多自由度平台 UDP 通信使用手册

概述

本手册描述了多自由度运动平台 UDP 通信协议的使用方法。UDP 服务器运行在多自由度平台控制软件中，接收客户端发送的控制命令并返回执行结果。

连接配置

服务器配置

- **默认IP地址**: 127.0.0.1（本地回环）
- **默认端口**: 1100
- **协议**: UDP
- **编码**: UTF-8

消息格式

- **请求格式**: `命令ID 参数1 参数2 ...`
- **响应格式**: `命令ID#结果`
- **分隔符**: 空格、制表符、逗号、分号
- **换行符**: 自动过滤 \r\n 和 \n

命令列表

基础控制命令

命令ID	名称	参数	说明	响应示例
1	初始化	无	初始化Stewart平台	`1#0`
2	复位	无	复位平台到初始状态	`2#0`
3	清除错误	无	清除系统错误状态	`3#0`
4	停止	无	缓慢停止平台运动	`4#0`
5	暂停	无	暂停当前运动	`5#0`
6	取消暂停	无	恢复暂停的运动	`6#0`

位置控制命令

命令ID	名称	参数	说明	响应示例
7	回零点	无	平台回到零点位置	`7#0`
8	到工作面	无	平台移动到工作面高度	`8#0`
9	单点移动	X Y Z RX RY RZ	移动到指定位置	`9#0`

单点移动示例:

```
9 10.5 20.0 30.0 5.0 10.0 15.0
```

轨迹控制命令

命令ID	名称	参数	说明	响应示例
10	路径点运动	点数 坐标序列	按路径点序列运动	`10#0`
11	带速度路径点	点数 位置序列 速度序列	路径点运动（含速度）	`11#0`
12	公式运动	公式1,公式2,...,公式6	基于公式的运动	`12#0`
13	PT模式	时间 坐标序列	位置-时间同步运动	`13#0`

路径点运动示例:

```
10 3 0 0 0 0 0 0 10 20 30 5 10 15 20 30 40 10 15 20
```

说明：3个点，每个点6个坐标(X Y Z RX RY RZ)

带速度路径点示例:

```
11 2 0 0 0 0 0 0 10 20 30 5 10 15 50 50 50 5 5 5 100 100 100 10 10 10
```

说明：2个点，先是位置数据，后是速度数据

公式运动示例:

```
12,10*sin(t),20*cos(t),5*t,0,0,2*t
```

说明：6个公式，对应X,Y,Z,RX,RY,RZ轴

文件执行命令

命令ID	名称	参数	说明	响应示例
33	执行文件	文件ID	执行预定义的运动文件	`33#0`

预定义文件: - 文件ID=1: 位置轨迹文件 (pos01.txt) - 文件ID=2: 路径点文件 (waypoints01.txt) - 文件ID=3: 运动公式文件 (formula01.txt)

状态查询命令

命令ID	名称	响应内容	说明
20	读取连接状态	`20#1`	1=已连接，0=未连接
21	读取归零状态	`21#1`	1=已归零，0=未归零
22	读取运动状态	`22#0`	1=运动中，0=停止
23	读取错误码	`23#0`	错误码值
24	读取平台位置	`24#0.000 0.000 0.000 0.000 0.000 0.000`	X Y Z RX RY RZ
25	读取平台速度	`25#0.000 0.000 0.000 0.000 0.000 0.000`	VX VY VZ VRX VRY VRZ
26	读取执行器位置	`26#0.000 0.000 0.000 0.000 0.000 0.000`	6个执行器位置
27	读取执行器速度	`27#0.000 0.000 0.000 0.000 0.000 0.000`	6个执行器速度
28	读取执行器扭矩	`28#0.000 0.000 0.000 0.000 0.000 0.000`	6个执行器扭矩
29	读取执行器电流	`29#0.000 0.000 0.000 0.000 0.000 0.000`	6个执行器电流
30	读取执行器电压	`30#0.000 0.000 0.000 0.000 0.000 0.000`	6个执行器电压
31	读取全部状态	`31#1;1;0;0;0.000 0.000...`	所有状态信息

错误码说明

常见错误码

错误码	含义	解决方法
0x0000	无错误	-
0x0001	无许可证	检查软件授权
0x0010	通信错误	检查网络连接和配置
0x0011	网络断开	检查网络连接
0x00C0	伺服错误	检查伺服系统
0x00C1	伺服未使能	启用伺服系统
0x00C2	需要归零	执行归零操作
0x00C3	急停状态	复位急停开关

限位错误

- 0x00A1-0x00AC: 平台各轴超限位
- 0x00AF-0x00B5: 平台各轴超速
- 0x00B6-0x00B8: 执行器超限位或超速

通信流程

1. 命令分类

UDP服务器采用**命令队列系统**，根据命令性质分为三类：

即时命令 - 立即执行并返回结果

命令ID	说明	特点
2	复位位置	不受运动状态影响
3	清除错误	不受运动状态影响
4	停止运动	紧急停止，随时可执行
5	暂停运动	随时可执行
6	取消暂停	随时可执行
20-32	各种查询命令	随时可查询，不影响运动

运动命令 - 运动完成后返回结果（需要排队）

命令ID	说明	特点
1	初始化平台	需要等待初始化完成
7	回零点	平台空闲时才能执行
8	到工作面	平台空闲时才能执行
9	单点移动	平台空闲时才能执行
10	路径点运动	平台空闲时才能执行
11	带速度路径点	平台空闲时才能执行
12	公式运动	平台空闲时才能执行
13	PT模式运动	平台空闲时才能执行
33	文件号运动	平台空闲时才能执行

重要说明： - **即时命令**：随时可执行，即使平台正在运动或队列中有待处理命令 - **运动命令**：只有当平台空闲且命令队列为空时才能被接受 - 如果平台正在运动或队列不为空，新的运动命令会被拒绝并返回错误码 `-2`

2. 运动命令反馈机制（2026年1月优化）

系统采用**命令队列机制**，支持运动过程中的暂停、恢复、停止操作，并提供完整的状态反馈。

反馈类型

运动命令可能收到以下反馈：

反馈码	类型	含义	命令状态
`0`	最终反馈	运动正常完成	命令结束
`-1`	立即反馈	格式错误/参数错误	命令结束
`-2`	立即反馈	平台忙碌，命令被拒绝	命令结束
`-4`	最终反馈	运动被停止命令中断	命令结束
`-5`	中间反馈	运动被暂停	命令保留，等待恢复或停止
`-6`	中间反馈	运动已恢复	命令保留，继续等待完成
`-7`	立即反馈	暂停状态下拒绝新命令	命令结束
正整数	最终反馈	硬件错误码	命令结束

说明： - **中间反馈**（-5、-6）：运动命令未结束，后续还会收到最终反馈（0或-4或错误码） - **最终反馈**：运动命令结束，不会再收到该命令的反馈

示例场景

正常运动流程:

```
客户端A -> 服务器: "33 1"     (文件运动命令)
[平台开始运动]
[运动完成]
服务器 -> 客户端A: "33#0"     (运动正常完成)
```

暂停后恢复运动:

```
客户端A -> 服务器: "33 1"     (文件运动命令)
[平台开始运动]

客户端B -> 服务器: "5"        (暂停命令)
服务器 -> 客户端B: "5#0"      (暂停命令执行成功)
服务器 -> 客户端A: "33#-5"    (通知A: 运动被暂停)

[平台暂停中...]

客户端B -> 服务器: "6"        (恢复命令)
服务器 -> 客户端B: "6#0"      (恢复命令执行成功)
服务器 -> 客户端A: "33#-6"    (通知A: 运动已恢复)

[运动继续执行]
[运动完成]
服务器 -> 客户端A: "33#0"     (运动正常完成)
```

暂停后停止运动:

```
客户端A -> 服务器: "33 1"     (文件运动命令)
[平台开始运动]

客户端B -> 服务器: "5"        (暂停命令)
服务器 -> 客户端B: "5#0"      (暂停命令执行成功)
服务器 -> 客户端A: "33#-5"    (通知A: 运动被暂停)

[平台暂停中...]

客户端B -> 服务器: "4"        (停止命令)
服务器 -> 客户端B: "4#0"      (停止命令执行成功)
服务器 -> 客户端A: "33#-4"    (通知A: 运动被停止，命令结束)
```

运动中直接停止:

```
客户端A -> 服务器: "33 1"     (文件运动命令)
[平台开始运动]

客户端B -> 服务器: "4"        (停止命令)
服务器 -> 客户端B: "4#0"      (停止命令执行成功)
服务器 -> 客户端A: "33#-4"    (通知A: 运动被停止，命令结束)
```

暂停状态下拒绝新运动命令:

```
客户端A -> 服务器: "33 1"     (文件运动命令)
[平台开始运动]

客户端B -> 服务器: "5"        (暂停命令)
服务器 -> 客户端B: "5#0"      (暂停命令执行成功)
服务器 -> 客户端A: "33#-5"    (通知A: 运动被暂停)

[平台暂停中...]

客户端C -> 服务器: "7"        (新的回零命令)
服务器 -> 客户端C: "7#-7"     (拒绝: 平台处于暂停状态)

[客户端C需要先让平台恢复(6)或停止(4)，才能发送新的运动命令]
```

系统特性

1. **排队机制**：同时只能执行一个运动命令，新命令需等待当前运动完成
2. **可靠反馈**：每个命令的反馈精确发送给发起该命令的客户端，不会混淆
3. **中间状态反馈**：暂停、恢复时立即通知原命令客户端，便于状态跟踪
4. **暂停保护**：暂停状态下拒绝新运动命令，返回 `-7`，需先恢复或停止
5. **无超时限制**：运动时间取决于轨迹参数，系统不设超时限制
6. **即时查询**：查询命令（20-32）随时可用，不受运动状态影响

3. 多客户端支持（2026年1月优化）

系统完全支持多客户端同时连接，采用**客户端EndPoint识别**机制：

客户端识别

- **识别方式**：`IP地址 + 端口号` 组合（即 UDP EndPoint）
- **端口说明**:
- 同一IP的不同端口被视为不同客户端
- UDP客户端端口通常是操作系统随机分配的临时端口
- 同一程序多次启动会使用不同端口，被识别为不同客户端

使用场景

场景1：控制客户端 + 监控客户端

```
[客户端A: 192.168.1.100:5000] 控制平台运动
客户端A -> 服务器: "7"              (回零命令)
[平台开始运动]

[客户端B: 192.168.1.101:6000] 实时监控
客户端B -> 服务器: "22"             (查询运动状态)
服务器 -> 客户端B: "22#1"           (返回: 运动中)

客户端B -> 服务器: "24"             (查询位置)
服务器 -> 客户端B: "24#0.5 1.2 3.4 0.0 0.0 0.0"

[运动完成]
服务器 -> 客户端A: "7#0"            (反馈发给A，不会发给B)
```

场景2：多个控制客户端竞争

```
[客户端A: 192.168.1.100:5000]
客户端A -> 服务器: "33 1"          (文件运动，约6秒)
[命令A在队列中，开始运动]

[客户端B: 192.168.1.100:5001]  ← 注意: 同IP但不同端口
客户端B -> 服务器: "7"             (回零命令)
[检测到队列不为空]
服务器 -> 客户端B: "7#-2"          (立即拒绝: 平台忙)

[6秒后]
服务器 -> 客户端A: "33#0"          (反馈给A，不会发给B)

[现在队列为空，B可以重新发送]
客户端B -> 服务器: "7"             (回零命令)
服务器 -> 客户端B: "7#0"           (运动完成后反馈给B)
```

场景3：查询命令不受影响

```
[客户端A运动中]
客户端A -> 服务器: "33 1"

[客户端B、C、D同时查询]
客户端B -> 服务器: "22"
服务器 -> 客户端B: "22#1"          (运动状态)

客户端C -> 服务器: "24"
服务器 -> 客户端C: "24#1.5 2.3 ..." (当前位置)

客户端D -> 服务器: "23"
服务器 -> 客户端D: "23#0"          (错误码)

[查询不影响运动，反馈各自返回正确客户端]
服务器 -> 客户端A: "33#0"          (A的运动完成)
```

关键特性

1. **反馈精准**：每个客户端的反馈独立返回，互不干扰
2. **并发查询**：多个客户端可同时查询状态和位置，互不影响
3. **运动互斥**：同时只能有一个运动命令在执行，其他命令需等待
4. **即时命令**：停止、暂停等命令随时可用，任何客户端都可执行
5. **灵活连接**：客户端可随时连接和断开，无需保持连接状态

最佳实践

- **监控程序**：使用查询命令（20-32），高频查询不影响控制
- **控制程序**：发送运动命令前先查询状态，收到`-2`说明平台忙，收到`-7`说明平台暂停中
- **紧急停止**：任何客户端都可发送停止命令（4），立即生效
- **暂停控制**：暂停后需发送恢复(6)或停止(4)才能接受新的运动命令
- **状态跟踪**：运动客户端应监听中间反馈（-5暂停、-6恢复），了解运动状态变化
- **长时间运动**：建议使用定时查询（命令22）监控运动进度

4. 统一错误码（2026年1月优化）

系统使用**标准化数字错误码**，更加清晰准确：

协议层错误码

错误码	含义	说明
`0`	成功	命令执行成功/运动正常完成
`-1`	格式错误/参数错误	命令格式不正确、参数超出范围
`-2`	平台正在运动	平台忙碌时发送运动命令会被拒绝
`-3`	运动过程出错	运动执行过程中发生错误
`-4`	运动被停止	运动被停止命令(4)中断，运动结束
`-5`	运动被暂停	运动被暂停命令(5)中断，等待恢复或停止
`-6`	运动已恢复	运动被恢复命令(6)继续，等待完成
`-7`	平台处于暂停状态	暂停状态下发送运动命令会被拒绝，请先恢复(6)或停止(4)

硬件错误码

除了上述协议层错误码，运动命令还可能返回底层硬件的错误代码（正整数）：

常见错误码范围: - `161-172` (0x00A1-0x00AC): 各轴超出限位 - `173-178` (0x00AD-0x00B2): 各轴超速 - `179-181` (0x00B3-0x00B5): 执行器超限位/超速 - `192-195` (0x00C0-0x00C3): 伺服系统错误

详细错误码列表请参考 错误码说明 章节。

错误响应示例

格式错误:

```
客户端 -> 服务器: "9 abc def"       (参数不是数字)
服务器 -> 客户端: "9#-1"            (格式错误)
```

平台忙碌:

```
客户端 -> 服务器: "7"               (平台正在运动时发送)
服务器 -> 客户端: "7#-2"            (平台正在运动)
```

底层硬件错误:

```
客户端 -> 服务器: "9 999 0 0 0 0 0" (X轴超限)
服务器 -> 客户端: "9#161"           (0x00A1 = 161, X轴负超限)
```

运动过程出错:

```
客户端 -> 服务器: "33 1"            (启动文件运动)
[运动中触发限位]
服务器 -> 客户端: "33#161"          (运动过程中出错)
```

无效命令:

```
客户端 -> 服务器："999"              （不存在的命令ID）
服务器 -> 客户端："999#-1"           （格式错误）
```

版本兼容说明

旧版本返回	新版本返回	说明
`AAAA`	`-1`	格式错误
`BBBB`	`-2`	平台忙碌
`Unknown Header#AAAA`	`命令ID#-1`	无效命令

使用说明

判断命令结果： - 返回 `0`：命令成功执行/运动正常完成 - 返回 `-1`：检查命令格式和参数是否正确 - 返回 `-2`：平台正在运动，请等待完成后重试 - 返回 `-3`：运动过程中出错 - 返回 `-4`：运动被停止命令中断（运动结束） - 返回 `-5`：运动被暂停（中间状态，等待恢复或停止） - 返回 `-6`：运动已恢复（中间状态，继续等待完成） - 返回 `-7`：平台处于暂停状态，请先发送恢复(6)或停止(4)命令 - 返回正整数：硬件错误码，请查阅错误码表

使用示例

SSCOM UDP 调试工具使用说明

SSCOM 是一款常用的串口和网络调试工具，支持 UDP 通信测试。

1. SSCOM 配置步骤

1. **下载安装 SSCOM** - 下载 SSCOM 调试工具 - 解压运行 sscom.exe
2. **配置 UDP 连接** - 选择 "网络助手" 标签 - 协议类型：选择 "UDP" - 本机 IP：127.0.0.1 （或实际IP） - 本机端口：设置一个空闲端口（如 2000） - 目标 IP：127.0.0.1 （Stewart 平台IP） - 目标端口：1100 （Stewart 平台端口）
3. **连接设置** - 点击 "打开网络" 按钮 - 状态显示 "网络已打开" 表示成功

2. 命令测试流程

步骤1：初始化平台

```
发送：1
期望响应：1#0
```

步骤2：查询连接状态

```
发送：20
期望响应：20#1
```

步骤3：查询运动状态

```
发送：22
期望响应：22#0
```

步骤4：单点移动测试

```
发送：9 10.5 20.0 30.0 5.0 10.0 15.0
期望响应：9#0 （运动完成后返回）
```

步骤5：查询当前位置

```
发送：24
期望响应：24#10.500 20.000 30.000 5.000 10.000 15.000
```

步骤6：回零测试

```
发送：7
期望响应：7#0 （运动完成后返回）
```

3. SSCOM 操作技巧

发送区域设置 - 勾选 "发送新行" 选项 - 编码选择：UTF-8 - 发送格式：字符串

接收区域设置 - 显示接收数据的时间戳 - 自动清除缓冲区 - 十六进制显示（可选）

批量测试 - 使用 "定时发送" 功能进行周期性查询 - 间隔设置：1000ms（1秒） - 命令示例：`22` （查询运动状态）

4. 常用测试命令组合

基础功能测试

```
1          # 初始化
20         # 查询连接状态
21         # 查询归零状态
22         # 查询运动状态
23         # 查询错误码
```

位置控制测试

```
7          # 回零点
8          # 到工作面
24         # 查询位置
```

简单运动测试

```
9 0 0 10 0 0 0     # Z轴向上10mm
24                 # 查询位置
9 0 0 0 0 0 0      # 回到原点
```

路径点测试

```
10 2 0 0 0 0 0 0 10 10 10 0 0 0    # 2个点的轨迹
```

5. 错误处理示例

命令格式错误

```
发送：abc
响应：abc#AAAA
```

运动中发送运动命令

```
发送：9 0 0 0 0 0 0    # 平台正在运动时
响应：9#BBBB           # 运动警告
```

无效命令ID

```
发送：99
响应：99#AAAA
```

6. 高级测试场景

连续路径测试

```
1. 发送：7           # 回零
2. 等待：7#0         # 运动完成确认
3. 发送：9 10 0 0 0 0 0    # 移动到第一点
4. 等待：9#0         # 运动完成确认
5. 发送：9 20 10 0 0 0 0   # 移动到第二点
6. 等待：9#0         # 运动完成确认
```

状态监控测试

```
# 使用定时发送，每500ms查询一次运动状态
定时命令：22
观察响应变化：22#1（运动中）-> 22#0（停止）
```

其他调试工具推荐

1. UDP Test Tool

- 轻量级 UDP 测试工具
- 支持十六进制和字符串格式
- 具有日志记录功能

2. NetAssist 网络调试助手

- 支持 TCP/UDP 协议
- 多连接管理
- 数据统计功能

3. SocketTest

- 专业的网络调试工具
- 支持脚本自动化测试
- 适用于批量命令测试

编程接口示例

C# 客户端示例

```
using System;
using System.Net;
using System.Net.Sockets;
using System.Text;

class UDPClient
{
    public static string SendCommand(string ip, int port, string command)
    {
        using (UdpClient client = new UdpClient())
        {
            IPEndPoint serverEndPoint = new IPEndPoint(IPAddress.Parse(ip), port);

            // 发送命令
            byte[] data = Encoding.UTF8.GetBytes(command);
            client.Send(data, data.Length, serverEndPoint);

            // 接收响应
            IPEndPoint remoteEndPoint = new IPEndPoint(IPAddress.Any, 0);
            byte[] response = client.Receive(ref remoteEndPoint);
            string result = Encoding.UTF8.GetString(response);

            Console.WriteLine($"发送: {command}");
            Console.WriteLine($"响应: {result}");
            return result;
        }
    }

    public static void Main()
    {
        string serverIP = "127.0.0.1";
        int serverPort = 1100;

        // 初始化
        SendCommand(serverIP, serverPort, "1");

        // 回零点
        SendCommand(serverIP, serverPort, "7");

        // 查询位置
        SendCommand(serverIP, serverPort, "24");
    }
}
```

注意事项

1. **运动安全**: 执行运动命令前确保平台已初始化且无错误
2. **并发限制**: 平台运动中不能执行新的运动命令，会返回 `-2` 错误
3. **参数范围**: 位置参数需在平台工作范围内
4. **网络超时**: 建议设置接收超时时间
5. **错误处理**: 检查响应中的错误码并及时处理
6. **位置精度**: 位置验证精度为 ±0.5mm
7. **多客户端**: 支持多客户端连接，查询命令不受运动状态影响
8. **反馈等待**: 运动命令需等待运动完成后才返回反馈，可能需要数秒

2026年1月系统优化说明

本次更新对UDP通信系统进行了重大优化，主要改进包括：

1. 可靠的反馈机制

- **问题**：旧版本多客户端同时使用时，可能出现反馈发错客户端的情况
- **改进**：采用命令队列系统，每条命令的反馈精确发送给对应客户端
- **效果**：完全解决多客户端反馈混乱问题

2. 标准化错误码

- **问题**：旧版本使用 `AAAA`、`BBBB` 等字符串错误码，不便于程序处理
- **改进**：使用标准数字错误码 `-1`、`-2`、`-3` 等
- **效果**：客户端程序更容易判断和处理错误

3. 更快的反馈速度

- **问题**：旧版本运动完成后最多需要1秒才能收到反馈
- **改进**：监控间隔优化到100ms
- **效果**：反馈延迟降低到100-200ms

4. 完善的多客户端支持

- **问题**：旧版本对多客户端支持不明确
- **改进**：明确说明客户端识别机制和使用场景
- **效果**：支持控制客户端+多个监控客户端同时工作

5. 命令分类清晰

- **改进**：明确区分即时命令、运动命令、查询命令
- **效果**：用户清楚哪些命令可以随时执行，哪些需要等待

升级建议

- 建议客户端程序更新错误码判断逻辑，使用数字错误码

- 旧版本客户端仍可正常使用，协议完全向后兼容
- 如需使用多客户端功能，建议参考本手册的多客户端使用场景

文件格式

位置轨迹文件 (*.txt)

```
时间1 X1 Y1 Z1 RX1 RY1 RZ1
时间2 X2 Y2 Z2 RX2 RY2 RZ2
...
```

路径点文件 (*.txt)

```
X1 Y1 Z1 RX1 RY1 RZ1
X2 Y2 Z2 RX2 RY2 RZ2
...
```

公式文件 (*.txt)

```
X轴公式
Y轴公式
Z轴公式
RX轴公式
RY轴公式
RZ轴公式
```

版本: v3.1 **更新日期**: 2026年1月29日 **适用软件**: 多自由度运动平台控制系统

更新记录: - v3.1 (2026.01.29): - **暂停/恢复中间状态反馈**：暂停时返回-5，恢复时返回-6，让运动客户端实时了解状态变化 - **暂停状态保护**：暂停状态下拒绝新运动命令，返回-7，需先恢复或停止 - **停止反馈优化**：运动被停止时返回-4，明确区分正常完成和被中断 - **多反馈支持**：一个运动命令可收到多次反馈（暂停-5、恢复-6、完成0或停止-4） - 新增错误码：-4(停止)、-5(暂停)、-6(恢复)、-7(暂停状态拒绝) - v3.0 (2026.01.28): - **重大优化**：引入命令队列系统，解决多客户端并发反馈混乱问题 - **反馈可靠性**：每条命令保证唯一反馈，精确发送给对应客户端 - **错误码标准化**：使用数字错误码替代字符串错误提示（-1, -2, -3） - **多客户端支持**：完善多客户端同时连接的说明和使用场景 - **反馈延迟优化**：监控间隔从1秒优化到100ms，反馈更及时 - **竞态条件修复**：修复运动结束时接受新命令导致反馈混乱的问题 - 修正文件执行命令号为33 - 添加命令分类说明（即时命令、运动命令、查询命令） - v2.0 (2026.01): - 简化运动命令反馈机制，每条命令只返回一条结果 - 添加多客户端支持说明 - v1.0 (2025): 初始版本