

V1.3



裁判系统学生串口协议附录

RoboMaster 组委会 编制

2021 年 12 月 发布

修改日志

日期	版本	改动记录
2021.12.31	V1.3	修改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

修改日志	2
1. 串口配置	4
2. 接口协议说明	4
3. 详细说明	6
4. 机器人间交互数据	18
5. ID 说明	19
6. 自定义控制器交互数据	26
7. 小地图交互信息	26
客户端下发信息	26
客户端接收信息	27
8. 图传遥控信息	27
客户端下发信息	28
9. CRC 校验代码示例	30

1. 串口配置

通信方式是串口，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。

2. 接口协议说明

表 2-1 通信协议格式

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

表 2-2 frame_header 格式

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

表 2-3 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

表 2-4 cmd_id 命令码 ID 说明

命令码	数据段长度	功能说明
0x0001	11	比赛状态数据，1Hz 周期发送
0x0002	1	比赛结果数据，比赛结束后发送
0x0003	28	比赛机器人血量数据，1Hz 周期发送
0x0004	3	飞镖发射状态，飞镖发射后发送
0x0005	11	人工智能挑战赛加成与惩罚状态，1Hz 周期发送
0x0101	4	场地事件数据，1Hz 周期发送

命令码	数据段长度	功能说明
0x0102	3	场地补给站动作标识数据，动作改变后发送
0x0103	2	请求补给站补弹数据，由参赛队发送，上限 10Hz。（RM 对抗赛尚未开放）
0x0104	2	裁判警告数据，警告发生后发送
0x0105	1	飞镖发射口倒计时，1Hz 周期发送
0x0201	15	机器人状态数据，10Hz 周期发送
0x0202	14	实时功率热量数据，50Hz 周期发送
0x0203	16	机器人位置数据，10Hz 发送
0x0204	1	机器人增益数据，增益状态改变后发送
0x0205	3	空中机器人能量状态数据，10Hz 周期发送，只有空中机器人主控发送
0x0206	1	伤害状态数据，伤害发生后发送
0x0207	6	实时射击数据，子弹发射后发送
0x0208	2	子弹剩余发送数，空中机器人以及哨兵机器人发送，1Hz 周期发送
0x0209	4	机器人 RFID 状态，1Hz 周期发送
0x020A	12	飞镖机器人客户端指令书，10Hz 周期发送
0x0301	n	机器人间交互数据，发送方触发发送，上限 10Hz
0x0302	n	自定义控制器交互数据接口，通过客户端触发发送，上限 30Hz
0x0303	15	客户端小地图交互数据，触发发送
0x0304	12	键盘、鼠标信息，通过图传串口发送
0x0305	10	客户端小地图接收信息

3. 详细说明

表 3-1 比赛状态数据：0x0001。发送频率：1Hz

字节偏移量	大小	说明
0	1	0-3 bit: 比赛类型 <ul style="list-style-type: none"> • 1: RoboMaster 机甲大师赛; • 2: RoboMaster 机甲大师单项赛; • 3: ICRA RoboMaster 人工智能挑战赛 • 4: RoboMaster 联盟赛 3V3 • 5: RoboMaster 联盟赛 1V1 4-7 bit: 当前比赛阶段 <ul style="list-style-type: none"> • 0: 未开始比赛; • 1: 准备阶段; • 2: 自检阶段; • 3: 5s 倒计时; • 4: 对战中; • 5: 比赛结算中
1	2	当前阶段剩余时间, 单位 s
3	8	机器人接收到该指令的精确 Unix 时间, 当机载端收到有效的 NTP 服务器授时后生效

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

表 3-2 比赛结果数据：0x0002。发送频率：比赛结束后发送

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;
} ext_game_result_t;
```

表 3-3 机器人血量数据：0x0003。发送频率：1Hz

字节偏移量	大小	说明
0	2	红 1 英雄机器人血量，未上场以及罚下血量为 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	红方前哨战血量
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

字节偏移量	大小	说明
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;
} ext_game_robot_HP_t;
```

表 3-4 人工智能挑战赛加成\惩罚区分布与潜伏模式状态：0x0005。发送频率：1Hz 周期发送，发送范围：所有机器人

字节偏移量	大小	说明
0	3	bit[0, 4, 8, 12, 16, 20]为 F1-F6 激活状态： 0 为未激活， 1 为可激活， bit[1-3, 5-7, 9-11, 13-15, 17-19, 21-23]为 F1-F1 的状态信息： 1 为红方回血区； 2 为红方弹药补给区； 3 为蓝方回血区； 4 为蓝方弹药补给区； 5 为禁止射击区； 6 为禁止移动区。
3	2	红方 1 号剩余弹量

字节偏移量	大小	说明
5	2	红方 2 号剩余弹量
7	2	蓝方 1 号剩余弹量
9	2	蓝方 2 号剩余弹量
11	1	0: 正常阶段; 1: 准备进入潜伏阶段; 2: 潜伏阶段
12	1	保留字节

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

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

表 3-5 场地事件数据：0x0101。发送频率：1Hz

字节偏移量	大小	说明
0	4	bit 0-2: bit 0: 己方补给站 1 号补血点占领状态 1 为已占领; bit 1: 己方补给站 2 号补血点占领状态 1 为已占领;

字节偏移量	大小	说明
		bit 2: 己方补给站 3 号补血点占领状态 1 为已占领; bit 3-5: 己方能量机关状态: <ul style="list-style-type: none"> • bit 3 为打击点占领状态, 1 为占领; • bit 4 为小能量机关激活状态, 1 为已激活; • bit 5 为大能量机关激活状态, 1 为已激活; bit 6: 己方侧 R2/B2 环形高地占领状态 1 为已占领; bit 7: 己方侧 R3/B3 梯形高地占领状态 1 为已占领; bit 8: 己方侧 R4/B4 梯形高地占领状态 1 为已占领; bit 9: 己方基地护盾状态: <ul style="list-style-type: none"> • 1 为基地有虚拟护盾血量; • 0 为基地无虚拟护盾血量; bit 10: 己方前哨战状态: <ul style="list-style-type: none"> • 1 为前哨战存活; • 0 为前哨战被击毁; bit 10 -31: 保留

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

表 3-6 补给站动作标识: 0x0102。发送频率: 动作改变后发送, 发送范围: 己方机器人

字节偏移量	大小	说明
0	1	补给站口 ID: 1: 1 号补给口; 2: 2 号补给口

字节偏移量	大小	说明
1	1	补弹机器人 ID: 0 为当前无机器人补弹, 1 为红方英雄机器人补弹, 2 为红方工程机器人补弹, 3/4/5 为红方步兵机器人补弹, 101 为蓝方英雄机器人补弹, 102 为蓝方工程机器人补弹, 103/104/105 为蓝方步兵机器人补弹
2	1	出弹口开闭状态: 0 为关闭, 1 为子弹准备中, 2 为子弹下落
3	1	补弹数量: 50: 50 颗子弹; 100: 100 颗子弹; 150: 150 颗子弹; 200: 200 颗子弹。

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

表 3-7 裁判警告信息: cmd_id (0x0104)。发送频率: 己方警告发生后发送

字节偏移量	大小	说明
0	1	警告等级: 1: 黄牌 2: 红牌 3: 判负
1	1	犯规机器人 ID: 判负时, 机器人 ID 为 0 黄牌、红牌时, 机器人 ID 为犯规机器人 ID

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

表 3-8 飞镖发射口倒计时: cmd_id (0x0105)。发送频率: 1Hz 周期发送, 发送范围: 己方机器人

字节偏移量	大小	说明
0	1	15s 倒计时

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

表 3-9 比赛机器人状态: 0x0201。发送频率: 10Hz

字节偏移量	大小	说明
0	1	本机器人 ID: 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; 6: 红方空中机器人; 7: 红方哨兵机器人; 8: 红方飞镖机器人; 9: 红方雷达站; 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; 106: 蓝方空中机器人; 107: 蓝方哨兵机器人; 108: 蓝方飞镖机器人; 109: 蓝方雷达站。
1	1	机器人等级: 1: 一级; 2: 二级; 3: 三级。
2	2	机器人剩余血量

字节偏移量	大小	说明
4	2	机器人上限血量
6	2	机器人 1 号 17mm 枪口每秒冷却值
8	2	机器人 1 号 17mm 枪口热量上限
10	2	机器人 1 号 17mm 枪口上限速度 单位 m/s
12	2	机器人 2 号 17mm 枪口每秒冷却值
14	2	机器人 2 号 17mm 枪口热量上限
16	2	机器人 2 号 17mm 枪口上限速度 单位 m/s
18	2	机器人 42mm 枪口每秒冷却值
20	2	机器人 42mm 枪口热量上限
22	2	机器人 42mm 枪口上限速度 单位 m/s
24	2	机器人底盘功率限制上限
26	1	主控电源输出情况： 0 bit: gimbal 口输出： 1 为有 24V 输出， 0 为无 24v 输出； 1 bit: chassis 口输出： 1 为有 24V 输出， 0 为无 24v 输出； 2 bit: shooter 口输出： 1 为有 24V 输出， 0 为无 24v 输出；

```

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

    uint16_t shooter_id2_17mm_cooling_rate;
    uint16_t shooter_id2_17mm_cooling_limit;
    uint16_t shooter_id2_17mm_speed_limit;

```

```
uint16_t shooter_id1_42mm_cooling_rate;
uint16_t shooter_id1_42mm_cooling_limit;
uint16_t shooter_id1_42mm_speed_limit;

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

表 3-10 实时功率热量数据：0x0202。发送频率：50Hz

字节偏移量	大小	说明
0	2	底盘输出电压 单位 毫伏
2	2	底盘输出电流 单位 毫安
4	4	底盘输出功率 单位 W 瓦
8	2	底盘功率缓冲 单位 J 焦耳 备注：飞坡根据规则增加至 250J
10	2	1 号 17mm 枪口热量
12	2	2 号 17mm 枪口热量
14	2	42mm 枪口热量

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

表 3-11 机器人位置：0x0203。发送频率：10Hz

字节偏移量	大小	说明
0	4	位置 x 坐标，单位 m
4	4	位置 y 坐标，单位 m

字节偏移量	大小	说明
8	4	位置 z 坐标, 单位 m
12	4	位置枪口, 单位度

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

表 3-12 机器人增益: 0x0204。发送频率: 1Hz

字节偏移量	大小	说明
0	1	bit 0: 机器人血量补血状态 bit 1: 枪口热量冷却加速 bit 2: 机器人防御加成 bit 3: 机器人攻击加成 其他 bit 保留

```
typedef __packed struct
{
    uint8_t power_rune_buff;
} ext_buff_t;
```

表 3-13 空中机器人能量状态: 0x0205。发送频率: 10Hz

字节偏移量	大小	说明
0	1	可攻击时间 单位 s。30s 递减至 0

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

表 3-14 伤害状态: 0x0206。发送频率: 伤害发生后发送

字节偏移量	大小	说明
0	1	bit 0-3: 当血量变化类型为装甲伤害, 代表装甲 ID, 其中数值为 0-4 号代表机器人的五个装甲片, 其他血量变化类型, 该变量数值为 0。 bit 4-7: 血量变化类型

字节偏移量	大小	说明
		0x0 装甲伤害扣血; 0x1 模块掉线扣血; 0x2 超射速扣血; 0x3 超枪口热量扣血; 0x4 超底盘功率扣血; 0x5 装甲撞击扣血

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

表 3-15 实时射击信息：0x0207。发送频率：射击后发送

字节偏移量	大小	说明
0	1	子弹类型: 1: 17mm 弹丸 2: 42mm 弹丸
1	1	发射机构 ID: 1: 1 号 17mm 发射机构 2: 2 号 17mm 发射机构 3: 42mm 发射机构
2	1	子弹射频 单位 Hz
3	4	子弹射速 单位 m/s

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

表 3-16 子弹剩余发射数：0x0208。发送频率：10Hz 周期发送，所有机器人发送

17mm 子弹剩余发射数量 含义说明	联盟赛	对抗赛
-----------------------	-----	-----

步兵机器人	全队步兵与英雄剩余可发射 17mm 弹丸总量	全队 17mm 弹丸剩余可兑换数量
英雄机器人	全队步兵与英雄剩余可发射 17mm 弹丸总量	全队 17mm 弹丸剩余可兑换数量
空中机器人、哨兵机器人	该机器人剩余可发射 17mm 弹丸总量	该机器人剩余可发射 17mm 弹丸总量

字节偏移量	大小	说明
0	2	17mm 子弹剩余发射数目
2	2	42mm 子弹剩余发射数目
4	2	剩余金币数量

```
typedef __packed struct
{
    uint16_t bullet_remaining_num_17mm;
    uint16_t bullet_remaining_num_42mm;
    uint16_t coin_remaining_num;
} ext_bullet_remaining_t;
```

表 3-17 机器人 RFID 状态：0x0209。发送频率：1Hz，发送范围：单一机器人

字节偏移量	大小	说明
0	4	bit 0: 基地增益点 RFID 状态; bit 1: 高地增益点 RFID 状态; bit 2: 能量机关激活点 RFID 状态; bit 3: 飞坡增益点 RFID 状态; bit 4: 前哨岗增益点 RFID 状态; bit 6: 补血点增益点 RFID 状态; bit 7: 工程机器人复活卡 RFID 状态; bit 8-31: 保留

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

表 3-18 飞镖机器人客户端指令数据：0x020A。发送频率：10Hz，发送范围：单一机器人

字节偏移量	大小	说明
0	1	当前飞镖发射口的状态 1: 关闭; 2: 正在开启或者关闭中 0: 已经开启
1	1	飞镖的打击目标, 默认为前哨站; 0: 前哨站; 1: 基地。
2	2	切换打击目标时的比赛剩余时间, 单位秒, 从未切换默认为 0。
4	2	最近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 dart_attack_target;
    uint16_t target_change_time;
    uint16_t operate_launch_cmd_time;
} ext_dart_client_cmd_t;
```

4. 机器人间交互数据

交互数据包括一个统一的数据段头结构。数据段包含了内容 ID，发送者以及接收者的 ID 和内容数据段，整个交互数据的包总共长最大为 128 个字节，减去 `frame_header`, `cmd_id` 和 `frame_tail` 共 9 个字节以及数据段头结构的 6 个字节，故而发送的内容数据段最大为 113。整个交互数据 0x0301 的字节限制如下表所示，其中数据量包括 `frame-header`, `cmd_id`, `frame_tail` 以及数据段头结构的字节数量。

每个机器人交互数据与自定义控制器数据上下行合计带宽不超过 5000 Byte。上下行发送频率分别不超过 30Hz。

表 4-1 交互数据接收信息：0x0301

字节偏移量	大小	说明	备注
0	2	数据段的内容 ID	-
2	2	发送者的 ID	需要校验发送者的 ID 正确性，例如红 1 发送给红 5，此项需要校验红 1
4	2	接收者的 ID	需要校验接收者的 ID 正确性，例如不能发送到敌对机器人的 ID
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;
}ext_student_interactive_header_data_t;
```

内容 ID	长度 (头结构长度+内容数据段长度)	功能说明
0x0200~0x02FF	6+n	己方机器人间通信
0x0100	6+2	客户端删除图形
0x0101	6+15	客户端绘制一个图形
0x0102	6+30	客户端绘制二个图形
0x0103	6+75	客户端绘制五个图形
0x0104	6+105	客户端绘制七个图形
0x0110	6+45	客户端绘制字符图形

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。

5. ID 说明

机器人 ID：1，英雄(红)；2，工程(红)；3/4/5，步兵(红)；6，空中(红)；7，哨兵(红)；9，雷达站（红）；101，英雄(蓝)；102，工程(蓝)；103/104/105，步兵(蓝)；106，空中(蓝)；107，哨兵(蓝)；109，雷达站（蓝）。

客户端 ID: 0x0101 为英雄操作手客户端(红); 0x0102, 工程操作手客户端((红); 0x0103/0x0104/0x0105, 步兵操作手客户端(红); 0x0106, 空中操作手客户端((红); 0x0165, 英雄操作手客户端(蓝); 0x0166, 工程操作手客户端(蓝); 0x0167/0x0168/0x0169, 步兵操作手客户端步兵(蓝); 0x016A, 空中操作手客户端(蓝)。

学生机器人间通信 cmd_id 0x0301, 内容 ID:0x0200~0x02FF

表 5-1 交互数据 机器人间通信: 0x0301

字节偏移量	大小	说明	备注
0	2	数据的内容 ID	0x0200~0x02FF 可以在以上 ID 段选取, 具体 ID 含义由参赛队自定义
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, 例如不能发送到敌对机器人的 ID
6	n	数据段	n 需要小于 113

```
typedef __pack struct
```

```
{
uint8_t data[]
} robot_interactive_data_t
```

表 5-2 客户端删除图形 机器人间通信: 0x0301

字节偏移量	大小	说明	备注
0	2	数据的内容 ID	0x0100
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客户端
6	1	图形操作	包括: 0: 空操作; 1: 删除图层; 2: 删除所有;
7	1	图层数	图层数: 0~9

```
typedef __packed struct
{
uint8_t operate_tpye;
uint8_t layer;
} ext_client_custom_graphic_delete_t
```

表 5-3 图形数据

字节偏移量	大小	说明	备注
0	3	图形名	在删除，修改等操作中，作为客户端的索引。
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: 整圆； 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: 红蓝主色； 1: 黄色； 2: 绿色； 3: 橙色；

字节偏移量	大小	说明	备注
			4: 紫红色; 5: 粉色; 6: 青色; 7: 黑色; 8: 白色; Bit 14-22: 起始角度, 单位: °, 范围[0,360]; Bit 23-31: 终止角度, 单位: °, 范围[0,360]。
7	4	图形配置 2	Bit 0-9: 线宽; Bit 10-20: 起点 x 坐标; Bit 21-31: 起点 y 坐标。
11	4	图形配置 3	Bit 0-9: 字体大小或者半径; Bit 10-20: 终点 x 坐标; Bit 21-31: 终点 y 坐标。

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

图形配置详见下表，其中空代表该字段的数据对该图形无影响，推荐字体大小与线宽比例为 10:1

类型	start_angle	end_angle	width	start_x	start_y	radius	end_x	end_y
直线	空	空	线条宽度	起点 x 坐标	起点 y 坐标	空	终点 x 坐标	终点 y 坐标
矩形	空	空	线条宽度	起点 x 坐标	起点 y 坐标	空	对角顶点 x 坐标	对角顶点 y 坐标
正圆	空	空	线条宽度	圆心 x 坐标	圆心 y 坐标	半径	空	空
椭圆	空	空	线条宽度	圆心 x 坐标	圆心 y 坐标	空	x 半轴长度	y 半轴长度
圆弧	起始角度	终止角度	线条宽度	圆心 x 坐标	圆心 y 坐标	空	x 半轴长度	y 半轴长度
浮点数	字体大小	小数位有效个数	线条宽度	起点 x 坐标	起点 y 坐标	乘以 1000 后, 以 32 位整型数, int32_t		
整型数	字体大小	空	线条宽度	起点 x 坐标	起点 y 坐标	32 位整型数, int32_t		
字符	字体大小	字符长度	线条宽度	起点 x 坐标	起点 y 坐标	空	空	空

表 5-4 客户端绘制一个图形 机器人间通信: 0x0301

字节偏移量	大小	说明	备注
0	2	数据的内容 ID	0x0101
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客户端
6	15	图形 1	详见图形数据介绍

```
typedef __packed struct
```

```
{
    graphic_data_struct_t graphic_data_struct;
} ext_client_custom_graphic_single_t;
```

表 5-5 客户端绘制二个图形 机器人间通信：0x0301

字节偏移量	大小	说明	备注
0	2	数据的内容 ID	0x0102
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客户端
6	15	图形 1	详见图形数据介绍
21	15	图形 2	详见图形数据介绍

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

表 5-6 客户端绘制五个图形 机器人间通信：0x0301

字节偏移量	大小	说明	备注
0	2	数据的内容 ID	0x0103
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客户端
6	15	图形 1	详见图形数据介绍
21	15	图形 2	详见图形数据介绍
36	15	图形 3	详见图形数据介绍
51	15	图形 4	详见图形数据介绍
66	15	图形 5	详见图形数据介绍

```
typedef __packed struct
{
    graphic_data_struct_t graphic_data_struct[5];
}
```



```
} ext_client_custom_graphic_five_t;
```

表 5-7 客户端绘制七个图形 机器人间通信：0x0301

字节偏移量	大小	说明	备注
0	2	数据的内容 ID	0x0104
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客户端
6	15	图形 1	详见图形数据介绍
21	15	图形 2	详见图形数据介绍
36	15	图形 3	详见图形数据介绍
51	15	图形 4	详见图形数据介绍
66	15	图形 5	详见图形数据介绍
81	15	图形 6	详见图形数据介绍
96	15	图形 7	详见图形数据介绍

```
typedef __packed struct
{
    graphic_data_struct_t graphic_data_struct[7];
} ext_client_custom_graphic_seven_t;
```

表 5-8 客户端绘制字符 机器人间通信：0x0301

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客户端
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  graptic_data_struct;
    uint8_t data[30];
} ext_client_custom_character_t;
```

6. 自定义控制器交互数据

自定义控制器数据包括一个统一的数据段头结构。数据段为内容数据段，整个交互数据的包总共长最大为 39 个字节，减去 frame_header,cmd_id 和 frame_tail 共 9 个字节，故而发送的内容数据段最大为 30 字节。整个交互数据 0x0302 的包下行频率为 30Hz。

表 6-1 交互数据接收信息：0x0302。发送频率：上限 30Hz

字节偏移量	大小	说明	备注
0	x	内容段数据	x 最大为 30

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

7. 小地图交互信息

小地图交互信息包括一个统一的数据段头结构。

客户端下发信息

小地图下发信息标识：0x0303。发送频率：触发时发送。

表 7-1 客户端下发信息

字节偏移量	大小	说明	备注
0	4	目标 x 位置坐标，单位 m	当发送目标机器人 ID 时，该项为 0
4	4	目标 y 位置坐标，单位 m	当发送目标机器人 ID 时，该项为 0
8	4	目标 z 位置坐标，单位 m	当发送目标机器人 ID 时，该项为 0
12	1	发送指令时，云台手按下的键盘信息	无按键按下则为 0
13	2	要作用的目标机器人 ID	当发送位置信息时，该项为 0

```
typedef __pack struct
{
```

```
float target_position_x;
float target_position_y;
float target_position_z;
uint8_t commd_keyboard;
uint16_t target_robot_ID;
} ext_robot_command_t
```

机器人 ID: 机器人 ID: 1, 英雄(红); 2, 工程(红); 3/4/5, 步兵(红); 6, 空中(红); 7, 哨兵(红); 9, 雷达站(红); 10, 前哨站(红); 11, 基地(红); 101, 英雄(蓝); 102, 工程(蓝); 103/104/105, 步兵(蓝); 106, 空中(蓝); 107, 哨兵(蓝); 109, 雷达站(蓝); 110, 前哨站(蓝); 111, 基地(蓝)。

客户端接收信息

小地图接收信息标识: 0x0305。最大接收频率: 10Hz。

雷达站发送的坐标信息可以被所有己方操作手在第一视角小地图看到。

表 7-2 客户端接收信息

字节偏移量	大小	说明	备注
0	2	目标机器人 ID	当 x,y 超出界限时则不显示。
2	4	目标 x 位置坐标, 单位 m	当 x,y 超出界限时则不显示。
6	4	目标 y 位置坐标, 单位 m	当 x,y 超出界限时则不显示。

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

8. 图传遥控信息

图传遥控信息, 是通过图传模块下发。

图传遥控信息标识: 0x0304。发送频率: 30Hz。

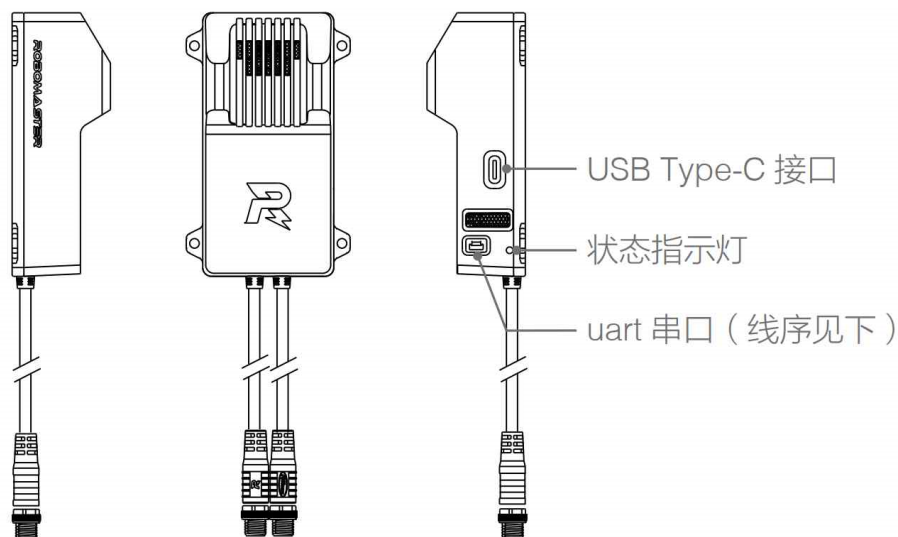


图 8-1 发送端示意图

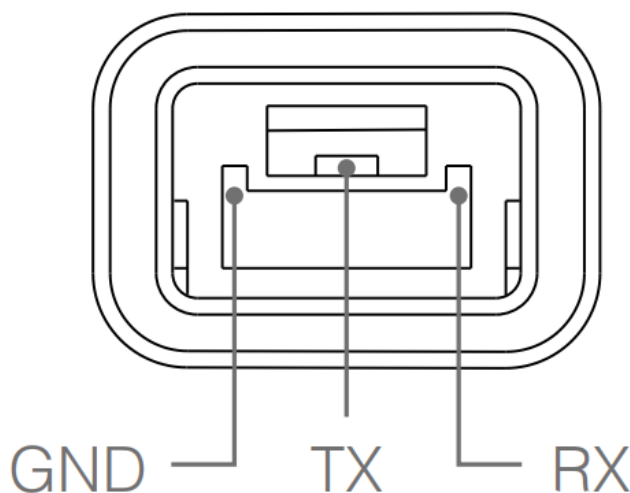


图 8-2 发送端 uart 串口线序

客户端下发信息

表 8-1 客户端下发信息

字节偏移量	大小	说明
0	2	鼠标 X 轴信息
2	2	鼠标 Y 轴信息
4	2	鼠标滚轮信息
6	1	鼠标左键
7	1	鼠标右键按下

字节偏移量	大小	说明
8	2	键盘信息 bit 0: 键盘 W 是否按下 bit 1: 键盘 S 是否按下 bit 2: 键盘 A 是否按下 bit 3: 键盘 D 是否按下 bit 4: 键盘 SHIFT 是否按下 bit 5: 键盘 CTRL 是否按下 bit 6: 键盘 Q 是否按下 bit 7: 键盘 E 是否按下 bit 8: 键盘 R 是否按下 bit 9: 键盘 F 是否按下 bit 10: 键盘 G 是否按下 bit 11: 键盘 Z 是否按下 bit 12: 键盘 X 是否按下 bit 13: 键盘 C 是否按下 bit 14: 键盘 V 是否按下 bit 15: 键盘 B 是否按下
10	2	保留位

```

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

```

9. CRC 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, 0xdb5e, 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,
    0xdec4, 0xcf44, 0xfddf, 0xec56, 0x98e9, 0x8960, 0xbbfb, 0xaa72,
    0x6306, 0x728f, 0x4014, 0x519d, 0x2522, 0x34ab, 0x0630, 0x17b9,
    0xef4e, 0xfec7, 0xcc5c, 0xdd5, 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,

```

```

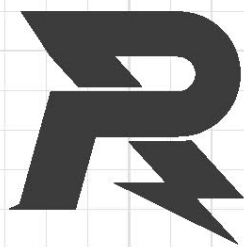
0xc60c, 0xd785, 0xe51e, 0xf497, 0x8028, 0x91a1, 0xa33a, 0xb2b3,
0x4a44, 0x5bcd, 0x6956, 0x78df, 0x0c60, 0x1de9, 0x2f72, 0x3efb,
0xd68d, 0xc704, 0xf59f, 0xe416, 0x90a9, 0x8120, 0xb3bb, 0xa232,
0x5ac5, 0x4b4c, 0x79d7, 0x685e, 0x1ce1, 0x0d68, 0x3ff3, 0x2e7a,
0xe70e, 0xf687, 0xc41c, 0xd595, 0xa12a, 0xb0a3, 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);
```



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镇茶光路1089号集成电路设计应用产业园2楼202