MCU 通信协议需求文档

本文档描述上位机与MCU之间的串口通信协议规范，供嵌入式工程师实现MCU侧代码。

---

1. 物理层规范

1.1 串口配置

参数	值
接口	USB虚拟串口（CDC）
波特率	115200
数据位	8
校验位	无（None）
停止位	1
流控制	无（None）

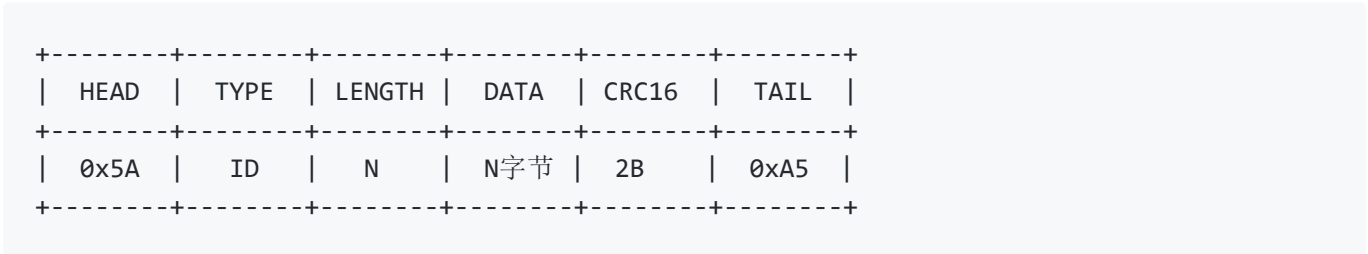
1.2 设备识别

- Linux下设备路径：`/dev/ttyACM*` 或 `/dev/ttyUSB*`
- 建议使用CDC模式，ACM设备号固定

---

2. 数据链路层协议

2.1 帧格式



字段	大小	说明
HEAD	1字节	固定为0x5A
TYPE	1字节	消息类型ID
LENGTH	1字节	DATA字段长度
DATA	N字节	消息数据
CRC16	2字节	从HEAD到DATA末尾的CRC校验
TAIL	1字节	固定为0xA5

2.2 字节序

- CRC16：**小端序**（低字节在前）
- 多字节浮点数/整数：**小端序**

2.3 CRC16计算

使用 **Modbus CRC-16** 算法：

```
// 初始值
uint16_t crc = 0xFFFF;

// 多项式（反向）
const uint16_t POLY = 0xA001;

uint16_t calculate_crc(const uint8_t* data, uint16_t length)
{
    uint16_t crc = 0xFFFF;
    for (uint16_t i = 0; i < length; i++)
    {
        crc ^= data[i];
        for (uint8_t j = 0; j < 8; j++)
        {
            if (crc & 0x0001)
            {
                crc = (crc >> 1) ^ 0xA001;
            }
            else
            {
                crc >>= 1;
            }
        }
    }
    return crc;
}
```

---

3. 消息类型定义

3.1 TypeID 分配

TypeID	方向	消息名称	数据长度
0x10	上→下	CHASSIS_CONTROL	13字节
0x11	下→上	CHASSIS_STATUS	24字节
0x12	下→上	BATTERY_INFO	14字节

---

4. 消息数据结构

4.1 底盘控制 (0x10) - 上位机→MCU

```c
typedef struct __attribute__((packed)) {
    uint8_t  control_type;  // 控制类型: 1=速度, 2=跟随, 3=摆动, 4=旋转
    float    vx;            // 前向速度 (m/s)
    float    vy;            // 横向速度 (m/s)
    float    wz;            // 旋转速度 (rad/s)
} ChassisControlData;
```

字段	偏移	类型	说明
control_type	0	uint8_t	1=VELOCITY, 2=FOLLOW, 3=SWING, 4=SPIN
vx	1	float	前向速度，范围[-3.0, 3.0] m/s
vy	5	float	横向速度，范围[-3.0, 3.0] m/s
wz	9	float	旋转速度，范围[-2.0, 2.0] rad/s

浮点数格式： IEEE 754 单精度（32位）

示例数据：

```
HEAD: 0x5A
TYPE: 0x10
LENGTH: 0x0D (13)
DATA:
  control_type: 0x01
  vx: 0x3F 0xE0 0x00 0x00 (0.5)
  vy: 0x00 0x00 0x00 0x00 (0.0)
  wz: 0x3E 0x99 0x99 0x9A (0.3)
CRC16: 0xXX 0xXX (计算得出)
TAIL: 0xA5
```

4.2 底盘状态 (0x11) - MCU→上位机

```c
typedef struct __attribute__((packed)) {
    float    vx;            // 实际速度X (m/s)
    float    vy;            // 实际速度Y (m/s)
    float    wz;            // 实际角速度 (rad/s)
    float    x;             // 位置X (m)
    float    y;             // 位置Y (m)
    float    theta;         // 航向角 (rad)
} ChassisStatusData;
```

发送频率建议： 50Hz（20ms一次）

4.3 电池信息 (0x12) - MCU→上位机

```c
typedef struct __attribute__((packed)) {
    float    voltage;       // 电压 (V)
    float    current;       // 电流 (A)，正值为放电
    float    percentage;    // 电量百分比 (0-100)
    uint8_t  status;        // 0=未充电，1=充电中，2=充满
} BatteryInfoData;
```

发送频率建议： 10Hz（100ms一次）

---

5. 通信时序

5.1 正常通信流程

```
上位机                              MCU
  |                                  |
  |--[CHASSIS_CONTROL]------------>|
  |                                  | 解析命令
  |                                  | 控制电机
  |                                  |
  |<--[CHASSIS_STATUS]-------------|
  |<--[BATTERY_INFO]---------------|
  |                                  |
  |--[CHASSIS_CONTROL]------------>|
  |                                  |
 ...                                ...
```

5.2 建议发送频率

消息	方向	频率
CHASSIS_CONTROL	上→下	按需（接命令发送）
CHASSIS_STATUS	下→上	50Hz
BATTERY_INFO	下→上	10Hz

---

6. 测试用例

6.1 测试数据示例

测试1：基本控制命令

```
HEX: 5A 10 0D 01 00 00 00 00 00 00 00 00 00 00 XX XX A5
     ^  ^  ^  ^  ^-------------^  ^-------------^  ^  ^  ^
     |  |  |  |       vx=0.0           wz=0.0       |  |  |
     |  |  |  +--------------------------------------+  |  |
     |  |  |          vy=0.0                            |  |
     |  |  |                                            |  |
     |  |  +-- control_type=1                           |  |
     |  |                                               |  |
HEAD  +-- TYPE=0x10                          CRC16    TAIL
```

测试2：前进+右转

```
HEX: 5A 10 0D 01 3E 00 00 00 00 00 00 00 3D CC CC CD XX XX A5
                   |------------|  |------------|
                   vx=0.125 m/s  wz=0.1 rad/s
```

6.2 CRC验证示例

使用在线工具验证：CRC-16/MODBUS计算器

或使用以下C代码验证：

```c
#include <stdio.h>
#include <stdint.h>

uint16_t calc_crc(const uint8_t* data, uint16_t len)
{
    uint16_t crc = 0xFFFF;
    for (uint16_t i = 0; i < len; i++) {
        crc ^= data[i];
        for (uint8_t j = 0; j < 8; j++) {
            if (crc & 0x0001) {
                crc = (crc >> 1) ^ 0xA001;
            } else {
                crc >>= 1;
            }
        }
    }
    return crc;
}

int main()
{
    // 测试数据: HEAD TYPE LENGTH DATA
    uint8_t data[] = {0x5A, 0x10, 0x0D, 0x01, 0x00, 0x00, 0x00,
                      0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00};

    uint16_t crc = calc_crc(data, sizeof(data));
    printf("CRC16: 0x%04X\n", crc);  // 应输出特定值

    // 小端序存储
    printf("CRC_L: 0x%02X, CRC_H: 0x%02X\n", crc & 0xFF, (crc >> 8) & 0xFF);

    return 0;
}
```

---

7. MCU实现建议

7.1 接收缓冲区

建议使用环形缓冲区处理接收数据：

```c
#define RING_BUFFER_SIZE 256

typedef struct {
    uint8_t buffer[RING_BUFFER_SIZE];
    uint16_t head;
    uint16_t tail;
} RingBuffer;

// 写入字节到缓冲区
void ring_buffer_write(RingBuffer* rb, uint8_t data);

// 从缓冲区读取字节
bool ring_buffer_read(RingBuffer* rb, uint8_t* data);
```

7.2 帧解析状态机

```
typedef enum {
    STATE_WAIT_HEAD,
    STATE_READ_TYPE,
    STATE_READ_LENGTH,
    STATE_READ_DATA,
    STATE_READ_CRC_L,
    STATE_READ_CRC_H,
    STATE_READ_TAIL,
    STATE_PROCESS_FRAME
} ParserState;

ParserState state = STATE_WAIT_HEAD;
uint8_t frame_buffer[256];
uint8_t data_length = 0;
uint8_t data_index = 0;
uint16_t received_crc = 0;

void parse_byte(uint8_t byte)
{
    switch (state) {
        case STATE_WAIT_HEAD:
            if (byte == 0x5A) {
                frame_buffer[0] = byte;
                state = STATE_READ_TYPE;
            }
            break;
        case STATE_READ_TYPE:
            frame_buffer[1] = byte;
            state = STATE_READ_LENGTH;
            break;
        // ... 其他状态
    }
}
```

7.3 错误处理

错误类型	处理方式
帧头错误	丢弃字节，继续查找0x5A
数据长度超限	丢弃当前帧，重新开始
CRC校验失败	丢弃当前帧，重新开始
帧尾错误	丢弃当前帧，重新开始

8. 联调检查清单

MCU侧完成后，请检查以下项目：

- [ ] 串口配置正确（115200, 8N1）
- [ ] 能接收并正确解析CHASSIS_CONTROL消息
- [ ] 能正确发送CHASSIS_STATUS消息
- [ ] 能正确发送BATTERY_INFO消息
- [ ] CRC16计算与上位机一致
- [ ] 浮点数使用IEEE 754格式
- [ ] 数据字节序为小端
- [ ] 发送频率符合建议值
- [ ] 错误处理机制完善

---

9. 联系方式

如有疑问，请联系上位机开发人员。