29 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
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 2-30 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 2-31 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。

字节偏移量	大小	说明	备注
			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。 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 21-31：保留

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

表 2-32 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。

```
typedef _packed struct
{
  uint8_t radar_cmd;
} radar_cmd_t;
```


3. 小地图交互数据

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。

- 选择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。

一台选择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。

表 3-1 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

字节偏移量	大小	说明	备注
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;
  uint8_t cmd_source;
}map_command_t;
```

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 3-2 命令码 ID：0x0305

字节偏移量	大小	说明	备注
0	2	目标机器人 ID	-
2	4	目标 x 位置坐标，单位：m	当 x、y 超出边界时则不显示
6	4	目标 y 位置坐标，单位：m	当 x、y 超出边界时则不显示

```
typedef _packed struct
{
  uint16_t target_robot_id;
  float target_position_x;
  float target_position_y;
}map_robot_data_t;
```

哨兵机器人或选择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 3-3 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	-

字节偏移量	大小	说明	备注
		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 3-4 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;
```


4. 图传链路数据说明

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 4-1 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;
```

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 4-2 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

字节偏移量	大小	说明
		● 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 ● 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;
```


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 5-1 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;
```

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

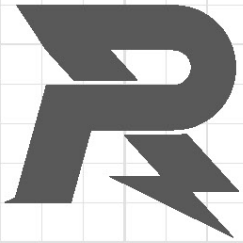
机器人 ID 编号如下所示：

- 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）



邮箱：robomaster@dji.com
论坛：http://bbs.robomaster.com
官网：http://www.robomaster.com
电话：0755-36383255（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