EasySmc 5470 运动控制器

使

用

说

明

厦门华菱工控技术有限公司

Xiamen Hualing Industry Control Technology Co.,Ltd

一、概述 1
二、性能指标 1
三、运动控制系统组成框图 1
3.1 PLC作为上位机，如松下的FP0-C10CR，带RS-232C串口 1
3.2 工业控制计算机作为上位机，与 EasySmc 5470 构成主从系统 2
3.3 EasyHmi作为上位机，可最大限度的提高性能价格比，需要用户订制 2
四、接线说明和跳线定义 2
4.1 外形尺寸说明 3
4.2 接线端说明 3
4.2.1 电源输入输出信号 3
4.2.2 通信口 3
4.2.3 电机驱动信号 5
4.2.4 下限输入，上限输入，原点输入……16 输入信号 5
4.2.5 每轴 4 路输出信号，X，Y，Z，U共 16 输出信号 6
4.2.6 紧急故障停止输入信号（EEM） 6
4.2.7 手动信号输入（A/M，L/H） 6
4.3 手动/联机跳线说明 6
4.4 状态灯输出等等 7
五、控制器Modbus寻址空间分配 7
5.1 读离散量输入（地址范围 0…68） 7
5.2 读单个线圈（地址范围 0…15） 7
5.3 写单个线圈（地址范围 0…15） 7
5.4 写多个线圈（地址范围 0…15） 7
5.5 输入寄存器（地址范围 0…199） 7
5.6 读、写保持寄存器（地址范围 0…25343） 8
5.7 读文件记录 12
5.8 写文件记录 12
六、串行口通信modbus_serial.dll使用说明 13
6.1 modbus_serial.dll Visual C++函数声明 13
6.2 modbus_serial.dll Delphi 函数声明 18
6.3 modbus_serial.dll Visual Basic 函数声明 20
6.4 modbus_serial.dll 调用顺序 22
七、USB通信modbus_usb.dll使用说明 23
7.1 modbus_usb.dll Delphi函数声明 23
7.2 modbus_usb.dll Visual C++函数声明 27
7.3 modbus_usb.dll Visual Basic函数声明 29
八、以太网通信modbus_tcp.dll使用说明 31
8.1 modbus_tcp.dll Delphi函数声明 31
8.2 modbus_tcp.dll Visual C++函数声明 36
8.3 modbus_tcp.dll Visual Basic函数声明 39
8.4 modbus_tcp.dll调用顺序 41
九、G代码、M、F、T代码说明 42
十、故障检查、维修保养、售后服务 43

EasySmc5470（步进/伺服电机）运动控制器 使用说明

V2011-07-16

一、概述

EasySmc 5470 采用 208MHz 高速 ARM 32 位 处理器，配合 FPGA 实现 4 轴运动控制。

控制器使用标准 Modbus 通信协议,物理层采用异步串行接口（RS232、RS422、RS485）或者 USB 或者以太网与外部通信，可以和 PLC，IPC，单片机系统组成运动控制系统。控制器执行运动控制的所有细节，包括脉冲和方向信号的产生，原点的复位，上下限位开关的检测，自动加减速，插补运算等等，能很方便的替代 PLC 的位置控制模块，IPC 的运动控制卡，极大的提高性能价格比。由于采用集散控制，上位机（PLC、IPC、单片机系统）能从实时的运动控制任务中摆脱出来，更好的执行人机交互和其它控制任务。

控制器直接接收执行 G 代码，执行代码易读，易修改。

控制器执行 8 种复位运动，独立运动，（2-4 轴）直线插补，两轴圆弧插补，螺旋插补。

控制器具有“小线段连续插补”功能，极大的提高小线段插补的执行效率。

控制器具有脱机执行功能（最多 100 个文件），脱机程序存储在 U 盘中，G 代码长度没有限制。并且控制器为每个脱机文件分别建立 6 个工件偏移量和 6 个工具偏移量。

控制器具有提前输出功能，提前量用毫秒级时间控制，特别适用于点胶机系统的提前关胶。

控制器固件通过 U 盘升级，可以快速，准确的为用户订制特殊功能。

二、性能指标

1、单电源供电：+24V 400mA 直流；

2、X，Y，Z，U 四轴独立、直线插补、圆弧插补控制；

3、脉冲输出频率最高 2MHz，旋转编码器接收最高频率 2MHz；

4、梯形加减速度，防止运动过程中的失步和过冲；

5、单轴最大步进步数 －134，217，728 — ＋ 134，217，727；

6、每轴各带一上限，下限的软件限位寄存器；旋转编码器输入。

7、每轴带一原点信号，上限、下限信号输入，外部启动联锁信号；系统带一紧急故障输入信号；

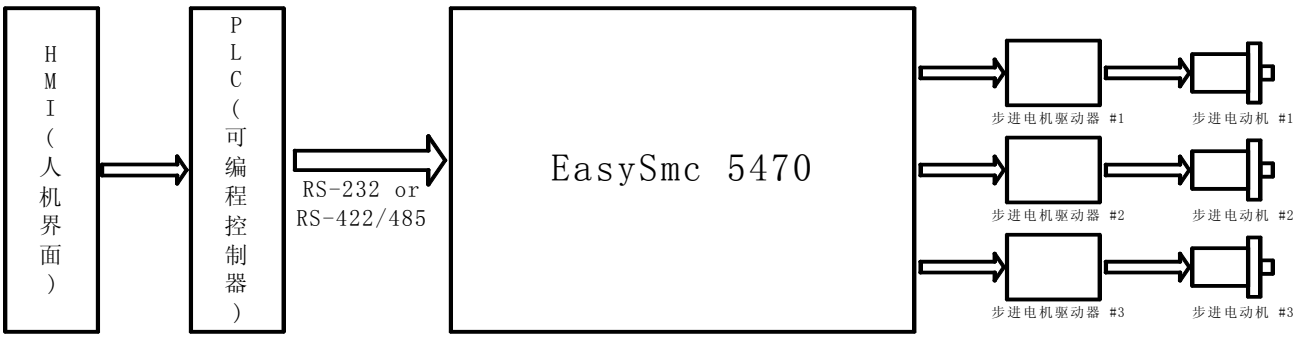
8、控制器具有手/自动切换功能，方便调试；控制器可以脱机执行；

9、可选择 RS232，RS422，RS485 通信接口；或选择 USB 通信；或选择以太网通信；

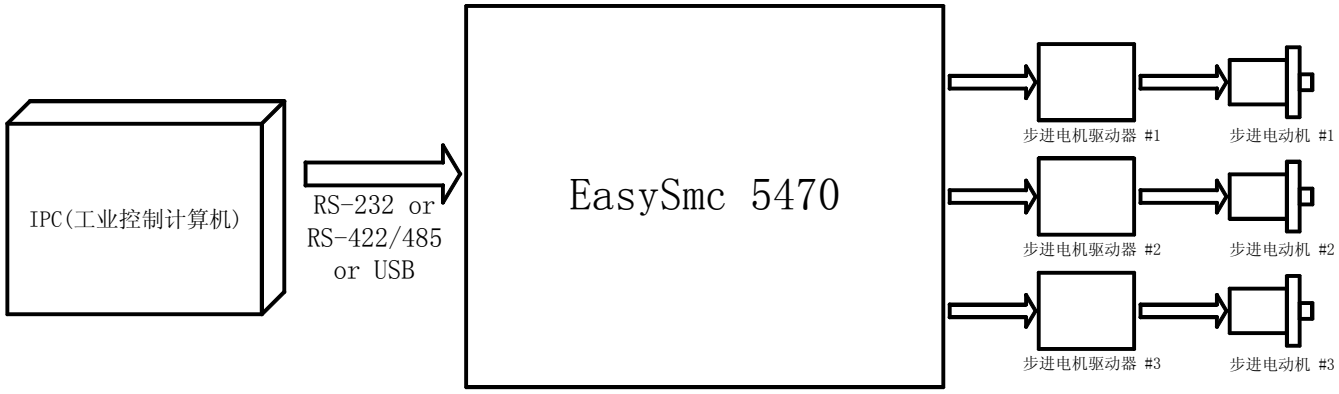
10、所有 I/O 使用光电隔，提高抗干扰性能；预留 16 路通用输出、16 路通用输入；

三、运动控制系统组成框图

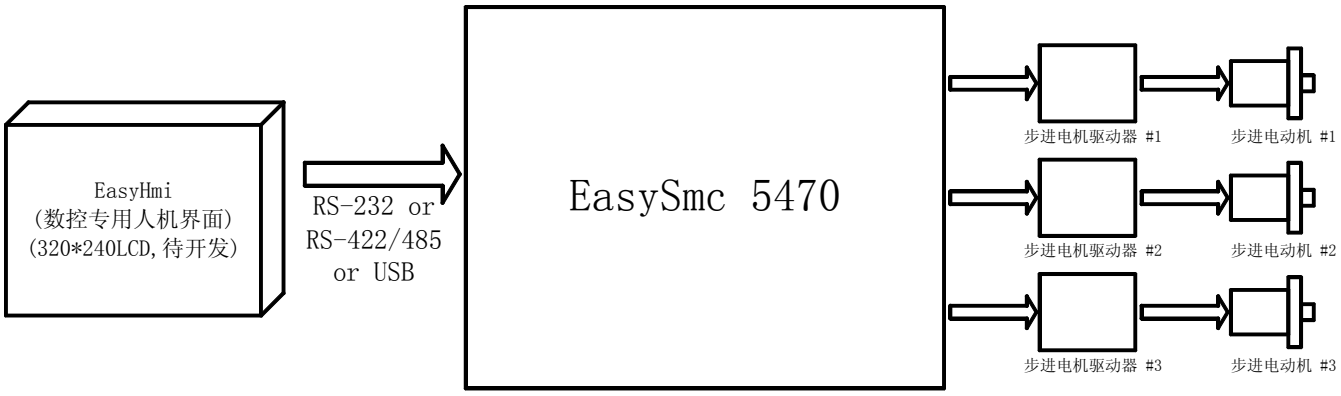
3.1 PLC作为上位机，如松下的FP0-C10CR，带RS-232C串口



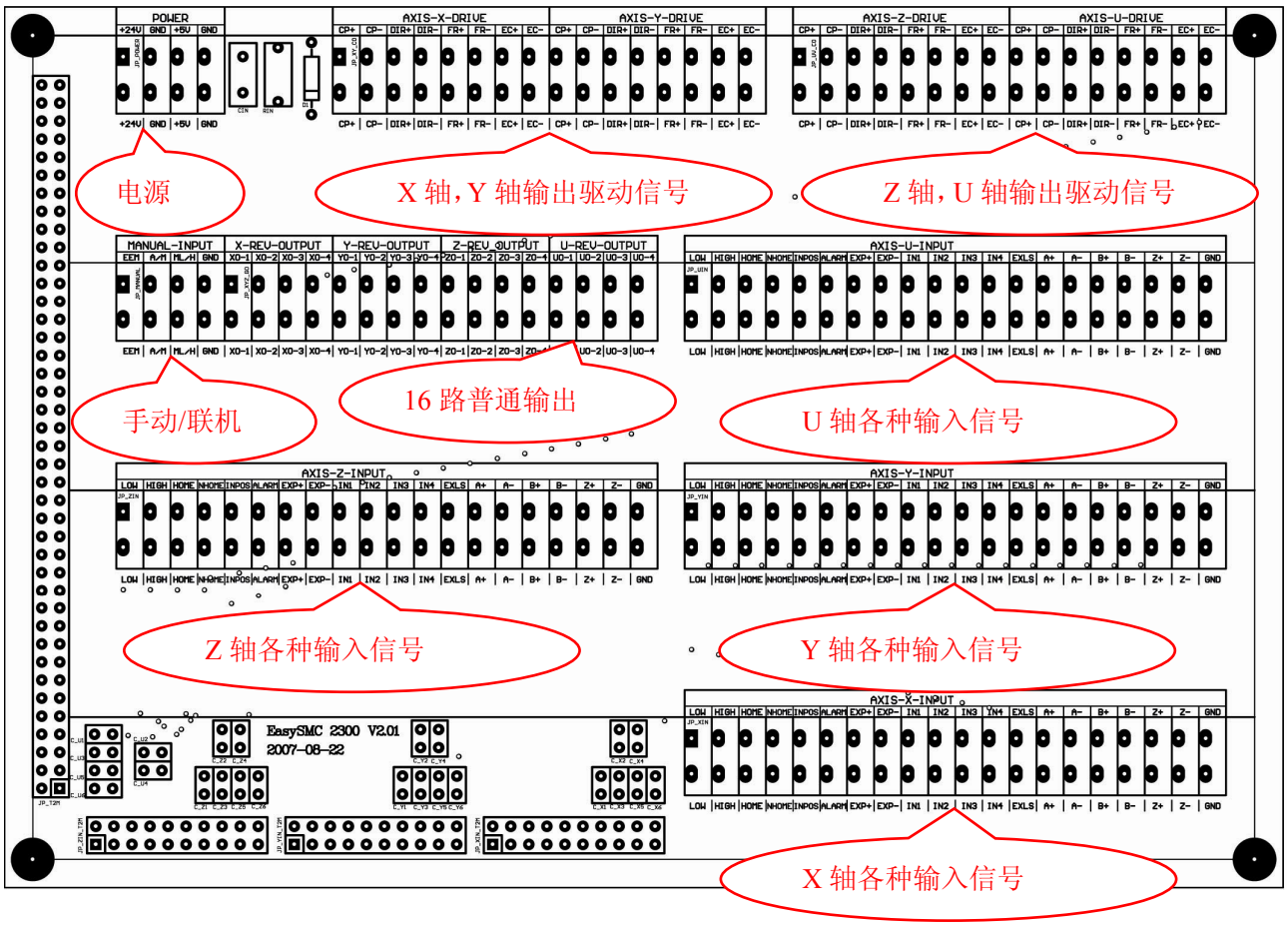
3.2 工业控制计算机作为上位机，与 EasySmc 5470 构成主从系统



3.3 EasyHmi作为上位机，可最大限度的提高性能价格比，需要用户订制



四、接线说明和跳线定义



4.1 外形尺寸说明

控制器外形为：198mm*133mm；安装孔为 Φ3mm，孔中心距边沿 3.80mm。

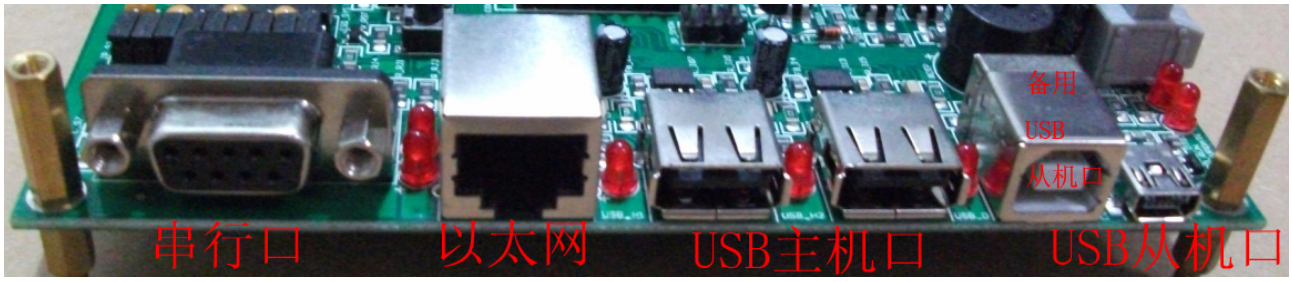
4.2 接线端说明

4.2.1 电源输入输出信号

接线端“+24V”，接+24VDC 的正端（+），“GND”接+24VDC 的负端（-）；

控制器提供 5V 电源输出（严禁接入 5V 电源），电流输出不超过 100mA。

4.2.2 通信口

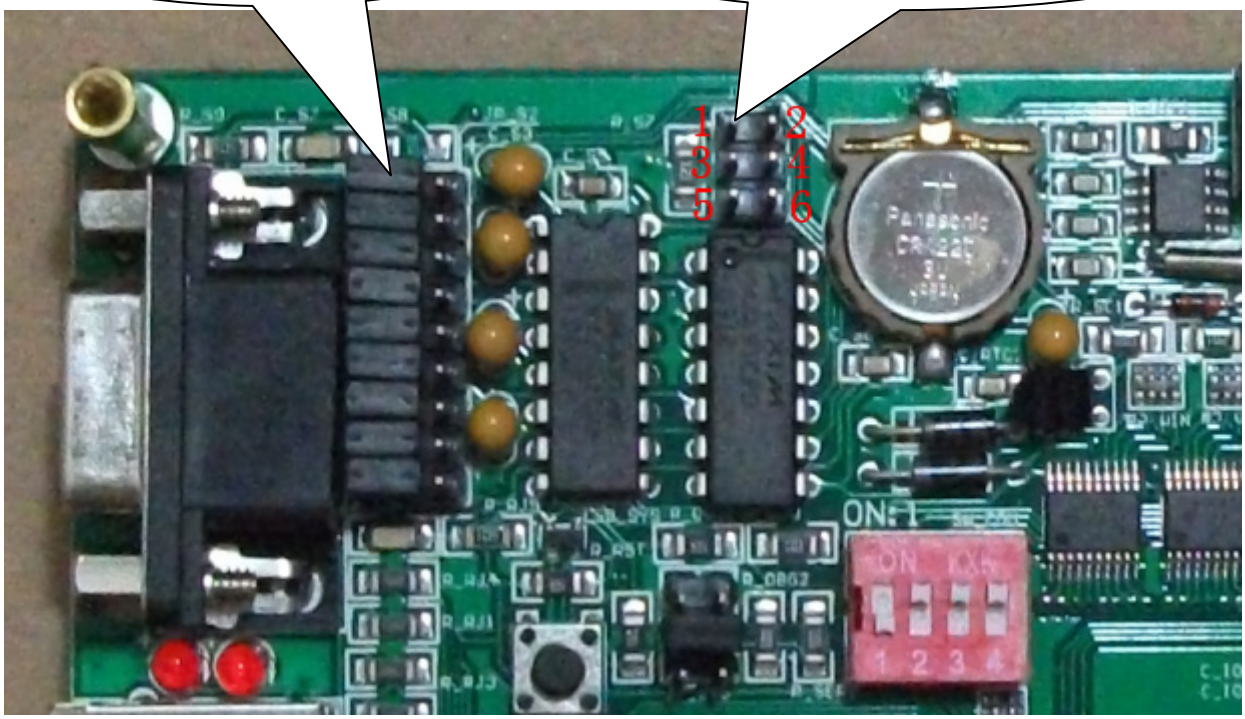


4.2.21 串行口信号 如果碰到强烈干扰,通信电缆请使用屏蔽线。

控制器出厂默认使用 RS232-Modbus，跳线已经设置如下

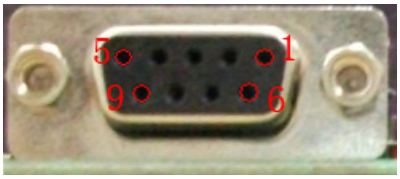
JP_S2（RS232/RS485 切换）
1、 RS232，9 针跳线跳到左边
2、 RS485，9 针跳线跳到右边

JP_S1（RS485）
引脚 1-2，3-4：2 线制（闭合）；4 线制（断开）。
引脚 5-6：终端电阻 120Ω，使用闭合；



RS232-Modbus 9 针 D-型连接器引脚分配：

2：TXD 发送数据
3：RXD 接收数据
5：公共端



2 线 RS485-Modbus 9 针 D-型连接器引脚分配：

1：公共端
5：D1（电压 D1>D0，对应于二进制 1）
9：D0

4 线 RS485-Modbus 9 针 D-型连接器引脚分配：

1：公共端

5：TXD1（电压 TXD1>TXD0，对应于二进制 1）

9：TXD0

4：RXD1（电压 RXD1>RXD0，对应于二进制 1）

8：RXD0

通信地址码选择：SW_COM，ON=1，OFF=0

（1）通信地址码选择：SEL1、SEL2，00，01，02，03；

特别说明的是，Modbus 采用地址 0 作为广播地址，控制器不使用拨码开关定义地址 0。

目前拨码开关设置的地址 1-4 仅对串行通信有效，对 usb，以太网无效。

拨码位 \ 地址码	01	02	03	04
SEL1，SEL2	0、0	0、1	1、0	1、1

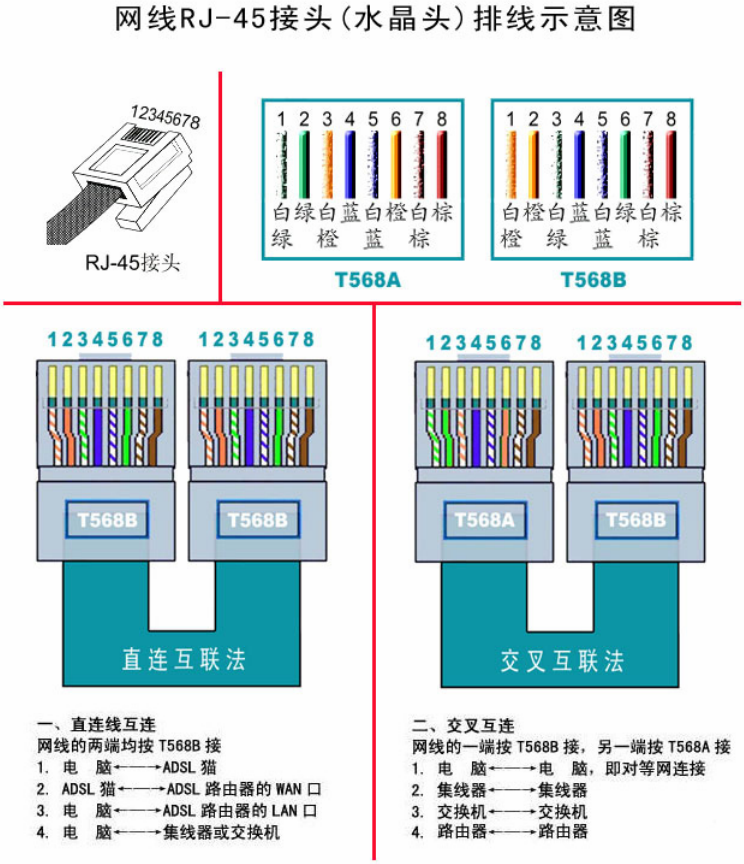
（2）通信波特率选择：SEL3，SEL4，2400，4800，9600，19200bps。

拨码位 \ 波特率	2400bps	4800bps	9600bps	19200bps
SEL3，SEL4	0、0	0、1	1、0	1、1

4.2.22 USB

采用标准 mini-B 线和控制器连接。

4.2.23 以太网



当电脑通过路由器、集线器、交换机和控制器连接时，采用“直连线互连”；

当电脑直接和控制器连接时，采用“交叉互联”。

4.2.3 电机驱动信号

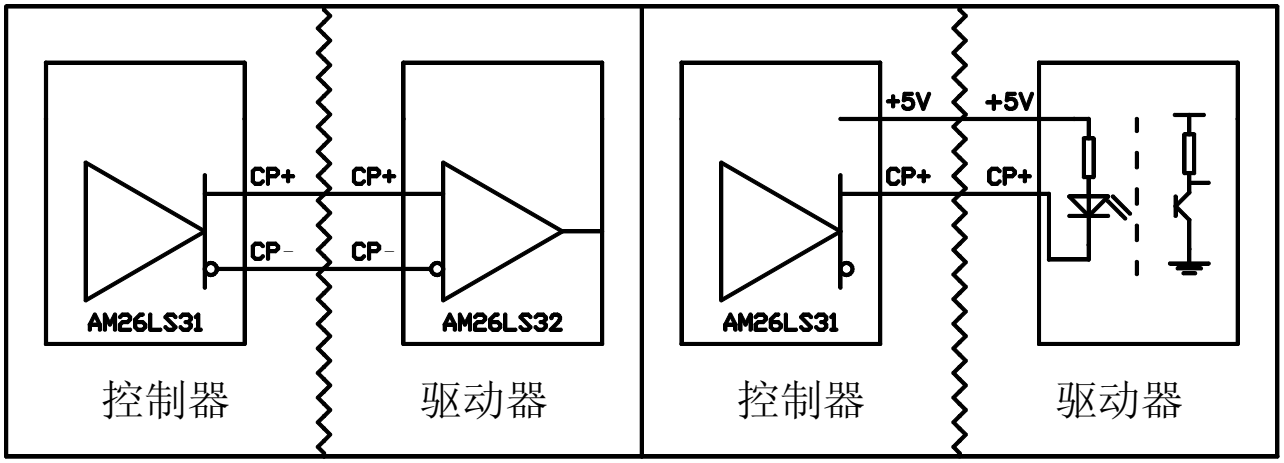
驱动信号采用差动输出；

CP+，CP-：（1）输出模式为脉冲信号+方向信号时，CP+和 CP-输出一个脉冲到驱动器，电机旋转一个脉冲角，电机运动方向由 DIR+和 DIR-决定。（2）输出模式为正向脉冲信号+反向脉冲信号，CP+和 CP-输出一个脉冲信号到驱动器，电机正向旋转一个脉冲角。

DIR+，DIR-：（1）输出模式为脉冲信号+方向信号时，DIR+和 DIR-决定电机运动方向。（2）输出模式为正向脉冲信号+反向脉冲信号，DIR+和 DIR-输出一个脉冲信号到驱动器，电机反向旋转一个脉冲角。

ECLR+，ECLR-：误差清除信号，暂时做普通 I/O；

FREE+，FREE-：脱机信号。可以接步进驱动器的 FREE 信号，或者接伺服驱动器的 SERVO ON/OFF 信号。



驱动器“差动接收”接线图，　　驱动器“单端接收”接线图

4.2.4 下限输入，上限输入，原点输入……16 输入信号

下限，上限，原点输入在复位运动中使用。当控制器做独立，直线，圆弧运动时，如果遇到下限有效或者上限有效，视为“故障”，控制器将立即停止输出，等待清除命令缓冲区。

4.2.41　X 轴 16 路外部输入信号说明

（1）LOW：机械下限位输入信号；（2）HIGH：机械上限位输入信号；

（3）HOME：机械原点输入信号；（4）NHOME：靠近原点输入信号；

（5）INPOS：驱动器到位信号，供上位机查询，也可做普通输入信号；

（6）ALARM：驱动器报警信号，供上位机查询，也可做普通输入信号；

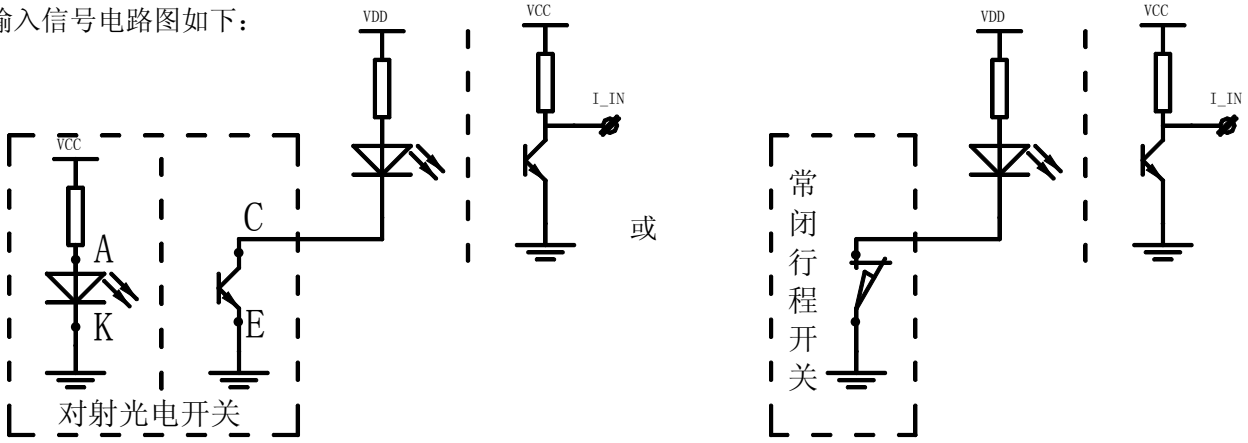
（7-8）EXP+ 、EXP-：手动输入信号 1、2，工作模式不同，用途不同；

（9-12）IN1，IN2，IN3，IN4：普通输入信号，供上位机查询；

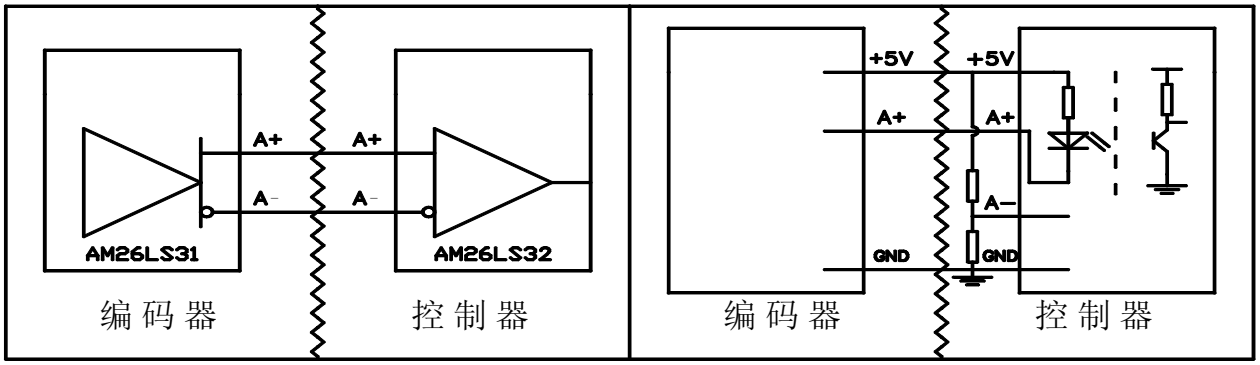
（13）EXLS：外部联锁信号；

（14-16）A+，A- ，B+，B- ,Z+，Z-：旋转编码器 A、B、Z 相差动输入信号；

输入信号电路图如下：



光电编码器信号“差动信号”的接法和“单端信号”的接法：



“差动信号”接收　　“单端信号”接收

说明：当使用编码器使用的是“单端信号”的时候，应该将 A-，B-，Z-的电压钳位在 A+，B+，C+信号的 VOH 和 VOL 之间，比如 AM26LS31 的 VOH>2.5V，VOL<0.5V，那么，A-，B-，Z-的输入电压应该是(0.5+2.5)2=1.5V，方法是如上图所示，使用外部电阻分压后，输入控制器。

4.2.42　Y 轴 16 路外部输入信号说明：同 X 轴

4.2.43　Z 轴 16 路外部输入信号说明：同 X 轴

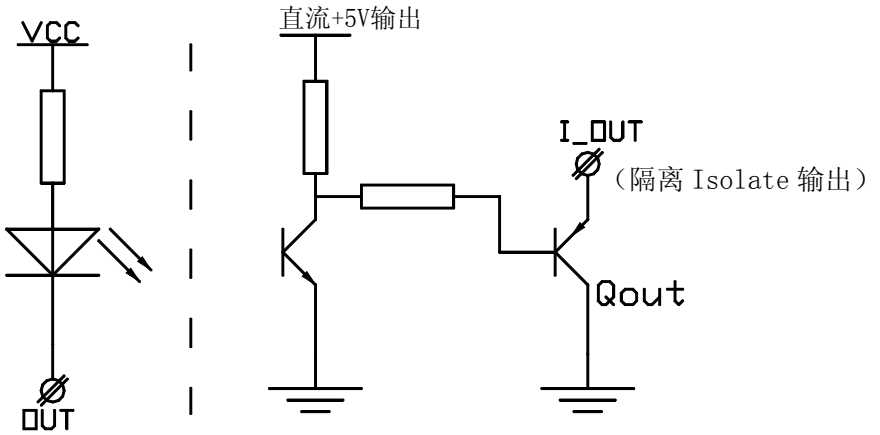
4.2.44　U 轴 16 路外部输入信号说明：同 X 轴

4.2.5 每轴 4 路输出信号，X，Y，Z，U共 16 输出信号

接线端号码为 XO-1，XO-2，XO-3，XO-4，YO-1，YO-2，YO-3，YO-4，

ZO-1，ZO-2，ZO-3，ZO-4，UO-1，UO-2，UO-3，UO-4。

输出信号的输出电路图如下：三极管 Qout 的吸收电流在 100mA 以下。**如果没有限流，可能导致三极管烧毁，控制器内部 DC/DC 电源模块烧毁。**



当输出为“0”时，对应的 Qout 对地导通；

当输出高“1”时，对应的 Qout 对地截止。

4.2.6 紧急故障停止输入信号（EEM）

4.2.7 手动信号输入（A/M，L/H）

为了方便调试，设置手动输入信号。

A/M：手动/联机选择。断开：联机；闭合：手动。

M_L/H：手动方式选择。断开：电平触发；闭合：脉冲触发。

4.3 手动/联机跳线说明

4.3.1、手动状态

当跳线 S_A/M 闭合时，控制器处于手动状态，手动输入有效，串行口的命令无效。

当从联机状态切换到手动状态时，必须等到命令缓冲区的命令执行完毕，否则手动无效。

（1）当 SW_ML/H 断开时，**电平触发**，如果手动信号输入，相应轴按默认速度做匀速运动；

（2）当 SW_ML/H 闭合时，**脉冲触发**，当手动信号输入一个脉冲，相应轴运动一个点动距离。

4.3.2、联机状态

当跳线 S_A/M 断开时，控制器处于联机状态。

（1）当 SW_ML/H 断开时，控制器处于“**在线运行**”状态，实时接收，执行来自上位机的命令。

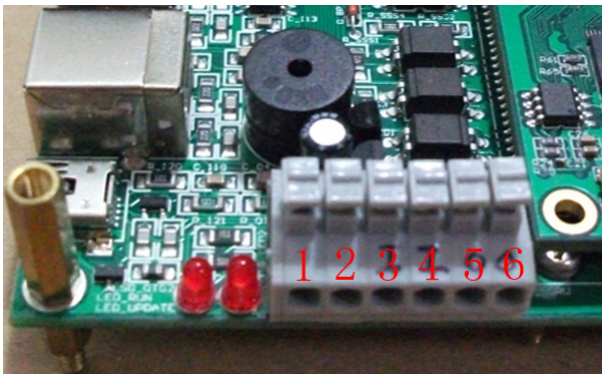
（2）当 SW_ML/H 闭合时，控制器处于“**脱机运行**”状态，上位机在此状态下编辑修改控制器 ROM 中的程序；修改完成后，控制器可以脱离上位机执行控制器 ROM 中的程序。

X_EXP+，执行程序；

Z_EXP+，停止运行；

U_EXP+，暂停。

4.4 状态灯输出等等



从接线端 1-6 分别为（1）+24V 输出，（2）GND，（3）控制器固件升级输入，（4）控制器 U 盘弹出输入，（5）控制器状态指示灯输出，（6）控制器 U 盘枚举灯输出。

输入信号电路参考 **4.2.41　X 轴 16 路外部输入信号说明；**

输出信号电路参考 **4.2.5 每轴 4 路输出信号，X，Y，Z，U 共 16 输出信号；**

控制器固件升级：将升级文件放置于 U 盘中，按下升级键盘，大约 1 分钟，控制器固件升级完成。

控制器 U 盘弹出：当需要从控制器中拔出 U 盘时，需要按下弹出键，等待 U 盘指示灯熄灭后，可以拔出 U 盘。

控制器状态指示灯：手动，指示灯大约每 2 秒闪烁 1 次；联机，指示灯大约每 0.5 秒闪烁 1 次。

五、控制器Modbus寻址空间分配

5.1 读离散量输入（地址范围 0…68）

X,Y,Z,U 轴通用输入（Low，High,Home,NHome,Inpos,Alarm,Expp,Expn，In1,In2,In3,In4,Exls,A,B,Z）：地址 0-63；

Eem：紧急制动输入；地址 64；

a/m：手动、自动切换；地址 65；

al/h：手动方式切换；地址 66；

sss1:固件升级；地址 67；

sss2:U 盘弹出；地址 68；

5.2 读单个线圈（地址范围 0…15）

X0-1…X0-4，Y0-1…Y0-4，Z0-1…Z0-4，U0-1…U0-4；

5.3 写单个线圈（地址范围 0…15）

X0-1…X0-4，Y0-1…Y0-4，Z0-1…Z0-4，U0-1…U0-4；

5.4 写多个线圈（地址范围 0…15）

X0-1…X0-4，Y0-1…Y0-4，Z0-1…Z0-4，U0-1…U0-4；

5.5 输入寄存器（地址范围 0…199）

对应于 100 个脱机执行文件的大小，每个脱机执行文件的大小为 4 字节，先传送高 16 位，后传送低 16 位。

modbus 读文件记录都是偶数字节，所以需要这个功能，因为脱机文件的大小可能是奇数。此功能配合读文件记录读取脱机执行文件的内容。

5.6 读、写保持寄存器（地址范围 0…25343）

地址	名称	默认值	用途
0	assign	0x0123	逻辑轴分配到物理轴(G 代码逻辑轴 X,Y,Z,U 对应控制器接线端的 X（0）,Y（1）,Z（2）,U（4）轴)；可以将 G 代码 Y 轴映射到接线端的 U(4)轴，实现双 Y 轴的直角坐标机器人。
1	input_assign （高 8 位） output_assign（低 8 位）	1 0	输入信号是否使用逻辑轴到物理轴的映射 输出信号是否使用逻辑轴到物理轴的映射
2	fpga_xach	0	X 轴实际位置寄存器（高 16 位）
3	fpga_xacl	0	X 轴实际位置寄存器（低 16 位）
4	fpga_yach	0	Y 轴实际位置寄存器（高 16 位）
5	fpga_yacl	0	Y 轴实际位置寄存器（低 16 位）
6	fpga_zach	0	Z 轴实际位置寄存器（高 16 位）
7	fpga_zacl	0	Z 轴实际位置寄存器（低 16 位）
8	fpga_uach	0	U 轴实际位置寄存器（高 16 位）
9	fpga_uacl	0	U 轴实际位置寄存器（低 16 位）
10	fpga_xlch	0	X 轴逻辑位置寄存器（高 16 位）
11	fpga_xlcl	0	X 轴逻辑位置寄存器（低 16 位）
12	fpga_ylch	0	Y 轴逻辑位置寄存器（高 16 位）
13	fpga_ylcl	0	Y 轴逻辑位置寄存器（低 16 位）
14	fpga_zlch	0	Z 轴逻辑位置寄存器（高 16 位）
15	fpga_zlcl	0	Z 轴逻辑位置寄存器（低 16 位）
16	fpga_ulch	0	U 轴逻辑位置寄存器（高 16 位）
17	fpga_ulcl	0	U 轴逻辑位置寄存器（低 16 位）
18	busy_status_x（高 8 位） busy_status_y（低 8 位）	0 0	X 轴忙状态（0：没有运动；1：运动中） Y 轴忙状态
19	busy_status_z（高 8 位） busy_status_u（低 8 位）	0 0	X 轴忙状态 Y 轴忙状态
20、21	run_gcode_no	0	正在执行的 G 代码序号，上电时为 0；每接收到 1 条 G 代码命令，run_gcode_no 加 1； 上位机可以把 run_gcode_no 清零，发送 N 条运动命令到控制器后，通过监控 run_gcode_no 和 busy_status_x,y,z,u 可以知道控制器是否将所有命令执行完毕。
22	tool_used （高 8 位） tool_curr（低 8 位）	1 1	是否使用工具偏移量 当前使用的工具号（取值 1-6）
23	ctrl_mode(高 8 位) dist_mode(低 8 位)	0 90	工作模式:标准模式 0,连续模式 1 距离模式(90：绝对坐标；91：相对坐标)
24 25 26 27	axis_mode_reg_x axis_mode_reg_y axis_mode_reg_z axis_mode_reg_u	0x0000 0x0000 0x0000 0x0000	工作模式寄存器（限位,有效电平等等） Bit0：Low 有效方式；当前 Bit0=0，Low 未动作电平为低电平，Low 动作电平为高电平。Bit0=1，Low 未动作电平为高电平，Low 动作电平为低电平。 Bit1：High 有效方式，同 Bit0；

			Bit2：Home 有效方式，同 Bit0； Bit3：NHome 有效方式，同 Bit0； Bit4：保留； Bit5：单脉冲,双脉冲选择；Bit5=0，控制器的输出信号为单脉冲，即“脉冲信号+方向信号”；Bit5=1，控制器的输出信号为双脉冲，即“正向脉冲信号+负向脉冲信号”； Bit6：硬限位选择；Bit6=0，接入控制器的 Low，High 硬限位信号无效；Bit6=1，接入控制器的 Low，High 硬限位信号有效，Low，High 有效动作，控制器将停止发送脉冲。 Bit7：软限位选择；Bit7=0，软件限位功能无效；Bit7=1，软件限位功能启动，当逻辑位置坐标超过软件限位坐标，控制器停止发送脉冲。 Bit8：保留； Bit9：保留； Bit10：EEM 有效方式，仅 X 轴工作模式寄存器有效；Bit10=0，EEM 未动作电平为低电平，EEM=1，控制器停止发任何脉冲；Bit10=1，EEM 未动作电平为高电平，EEM=0，控制器停止发任何脉冲。 Bit11：保留。
28 29 30 31	ok1delay,ok2delay, ok3delay,ok4delay, ok5delay,ok6delay, ok7delay,ok8delay,	0x0505 0x0505 0x0505 0x0505	按键 1，按键 2 检测抗抖动延时 按键 3，按键 4 检测抗抖动延时 按键 5，按键 6 检测抗抖动延时 按键 7，按键 8 检测抗抖动延时
32 33 34 35	k1indi,k2indi, k3indi,k4indi, k5indi,k6indi, k7indi,k8indi,	0x0001 0x0203 0x0405 0x0607	按键 1，按键 2 指示灯 按键 3，按键 4 指示灯 按键 5，按键 6 指示灯 按键 7，按键 8 指示灯 值 00-07 对应输出 xout1-xout4,yout1-yout7
36	app_fileno	100	脱机执行文件号 当 app_fileno<100 时，脱机文件号由 app_fileno 确定 当 app_fileno>=100 时，脱机文件号由外接 bcd 码确定，外接 bcd 码选择电路将占用 xin1-xin4,uout3,uout4。
37	StopProgLen	0	STOP 程序的长度（保留）
38	ResetProgLen	0	RESET 程序的长度（保留）
39 40	Resmode_x（高 8 位） Resmode_y（低 8 位） Resmode_z（高 8 位） Resmode_u（低 8 位）	1 1 1 1	默认复位模式,取值范围 1-8 1：向上原点复位 2：向下原点复位 3：下限位复位 4：上限位复位 5：向上Z相复位 6：向下Z相复位 7：快速向上原点复位

			8：快速向下原点复位
41	xout_pt	0x3333	暂停时 X_OUT1-OUT4 变换方式 0：暂停时，X_OUT 强制为 0，低电平 1：暂停时，X_OUT 强制为 1，高电平 2：暂停时，X_OUT 取反 3：暂停时，X_OUT 保持不变
42	yout_pt	0x3333	暂停时 Y_OUT-OUT4 变换方式，同 xout_pt
43	zout_pt	0x3333	暂停时 Z_OUT-OUT4 变换方式，同 xout_pt
44	uout_pt	0x3333	暂停时 U_OUT-OUT4 变换方式，同 xout_pt
45 46 47	pmac1,pmac2 pmac3,pmac4 pmac5,pmac6	0x0A,0B 0x0C,0D 0x0E,0F	网卡 mac 硬件地址
48、49	IP_ADDR_THIS_HOST	192.168 .1.175	控制器 IP 地址
50、51	IP_ADDR_NET_MASK	255.255 .255.0	子网掩码
52、53	IP_ADDR_DFLT_GATEWAY	192.168 .1.1	默认网关
54	arm_ver_year	不定	arm 软件版本号_年份
55	arm_ver_monday	不定	arm 软件版本号_月份日期
56	arm_ver_hourmin	不定	arm 软件版本号_分钟小时
57	fpga_ver_year	不定	fpga 软件版本号_年份
58	fpga_ver_monday	不定	fpga 软件版本号_月份日期
59	fpga_ver_hourmin	不定	fpga 软件版本号_分钟小时
768	com_buf_clr	0	停止,并且清除缓冲区
769	parameter_restore_cmd	0	恢复出厂参数
770	write_flash_cmd	0	保存参数到 flash
771	act_fileno	不定	实际执行的文件号,可能由 BCD 码决定
772	pd_filesize_h16	0	写文件记录,文件大小(文件长度肯能是奇数)
773	pd_filesize_l16	0	写文件记录,文件大小(文件长度肯能是奇数)
774	rtc_year	不定	年份
775	rtc_month （高 8 位） rtc_week （低 8 位）	不定 不定	月份 星期几
776	rtc_day （高 8 位） rtc_hour （低 8 位）	不定 不定	日期 小时
777	rtc_minute（高 8 位） rtc_second（低 8 位）	不定 不定	分钟 秒
778、779	rec_gcode_no	0	控制器接收(receive)到的 fifo 命令的总数量
780	fifo_max_len	199	fifo 缓冲区长度
781	fifo_rem_len	199	fifo 剩余缓冲区长度
1024	System_cmd		fifo(G 代码命令)写地址 此地址写入 G 代码命令,G 代码必须以’/0’或者空格结束。如果 添加’/0’或者空格后 G 代码长度是

			奇数，必须再添加需要添加’/0’或者空格，使 G 代码长度变成偶数，因为 modbus 写寄存器都是以双字节为单位。
2047	StopProg[1024]		保留
3072	ResetProg[1024]		保留
以下地址为浮点变量区			
4096	mcdl[4]	0.00125	脉冲当量
4112	ppmax[4]	1000	正向软件限位寄存器
4128	npmin[4]	-1000	反向软件限位寄存器
4114	resfxfw[4]	2	反向复位距离
4160	reskdd[4]	1	电平检测,抗抖动距离,用于确认电平信号
4176	resmin[4]	1	复位最小距离
4192	vin_res[4]	2.5	起跳速度（复位运动）
4028	vin_iso[4]	2.5	起跳速度（独立运动速度参数）
4224	vmax_iso[4]	25	最高速度
4240	vout_iso[4]	2.5	出口速度
4256	acc1_iso[4]	25	加速度 1
4272	acc2_iso[4]	25	加速度 2
4228	vin_ipl	2.5	起跳速度（插补运动速度参数）
4292	vmax_ipl	25	最高速度
4296	vout_ipl	2.5	出口速度
4300	acc1_ipl	25	加速度 1
4304	acc2_ipl	25	加速度 2
4308	gjacc_ipl	75	拐角加速度
4312	ggwc_ipl	0.1	弓高误差
4316	vin_man[4]	2.5	手动速度
4332	s_man[4]	1	点动距离
4348	work_offset[4]	0	当前工件偏移量,"物理轴"顺序存放
4364	tool_offset[4]	0	当前工具偏移量,"物理轴"顺序存放
4380	work_offset1[4]	0	当前使用的制具偏移量组
4396	work_offset2[4]	0	
4412	work_offset3[4]	0	
4428	work_offset4[4]	0	
4444	work_offset5[4]	0	
4460	work_offset6[4]	0	
4476	tool_offset1[4]	0	当前使用的工具偏移量组
4492	tool_offset2[4]	0	
4508	tool_offset3[4]		
4524	tool_offset4[4]	0	
4540	tool_offset5[4]	0	
4556	tool_offset6[4]	0	
6144	work_offset1_file0[4]	0	制具偏移量 1（脱机文件 0 使用）
6160	work_offset2_file0[4]	0	

6176	work_offset3_file0[4]	0	
6192	work_offset4_file0[4]	0	
6208	work_offset5_file0[4]	0	
6224	work_offset6_file0[4]	0	制具偏移量 6（脱机文件 0 使用）
6240	tool_offset1_file0[4]	0	工具偏移量 1（脱机文件 0 使用）
6256	tool_offset2_file0[4]	0	
6272	tool_offset3_file0[4]	0	
6288	tool_offset4_file0[4]	0	
6304	tool_offset5_file0[4]	0	
6320	tool_offset6_file0[4]	0	工具偏移量 6（脱机文件 0 使用）
	……		
25152	work_offset1_file99[4]	0	制具偏移量 1（脱机文件 99 使用）
25168	work_offset2_file99[4]	0	
25184	work_offset3_file99[4]	0	
25200	work_offset4_file99[4]	0	
25216	work_offset5_file99[4]	0	
25232	work_offset6_file99[4]	0	制具偏移量 6（脱机文件 99 使用）
25248	tool_offset1_file99[4]	0	工具偏移量 1（脱机文件 99 使用）
25264	tool_offset2_file99[4]	0	
25280	tool_offset3_file99[4]	0	
25296	tool_offset4_file99[4]	0	
25312	tool_offset5_file99[4]	0	
25328	tool_offset6_file99[4]	0	工具偏移量 6（脱机文件 99 使用）

5.7 读文件记录

modbus 协议文件采用记录的结构，每个文件包括 10000 个记录，每个记录 2 个字节，因此 modbus 最大的文件大小为 20000 个字节。

本控制器有 100 个脱机文件，每个脱机文件占用 modbus 协议文件 512 个文件号，最大的脱机文件大小为 20000*512=10240000，大约是 9.8M。

5.8 写文件记录

参考“读、写保持寄存器”地址 772：pd_filesize_h16；773：pd_filesize_l16，写文件记录,文件大小(因为文件长度肯能是奇数)。

写文件记录之前必须先写文件大小，再使用循环语句写文件记录的内容。

六、串行口通信modbus_serial.dll使用说明

如果不使用 modbus 串行口通信，可跳过阅读此节。

6.1 modbus_serial.dll Visual C++函数声明

modbus_serial.dll 采用 visual c++编写，配套有 modbus_serial.h 和 modbus_serial.lib 文件，可以直接使用 modbus_serial.h 做函数声明，链接库依赖 modbus_serial.lib。

long __stdcall mbserial_init(unsigned char Port,long Baud,unsigned char DataBits,unsigned char Parity,unsigned char StopBits,long TimeoutM,long TimeoutC);//串行口初始化

参数：

Port：串行口端口号；

Baud：串行波特率；

DataBits：数据位；

Parity：校验位；0：无校验；1：偶检验；2：奇校验；

StopBits：停止位；0：1 位停止位；1：1.5 位停止位；2：2 位停止位；

TimeoutM：每字符间的超时；

TimeoutC：一次读取串口数据的固定超时；

返回值：

0：成功；1：失败。

例程：

mbserial_init(2,19200,8,0,2,2,50)；//本控制器使用 8 位数据位，无校验，2 位停止位。

long __stdcall mbserial_connect();//串行口连接（打开串口）

返回值：

0：成功；1：失败。

long __stdcall mbserial_readcoils(unsigned char addr,unsigned char * prec,long staddr,long quantity);//读线圈

参数：

addr：控制器 modbus 地址

prec：接收数据指针

staddr：读线圈的开始地址

quantity：读线圈的数量

返回值：

-1：未知错误；-2：串口没有打开；　　-3：staddr 错误；　　-4：quantity 错误；

-5：超时错误；-6：响应 CRC 错误；-7：异常响应 CRC 错误

0：正确返回；

1：非法功能；2：非法数据地址；3：非法数据值；4：从站设备故障

long __stdcall mbserial_readdiscreteinputs(unsigned char addr,unsigned char * prec,long staddr,long quantity);//读离散量输入

参数：

addr：控制器 modbus 地址

prec：接收数据指针

staddr：读离散量输入的开始地址

quantity：读离散量输入的数量

返回值：

-1：未知错误；-2：串口没有打开；　　-3：staddr 错误；　　-4：quantity 错误；
-5：超时错误；-6：响应 CRC 错误；-7：异常响应 CRC 错误
0：正确返回；
1：非法功能；2：非法数据地址；3：非法数据值；4：从站设备故障

long __stdcall mbserial_readholdingreg(unsigned char addr,unsigned char * prec,long staddr,long quantity);//读保持寄存器

参数：

addr：控制器 modbus 地址
prec：接收数据指针
staddr：读保持寄存器的开始地址
quantity：读保持寄存器的数量

返回值：

-1：未知错误；-2：串口没有打开；　　-3：staddr 错误；　　-4：quantity 错误；
-5：超时错误；-6：响应 CRC 错误；-7：异常响应 CRC 错误
0：正确返回；
1：非法功能；2：非法数据地址；3：非法数据值；4：从站设备故障

long __stdcall mbserial_readinputreg(unsigned char addr,unsigned char * prec,long staddr,long quantity);//读输入寄存器

参数：

addr：控制器 modbus 地址
prec：接收数据指针
staddr：读输入寄存器的开始地址
quantity：读输入寄存器的数量

返回值：

-1：未知错误；-2：串口没有打开；　　-3：staddr 错误；　　-4：quantity 错误；
-5：超时错误；-6：响应 CRC 错误；-7：异常响应 CRC 错误
0：正确返回；
1：非法功能；2：非法数据地址；3：非法数据值；4：从站设备故障

long __stdcall mbserial_readfilerecord(unsigned char addr,unsigned char * prec,long fileno,long staddr,long quantity);//读文件记录

参数：

addr：控制器 modbus 地址
prec：接收数据指针
fileno：文件号
staddr：读文件记录的开始地址
quantity：读文件记录的数量

返回值：

-1：未知错误；-2：串口没有打开；　　-3：staddr 错误；　　-4：quantity 错误；
-5：超时错误；-6：响应 CRC 错误；-7：异常响应 CRC 错误
0：正确返回；
1：非法功能；2：非法数据地址；3：非法数据值；4：从站设备故障

long __stdcall mbserial_writesinglecoil(unsigned char addr,unsigned char value,long

staddr);//写单个线圈

参数：

addr：控制器 modbus 地址

value：输出值 0 或 1

staddr：写单个线圈地址

返回值：

-1：未知错误；-2：串口没有打开；　　-3：staddr 错误；　　-4：quantity 错误；

-5：超时错误；-6：响应 CRC 错误；-7：异常响应 CRC 错误

0：正确返回；

1：非法功能；2：非法数据地址；3：非法数据值；4：从站设备故障

long __stdcall mbserial_writemultiplecoils(unsigned char addr,unsigned char * psend,long staddr,long quantity);//写多个线圈

参数：

addr：控制器 modbus 地址

psend：发送数据指针

staddr：写线圈的开始地址

quantity：写线圈的数量

返回值：

-1：未知错误；-2：串口没有打开；　　-3：staddr 错误；　　-4：quantity 错误；

-5：超时错误；-6：响应 CRC 错误；-7：异常响应 CRC 错误

0：正确返回；

1：非法功能；2：非法数据地址；3：非法数据值；4：从站设备故障

long __stdcall mbserial_writefilerecord(unsigned char addr,unsigned char * psend,long fileno,long staddr,long quantity,long anothertimeout);//写文件记录

参数：

addr：控制器 modbus 地址

psend：发送数据指针

fileno：文件号

quantity：写文件记录的数量

anothertimeout：写文件记录专用超时时间；写文件记录时，控制器写入时间比较长，特殊使用一个超时时间；

返回值：

-1：未知错误；-2：串口没有打开；　　-3：staddr 错误；　　-4：quantity 错误；

-5：超时错误；-6：响应 CRC 错误；-7：异常响应 CRC 错误

0：正确返回；

1：非法功能；2：非法数据地址；3：非法数据值；4：从站设备故障

long __stdcall mbserial_writemultiplereg(unsigned char addr,unsigned char * psend,long staddr,long quantity);//写多个寄存器（写保持寄存器）

参数：

addr：控制器 modbus 地址

psend：发送数据指针

quantity：写保持寄存器的数量

返回值：

-1：未知错误；-2：串口没有打开；　　-3：staddr 错误；　　-4：quantity 错误；
-5：超时错误；-6：响应 CRC 错误；-7：异常响应 CRC 错误
0：正确返回；
1：非法功能；2：非法数据地址；3：非法数据值；4：从站设备故障

```
void __stdcall mbserial_disconnect();//串行口断开（关闭串口）

void __stdcall float_b2l(unsigned char *Buff_addr,unsigned short *ptr_float);
```

参数：

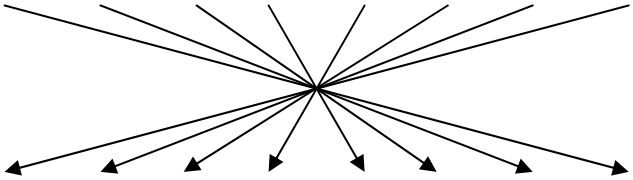
Buffer_addr：待转换的大端格式浮点数存放地址

ptr_float:转换后的小端格式浮点数存放地址

modbus 协议采用大端格式传送数据。由于 modbus 没有定义 IEEE-754 双精度浮点数（64 位）的传送格式，本控制器规定 IEEE-754 双精度浮点数（64 位）也采用大端格式传送。PC 机一般是使用小端格式保存浮点数，因此当 PC 机读取控制器的浮点数时，需要将大端格式的浮点数转换成小端格式的浮点数。

例如：123.456 大端格式为 0x40　0x5E　0xDD　0x2F　0x1A　0x9F　0xBE　0x77　**（Buffer_addr）**

使用 float_b2l 转换结果为



小端格式为 0x77　0xBE　0x9F　0x1A　0x2F　0xDD　0x5E　0x40　**（ptr_float）**

基于 float_b2l 的工作原理，当 PC 机要写入浮点数到控制器时，也可以使用 float_b2l 将小端格式的浮点数转换成大端格式的浮点数，然后写入控制器。

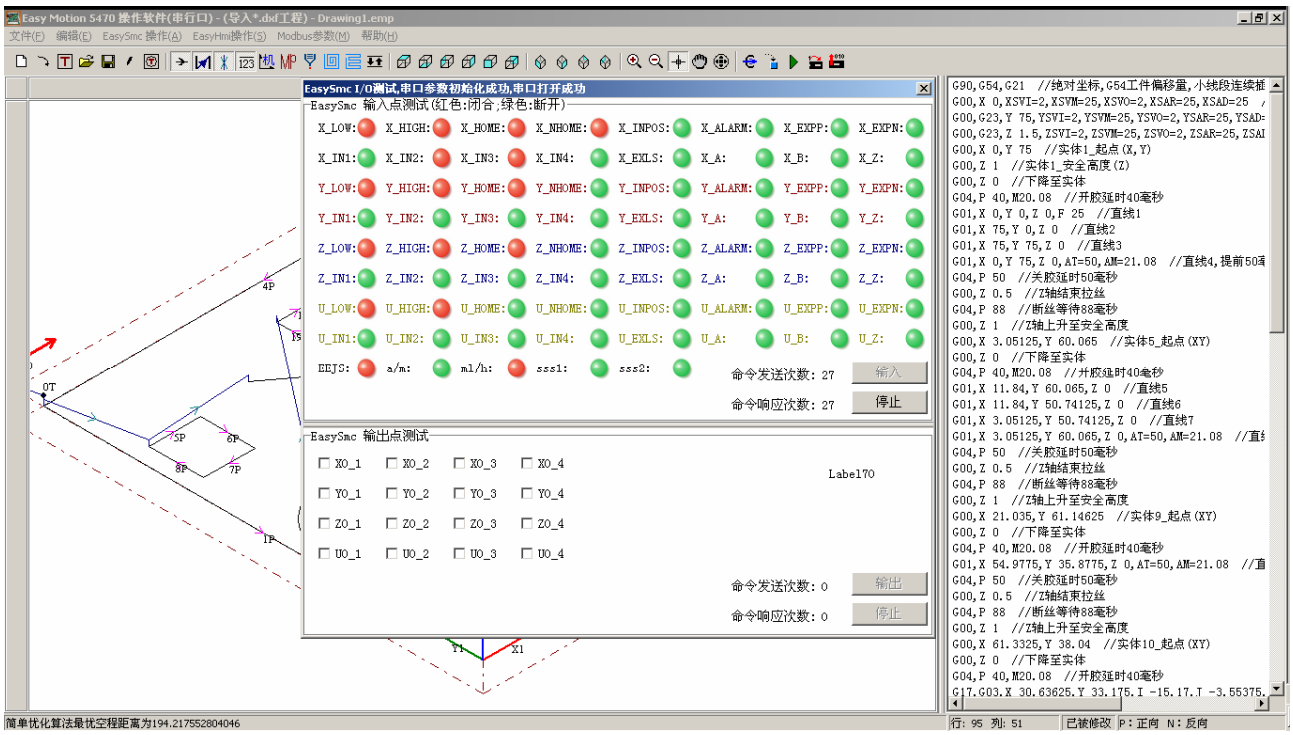
EasyMotion5470(串行口) Visual C++版本，提供源代码

6.2 modbus_serial.dll Delphi 函数声明

参数和返回值请参考 6.1 modbus_serial.dll Visual C++函数声明。

```
type
blink=^byte;
……
function mbserial_init(Port:byte;Baud:integer;DataBits:byte;Parity:byte;StopBits:byte;
TimeoutM:integer;TimeoutC:integer):integer;stdcall;external 'modbus_serial.dll';
function mbserial_connect():integer;stdcall;external 'modbus_serial.dll';

function mbserial_readcoils(addr:byte;prec:blink;staddr:integer;quantity:integer)
:integer;stdcall;external 'modbus_serial.dll';
function mbserial_readdiscreteinputs(addr:byte;prec:blink;staddr:integer;
quantity:integer):integer;stdcall;external 'modbus_serial.dll';
function mbserial_readholdingreg(addr:byte;prec:blink;staddr:integer;quantity:integer)
:integer;stdcall;external 'modbus_serial.dll';
function mbserial_readinputreg(addr:byte;prec:blink;staddr:integer;quantity:integer)
:integer;stdcall;external 'modbus_serial.dll';
function mbserial_readfilerecord(addr:byte;prec:blink;fileno:integer;staddr:integer;
quantity:integer):integer;stdcall;external 'modbus_serial.dll';
function mbserial_writefilerecord(addr:byte;psend:blink;fileno:integer;staddr:integer;
quantity:integer;anothertimeout:integer):integer;stdcall;external 'modbus_serial.dll';
function mbserial_writesinglecoil(addr:byte;psend:byte;staddr:integer):integer;
stdcall;external 'modbus_serial.dll';
function mbserial_writemultiplecoils(addr:byte;psend:blink;staddr:integer;
quantity:integer):integer;stdcall;external 'modbus_serial.dll';
function mbserial_writemultiplereg(addr:byte;psend:blink;staddr:integer;
quantity:integer):integer;stdcall;external 'modbus_serial.dll';

procedure mbserial_disconnect();stdcall;external 'modbus_serial.dll';

procedure float_b2l(Buff_addr:blink;ptr_float:blink);stdcall;
external 'modbus_serial.dll';
```

EasyMotion5470(串行口) Delphi 版本

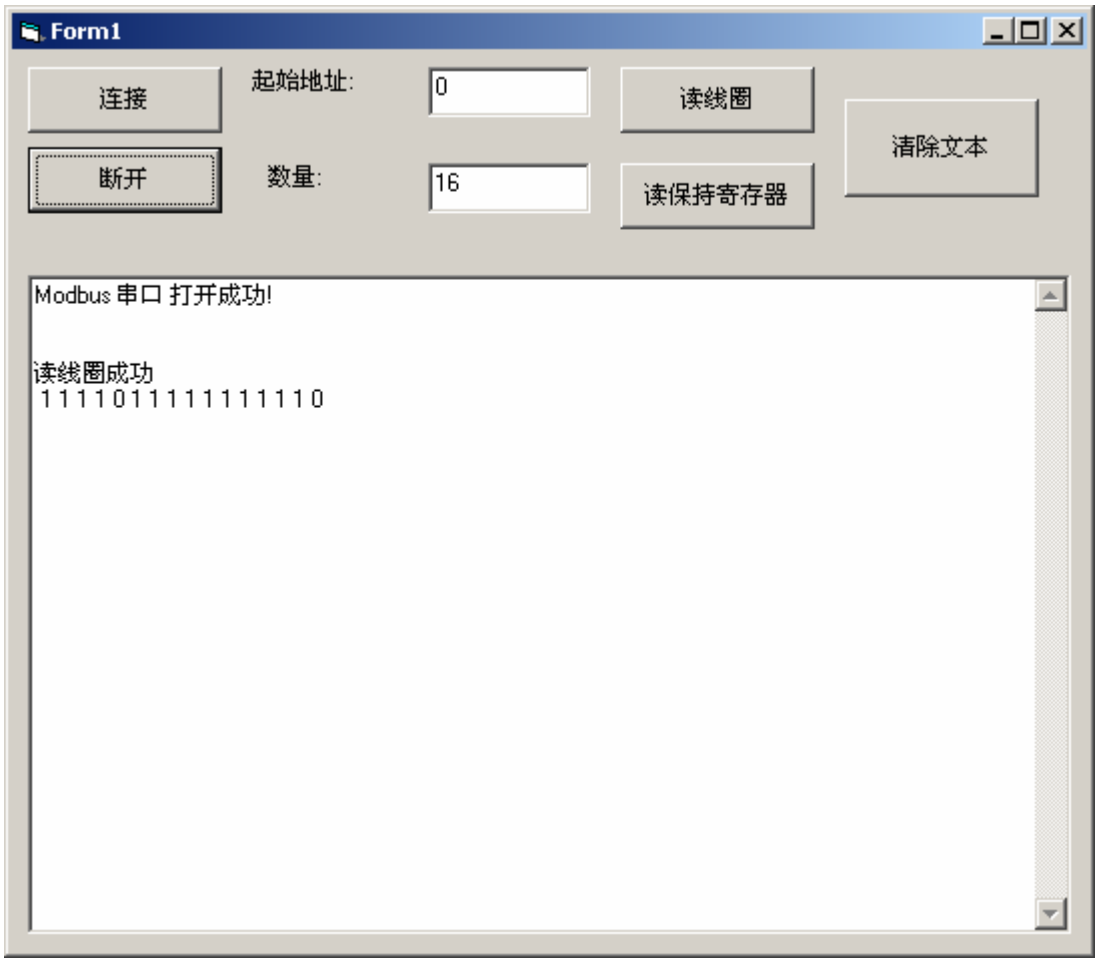


6.3 modbus_serial.dll Visual Basic 函数声明

参数和返回值请参考 6.1 modbus_serial.dll Visual C++函数声明。

```
Private Declare Function mbserial_init Lib "modbus_serial.dll" _
(ByVal Port As Byte, ByVal Baud As Integer, ByVal Databits As Byte, ByVal Parity As Byte, ByVal StopBits As Byte, ByVal TimeoutM As Integer, ByVal TimeoutC As Integer) As Long

Private Declare Function mbserial_connect Lib "modbus_serial.dll" () As Long

Private Declare Function mbserial_readcoils Lib "modbus_serial.dll" (ByVal addr As Byte, ByRef prec As Byte, ByVal staddr As Long, ByVal quantity As Long) As Long

Private Declare Function mbserial_readdiscreteinputs Lib "modbus_serial.dll" (ByVal addr As Byte, ByRef prec As Byte, ByVal staddr As Long, ByVal quantity As Long) As Long

Private Declare Function mbserial_readholdingreg Lib "modbus_serial.dll" (ByVal addr As Byte, ByRef prec As Byte, ByVal staddr As Long, ByVal quantity As Long) As Long

Private Declare Function mbserial_readinputreg Lib "modbus_serial.dll" (ByVal addr As Byte, ByRef prec As Byte, ByVal staddr As Long, ByVal quantity As Long) As Long

Private Declare Function mbserial_readfilerecord Lib "modbus_serial.dll" (ByVal addr As Byte, ByRef prec As Byte, ByVal fileno As Long, ByVal staddr As Long, ByVal quantity As Long) As Long

Private Declare Function mbserial_writefilerecord Lib "modbus_serial.dll" (ByVal addr As Byte, ByRef prec As Byte, ByVal fileno As Long, ByVal staddr As Long, ByVal quantity As Long, ByVal anothertimeout As Long) As Long

Private Declare Function mbserial_writesinglecoil Lib "modbus_serial.dll" (ByVal addr As Byte, ByVal value As Byte, ByVal staddr As Long) As Long

Private Declare Function mbserial_writemultiplecoils Lib "modbus_serial.dll" (ByVal addr As Byte, ByRef psend As Byte, ByVal staddr As Long, ByVal quantity As Long) As Long

Private Declare Function mbserial_writemultiplereg Lib "modbus_serial.dll" (ByVal addr As Byte, ByRef psend As Byte, ByVal staddr As Long, ByVal quantity As Long) As Long

Private Declare Sub mbserial_disconnect Lib "modbus_serial.dll" ()

Private Declare Sub float_b21 Lib "modbus_serial.dll" (ByRef Buff_addr As Byte, ByRef ptr_float As Byte)
```

EasyMotion5470(串行口) Visual Basic 版本，提供源代码



6.4 modbus_serial.dll 调用顺序

开始使用 mbserial_init，再使用 mbserial_connect，而后可以使用 mbserial_readcoils 等功能函数，结束使用 mbserial_disconnect。

七、USB通信modbus_usb.dll使用说明

如果不使用 modbus USB 通信，可跳过阅读此节。

7.1 modbus_usb.dll Delphi函数声明

```
function mbusb_readcoils(addr:byte;prec:blink;staddr:integer;quantity:integer)
:integer;stdcall;external 'modbus_usb.dll';
```

参数：

addr：控制器 modbus 地址

prec：接收数据指针

staddr：读线圈的开始地址

quantity：读线圈的数量

返回值：

-1：未知错误；-3：staddr 错误；-4：quantity 错误；-5：超时错误；

0：正确返回；

1：非法功能；2：非法数据地址；3：非法数据值；4：从站设备故障

```
function mbusb_readdiscreteinputs(addr:byte;prec:blink;staddr:integer;quantity:integer)
:integer;stdcall;external 'modbus_usb.dll';
```

参数：

addr：控制器 modbus 地址

prec：接收数据指针

staddr：读离散量输入的开始地址

quantity：读离散量输入的数量

返回值：

-1：未知错误；-3：staddr 错误；-4：quantity 错误；-5：超时错误；

0：正确返回；

1：非法功能；2：非法数据地址；3：非法数据值；4：从站设备故障

```
function mbusb_readholdingreg(addr:byte;prec:blink;staddr:integer;quantity:integer)
:integer;stdcall;external 'modbus_usb.dll';
```

参数：

addr：控制器 modbus 地址

prec：接收数据指针

staddr：读保持寄存器的开始地址

quantity：读保持寄存器的数量

返回值：

-1：未知错误；-3：staddr 错误；-4：quantity 错误；-5：超时错误；

0：正确返回；

1：非法功能；2：非法数据地址；3：非法数据值；4：从站设备故障

```
function mbusb_readinputreg(addr:byte;prec:blink;staddr:integer;quantity:integer)
:integer;stdcall;external 'modbus_usb.dll';
```

参数：

addr：控制器 modbus 地址

prec：接收数据指针

staddr：读输入寄存器的开始地址

quantity：读输入寄存器的数量

返回值：

-1：未知错误；-3：staddr 错误；-4：quantity 错误；-5：超时错误；

0：正确返回；

1：非法功能；2：非法数据地址；3：非法数据值；4：从站设备故障

function mbusb_readfilerecord(addr:byte;prec:blink;fileno:integer;staddr:integer;quantity:integer):integer;stdcall;external 'modbus_usb.dll';

参数：

addr：控制器 modbus 地址

prec：接收数据指针

fileno：文件号

staddr：读文件记录的开始地址

quantity：读文件记录的数量

返回值：

-1：未知错误；-3：staddr 错误；-4：quantity 错误；-5：超时错误；

0：正确返回；

1：非法功能；2：非法数据地址；3：非法数据值；4：从站设备故障

Function mbusb_writefilerecord(addr:byte;prec:blink;fileno:integer;staddr:integer;quantity:integer;anothertimeout:integer):integer;stdcall;external 'modbus_usb.dll';

参数：

addr：控制器 modbus 地址

psend：发送数据指针

fileno：文件号

quantity：写文件记录的数量

anothertimeout：写文件记录专用超时时间；写文件记录时，控制器写入时间比较长，特殊使用一个超时时间；

返回值：

-1：未知错误；-3：staddr 错误；-4：quantity 错误；-5：超时错误；

0：正确返回；

1：非法功能；2：非法数据地址；3：非法数据值；4：从站设备故障

function mbusb_writemultiplereg(addr:byte;psend:blink;staddr:integer;quantity:integer):integer;stdcall;external 'modbus_usb.dll';

参数：

addr：控制器 modbus 地址

psend：发送数据指针

quantity：写保持寄存器的数量

返回值：

-1：未知错误；-3：staddr 错误；-4：quantity 错误；-5：超时错误；

0：正确返回；

1：非法功能；2：非法数据地址；3：非法数据值；4：从站设备故障

function mbusb_writesinglecoil(addr:byte;value:byte;staddr:integer):integer;stdcall;external 'modbus_usb.dll';

参数：

addr：控制器 modbus 地址

value：输出值 0 或 1

staddr：写单个线圈地址

返回值：

-1：未知错误；-3：staddr 错误；-4：quantity 错误；-5：超时错误；

0：正确返回；

1：非法功能；2：非法数据地址；3：非法数据值；4：从站设备故障

function mbusb_writemultiplecoils(addr:byte;psend:blink;staddr:integerquantity:integer):integer;stdcall;external 'modbus_usb.dll';

参数：

addr：控制器 modbus 地址

psend：发送数据指针

staddr：写线圈的开始地址

quantity：写线圈的数量

返回值：

-1：未知错误；-3：staddr 错误；-4：quantity 错误；-5：超时错误；

0：正确返回；

1：非法功能；2：非法数据地址；3：非法数据值；4：从站设备故障

procedure float_b2l(Buff_addr:blink;ptr_float:blink);stdcall;external 'modbus_usb.dll';

参数：

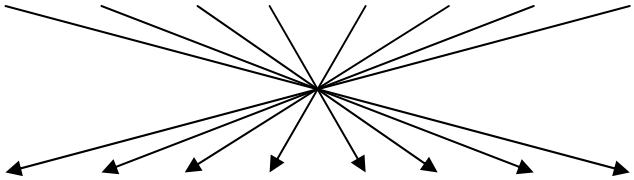
Buffer_addr：待转换的大端格式浮点数存放地址

ptr_float:转换后的小端格式浮点数存放地址

modbus 协议采用大端格式传送数据。由于 modbus 没有定义 IEEE-754 双精度浮点数（64 位）的传送格式，本控制器规定 IEEE-754 双精度浮点数（64 位）也采用大端格式传送。PC 机一般是使用小端格式保存浮点数，因此当 PC 机读取控制器的浮点数时，需要将大端格式的浮点数转换成小端格式的浮点数。

例如：123.456 大端格式为 0x40 0x5E 0xDD 0x2F 0x1A 0x9F 0xBE 0x77 **（Buffer_addr）**

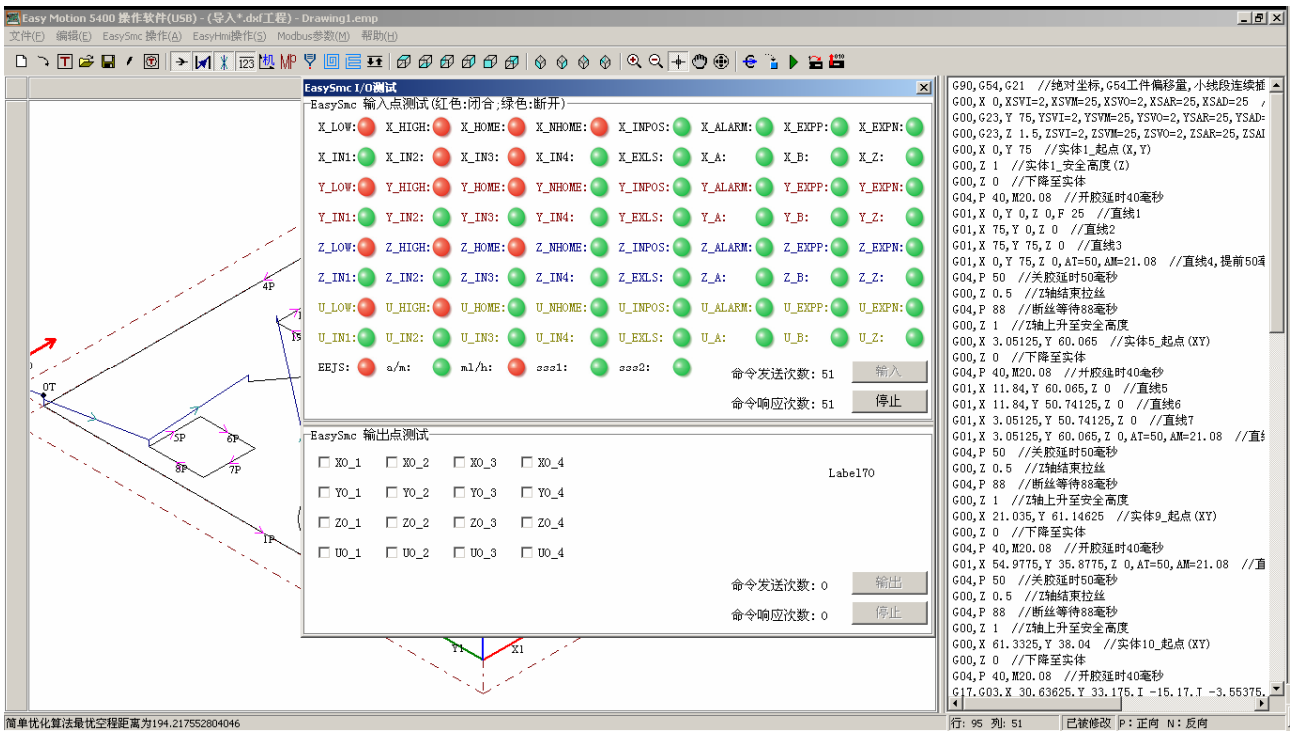
使用 float_b2l 转换结果为



小端格式为 0x77 0xBE 0x9F 0x1A 0x2F 0xDD 0x5E 0x40 **（ptr_float）**

基于 float_b2l 的工作原理，当 PC 机要写入浮点数到控制器时，也可以使用 float_b2l 将小端格式的浮点数转换成大端格式的浮点数，然后写入控制器。

EasyMotion5470(USB) Delphi 版本



7.2 modbus_usb.dll Visual C++函数声明

参数和返回值请参考 7.1 modbus_usb.dll Delphi 函数声明。

```
typedef long (_stdcall *lpmbusb_readcoils)(long addr,unsigned char * prec,long staddr,long quantity);//读线圈
typedef long (_stdcall *lpmbusb_readdiscreteinputs)(unsigned char addr,unsigned char * prec,long staddr,long quantity);//读离散量输入
typedef long (_stdcall *lpmbusb_readholdingreg)(unsigned char addr,unsigned char * prec,long staddr,long quantity);//读保持寄存器
typedef long (_stdcall *lpmbusb_readinputreg)(unsigned char addr,unsigned char * prec,long staddr,long quantity);//读输入寄存器
typedef long (_stdcall *lpmbusb_readfilerecord)(unsigned char addr,unsigned char * prec,long fileno,long staddr,long quantity);//读文件记录
typedef long (_stdcall *lpmbusb_writesinglecoil)(unsigned char addr,unsigned char value,long staddr);//写单个线圈
typedef long (_stdcall *lpmbusb_writemultiplecoils)(unsigned char addr,unsigned char * psend,long staddr,long quantity);//写多个线圈
typedef long (_stdcall *lpmbusb_writefilerecord)(unsigned char addr,unsigned char * psend,long fileno,long staddr,long quantity,long anothertimeout);//写文件记录
typedef long (_stdcall *lpmbusb_writemultiplereg)(unsigned char addr,unsigned char * psend,long staddr,long quantity);//写多个寄存器
typedef void (_stdcall *lpfloat_b2l)(unsigned char *Buff_addr,unsigned short *ptr_float);
……
HINSTANCE hDll; //句柄

lpmbusb_readcoils mbusb_readcoils;//函数指针
lpmbusb_readdiscreteinputs mbusb_readdiscreteinputs;
lpmbusb_readholdingreg mbusb_readholdingreg;
lpmbusb_readinputreg mbusb_readinputreg;
lpmbusb_readfilerecord mbusb_readfilerecord;
lpmbusb_writesinglecoil mbusb_writesinglecoil;
lpmbusb_writemultiplecoils mbusb_writemultiplecoils;
lpmbusb_writefilerecord mbusb_writefilerecord;
lpmbusb_writemultiplereg mbusb_writemultiplereg;
lpfloat_b2l float_b2l;
……
hDll = LoadLibrary("modbus_usb.dll");//加载 dll

mbusb_readcoils = (lpmbusb_readcoils)GetProcAddress(hDll,"mbusb_readcoils");
mbusb_readdiscreteinputs = (lpmbusb_readdiscreteinputs)GetProcAddress
                           (hDll,"mbusb_readdiscreteinputs");
mbusb_readholdingreg = (lpmbusb_readholdingreg)GetProcAddress
                           (hDll,"mbusb_readholdingreg");
mbusb_readinputreg = (lpmbusb_readinputreg)GetProcAddress
                           (hDll,"mbusb_readinputreg");
mbusb_readfilerecord = (lpmbusb_readfilerecord)GetProcAddress
                           (hDll,"mbusb_readfilerecord");
```

```
mbusb_writesinglecoil = (lpmbusb_writesinglecoil)GetProcAddress
                        (hDll,"mbusb_writesinglecoil");
mbusb_writemultiplecoils = (lpmbusb_writemultiplecoils)GetProcAddress
                        (hDll,"mbusb_writemultiplecoils");
mbusb_writefilerecord = (lpmbusb_writefilerecord)GetProcAddress
                        (hDll,"mbusb_writefilerecord");
mbusb_writemultiplereg = (lpmbusb_writemultiplereg)GetProcAddress
                        (hDll,"mbusb_writemultiplereg");
float_b2l = (lpfloat_b2l)GetProcAddress(hDll,"float_b2l");
……
FreeLibrary(hDll);//释放 dll
……
```

EasyMotion5470(USB) Visual C++版本，提供源代码

7.3 modbus_usb.dll Visual Basic函数声明

参数和返回值请参考 7.1 modbus_usb.dll Delphi 函数声明。

```
Private Declare Function mbusb_readcoils Lib "modbus_usb.dll" (ByVal addr As Byte, ByRef prec As Byte, ByVal staddr As Long, ByVal quantity As Long) As Long

Private Declare Function mbusb_readdiscreteinputs Lib "modbus_usb.dll" (ByVal addr As Byte, ByRef prec As Byte, ByVal staddr As Long, ByVal quantity As Long) As Long

Private Declare Function mbusb_readholdingreg Lib "modbus_usb.dll" (ByVal addr As Byte, ByRef prec As Byte, ByVal staddr As Long, ByVal quantity As Long) As Long

Private Declare Function mbusb_readinputreg Lib "modbus_usb.dll" (ByVal addr As Byte, ByRef prec As Byte, ByVal staddr As Long, ByVal quantity As Long) As Long

Private Declare Function mbusb_readfilerecord Lib "modbus_usb.dll" (ByVal addr As Byte, ByRef prec As Byte, ByVal fileno As Long, ByVal staddr As Long, ByVal quantity As Long) As Long

Private Declare Function mbusb_writefilerecord Lib "modbus_usb.dll" (ByVal addr As Byte, ByRef prec As Byte, ByVal fileno As Long, ByVal staddr As Long, ByVal quantity As Long, ByVal anothertimeout As Long) As Long

Private Declare Function mbusb_writesinglecoil Lib "modbus_usb.dll" (ByVal addr As Byte, ByVal value As Byte, ByVal staddr As Long) As Long

Private Declare Function mbusb_writemultiplecoils Lib "modbus_usb.dll" (ByVal addr As Byte, ByRef psend As Byte, ByVal staddr As Long, ByVal quantity As Long) As Long

Private Declare Function mbusb_writemultiplereg Lib "modbus_usb.dll" (ByVal addr As Byte, ByRef psend As Byte, ByVal staddr As Long, ByVal quantity As Long) As Long

Private Declare Sub float_b2l Lib "modbus_usb.dll" (ByRef Buff_addr As Byte, ByRef ptr_float As Byte)
```

EasyMotion5470(USB) Visual Basic 版本，提供源代码

八、以太网通信modbus_tcp.dll使用说明

如果不使用 modbus 以太网通信，可跳过阅读此节。

8.1 modbus_tcp.dll Delphi函数声明

```
function mbtcp_init(socket:byte;pc_ip1:byte;pc_ip2:byte;pc_ip3:byte;pc_ip4:byte;
pc_port:integer;host_ip1:byte;host_ip2:byte;host_ip3:byte;host_ip4:byte;host_port:integer;
readtimeout:integer):integer;stdcall;external 'modbus_tcp.dll';//TCP 初始化
```

参数：

socket：modbus_tcp 连接 socket 索引号（控制器 1 用数值 1，控制器 2 用数值 2……）

pc_ip1-pc_ip4：上位机（电脑）ip 地址；

pc_port：上位机（电脑）端口地址；

host_ip1-host_ip4：控制器 ip 地址；

host_port：控制器端口地址；固定为 502 端口；

readtimeout：读写超时时间；

返回值：

0：成功；1：失败；

```
function mbtcp_connect(socket:byte;timeout:integer):integer;stdcall;
external 'modbus_tcp.dll';//TCP 连接
```

参数：

socket：modbus_tcp 连接 socket 索引号（控制器 1 用数值 1，控制器 2 用数值 2……）

timeout：tcp 连接超时时间；

返回值：

0：成功；1：失败；-1：超时。

```
function mbtcp_readcoils(socket:byte;addr:byte;prec:blink;staddr:integer;
quantity:integer):integer;stdcall;external 'modbus_tcp.dll';//读线圈
```

参数：

socket：modbus_tcp 连接 socket 索引号（控制器 1 用数值 1，控制器 2 用数值 2……）

addr：控制器 modbus 地址

prec：接收数据指针

staddr：读线圈的开始地址

quantity：读线圈的数量

返回值：

-1：未知错误；-2：tcp 没有连接； -3：staddr 错误； -4：quantity 错误；-5：超时错误；

0：正确返回；

1：非法功能；2：非法数据地址；3：非法数据值；4：从站设备故障

```
function mbtcp_readdiscreteinputs(socket:byte;addr:byte;prec:blink;staddr:integer;
quantity:integer):integer;stdcall;external 'modbus_tcp.dll';//读离散量输入
```

参数：

socket：modbus_tcp 连接 socket 索引号（控制器 1 用数值 1，控制器 2 用数值 2……）

addr：控制器 modbus 地址

prec：接收数据指针

staddr：读离散量输入的开始地址

quantity：读离散量输入的数量

返回值：

-1：未知错误；-2：tcp 没有连接； -3：staddr 错误； -4：quantity 错误；-5：超时错误；
0：正确返回；
1：非法功能；2：非法数据地址；3：非法数据值；4：从站设备故障

function mbtcp_readholdingreg(socket:byte;addr:byte;prec:blink;staddr:integer;quantity:integer):integer;stdcall;external 'modbus_tcp.dll';//读保持寄存器

参数：

socket：modbus_tcp 连接 socket 索引号（控制器 1 用数值 1，控制器 2 用数值 2……）
addr：控制器 modbus 地址
prec：接收数据指针
staddr：读保持寄存器的开始地址
quantity：读保持寄存器的数量

返回值：

-1：未知错误；-2：tcp 没有连接； -3：staddr 错误； -4：quantity 错误；-5：超时错误；
0：正确返回；
1：非法功能；2：非法数据地址；3：非法数据值；4：从站设备故障

function mbtcp_readinputreg(socket:byte;addr:byte;prec:blink;staddr:integer;quantity:integer):integer;stdcall;external 'modbus_tcp.dll';//读输入寄存器

参数：

socket：modbus_tcp 连接 socket 索引号（控制器 1 用数值 1，控制器 2 用数值 2……）
addr：控制器 modbus 地址
prec：接收数据指针
staddr：读输入寄存器的开始地址
quantity：读输入寄存器的数量

返回值：

-1：未知错误；-2：tcp 没有连接； -3：staddr 错误； -4：quantity 错误；-5：超时错误；
0：正确返回；
1：非法功能；2：非法数据地址；3：非法数据值；4：从站设备故障

function mbtcp_readfilerecord(socket:byte;addr:byte;prec:blink;fileno:integer;staddr:integer;quantity:integer):integer;stdcall;external 'modbus_tcp.dll';//读文件记录

参数：

socket：modbus_tcp 连接 socket 索引号（控制器 1 用数值 1，控制器 2 用数值 2……）
addr：控制器 modbus 地址
prec：接收数据指针
fileno：文件号
staddr：读文件记录的开始地址
quantity：读文件记录的数量

返回值：

-1：未知错误；-2：tcp 没有连接； -3：staddr 错误； -4：quantity 错误；-5：超时错误；
0：正确返回；
1：非法功能；2：非法数据地址；3：非法数据值；4：从站设备故障

function mbtcp_writefilerecord(socket:byte;addr:byte;prec:blink;fileno:integer;staddr:integer;quantity:integer;anothertimeout:integer):integer;stdcall;external

'modbus_tcp.dll';//写文件记录

参数：

socket：modbus_tcp 连接 socket 索引号（控制器 1 用数值 1，控制器 2 用数值 2……）

addr：控制器 modbus 地址

psend：发送数据指针

fileno：文件号

quantity：写文件记录的数量

anothertimeout：写文件记录专用超时时间；写文件记录时，控制器写入时间比较长，特殊使用一个超时时间；

返回值：

-1：未知错误；-2：tcp 没有连接； -3：staddr 错误； -4：quantity 错误；-5：超时错误；

0：正确返回；

1：非法功能；2：非法数据地址；3：非法数据值；4：从站设备故障

function mbtcp_writesinglecoil(socket:byte;addr:byte;value:byte;staddr:integer):integer;stdcall;external 'modbus_tcp.dll';//写单个线圈

参数：

socket：modbus_tcp 连接 socket 索引号（控制器 1 用数值 1，控制器 2 用数值 2……）

addr：控制器 modbus 地址

value：输出值 0 或 1

staddr：写单个线圈地址

返回值：

-1：未知错误；-2：tcp 没有连接； -3：staddr 错误； -4：quantity 错误；-5：超时错误；

0：正确返回；

1：非法功能；2：非法数据地址；3：非法数据值；4：从站设备故障

function mbtcp_writemultiplecoils(socket:byte;addr:byte;psend:blink;staddr:integer;quantity:integer):integer;stdcall;external 'modbus_tcp.dll';//写多个线圈

参数：

socket：modbus_tcp 连接 socket 索引号（控制器 1 用数值 1，控制器 2 用数值 2……）

addr：控制器 modbus 地址

psend：发送数据指针

staddr：写线圈的开始地址

quantity：写线圈的数量

返回值：

-1：未知错误；-2：tcp 没有连接； -3：staddr 错误； -4：quantity 错误；-5：超时错误；

0：正确返回；

1：非法功能；2：非法数据地址；3：非法数据值；4：从站设备故障

function mbtcp_writemultiplereg(socket:byte;addr:byte;psend:blink;staddr:integer;quantity:integer):integer;stdcall;external 'modbus_tcp.dll';//写保持寄存器

参数：

socket：modbus_tcp 连接 socket 索引号（控制器 1 用数值 1，控制器 2 用数值 2……）

addr：控制器 modbus 地址

psend：发送数据指针

quantity：写保持寄存器的数量

返回值：

-1：未知错误；-2：tcp 没有连接； -3：staddr 错误； -4：quantity 错误；-5：超时错误；

0：正确返回；

1：非法功能；2：非法数据地址；3：非法数据值；4：从站设备故障

procedure mbtcp_free(socket:byte);stdcall;external 'modbus_tcp.dll';//释放 TCP 连接资源

参数：

socket：modbus_tcp 连接 socket 索引号（控制器 1 用数值 1，控制器 2 用数值 2……）

procedure mbtcp_disconnect(socket:byte);stdcall;external 'modbus_tcp.dll';//TCP 断开

参数：

socket：modbus_tcp 连接 socket 索引号（控制器 1 用数值 1，控制器 2 用数值 2……）

procedure float_b2l(Buff_addr:blink;ptr_float:blink);
stdcall;external 'modbus_tcp.dll';//浮点数大端格式切换到小端格式

参数：

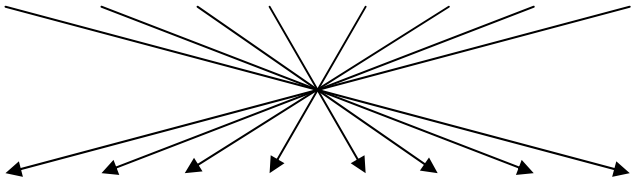
Buffer_addr：待转换的大端格式浮点数存放地址

ptr_float:转换后的小端格式浮点数存放地址

modbus 协议采用大端格式传送数据。由于 modbus 没有定义 IEEE-754 双精度浮点数（64 位）的传送格式，本控制器规定 IEEE-754 双精度浮点数（64 位）也采用大端格式传送。PC 机一般是使用小端格式保存浮点数，因此当 PC 机读取控制器的浮点数时，需要将大端格式的浮点数转换成小端格式的浮点数。

例如：123.456 大端格式为 0x40 0x5E 0xDD 0x2F 0x1A 0x9F 0xBE 0x77 **（Buffer_addr）**

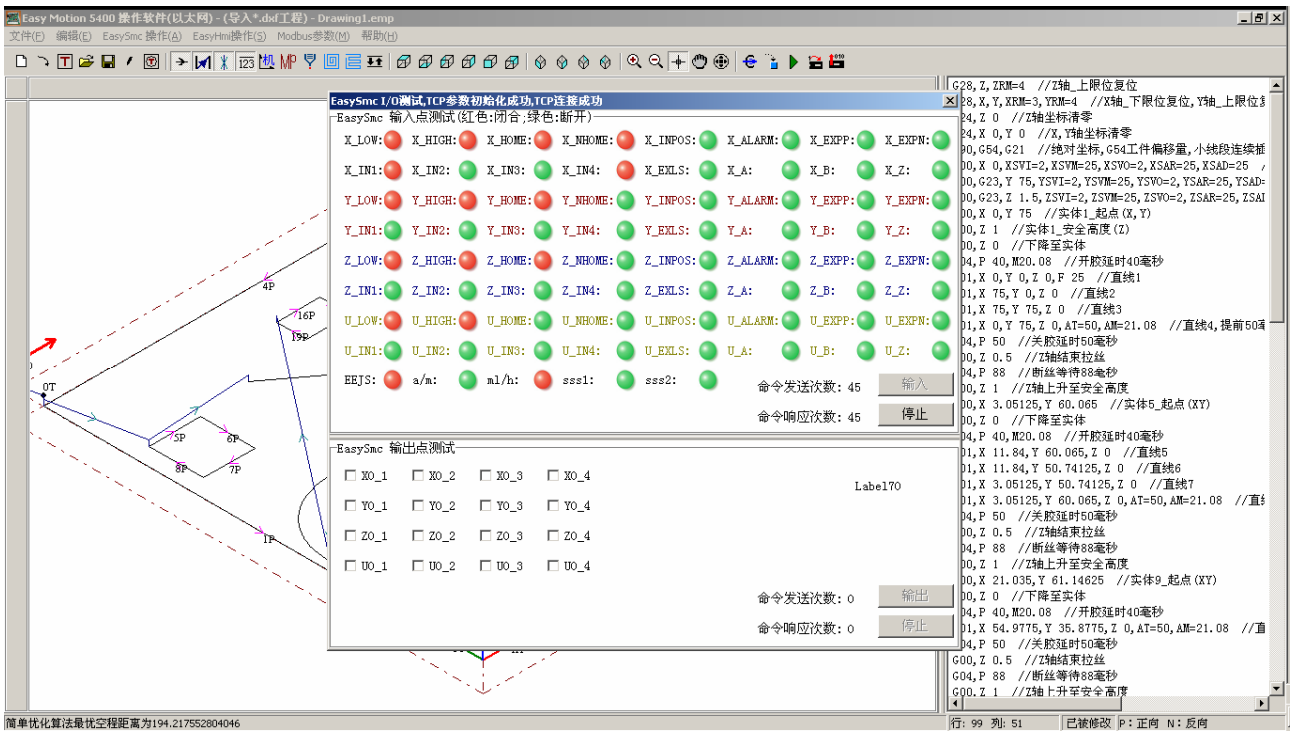
使用 float_b2l 转换结果为



小端格式为 0x77 0xBE 0x9F 0x1A 0x2F 0xDD 0x5E 0x40 **（ptr_float）**

基于 float_b2l 的工作原理，当 PC 机要写入浮点数到控制器时，也可以使用 float_b2l 将小端格式的浮点数转换成大端格式的浮点数，然后写入控制器。

演示软件 EasyMotion5470(以太网) Delphi 版本截图



8.2 modbus_tcp.dll Visual C++函数声明

参数和返回值请参考 8.1 modbus_tcp.dll Delphi 函数声明。

```
typedef long (_stdcall *lpmbtcp_init)( unsigned char socket,unsigned char pc_ip1,unsigned char pc_ip2,unsigned char pc_ip3,unsigned char pc_ip4,long pc_port,unsigned char host_ip1,unsigned char host_ip2,unsigned char host_ip3,unsigned char host_ip4,long host_port,long readtimeout);//初始化
typedef long (_stdcall *lpmbtcp_connect)( unsigned char socket,long timeout);//连接
typedef long (_stdcall *lpmbtcp_readcoils)( unsigned char socket,long addr,unsigned char * prec,long staddr,long quantity);//读线圈
typedef long (_stdcall *lpmbtcp_readdiscreteinputs)( unsigned char socket,unsigned char addr,unsigned char * prec,long staddr,long quantity);//读离散量输入
typedef long (_stdcall *lpmbtcp_readholdingreg)( unsigned char socket,unsigned char addr,unsigned char * prec,long staddr,long quantity);//读保持寄存器
typedef long (_stdcall *lpmbtcp_readinputreg)( unsigned char socket,unsigned char addr,unsigned char * prec,long staddr,long quantity);//读输入寄存器
typedef long (_stdcall *lpmbtcp_readfilerecord)( unsigned char socket,unsigned char addr,unsigned char * prec,long fileno,long staddr,long quantity);//读文件记录
typedef long (_stdcall *lpmbtcp_writesinglecoil)( unsigned char socket,unsigned char addr,unsigned char value,long staddr);//写单个线圈
typedef long (_stdcall *lpmbtcp_writemultiplecoils)( unsigned char socket,unsigned char addr,unsigned char * psend,long staddr,long quantity);//写多个线圈
typedef long (_stdcall *lpmbtcp_writefilerecord)( unsigned char socket,unsigned char addr,unsigned char * psend,long fileno,long staddr,long quantity,long anothertimeout);//写文件记录
typedef long (_stdcall *lpmbtcp_writemultiplereg)( unsigned char socket,unsigned char addr,unsigned char * psend,long staddr,long quantity);//写多个寄存器
typedef void (_stdcall *lpmbtcp_free)( unsigned char socket);//释放 TCP 连接所用资源
typedef void (_stdcall *lpmbtcp_disconnect)( unsigned char socket);//TCP 断开

typedef void (_stdcall *lpfloat_b2l)(unsigned char *Buff_addr,unsigned short *ptr_float);
……
HINSTANCE hDll; //句柄
lpmbtcp_init mbtcp_init;//函数指针
lpmbtcp_connect mbtcp_connect;
lpmbtcp_readcoils mbtcp_readcoils;
lpmbtcp_readdiscreteinputs mbtcp_readdiscreteinputs;
lpmbtcp_readholdingreg mbtcp_readholdingreg;
lpmbtcp_readinputreg mbtcp_readinputreg;
lpmbtcp_readfilerecord mbtcp_readfilerecord;
lpmbtcp_writesinglecoil mbtcp_writesinglecoil;
lpmbtcp_writemultiplecoils mbtcp_writemultiplecoils;
lpmbtcp_writefilerecord mbtcp_writefilerecord;
lpmbtcp_writemultiplereg mbtcp_writemultiplereg;
lpmbtcp_disconnect mbtcp_disconnect;
lpmbtcp_free mbtcp_free;
lpfloat_b2l float_b2l;
```

```
……
HINSTANCE hDll; //句柄
lpmbtcp_init mbtcp_init;
lpmbtcp_connect mbtcp_connect;
lpmbtcp_readcoils mbtcp_readcoils;
lpmbtcp_readdiscreteinputs mbtcp_readdiscreteinputs;
lpmbtcp_readholdingreg mbtcp_readholdingreg;
lpmbtcp_readinputreg mbtcp_readinputreg;
lpmbtcp_readfilerecord mbtcp_readfilerecord;
lpmbtcp_writesinglecoil mbtcp_writesinglecoil;
lpmbtcp_writemultiplecoils mbtcp_writemultiplecoils;
lpmbtcp_writefilerecord mbtcp_writefilerecord;
lpmbtcp_writemultiplereg mbtcp_writemultiplereg;
lpmbtcp_disconnect mbtcp_disconnect;
lpmbtcp_free mbtcp_free;
lpfloat_b2l float_b2l;
……
hDll = LoadLibrary("modbus_tcp.dll");//加载 dll

mbtcp_init = (lpmbtcp_init)GetProcAddress(hDll,"mbtcp_init");
mbtcp_connect = (lpmbtcp_connect)GetProcAddress(hDll,"mbtcp_connect");
mbtcp_readcoils = (lpmbtcp_readcoils)GetProcAddress(hDll,"mbtcp_readcoils");
mbtcp_readdiscreteinputs = (lpmbtcp_readdiscreteinputs)GetProcAddress
                           (hDll,"mbtcp_readdiscreteinputs");
mbtcp_readholdingreg = (lpmbtcp_readholdingreg)GetProcAddress
                           (hDll,"mbtcp_readholdingreg");
mbtcp_readinputreg = (lpmbtcp_readinputreg)GetProcAddress
                           (hDll,"mbtcp_readinputreg");
mbtcp_readfilerecord = (lpmbtcp_readfilerecord)GetProcAddress
                           (hDll,"mbtcp_readfilerecord");
mbtcp_writesinglecoil = (lpmbtcp_writesinglecoil)GetProcAddress
                           (hDll,"mbtcp_writesinglecoil");
mbtcp_writemultiplecoils = (lpmbtcp_writemultiplecoils)GetProcAddress
                           (hDll,"mbtcp_writemultiplecoils");
mbtcp_writefilerecord = (lpmbtcp_writefilerecord)GetProcAddress
                           (hDll,"mbtcp_writefilerecord");
mbtcp_writemultiplereg = (lpmbtcp_writemultiplereg)GetProcAddress
                           (hDll,"mbtcp_writemultiplereg");
mbtcp_free = (lpmbtcp_free)GetProcAddress(hDll,"mbtcp_free");
mbtcp_disconnect = (lpmbtcp_disconnect)GetProcAddress(hDll,"mbtcp_disconnect");
float_b2l = (lpfloat_b2l)GetProcAddress(hDll,"float_b2l");
……
  FreeLibrary(hDll);//释放 dll
```

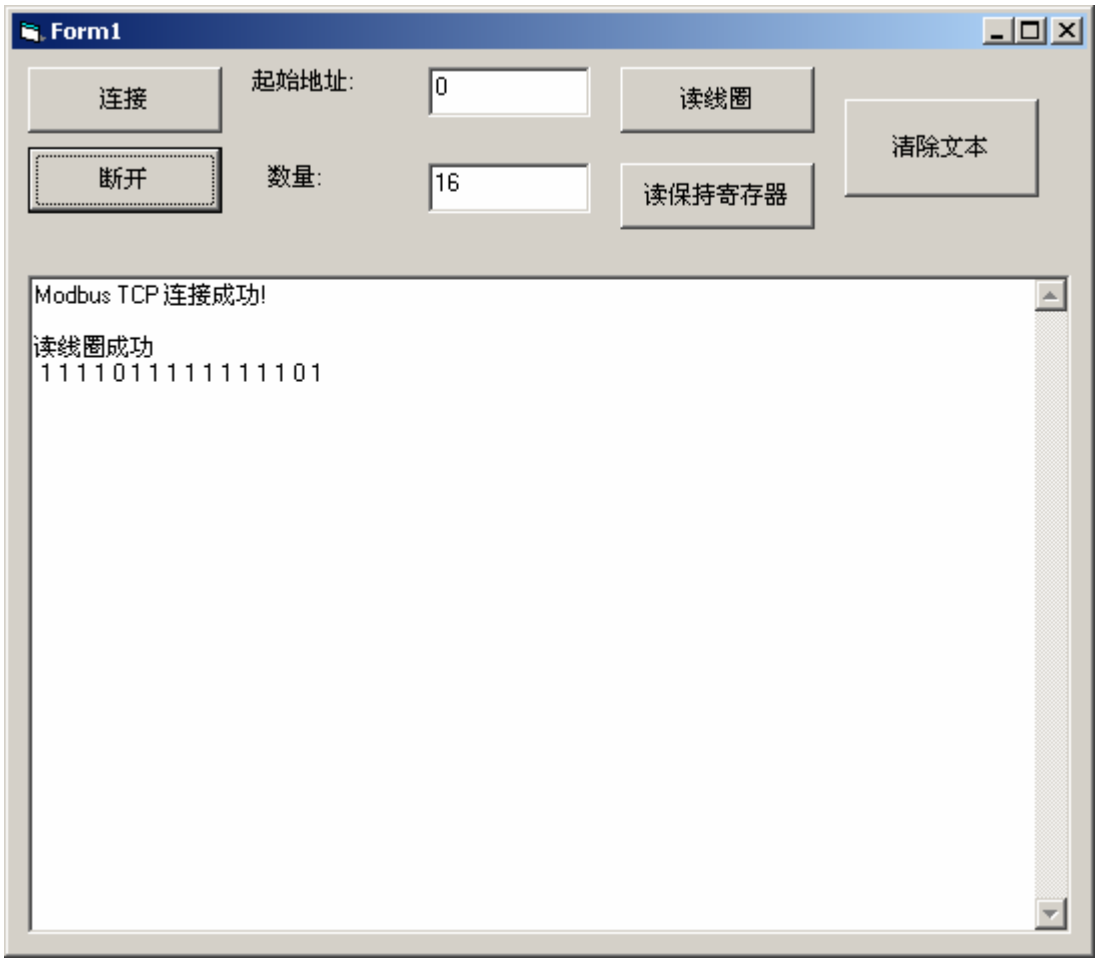
演示软件 EasyMotion5470(以太网) Visual C++版本截图，提供源代码

8.3 modbus_tcp.dll Visual Basic函数声明

参数和返回值请参考 8.1 modbus_ tcp.dll Delphi 函数声明。

```
Private Declare Function mbtcp_init Lib "modbus_tcp.dll" (ByVal socket As Byte, ByVal pc_ip1 As Byte, ByVal pc_ip2 As Byte, ByVal pc_ip3 As Byte, ByVal pc_ip4 As Byte, ByVal pc_port As Long, ByVal host_ip1 As Byte, ByVal host_ip2 As Byte, ByVal host_ip3 As Byte, ByVal host_ip4 As Byte, ByVal host_port As Long, ByVal readtimeout As Long) As Long

Private Declare Function mbtcp_connect Lib "modbus_tcp.dll" (ByVal socket As Byte, ByVal timeout As Long) As Long

Private Declare Function mbtcp_readcoils Lib "modbus_tcp.dll" (ByVal socket As Byte, ByVal addr As Byte, ByRef prec As Byte, ByVal staddr As Long, ByVal quantity As Long) As Long

Private Declare Function mbtcp_readdiscreteinputs Lib "modbus_tcp.dll" (ByVal socket As Byte, ByVal addr As Byte, ByRef prec As Byte, ByVal staddr As Long, ByVal quantity As Long) As Long

Private Declare Function mbtcp_readholdingreg Lib "modbus_tcp.dll" (ByVal socket As Byte, ByVal addr As Byte, ByRef prec As Byte, ByVal staddr As Long, ByVal quantity As Long) As Long

Private Declare Function mbtcp_readinputreg Lib "modbus_tcp.dll" (ByVal socket As Byte, ByVal addr As Byte, ByRef prec As Byte, ByVal staddr As Long, ByVal quantity As Long) As Long

Private Declare Function mbtcp_readfilerecord Lib "modbus_tcp.dll" (ByVal socket As Byte, ByVal addr As Byte, ByRef prec As Byte, ByVal fileno As Long, ByVal staddr As Long, ByVal quantity As Long) As Long

Private Declare Function mbtcp_writefilerecord Lib "modbus_tcp.dll" (ByVal socket As Byte, ByVal addr As Byte, ByRef prec As Byte, ByVal fileno As Long, ByVal staddr As Long, ByVal quantity As Long, ByVal anothertimeout As Long) As Long

Private Declare Function mbtcp_writesinglecoil Lib "modbus_tcp.dll" (ByVal socket As Byte, ByVal addr As Byte, ByVal value As Byte, ByVal staddr As Long) As Long

Private Declare Function mbtcp_writemultiplecoils Lib "modbus_tcp.dll" (ByVal socket As Byte, ByVal addr As Byte, ByRef psend As Byte, ByVal staddr As Long, ByVal quantity As Long) As Long

Private Declare Function mbtcp_writemultiplereg Lib "modbus_tcp.dll" (ByVal socket As Byte, ByVal addr As Byte, ByRef psend As Byte, ByVal staddr As Long, ByVal quantity As Long) As Long

Private Declare Sub mbtcp_free Lib "modbus_tcp.dll" (ByVal socket As Byte)

Private Declare Sub mbtcp_disconnect Lib "modbus_tcp.dll" (ByVal socket As Byte,)

Private Declare Sub float_b2l Lib "modbus_tcp.dll" (ByRef Buff_addr As Byte, ByRef ptr_float As Byte)
```

演示软件 EasyMotion5470(以太网) Visual Basic 版本截图，提供源代码



8.4 modbus_tcp.dll调用顺序

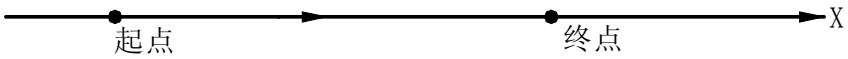
开始使用 mbtcp_init，再使用 mbtcp_connect，而后可以使用 mbtcp_readcoils 等功能函数，结束使用 mbtcp_disconnect，再使用 mbtcp_free。

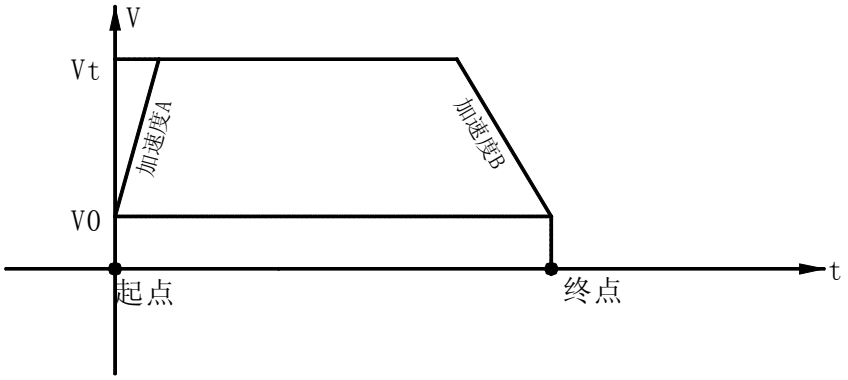
九、G代码、M、F、T代码说明

9.1 G00 独立运动

程序格式为：G00,X _ ,Y _ ,Z _ , U _ ,

特殊参数：XSVI,XSVM,XSVO,XSAR,XSAD：X 轴独立运动参数，起跳速度 V0，最高速度 Vt，出口速度 V0，加速度 A，加速度 B





YSVI,YSVM,YSVO,YSAR,YSAD：Y 轴独立运动参数，
ZSVI,ZSVM,ZSVO,ZSAR,ZSAD：Z 轴独立运动参数，
USVI,USVM,USVO,USAR,USAD：U 轴独立运动参数，

举例 1：G00,X 10.1,Y 20.2,Z 30.3,U 40.4

举例 2：G00,X 0,XSVI=2,XSVM=25,XSVO=2,XSAR=25,XSAD=25

9.2 G01 直线插补运动

程序格式为：G01,X _ ,Y _ ,Z _ , U _ ,

举例：G00,X 10.1,Y 20.2,Z 30.3,U 40.4

9.3 G02 顺时针圆弧插补运动、G03 逆时针圆弧插补运动

程序格式为：G17,G02（或者 G03）,X _ ,Y _ ,I _ , J _ ,

G18,G02（或者 G03）,X _ ,Z _ ,I _ , K _ ,

G19,G02（或者 G03）,Y _ ,Z _ ,J _ , K _ ,

X,Y,Z 分别为 X,Y,Z 轴圆弧终点

I,J,K 分别为 X,Y,Z 轴圆心相对圆弧起点（当前点）的坐标增量

如果 G17 选择 XY 平面圆弧插补，参数出现 Z 或者 U 坐标，则 Z 或者 U 坐标作为螺线插补第 3 轴

9.4 G04 暂停

程序格式为：G04,P _

P 为延时时间，单位毫秒

9.5 G10 设置偏移量

程序格式为：G10.101,X _ ,Y _ ,Z _ , U _ ,

G10.101-G10.106 设置工具偏移量 tool_offset1 - tool_offset6

G10.201-G10.206 设置夹具偏移量 work_offset1 - work_offset6

9.6 工作平面选择

G17 选择 XY 平面

G18 选择 XZ 平面

G19 选择 YZ 平面

9.7 插补模式选择

G20 标准插补模式

G21 小线段连续插补模式

9.8 G24 设置逻辑坐标

程序格式为：G24,X _ ,Y _ ,Z _ , U _ ,

9.9 复位运动 G28

程序格式为：G28,X _ ,Y _ ,Z _ , U _ ,

特殊参数：XRM，XRSN,XRSC,XRSM,XRVI：X 轴复位运动参数，复位模式，反向复位距离，抗抖动距离，复位最小距离，复位速度

复位模式说明：

1：向上原点复位

2：向下原点复位

3：下限位复位

4：上限位复位

5：向上 Z 相复位

6：向下 Z 相复位

7：快速向上原点复位

8：快速向下原点复位

YRM，YRSN,YRSC,YRSM,YRVI：Y 轴复位运动参数，

ZRM，ZRSN,ZRSC,ZRSM,ZRVI：Z 轴复位运动参数，

URM，URSN,URSC,URSM,URVI：U 轴复位运动参数，

举例：G28,X,Y,XRM=3,YRM=4 //X轴_下限位复位,Y轴_上限位复位

9.10 G53 选择机床坐标

当运动命令前出现 G53 时，工具偏移量无效，夹具偏移量无效，选择机床坐标系

9.11 G54、G55、G56、G57、G58、G59 选择夹具偏移量

G54 - G59 选择相应的 work_offset1 - work_offset6 夹具偏移量

9.12 坐标模式选择

G90 绝对坐标

G91 相对坐标

9.13 M、F、T 代码

M 代码：

M20.00-M20.15：输出 XO-1…UO-4 低电平

M21.00-M21.15：输出 XO-1…UO-4 高电平

M22.00-M22.63：等待 X、Y、Z、U 输入低电平

M23.00-M23.63：等待 X、Y、Z、U 输入高电平

F 代码：

插补运动进给速度，单位为 毫米/秒

T 代码：

工具选择 T1-T6，启用对应工具偏移量 tool_offset1-tool_offset6

9.14 提前输出代码

提前输出只适用于 G21,G02,G03，主要用于点胶机系统的提前关胶

AT：提前时间，单位毫秒

AM：提前输出，同 M20，M21

举例：G01,X 0,Y 75,Z 0,AT=50,AM=21.08 //提前 50 毫秒,关闭 ZO-1

十、故障检查、维修保养、售后服务

自出厂之日起 1 年内免费保修。

厦门华菱工控技术有限公司

Xiamen Hualing Industry Control Technology Co.,Ltd

地址： 福建省厦门市高科技产业开发区光厦楼北 5 楼

邮编： 361006

联系电话：0592-6798422 传真：0592-5660695

技术支持：13799757073（林先生）

E-mail：linrongfeng77@21cn.com QQ：49579631