智流形机器人
从 制 造 到 智 造

INSTRUCTION

智流形机器人控制系统

Modbus 通讯使用文档

版权声明

声明

深圳市智流形机器人技术有限公司保留在没有预先通知的情况下修改产品或其特性的权利。

深圳市智流形机器人技术有限公司并不承担由于使用产品不当而产生的直接或是间接的伤害或损坏的责任。

联系我们

深圳市智流形机器人技术有限公司

Shenzhen iManifold Robot Technology Ltd.

地址：深圳市宝安区华丰国际机器人产业园二期 B 栋 410

智云流形科技（江阴）有限公司

iManifold Technology (Jiangyin) Ltd.

地址：江苏省无锡市江阴市长山大道 55 号天安数码城 60 幢 102 室

目　录

1 Modbus 通讯协议介绍

本功能为工业机器人提供 modbus_tcp 通讯方式，提供工业机器人的系统配置参数，变量资源，运动控制指令，机器人工作状态等接口。

通过本功能可对机器人系统进行modbus_tcp 通讯，从而实现机器人实时运行状态的监控和收集，多设备互联互通和协同配合运行，远程操作控制和运维等需求。

部分特性：

1. 远程模式下生效
2. 更新周期:
    1. input 变量每 100ms 更新一次
    2. hold 中变量部分每 100ms 更新 100 个，循环更新（全部变量相当于 1000ms 更新一次）
    3. coil 变量每 100ms 更新一次
    4. discrete 变量每 100ms 更新一次
    5. 用戶触发命令通过信号槽立即更新
3. 详细地址请使用地址计算工具

2 寄存器地址表

Modbus 支持四种寄存器类型

寄存器种类	对象类型	访问类型	内容
Input_Register 状态寄存器	1-Bit	只读	I/O 系统提供这种类型数据
Coli 线圈 数字量输入	1-Bit	读写	通过应用程序改变这种类型数据
Holding_Registe 命令寄存器	16-Bit	读写	通过应用程序改变这种类型数据
Discrete_Input 数字量输入	16-Bit	只读	I/O 系统提供这种类型数据

2.1 Input_Register

状态寄存器：只读

名称	数据类型	地址	长度	备注
系统状态	16-bit integer	0x0000	1	0x0000：待机； 0x0001：使能； 0x0002：执行中； 0x0003：暂停； 0x0A00：警告； 0x0A01：伺服警告 0x0A02：执行中警告 0x0A03：暂停中警告 0x0A04：报警；
伺服状态	16-bit integer	0x0001	1	0：伺服关闭； 1：伺服打开； 2：上伺服报警； 3：驱动器错误；
工作模式	16-bit integer	0x0002	1	0：示教模式； 1：回放模式； 2：远程模式； 3：远程示教模式； 4：远程回放模式；

姿态号	16-bit integer	0x0003	1	当前姿态号
外部轴号	16-bit integer	0x0004	1	
回放速度	16-bit integer	0x0005	1	百分比[1-100]
示教速度	16-bit integer	0x0006	1	百分比[1-100]
关节位置：轴 1	32-bit float	0x0010	2	
关节位置：轴 2	32-bit float	0x0012	2	
关节位置：轴 3	32-bit float	0x0014	2	
关节位置：轴 4	32-bit float	0x0016	2	
关节位置：轴 5	32-bit float	0x0018	2	
关节位置：轴 6	32-bit float	0x001A	2	
关节位置：轴 7	32-bit float	0x001C	2	
关节位置：轴 8	32-bit float	0x001E	2	
外部轴位置：轴 1	32-bit float	0x0020	2	
外部轴位置：轴 2	32-bit float	0x0022	2	
外部轴位置：轴 3	32-bit float	0x0024	2	
外部轴位置：轴 4	32-bit float	0x0026	2	
外部轴位置：轴 5	32-bit float	0x0028	2	
外部轴位置：轴 6	32-bit float	0x002A	2	
外部轴位置：轴 7	32-bit float	0x002C	2	
外部轴位置：轴 8	32-bit float	0x002E	2	

工具在机器人坐标系下位置：x	32-bit float	0x0030	2	
工具在机器人坐标系下位置：y	32-bit float	0x0032	2	
工具在机器人坐标系下位置：z	32-bit float	0x0034	2	
工具在机器人坐标系下位置：r	32-bit float	0x0036	2	
工具在机器人坐标系下位置：p	32-bit float	0x0038	2	
工具在机器人坐标系下位置：y	32-bit float	0x003A	2	
工具在工件坐标系下位置：x	32-bit float	0x003C	2	
工具在工件坐标系下位置：y	32-bit float	0x003E	2	
工具在工件坐标系下位置：z	32-bit float	0x0040	2	
工具在工件坐标系下位置：r	32-bit float	0x0042	2	
工具在工件坐标系下位置：p	32-bit float	0x0044	2	

工具在工件坐标系下位置：y	32-bit float	0x0046	2	
报警信息	128-byte string	0x0050	64	以字节流保存报警信息
tcs_id	16-bit integer	0x0090	1	
wcs_id	16-bit integer	0x0091	1	
pcs1_id	16-bit integer	0x0092	1	
pcs2_id	16-bit integer	0x0093	1	
当前坐标系	16-bit integer	0x0094	1	0：关节坐标系； 1：笛卡尔坐标系； 2：工件坐标系 1； 3：工件坐标系 2； 4：世界坐标系； 5：工具坐标系； 6：扩展关节坐标系； 7：未定义
当前工程名	32-byte string	0x00A0	16	以字节流保存报警信息
当前程序名	32-byte string	0x00C0	16	以字节流保存报警信息
当前执行程序行数	32-bit integer	0x00E0	2	

当前执行程序模式	16-bit integer	0x00E2	1	0：一次； 1：步进； 2：循环；
AI1	32-bit float	0x00F0	2	偏移地址：0x0002 * n + 0x00EE
……	……	……		
AI32	32-bit float	0x012E	2	

2.2 Holding_Register

命令寄存器（加粗表示命令字，其余为参数）

注意：需要先修改参数

名称	数据类型	地址	长度	备注
伺服上使能	16-bit integer	0x0000	1	上升沿有效，生效后自动复位； 如当前未使能，则该信号触发后伺服使能
伺服去使能	16-bit integer	0x0001	1	上升沿有效，生效后自动复位； 如当前已经使能，则该信号触发后伺服去使能

清除报警	16-bit integer	0x0002	1	上升沿有效，生效后自动复位； 该信号触发后尝试清除报警信息
回放速度	16-bit integer	0x0003	1	百分比(1-100)，值改变立即生效
示教速度	16-bit integer	0x0004	1	百分比(1-100)，值改变立即生效
工作模式	16-bit integer	0x0005	1	3：远程点动； 4：远程回放 值改变立即生效
执行工程名	32-byte string	0x0010	16	以字节流保存报警信息
执行程序名	32-byte string	0x0020	16	以字节流保存报警信息
执行程序行数	16-bit integer	0x0030	1	
执行程序模式	16-bit integer	0x0031	1	0：一次； 1：步进； 2：循环；
执行程序	16-bit integer	0x0032	1	上升沿有效，生效后自动复位；该信号触发后执行程序

暂停程序	16-bit integer	0x0033	1	上升沿有效，生效后自动复位； 该信号触发后暂停程序
继续执行	16-bit integer	0x0034	1	上升沿有效，生效后自动复位； 该信号触发后停止程序
停止程序	16-bit integer	0x0035	1	上升沿有效，生效后自动复位； 该信号触发后停止程序
轴 1 点动	16-bit integer	0x0040	1	在当前的坐标系下示教 3：停止运动； 1：正向运动； 2：反向运动 沿信号生效，生效后自动复位 500ms 后自动停止运动，期间再次触发刷新计时
……	……	……	……	……
轴 8 点动	16-bit integer	0x0047	1	在当前的坐标系下示教 3：停止运动； 1：正向运动；

				2：反向运动 沿信号生效，生效后自动复位 500ms 后自动停止运动，期间再次触发刷新计时
外轴 1 点动	16-bit integer	0x0050	1	在当前的坐标系下示教 3：停止运动； 1：正向运动； 2：反向运动 沿信号生效，生效后自动复位 500ms 后自动停止运动，期间再次触发刷新计时
……	……	……	……	……
外轴 8 点动	16-bit integer	0x0057	1	在当前的坐标系下示教 3：停止运动； 1：正向运动； 2：反向运动 沿信号生效，生效后自动复位 500ms 后自动停止运动，期间再次触发刷新计时

tcs_id	16-bit integer	0x0058	1	值改变立即生效
wcs_id	16-bit integer	0x0059	1	值改变立即生效
pcs1_id	16-bit integer	0x005A	1	值改变立即生效
pcs2_id	16-bit integer	0x005B	1	值改变立即生效
当前坐标系	16-bit integer	0x005C	1	0：关节坐标系； 1：笛卡尔坐标系； 2：工件坐标系 1； 3：工件坐标系 2； 4：世界坐标系； 5：工具坐标系； 6：扩展关节坐标系； 7：未定义 值改变立即生效
AO1	32-bit float	0x0060	1	值改变立即生效
……	……	……	……	……
AO32	32-bit float	0x0080	1	值改变立即生效
全局全局位置型变量 0001: 坐标系类型	16-bit integer	0x0100	1	值改变立即生效，偏移地址: 0x0020 * n + 0x00E0

				参考上面 当前坐标系数值对应的坐标系
全局位置型变量 0001:坐标系编号	16-bit integer	0x0101	1	值改变立即生效，偏移地址: 0x0020 * n + 0x00E1
全局位置型变量 0001:关节 1	32-bit float	0x0102	2	值改变立即生效，偏移地址: 0x0020 * n + 0x00E2
全局位置型变量 0001:关节 2	32-bit float	0x0104	2	值改变立即生效，偏移地址: 0x0020 * n + 0x00E4
全局位置型变量 0001:关节 3	32-bit float	0x0106	2	值改变立即生效，偏移地址: 0x0020 * n + 0x00E6
全局位置型变量 0001:关节 4	32-bit float	0x0108	2	值改变立即生效，偏移地址: 0x0020 * n + 0x00E8
全局位置型变量 0001:关节 5	32-bit float	0x010A	2	值改变立即生效，偏移地址: 0x0020 * n + 0x00EA
全局位置型变量 0001:关节 6	32-bit float	0x010C	2	值改变立即生效，偏移地址: 0x0020 * n + 0x00EC
全局位置型变量	32-bit float	0x010E	2	值改变立即生效，偏移

0001:关节 7				地址: 0x0020 * n + 0x00EE
全局位置型变量 0001:关节 8	32-bit float	0x0110	2	值改变立即生效，偏移 地址: 0x0020 * n + 0x00F0
全局位置型变量 0001:笛卡尔 X	32-bit float	0x0112	2	值改变立即生效，偏移 地址: 0x0020 * n + 0x00F2
全局位置型变量 0001:笛卡尔 Y	32-bit float	0x0114	2	值改变立即生效，偏移 地址: 0x0020 * n + 0x00F4
全局位置型变量 0001:笛卡尔 Z	32-bit float	0x0116	2	值改变立即生效，偏移 地址: 0x0020 * n + 0x00F6
全局位置型变量 0001:笛卡尔 R	32-bit float	0x0118	2	值改变立即生效，偏移 地址: 0x0020 * n + 0x00F8
全局位置型变量 0001:笛卡尔 P	32-bit float	0x011A	2	值改变立即生效，偏移 地址: 0x0020 * n + 0x00FA
全局位置型变量 0001:笛卡尔 Y	32-bit float	0x011C	2	值改变立即生效，偏移 地址: 0x0020 * n + 0x00FC
……	……	……	……	……
全局位置型变量	16-bit	0x7DE0	1	值改变立即生效

1000:坐标系类型	integer			
全局位置型变量 1000:坐标系编号	16-bit integer	0x7DE1	1	值改变立即生效
全局位置型变量 1000:关节 1	32-bit float	0x7DE2	2	值改变立即生效
全局位置型变量 1000:关节 2	32-bit float	0x7DE4	2	值改变立即生效
全局位置型变量 1000:关节 3	32-bit float	0x7DE6	2	值改变立即生效
全局位置型变量 1000:关节 4	32-bit float	0x7DE8	2	值改变立即生效
全局位置型变量 1000:关节 5	32-bit float	0x7DEA	2	值改变立即生效
全局位置型变量 1000:关节 6	32-bit float	0x7DEC	2	值改变立即生效
全局位置型变量 1000:关节 7	32-bit float	0x7DEE	2	值改变立即生效
全局位置型变量 1000:关节 8	32-bit float	0x7DF0	2	值改变立即生效
全局位置型变量 1000:笛卡尔 X	32-bit float	0x7DF2	2	值改变立即生效
全局位置型变量	32-bit	0x7DF4	2	值改变立即生效

1000:笛卡尔 Y	float			
全局位置型变量 1000:笛卡尔 Z	32-bit float	0x7DF6	2	值改变立即生效
全局位置型变量 1000:笛卡尔 R	32-bit float	0x7DF8	2	值改变立即生效
全局位置型变量 1000:笛卡尔 P	32-bit float	0x7DFA	2	值改变立即生效
全局位置型变量 1000:笛卡尔 Y	32-bit float	0x7DFC	2	值改变立即生效
全局数值型变量 0001	32-bit float	0x7E10	2	值改变立即生效，偏移 地址: 0x0002 * n + 0x7E0E
……	……	……	……	……
全局数值型变量 0300	32-bit float	0x8066	2	值改变立即生效

2.3 Coil 线圈

数字量输出：4 组，每组 16 个，共计 48 个

名称	数据类型	地址	备注
DO1.1~DO1.16	1-bit	0x0000~0x000F	值改变立即生效
DO2.1~DO2.16	1-bit	0x0010~0x001F	值改变立即生效
DO3.1~DO3.16	1-bit	0x0020~0x002F	值改变立即生效

DO4.1~DO4.16	1-bit	0x0030~0x003F	值改变立即生效
DO5.1~DO5.16	1-bit	0x0040~0x004F	值改变立即生效
DO6.1~DO6.16	1-bit	0x0050~0x005F	值改变立即生效
DO7.1~DO7.16	1-bit	0x0060~0x006F	值改变立即生效
DO8.1~DO8.16	1-bit	0x0070~0x007F	值改变立即生效

2.4 Discrete_Input 只读

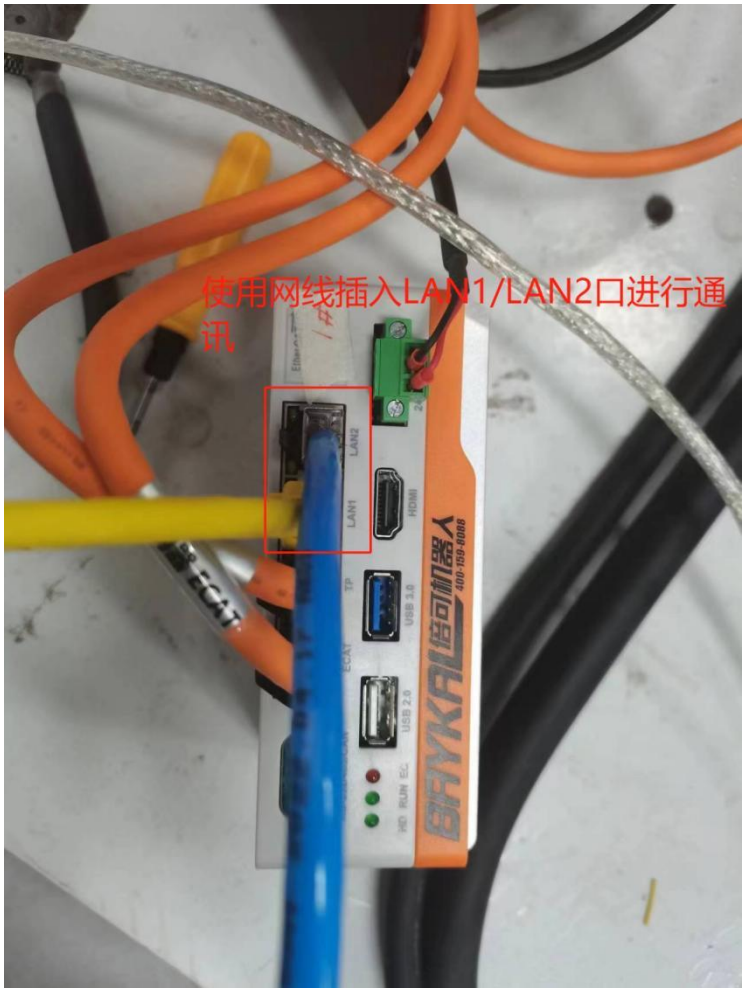
数字量输入：4 组，每组 16 个，共计 48 个

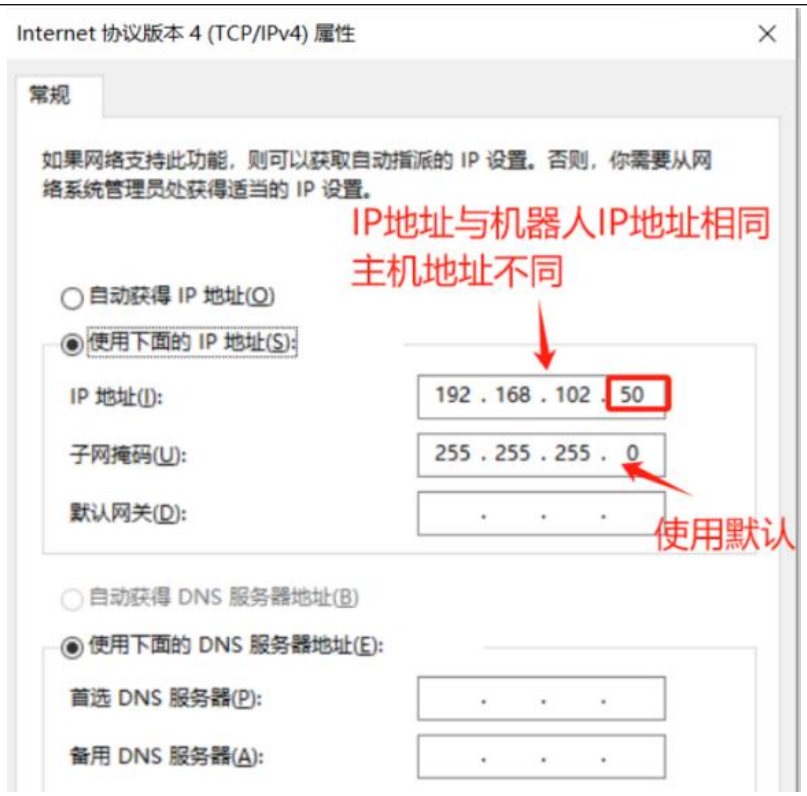
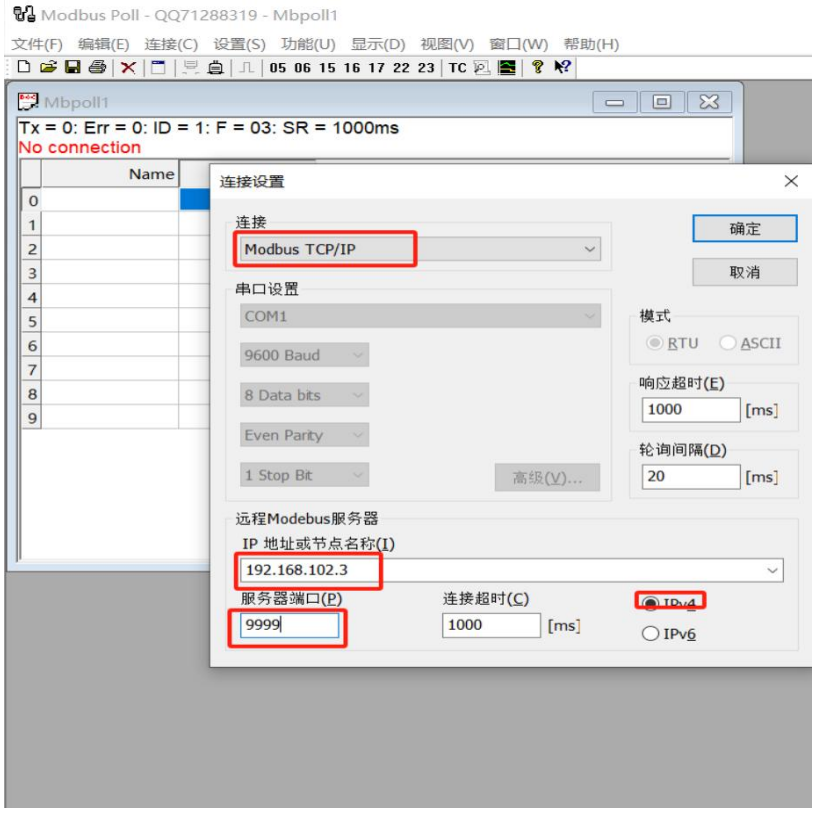
名称	数据类型	地址	备注
DI1.1~DI1.16	1-bit	0x0000~0x000F	
DI2.1~DI2.16	1-bit	0x0010~0x001F	
DI3.1~DI3.16	1-bit	0x0020~0x002F	
DI4.1~DI4.16	1-bit	0x0030~0x003F	
DI5.1~DI5.16	1-bit	0x0040~0x004F	
DI6.1~DI6.16	1-bit	0x0050~0x005F	
DI7.1~DI7.16	1-bit	0x0060~0x006F	
DI8.1~DI8.16	1-bit	0x0070~0x007F	

3 通讯示例

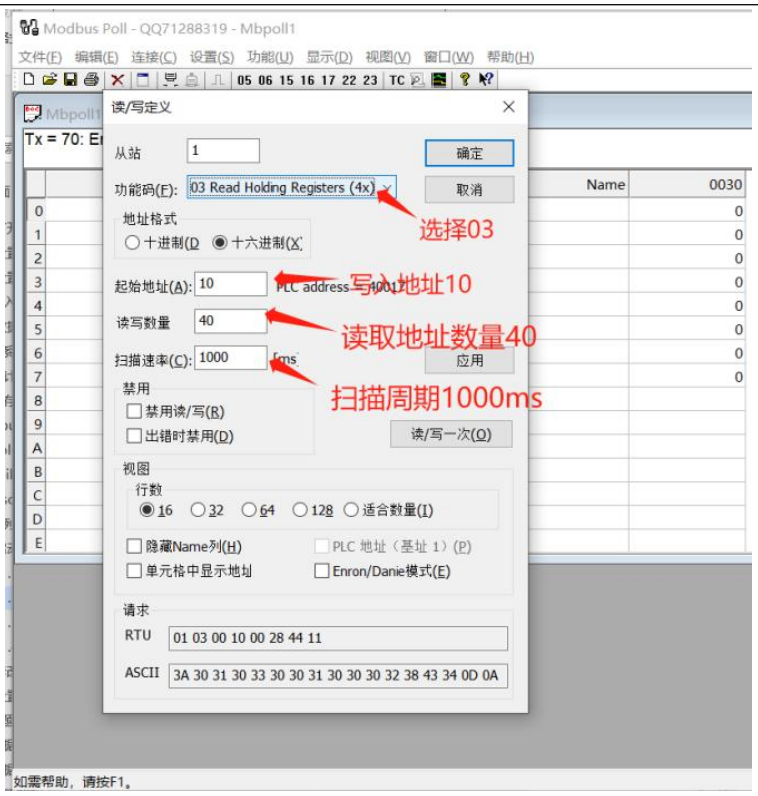
Modbus 有众多客户端，本文档中采用 Modbus poll 工具演示如何通过 Moubus 与机器人系统进行通讯。Modbus 通讯仅在机器人处于远程模式下进行。

3.1 操作示例

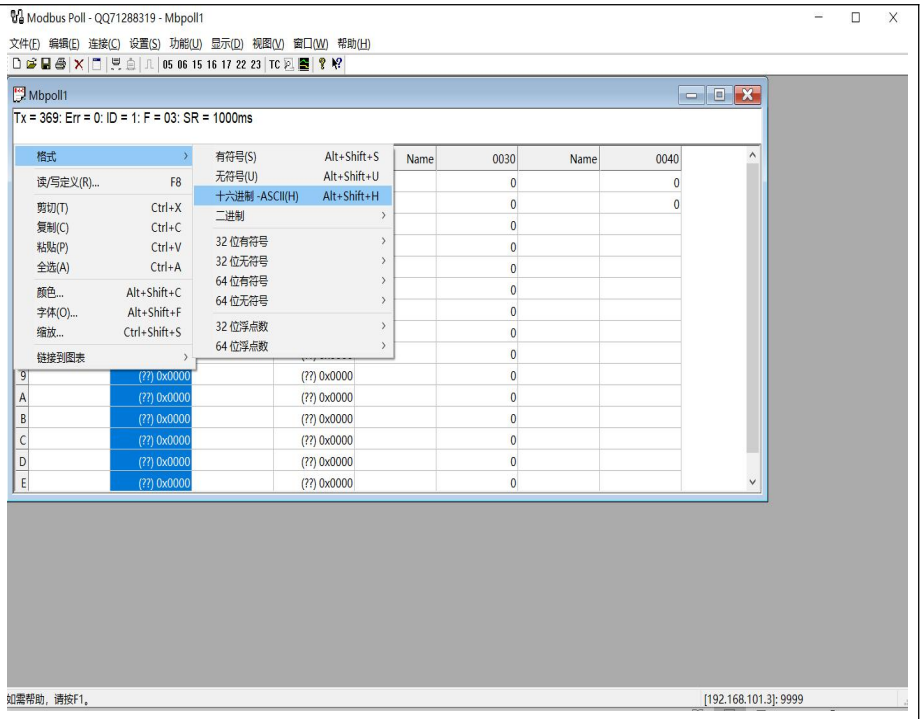
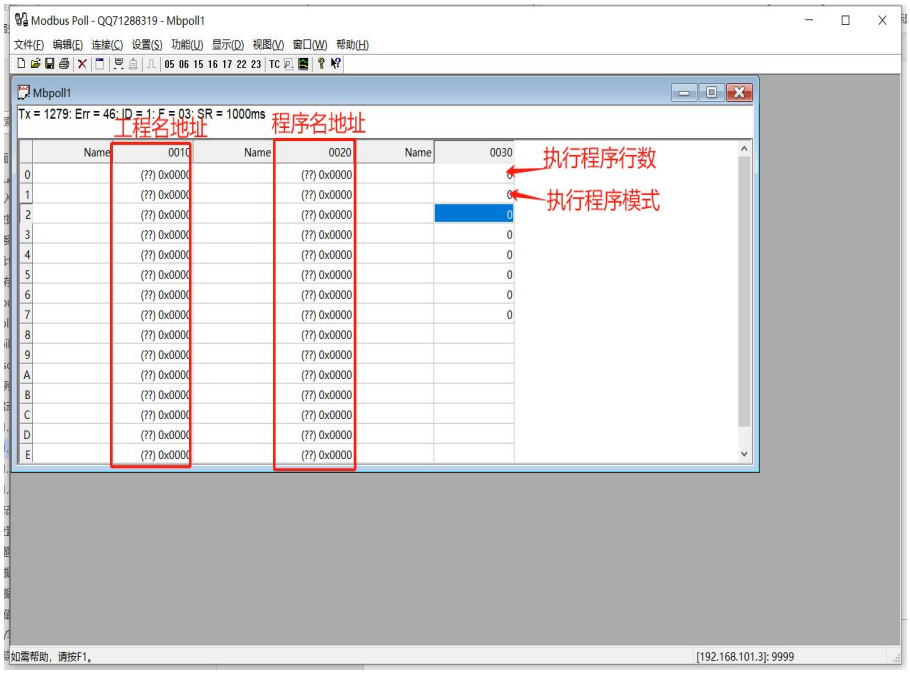
项目	操作步骤
连接控制器	 网口 Lan1 或 Lan2 与电脑通过网线连接

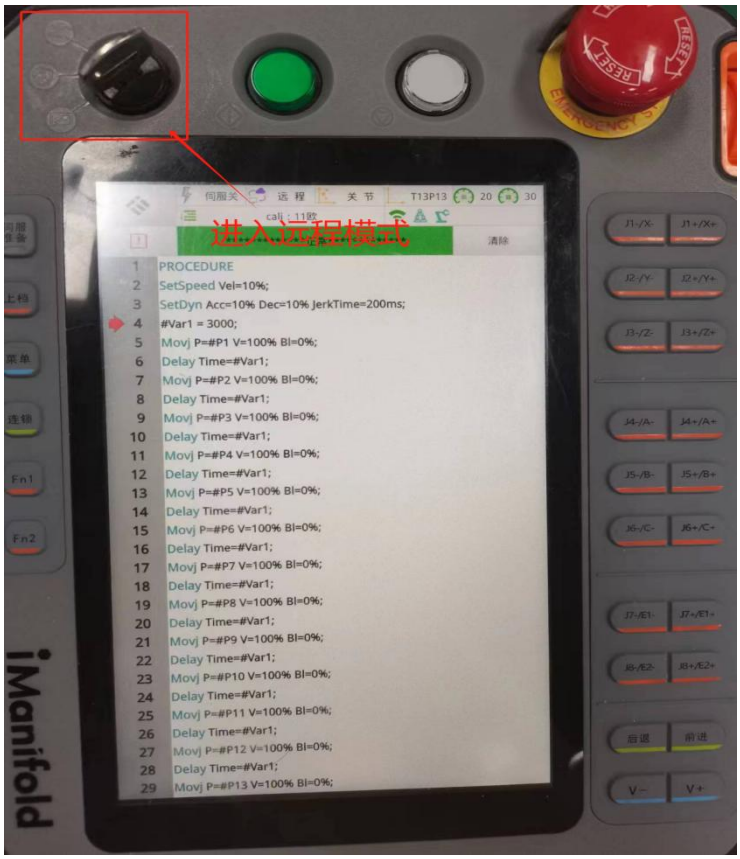
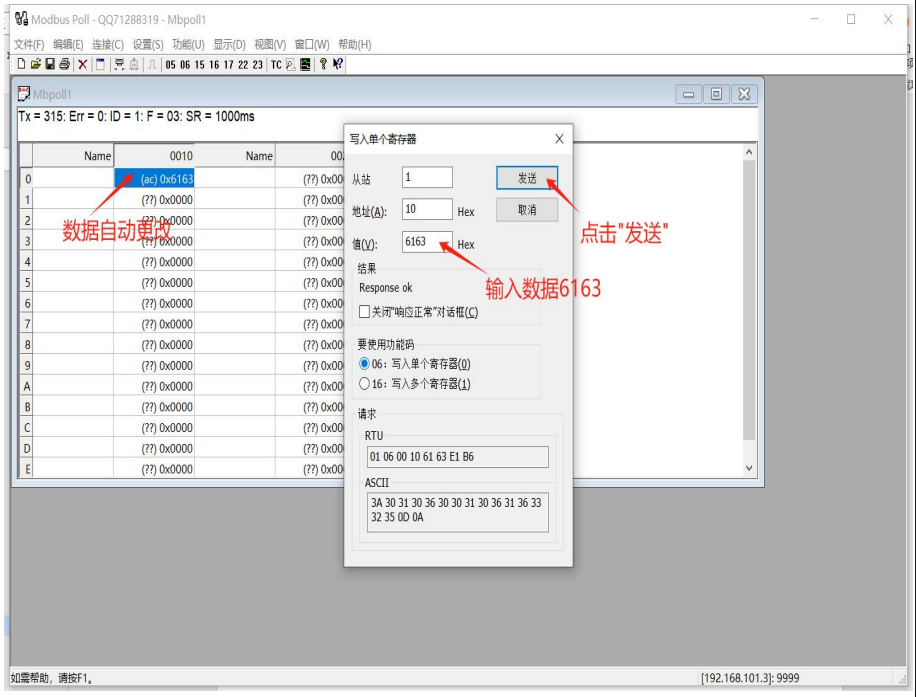
设置网络	 需要电脑 IP 与控制器 IP 保持一致
连接设置	 打开 Modbus Poll 后打开连接界面，连接协议为 Modbus TCP/IP；

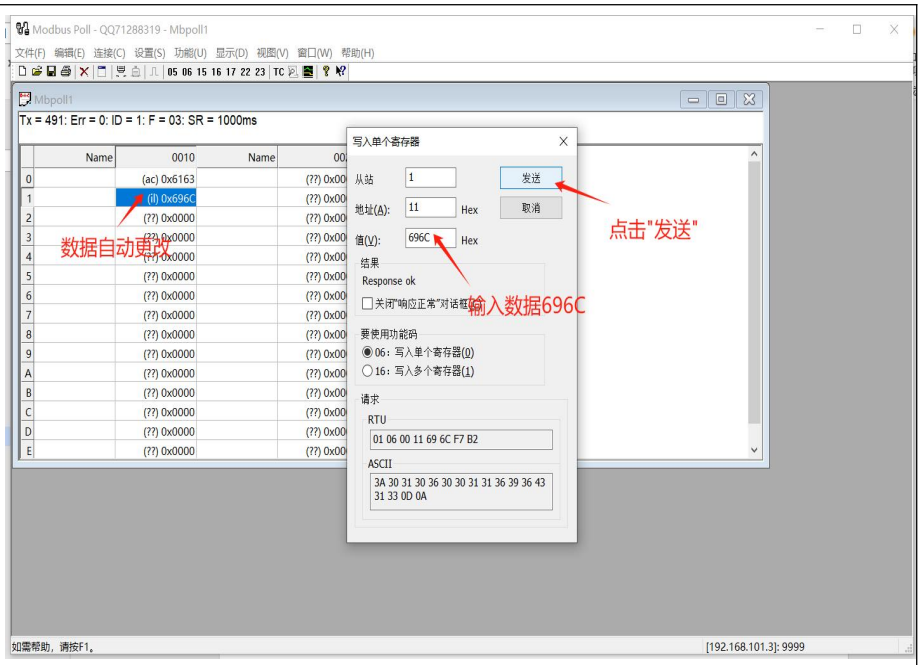
	IP 地址要与机器人 IP 地址相同；服务器端口为 9999；协议版本为 Ipv4；其余默认。
项目示例	 演示：用 Modbus poll 启动一个工程名为 cali，程序名为 11 欧，执行程序模式为多次循环,从程序第 11 行开始执行

<table>
<tr>
<td>设置读/写定义</td>
<td>


根据使用的功能码不同，需要找到对应的功能地址表进行对应；
地址格式默认为 10 进制，必需改为 16 进制，否则数据地址将会错误；</td>
</tr>
<tr>
<td>功能表地址切换</td>
<td>
<table>
<tr><td>执行工程名</td><td>32-byte string</td><td>0x0010</td><td>16</td><td>以字节流保存报警信息</td></tr>
<tr><td>执行程序名</td><td>32-byte string</td><td>0x0020</td><td>16</td><td>以字节流保存报警信息</td></tr>
<tr><td>执行程序行数</td><td>16-bit integer</td><td>0x0030</td><td>1</td><td></td></tr>
<tr><td>执行程序模式</td><td>16-bit integer</td><td>0x0031</td><td>1</td><td>0：一次；
1：步进；
2：循环；</td></tr>
</table>
根据功能地址表找到执行工程名，执行程序名，执行程序行数，执行模式地址不同地址需要输入的数据类型不同，需要转换数据类型。需要使用转换工具转换为 16 进制。
转换工具（http://www.kjson.com/transform/ox2str/）</td>
</tr>
</table>

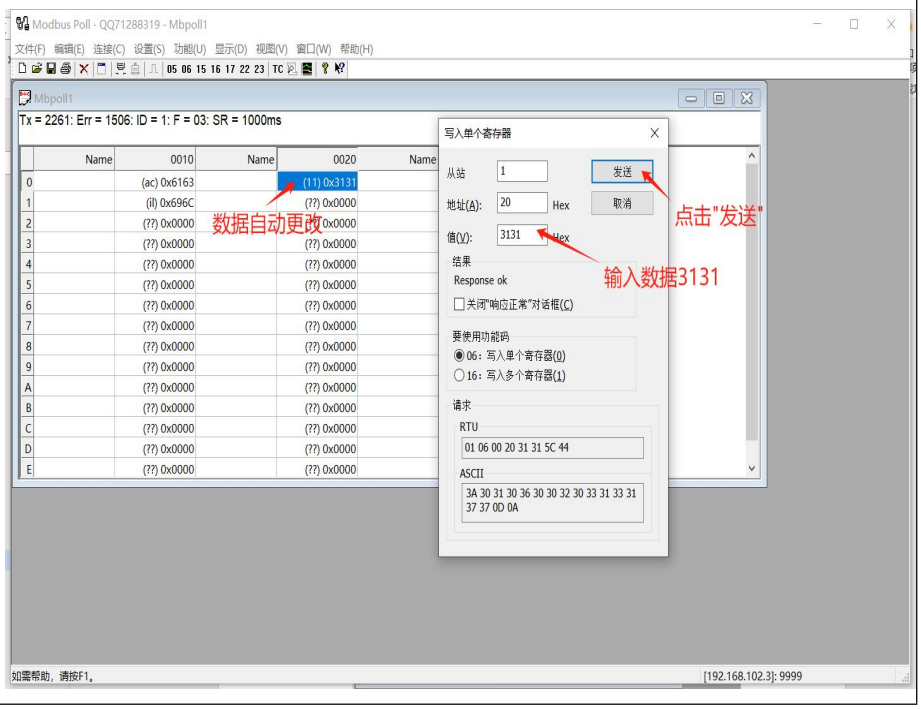
格式选择	 选中地址 0010、00020，更改地址格式为“16 进制-ASCII”
数据地址	 根据地址表，找到 0x0030 执行程序行数、0x0031 执行程序模式地址进行设置

切换模式	 进入远程模式 PROCEDURE SetSpeed Vel=10%; SetDyn Acc=10% Dec=10% JerkTime=200ms; #Var1 = 3000; Movj P=#P1 V=100% Bl=0%; Delay Time=#Var1; Movj P=#P2 V=100% Bl=0%; Delay Time=#Var1; Movj P=#P3 V=100% Bl=0%; Delay Time=#Var1; Movj P=#P4 V=100% Bl=0%; Delay Time=#Var1; Movj P=#P5 V=100% Bl=0%; Delay Time=#Var1; Movj P=#P6 V=100% Bl=0%; Delay Time=#Var1; Movj P=#P7 V=100% Bl=0%; Delay Time=#Var1; Movj P=#P8 V=100% Bl=0%; Delay Time=#Var1; Movj P=#P9 V=100% Bl=0%; Delay Time=#Var1; Movj P=#P10 V=100% Bl=0%; Delay Time=#Var1; Movj P=#P11 V=100% Bl=0%; Delay Time=#Var1; Movj P=#P12 V=100% Bl=0%; Delay Time=#Var1; Movj P=#P13 V=100% Bl=0%; iManifold
写入工程名	 Modbus Poll - QQ71288319 - Mbpoll1 Mbpoll1 Tx = 315: Err = 0: ID = 1: F = 03: SR = 1000ms 写入单个寄存器 从站 1 地址(A): 10 Hex 值(V): 6163 Hex 发送 取消 结果 Response ok 关闭"响应正常"对话框(C) 要使用功能码 06：写入单个寄存器(0) 16：写入多个寄存器(1) 请求 RTU 01 06 00 10 61 63 E1 B6 ASCII 3A 30 31 30 36 30 30 31 30 36 31 36 33 32 35 0D 0A 数据自动更改 点击"发送" 输入数据6163 如需帮助，请按F1。 [192.168.101.3]: 9999

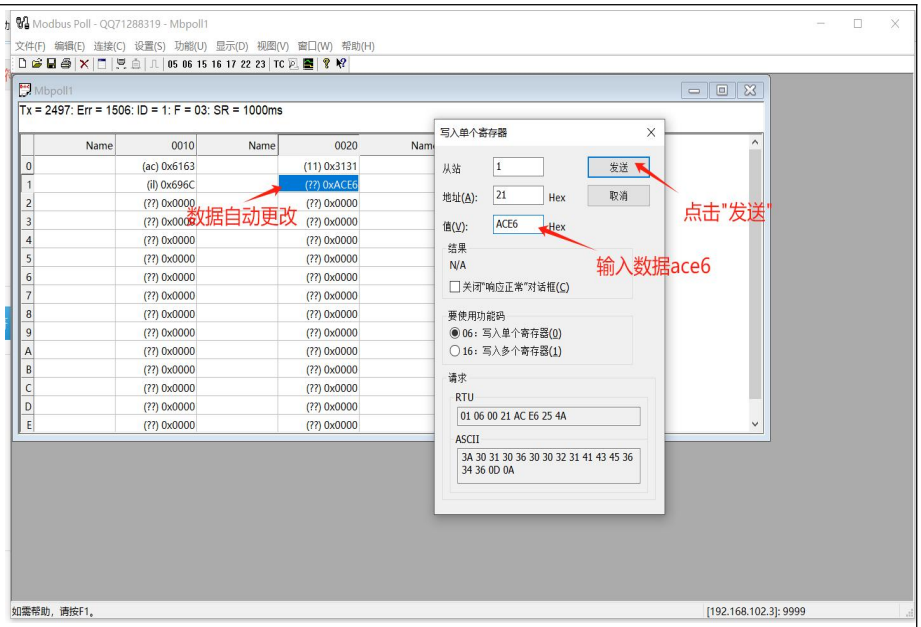
<table>
<tr>
<td></td>
<td>


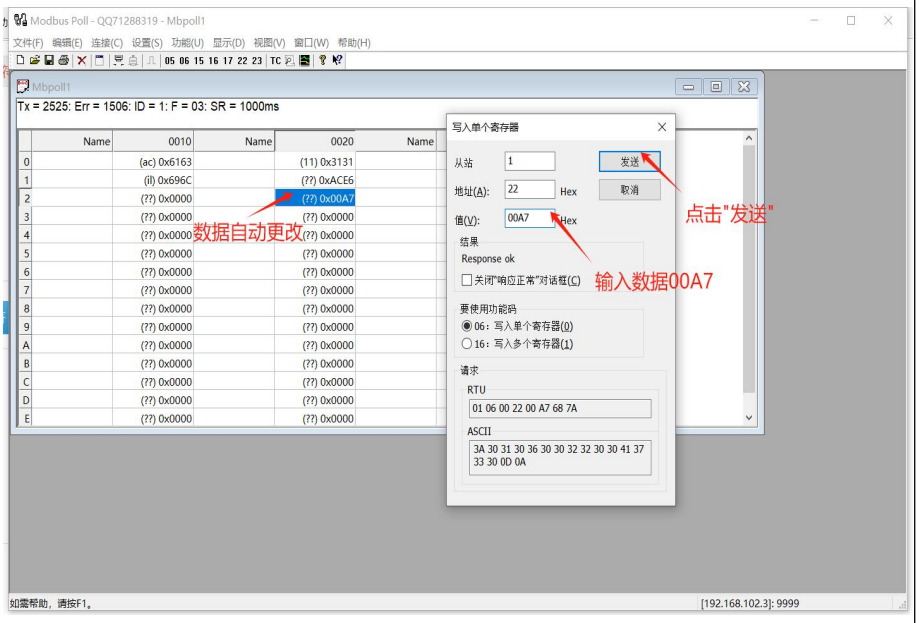
本功能数据传输格式使用 little_endian_swap，需要注意字节序。

将工程名“cali”用转换工具转换为 16 进制码。

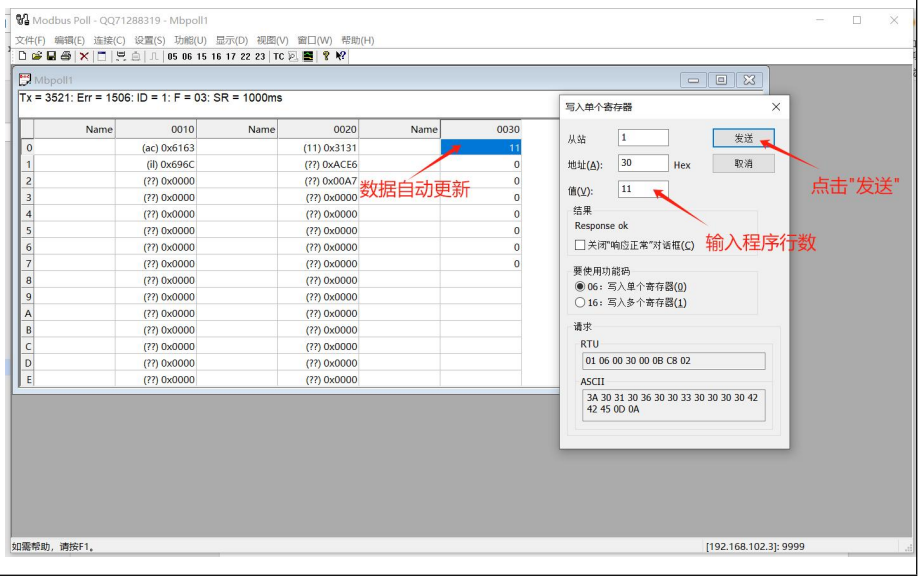
转换工具（http://www.kjson.com/transform/ox2str/）</td>
</tr>
<tr>
<td>写入程序名</td>
<td>


</td>
</tr>
</table>

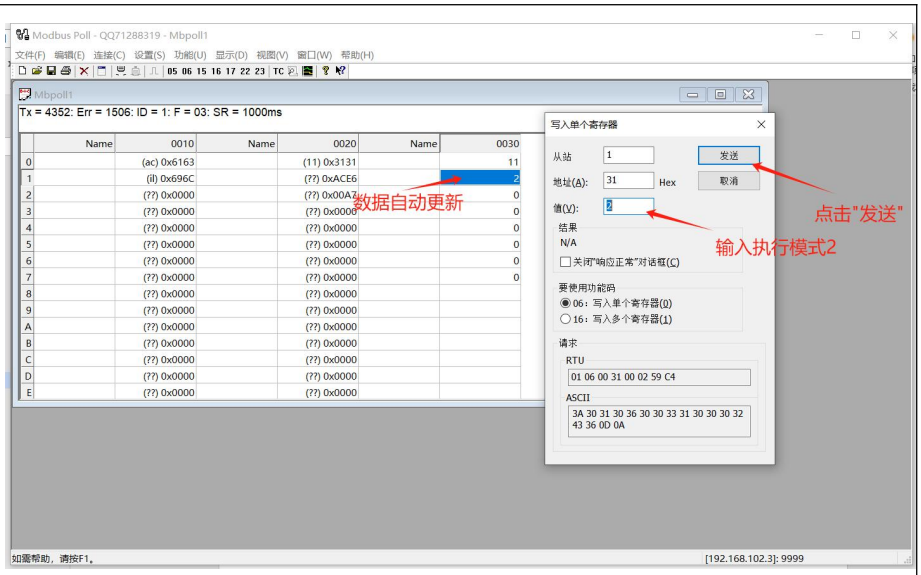

Modbus Poll - QQ71288319 - Mbpoll1
Mbpoll1
Tx = 2497: Err = 1506: ID = 1: F = 03: SR = 1000ms
写入单个寄存器
从站 1
地址(A): 21 Hex
值(V): ACE6 Hex
发送
取消
结果
N/A
关闭"响应正常"对话框(C)
要使用功能码
06：写入单个寄存器(0)
16：写入多个寄存器(1)
请求
RTU
01 06 00 21 AC E6 25 4A
ASCII
3A 30 31 30 36 30 30 32 31 41 43 45 36 34 36 0D 0A
数据自动更改
点击"发送"
输入数据ace6
如需帮助，请按F1。
[192.168.102.3]: 9999


Modbus Poll - QQ71288319 - Mbpoll1
Mbpoll1
Tx = 2525: Err = 1506: ID = 1: F = 03: SR = 1000ms
写入单个寄存器
从站 1
地址(A): 22 Hex
值(V): 00A7 Hex
发送
取消
结果
Response ok
关闭"响应正常"对话框(C)
要使用功能码
06：写入单个寄存器(0)
16：写入多个寄存器(1)
请求
RTU
01 06 00 22 00 A7 68 7A
ASCII
3A 30 31 30 36 30 30 32 32 30 30 41 37 33 30 0D 0A
数据自动更改
点击"发送"
输入数据00A7
如需帮助，请按F1。
[192.168.102.3]: 9999

写入执行程序行数

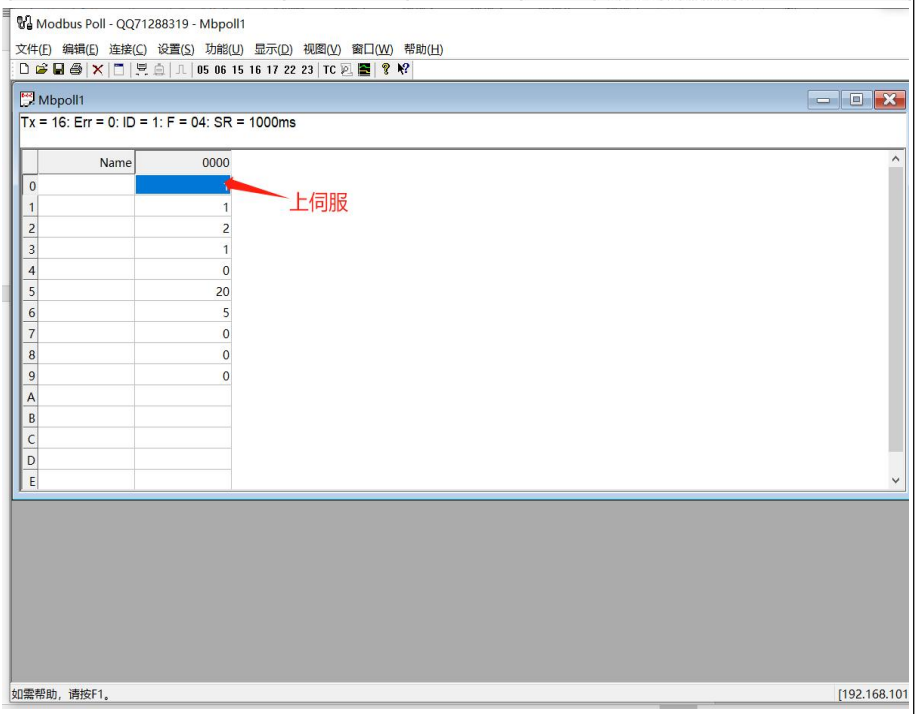

Modbus Poll - QQ71288319 - Mbpoll1
Mbpoll1
Tx = 3521: Err = 1506: ID = 1: F = 03: SR = 1000ms
写入单个寄存器
从站 1
地址(A): 30 Hex
值(V): 11
发送
取消
结果
Response ok
关闭"响应正常"对话框(C)
要使用功能码
06：写入单个寄存器(0)
16：写入多个寄存器(1)
请求
RTU
01 06 00 30 00 0B C8 02
ASCII
3A 30 31 30 36 30 30 33 30 30 30 30 42 42 45 0D 0A
数据自动更新
点击"发送"
输入程序行数
如需帮助，请按F1。
[192.168.102.3]: 9999

写入执行程序模式



执行伺服

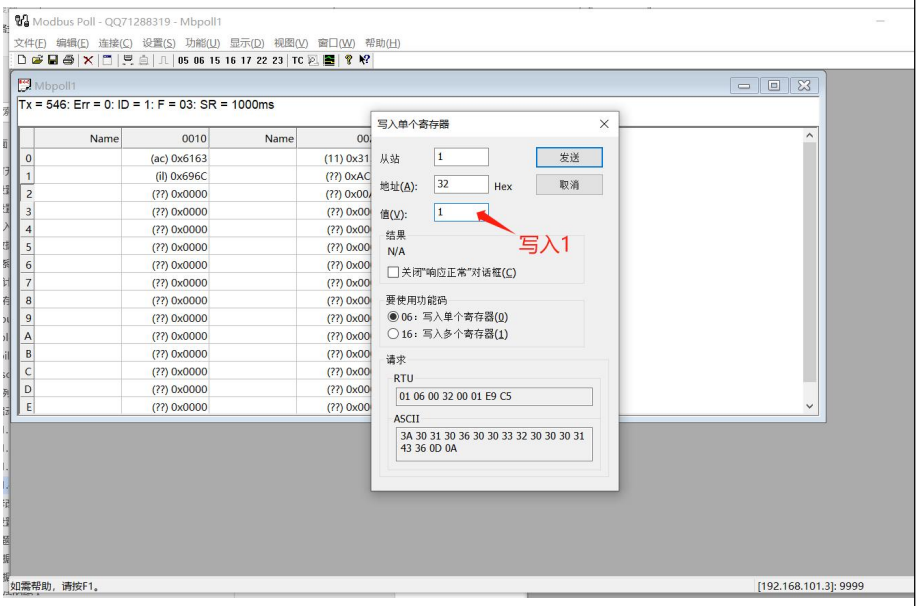
伺服上使能	16-bit integer	0x0000	1	上升沿有效，生效后自动复位； 如当前未使能，则该信号触发后伺服使能
伺服去使能	16-bit integer	0x0001	1	上升沿有效，生效后自动复位； 如当前已经使能，则该信号触发后伺服去使能



根据 Holding_Register 命令寄存器功能地址表，找到上伺服地址进行远程上伺服

执行程序	16-bit integer	0x0032	1	沿有效，生效后自动复位； 该信号触发后执行程序
暂停程序	16-bit integer	0x0033	1	上升沿有效，生效后自动复位； 该信号触发后暂停程序
继续执行	16-bit integer	0x0034	1	上升沿有效，生效后自动复位； 该信号触发后停止程序
停止程序	16-bit integer	0x0035	1	上升沿有效，生效后自动复位； 该信号触发后停止程序

执行程序



根据地址表，找到 0x0032 地址，写入数值 1，程序执行

4 常见问题

4.1 数据类型的转换

Modbus poll 不支持字符串、ACSII 码需要转换为 16 进制。

中文、中文符号需要使用字符串转 16 进制工具。

例： “圈”→“e59c88”

转换工具（http://www.kjson.com/transform/ox2str/）

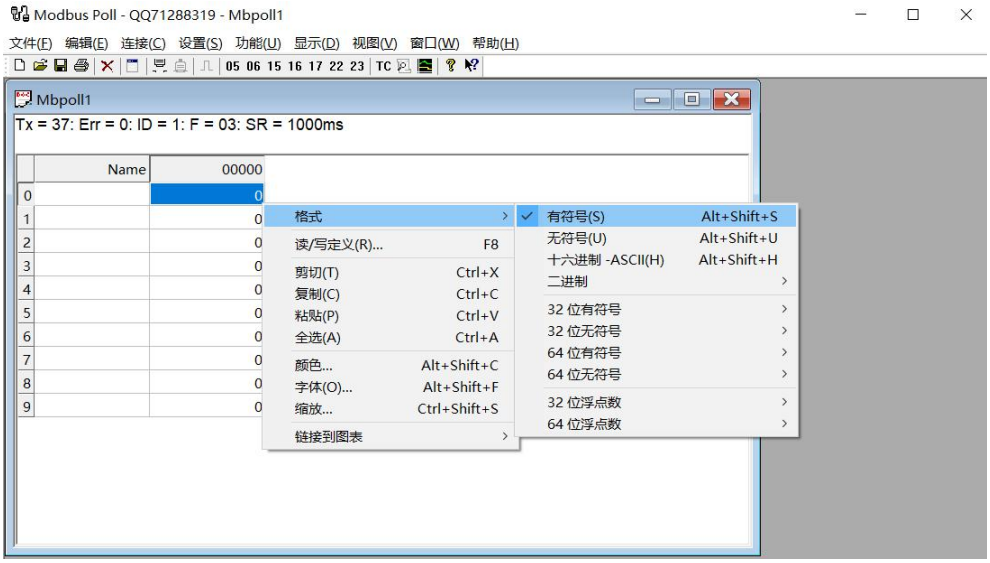
ACSII 码需要使用 ASCII 码转 16 进制工具。

例：“cali”→“63616C69”

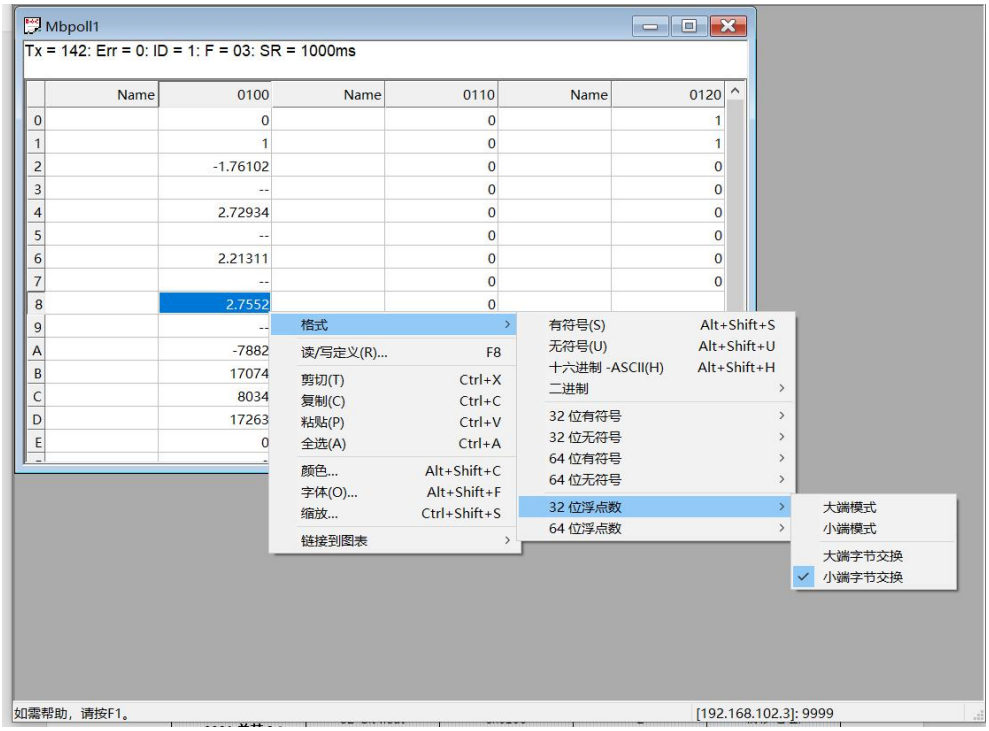
4.2 数据格式

本功能数据传输格式使用 little_endian_swap。功能表数据类型分为 16-bit intege、32-bit float、32-byte string、128-byte string 查看数据时需要注意数据类型。

Modbus Poll 默认为“16-bit intege（有符号）”。



若数据类型为“32-bit float”，则需要将格式设置为“32 位浮点数”→“小端字节交换”才可查看数据。



4.3 数值的输入

地址数值输入需要看功能码的备注栏进行输入，上升沿有效命令字只能输入 1。

字符串数据输入到地址需要更改格式为“16 进制-ASCII”，再双击需要更改的数值即可修改（使用“写入单个寄存器”功能默认是 10 进制输入，无法写入 16 进制数据）。

输入 16 进制码时，注意 4 个字为 1 组且需要注意字节序，数据超过 4 位时，需要输入多个地址。

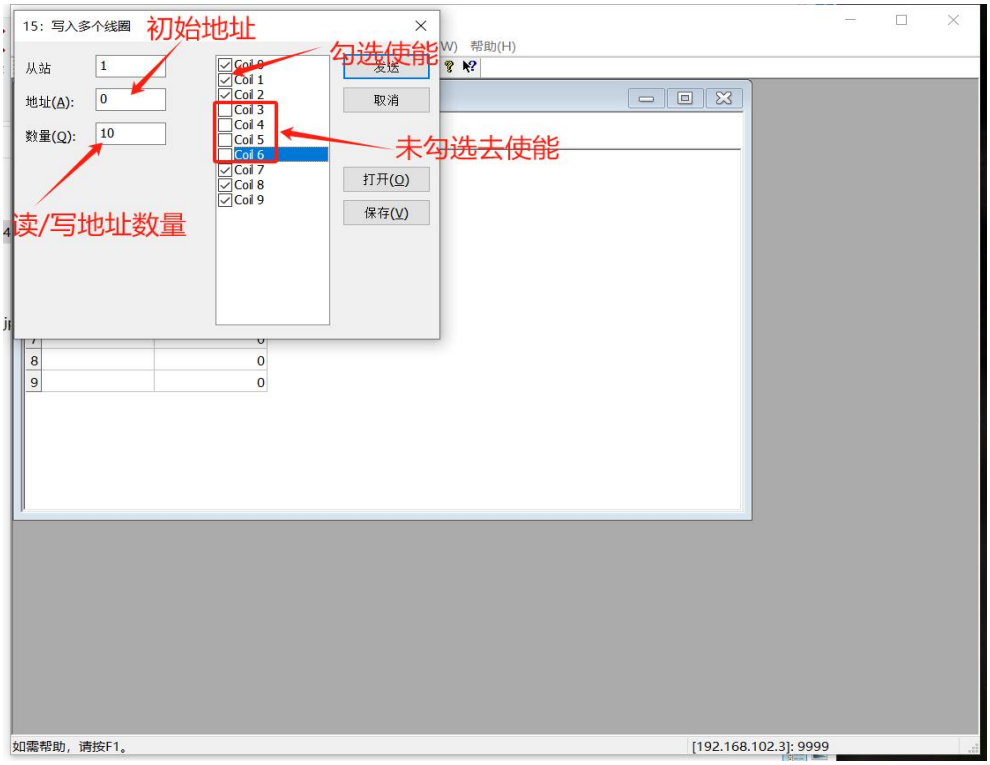
例：“圈”→“。e59c88”输入地址时写成“0x9ce5”“0x0088”

4.4 读/写定义

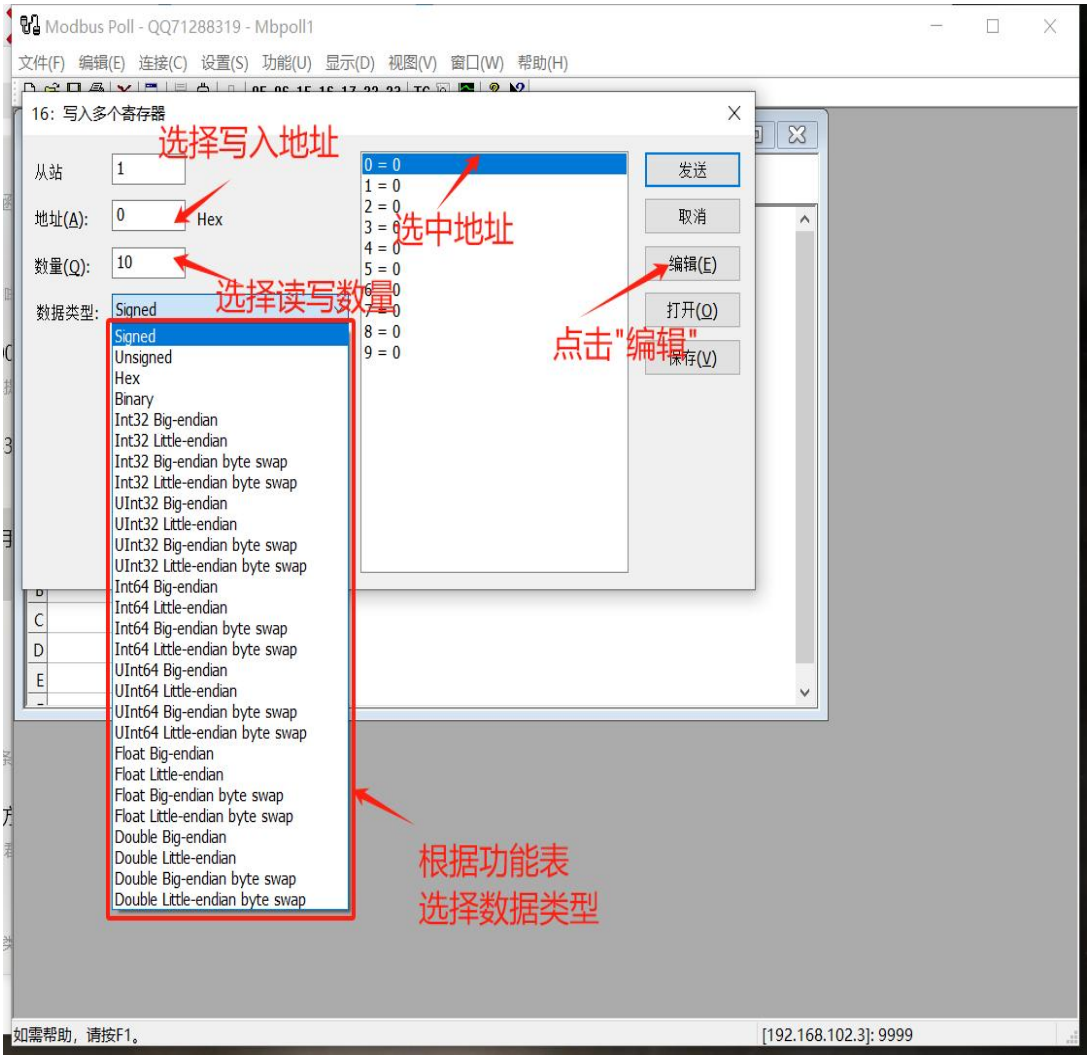
读/写定义中，从站固定为 1；功能码共 4 个：01、02、03、04；地址格式默认为 10 进制，需要改为 16 进制，否则数据地址将会错误；扫描速率默认 1000ms

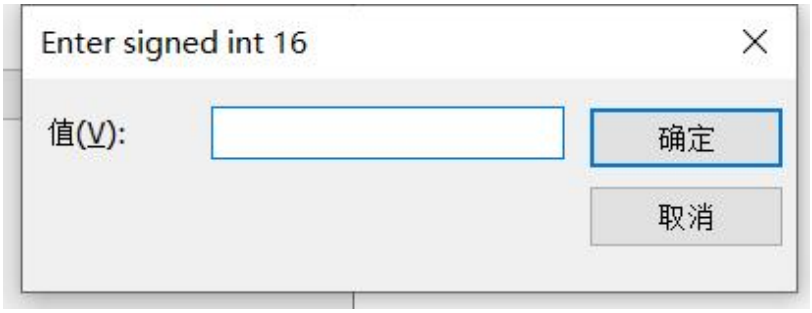
4.5 写入多个线圈/多个寄存器

Modbus 支持多个线圈/寄存器输入，例：多个线圈使能



例：多个寄存器写入


Modbus Poll - QQ71288319 - Mbpoll1
文件(F) 编辑(E) 连接(C) 设置(S) 功能(U) 显示(D) 视图(V) 窗口(W) 帮助(H)
16: 写入多个寄存器
选择写入地址
从站 1
地址(A): 0 Hex
数量(Q): 10
数据类型: Signed
选择读写数量
选中地址
点击"编辑"
根据功能表
选择数据类型
发送
取消
编辑(E)
打开(O)
保存(V)
0 = 0
1 = 0
2 = 0
3 = 0
4 = 0
5 = 0
6 = 0
7 = 0
8 = 0
9 = 0
Signed
Unsigned
Hex
Binary
Int32 Big-endian
Int32 Little-endian
Int32 Big-endian byte swap
Int32 Little-endian byte swap
UInt32 Big-endian
UInt32 Little-endian
UInt32 Big-endian byte swap
UInt32 Little-endian byte swap
Int64 Big-endian
Int64 Little-endian
Int64 Big-endian byte swap
Int64 Little-endian byte swap
UInt64 Big-endian
UInt64 Little-endian
UInt64 Big-endian byte swap
UInt64 Little-endian byte swap
Float Big-endian
Float Little-endian
Float Big-endian byte swap
Float Little-endian byte swap
Double Big-endian
Double Little-endian
Double Big-endian byte swap
Double Little-endian byte swap
如需帮助，请按F1。
[192.168.102.3]: 9999


Enter signed int 16
值(V):
确定
取消

4.6 Modbus poll 错误码

No connection	正常未连接的状态，正常 COM 口未被占用的情况下连接即可消除。
Timeout	所有由软件发出指令，从机设备未回复的情况，都会显示 Timeout。而从机设备不回复，可能性有很多，比如： 1、连接配置错误，主机的波特率、Slave ID 等信息跟从机设备对应不上，从机就不会回。 2、线路异常，电脑跟从机设备之前的通信线存在异常，也是无法正常收到回复。 3、从机设备解析异常不回复，这个具体可以查看 Modbus 协议详解。
Write error/Read error	使用的是 USB 转 485 的工具，调换 485 的 A、B 线，可能会出现这种情况。 使用的是 USB 转 232 或 TTL 的工具，则短接 Tx 和 Rx 就会出现这种情况。 使用网线连接时，断开网线会出现这种情况。 在发送的过程中，收到数据，即总线上数据冲突，也会出现类似的错误。

Checksum Error	从机设备返回的 CRC 校验不正确。 通信总线上存在干扰。 连接配置里的校验码、数据位、停止位配置错误。
Insufficient bytes received	接收的字节数不完整，可能由于线路或某种原因，导致返回的指令长度与理论上返回的指令长度对不上，则会报些错误。
Illegal Function	功能码异常，一般是当访问的从机设备不存在可操作的功能码时，会返回不存在此功能码不存在的 01 异常码，软件接收到此指令时就会报出这个错误。
Illegal Data Address	地址异常，一般是当访问的从机设备不存在要读取的寄存器/线圈地址时，会返回不存在此地址的 02 异常码，软件接收到此指令时就会报出这个错误。
Illegal Data Value	数据异常，一般是当前所要读/写的数据，从机设备不允许此地址数据的操作，会返回数据不可操作的 03 异常码，软件接收到此指令时就会报出这个错

	误。
Slave Device Busy	从机繁忙，一般是从机设备回复了 06 的异常码 ，软件接收到此指令时就会报出这个错误。
Response Error	响应异常，一般是从机设备回复了一些无法识别的指令，就会报这个异常，另外一个不知道算不算是这个软件的 Bug，一般广播指令（0 地址）是不可以读数据，只能广播写，但这个软件可以设置用广播指令读取，当这样设置时，就会出现这个"Response Error"的异常。