

MDCS-510

驱控一体用户手册



历史版本

版本号	修订内容	修订人	修订时间
V1.0	新建	王 鹏	2021-12-27

 注意	请确保仔细阅读过本手册或在使用之前受到过足够专业的培训，否则可能给机械或人身带来危险！
 触电风险	请勿将系统置于液体、金属碎屑飞溅的地方，如环境工况无法避免，应务必设置防护罩！

目 录

第一章 规格参数.....	1
1.1 规格参数表.....	1
第二章 控制器介绍.....	2
2.1 控制器 2D 尺寸图.....	2
2.2 系统接口说明.....	3
第三章 接线定义.....	4
3.1 DI 输入电路图.....	4
3.2 DO 输出电路图.....	4
3.3 本地 I/O 输入输出端口.....	5
3.4 EXTIO 端口接线定义.....	6
3.5 扩展 I/O 板接线定义（选件）.....	6
3.5.1 接线端说明.....	7
3.5.2 安装说明.....	7
3.6 RS232/485 端口接线定义.....	7
3.7 HMI 端口接线定义.....	8
3.8 连接电机接头接线定义.....	8
3.9 BRK 电机刹车端口接线定义.....	8
3.10 接地注意事项.....	9
3.10.1 接地标准.....	9
3.10.2 接地示意图.....	9
第四章 示教器介绍.....	10
4.1 示教器布局图.....	10
4.2 示教器控制开关功能说明.....	11
第五章 软件界面使用介绍.....	12
5.1 界面区域介绍.....	12
5.2 工程管理.....	13
5.2.1 工程文件管理.....	14
5.2.2 工程内容界面.....	14
5.2.3 工程任务配置.....	15
5.2.4 程序导入/导出界面（U 盘）.....	18
5.2.5 点位表.....	19
5.2.6 I/O 监控.....	23
5.2.7 变量（界面）.....	24

5.2.8 通讯.....	26
5.2.9 工艺.....	30
5.2.10 程序编辑.....	31
5.3 设置.....	35
5.3.1 坐标系.....	37
5.3.2 监控区.....	41
5.3.3 参数.....	44
5.3.4 远程配置.....	51
5.3.5 IO 配置.....	53
5.3.6 通信配置.....	53
5.3.7 版本.....	54
5.3.8 系统设置.....	56
5.3.9 续期.....	57
5.3.10 脱机.....	57
5.4 实时信息监控.....	58
5.4.1 坐标系统监控.....	58
5.4.2 电机监控.....	58
5.4.3 IO 监视.....	58
5.4.4 机器人电机轴编码器.....	58
5.5 历史信息.....	59
第六章 指令系统介绍.....	60
6.1 编程规范.....	60
6.1.1 工程文件结构.....	60
6.1.2 变量种类及作用范围.....	60
6.1.3 自定义变量声明种类.....	61
6.1.4 自定义函数.....	62
6.1.5 关键字列表.....	65
6.1.6 运算符列表.....	66
6.2 指令功能说明.....	67
6.2.1 运动类指令.....	67
6.2.2 语法指令.....	83
6.2.3 点操作类指令.....	94
6.2.4 系统类指令.....	115
6.2.5 I/O 指令.....	132

6.2.6 通信指令.....	140
6.2.7 字符串指令.....	169
6.2.8 其他指令.....	193
第七章 通讯功能介绍.....	210
7.1 通讯功能简介.....	210
7.2 串口（RS-485）/EXTIO 使用方法.....	210
7.2.1 接线.....	211
7.2.2 自由协议模式（COM1/COM2）	211
7.2.3 Modbus-RTU 模式（COM1）	212
7.2.4 扩展 I/O 的设置（EXTIO）	213
7.2.5 串口通信范例(自由协议).....	216
7.3 网口（Ethernet）使用方法.....	217
7.3.1 协议模式说明.....	217
7.3.2 自由协议模式(TCP/IP).....	217
7.3.3 Modbus-TCP 模式配置.....	218
7.3.3 网口通信范例（自由协议机器人做客户端）	219
7.3.4 网口通信范例（自由协议机器人做服务器）	220
第八章 附录.....	221
附录 A 常见报警.....	221
附录 B Modbus 通讯地址表.....	233

第一章 规格参数

1.1 规格参数表

型号	MDCS-510				
输入电压	单相 AC 200V~240V , 50/60Hz				
最大输入功率	4.0 KW				
最大输出功率	J1 750W	J2 750W	J3 750W	J4 750W	J5 750W
环境温度	-25℃~40℃				
相对湿度	20%~80% (无凝露)				
数字 I/O		输入		输出	
	数量	32 点		32 点	
	类型	NPN/PNP (拨码配置)		固定输出 0V	
	最大输出电流	N/A		开漏输出≤167mA	
通信接口	USB 接口	1 个 (版本升级)			
	串行通信	RS485 ×1 路 , RS232×1 路			
	网络通信	RJ45×1 个 (软件:4 路)			
	总线接口	EtherCat×1 个 (暂不支持使用)			
示教器	屏幕	8" TFT			
	分辨率	800*600			
	触摸屏类型	电阻式			
	模式切换开关	钥匙型 (手动、自动、远程)			
	急停开关	1 个			
	使能开关	3 档型 1 个			
	外部端口	USB x1 个			
重量	控制器	约 8 Kg			
	示教器	855g			
尺寸	控制器	长*宽*高 : 367*210*256mm			
	示教器	长*宽*高 : 270*194*59mm			

第二章 控制器介绍

2.1 控制器 2D 尺寸图

MDCS-510 控制系统结构尺寸图如下所示：

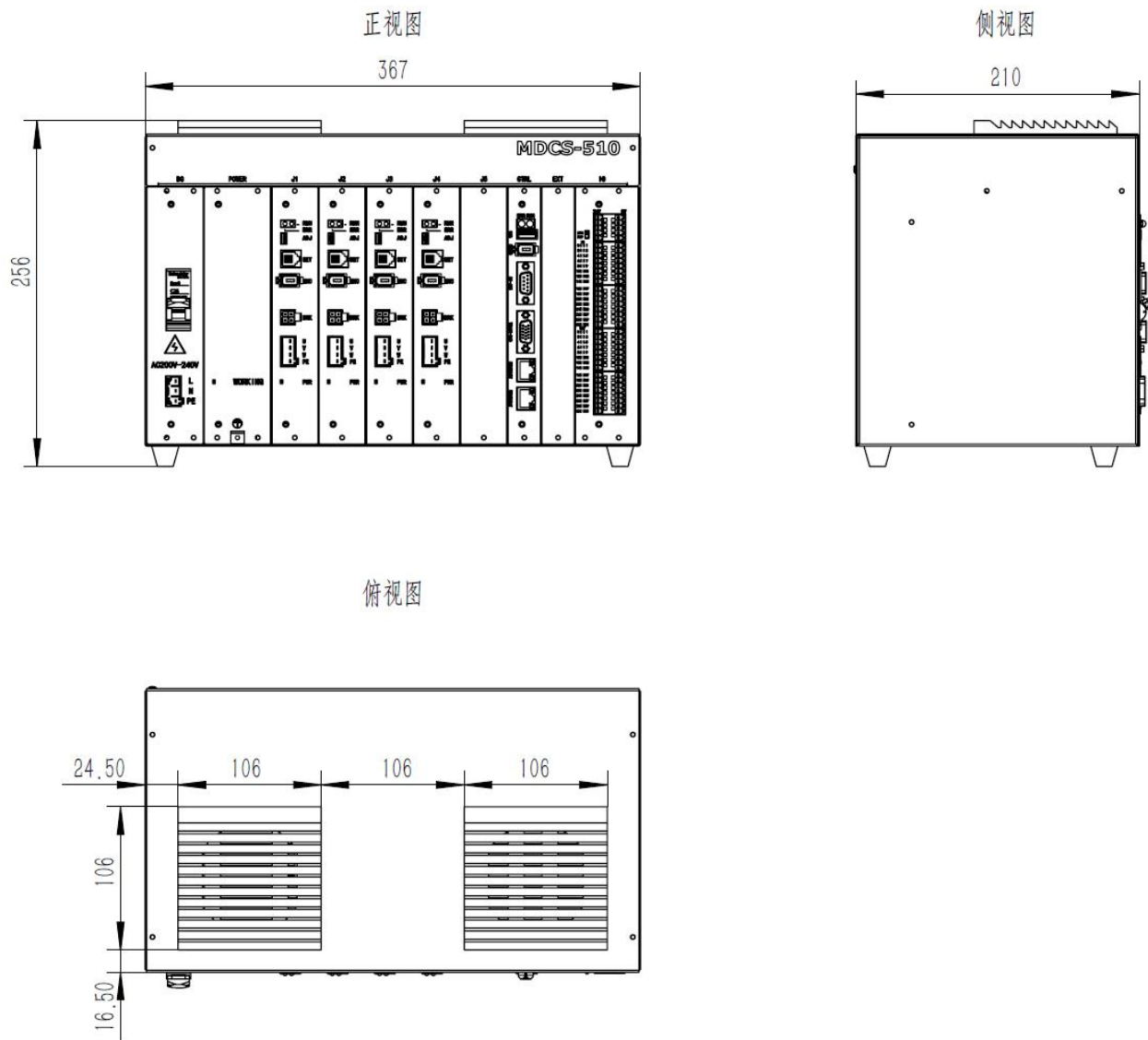


图 2-1 MDCS-510 外框尺寸图



注意

在柜子中实际摆放时：

- 1、需在接口面板侧预留足够的接头位置，约 50mm 以上；
- 2、不要遮挡控制柜侧面的散热风扇，预留 100mm 以上。

2.2 系统接口说明

系统接口面板的布局如图 2-2 所示。

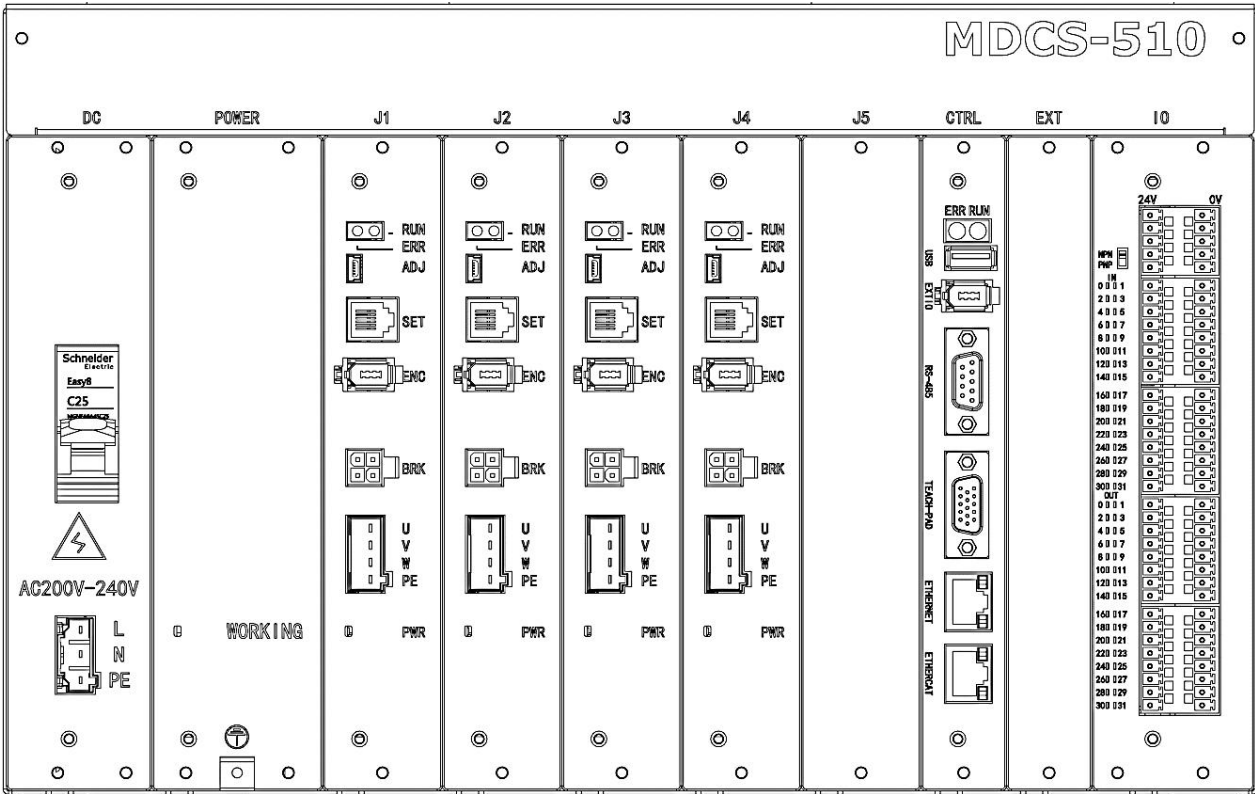


图 2-2 接口面板布局图

名称 / 标识	功能	名称 / 标识	功能
ON<->OFF	电源开关	EXTIO	扩展 I/O 接口
AC220V	电源接口	RS485	串行接口
ADJ	伺服调试接口	TEACH-PAD	示教器接口
SET	伺服调试板接口	ETHERNET	网口
ENC	电动机编码器接口	ETHERCAT	总线接口（暂不支持）
BRK	刹车线接口（J3）	NPN-PNP	输入电平拨码
U V W PE	电动机动力线接口	RUN	运行指示灯
USB	控制器软件升级接口	ERR	报警指示灯
24V	24V 电源接线端子	PWR	伺服电源指示灯
0V	0V 电源接线端子		接地端

第三章 接线定义

3.1 DI 输入电路图

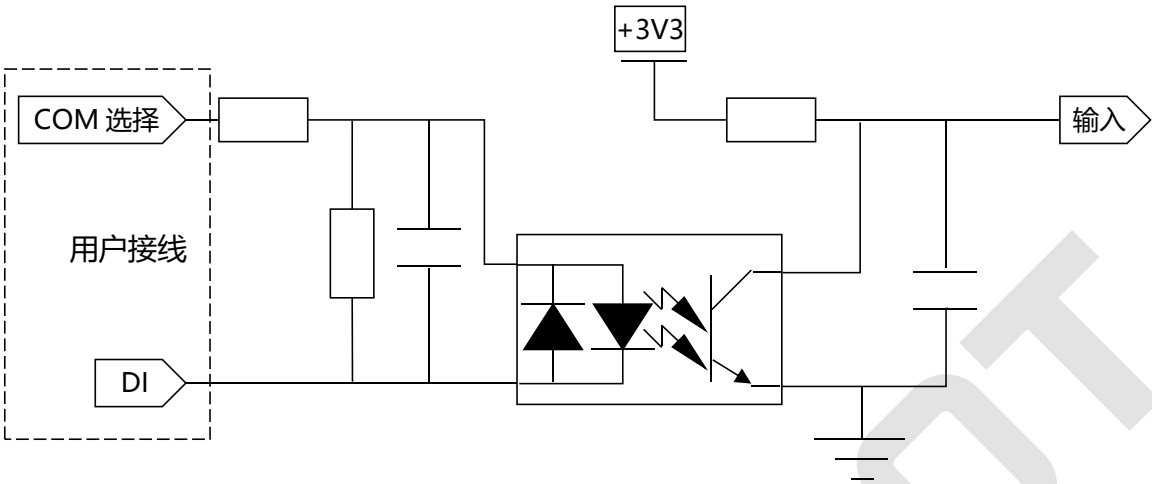


图 3-1 输入电路图

	NPN 型输入器件请将拨码拨到 NPN 侧；PNP 型将拨码拨到 PNP 侧； 系统内部 24V 有无供电，以实际测量为准！
注意	

3.2 DO 输出电路图

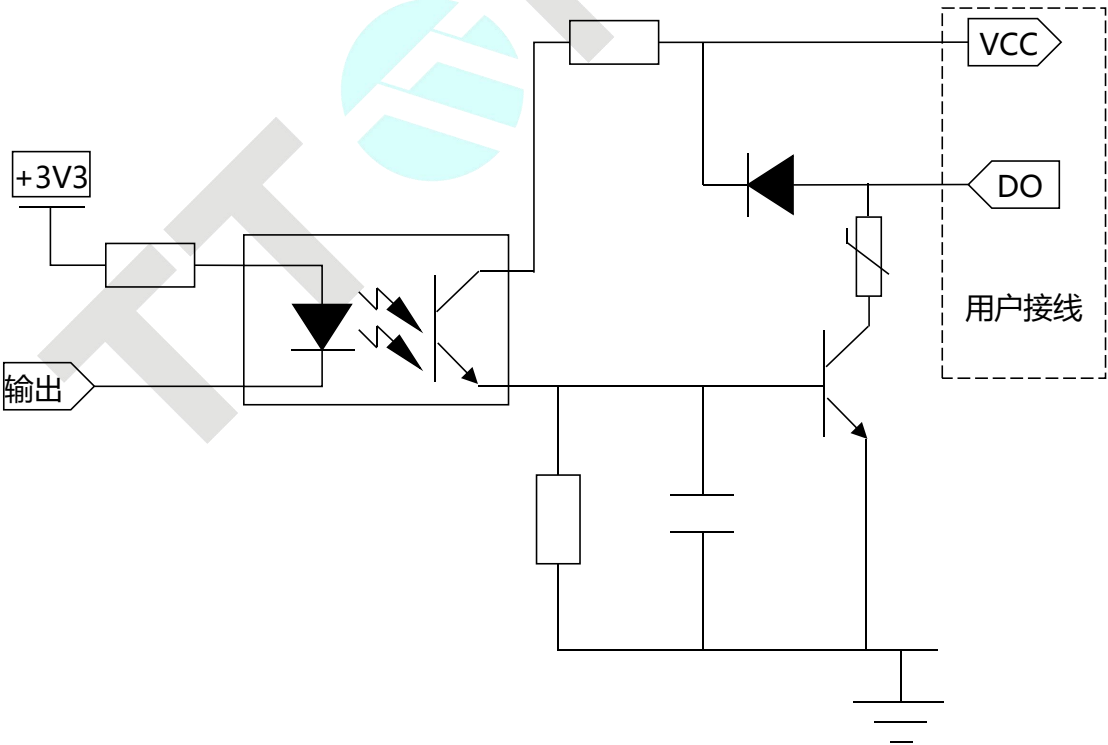


图 3-2 输出电路图

3.3 本地 I/O 输入输出端口

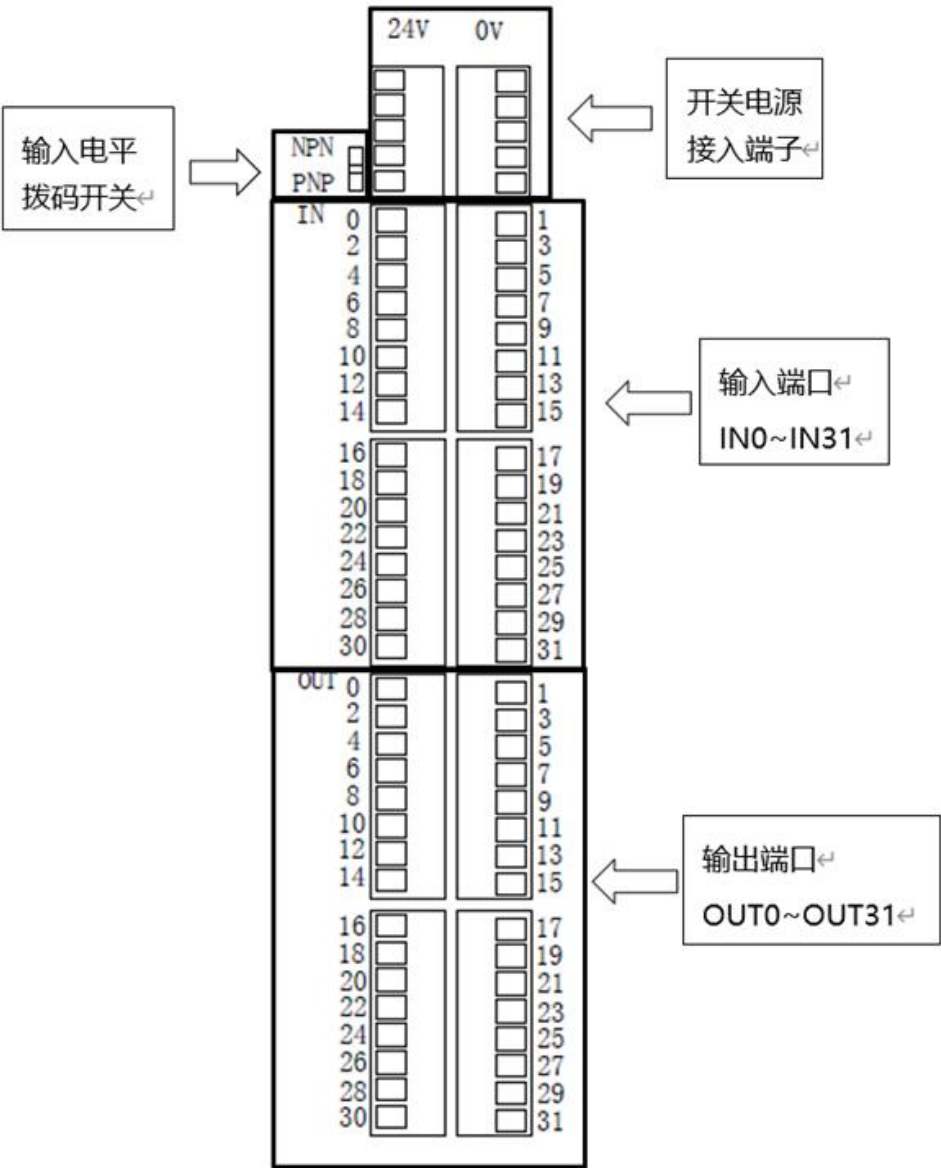


图 3-3 本地 I/O 布局图

端口	说明
开关电源接入端子	1.IO 内部无电源，需要外接电源（特殊订单里面预装电源） 2.输入端口，输出端口，共用电源
输入端口	1. 输入端口为 32 个 2. 输入端口可通过拨码开关选择输入电平为 NPN 或 PNP
输出端口	1. 输出端口为 32 个 2. 输出端口规定为 0V 输出

3.4 EXTIO 端口接线定义

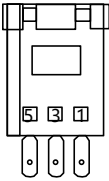
	引脚定义 EXTIO (1394 母头)			
	1	内部占用	4	EXT485_A-
	2	EXT485_A+	5	空
	3	内部占用	6	空

图 3-4-1 RS232/485 端口接线图



提示

关于 EXTIO 端口说明：485+和 485-分别构成 485 I/O 扩展总线，扩展 I/O 设备（选件）分别连入该总线即可。详细使用说明参考 7.2.4 扩展 I/O 的设置（EXTIO）

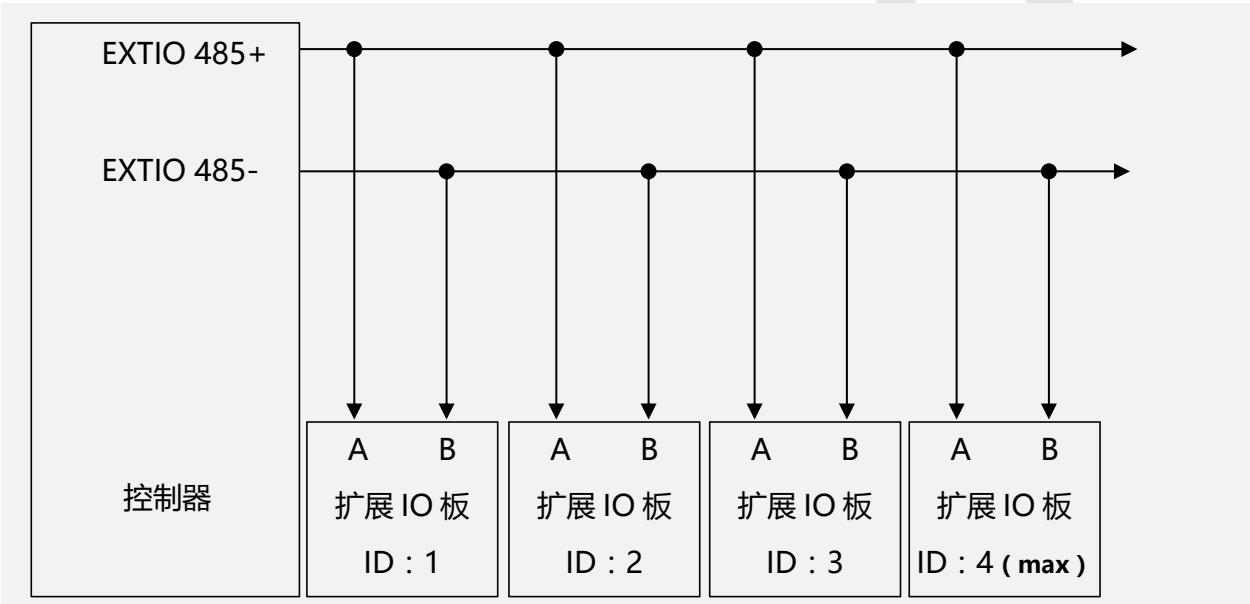


图 3-4-2 EXTIO 扩展总线示意图

3.5 扩展 I/O 板接线定义（选件）

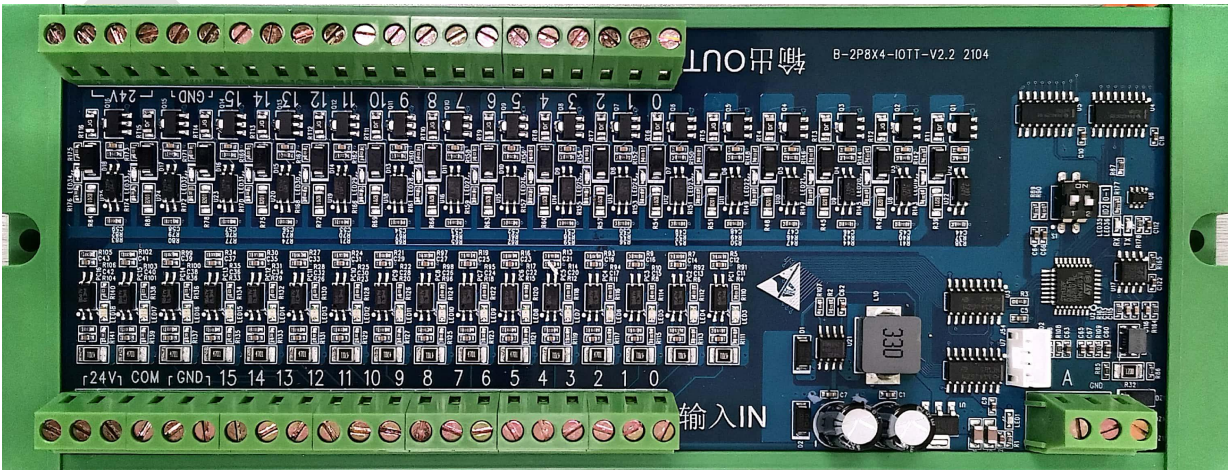


图 3-5 扩展 IO 板

3.5.1 接线端说明

接线端子	说明
24V	1、用户需额外配置 24V 电源给扩展 IO 板供电； 2、输入部分与输出部分的 24V 之间和 0V 之间内部是连通的； 3、扩展 I/O 板供电不要采用系统 IO 中的 24V，系统直流 24V 电源设计功率不包含扩展 I/O 板的消耗；
COM	1、NPN 类型（低电平有效）的输入器件，COM 端短接 24V； 2、PNP 类型（高电平有效）的输入器件，COM 端短接 0V； 3、输出固定为 0V 输出，与 COM 端子无关；
GND	1、0V（外置 24V 电源的 0V）； 2、扩展板内输入部分与输出部分的 0V 是连通的；
0....15	输入、输出接线引脚；
B A GND	485 通讯引脚，B 接 485-，A 接 485+； 本系统 485 通讯采用的是 2 线制，GND 无需接线；

3.5.2 安装说明

尺寸	说明
板长	218mm
板宽	86mm
安装方式	适合 35mm 标准导轨，卡扣式安装

 提示	关于扩展 I/O 的使用，请参考 7.2.4 扩展 I/O 的设置详细说明。
---	--

3.6 RS232/485 端口接线定义

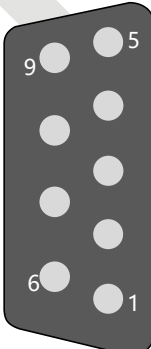
	引脚定义 RS232/485(DB9-F)			
	1	1#485+	6	1#485-
	2	232RX	7	空
	3	232TX	8	空
	4	空	9	空
	5	GND		

图 3-6 RS232/485 端口接线图

3.7 HMI 端口接线定义

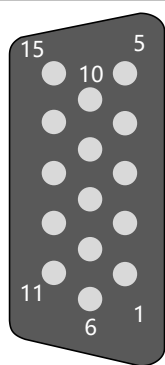


图 3-7 HMI 端口接线图

引脚定义 HMI(DB15-3R-F)

1	NET-RX+	6	NET-TX+	11	NET-TX-
2	HMI485+	7	NET-RX-	12	空
3	IN4	8	HMI485-	13	空
4	IN2	9	IN3	14	空
5	GND	10	IN1	15	24V

3.8 连接电机接头接线定义

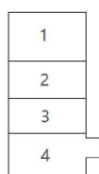


图 3-8-1 电机动力线接线图

引脚定义 HMI(DB15-3R-F)

1	U
2	V
3	W
4	PE
1	5V
2	D+
3	0V
4	D-

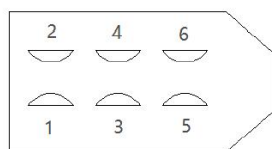


图 3-8-1 电机动力线接线图

3.9 BRK 电机刹车端口接线定义

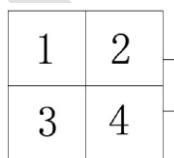


图 3-9 BRK 电机刹车接线图

引脚定义 HMI(DB15-3R-F)

1	BRK-
2	BRK+
3	24V
4	空

3.10 接地注意事项

3.10.1 接地标准

- 就近接地，控制器接地安装螺丝到接地端长度不超过 20cm
- 接地线规格：不小于 0.75 平方的黄绿线

3.10.2 接地示意图

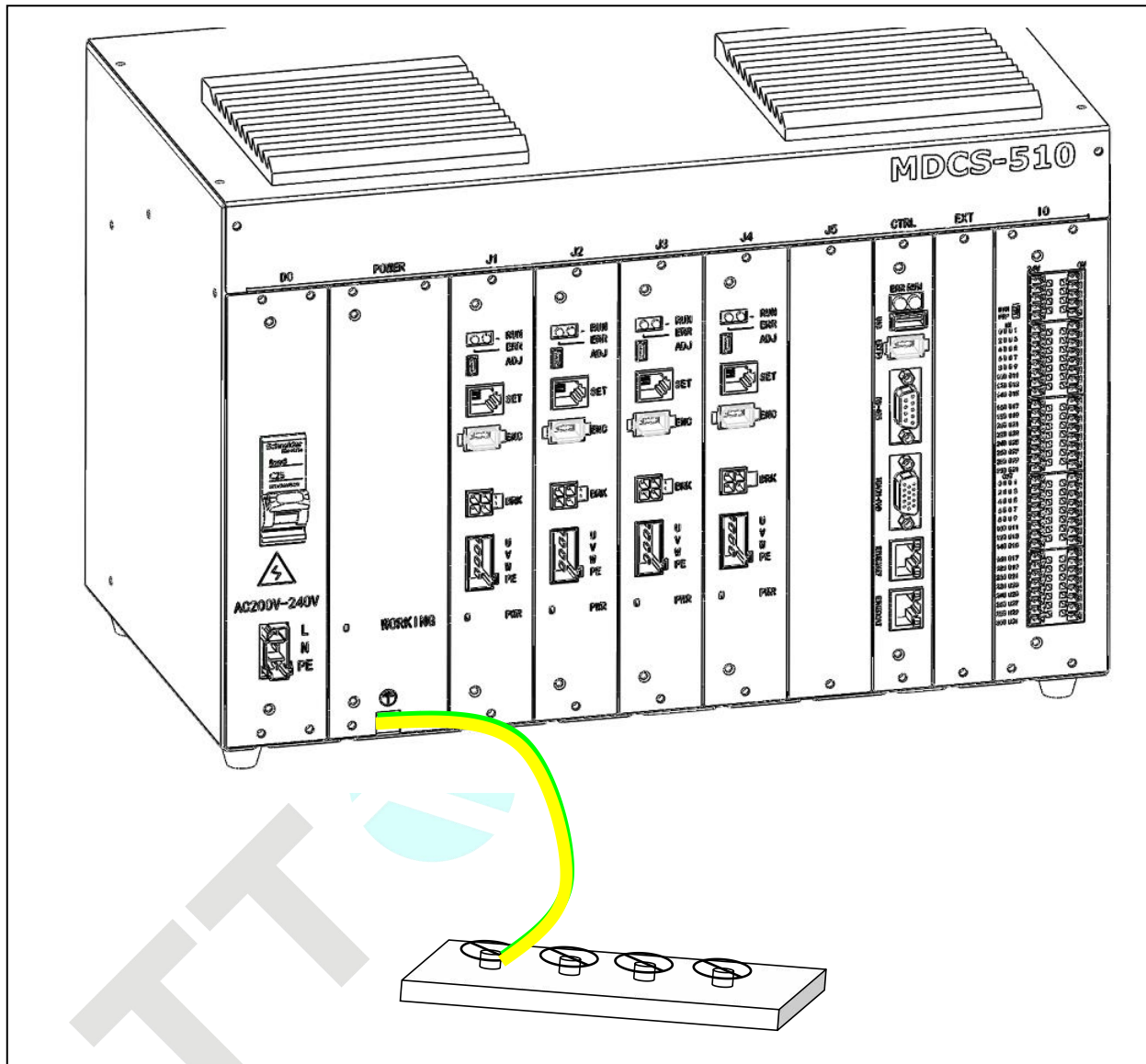


图 3-10 控制器接地示意图



注意

请确保系统可靠接地，否则可能引发通讯相关故障。

第四章 示教器介绍

4.1 示教器布局图

如图 4-1，正面布局从左到右依次是指示灯、屏幕、按键、模式切换开关和急停开关。



图 4-1 示教器正面效果图

如图 4-2，在示教器底部配备 USB 接口和触摸笔。



图 4-2 示教器底部效果图

如图 4-3，在示教器背面配备手动机器人使能开关（图中黄色装置）。

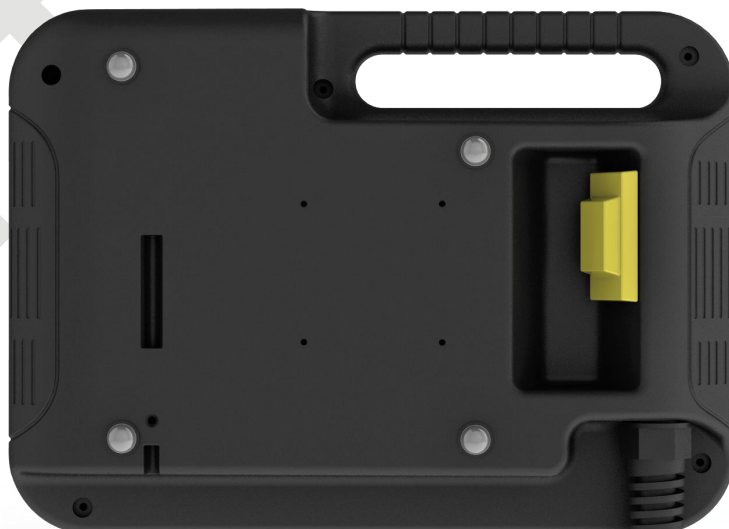


图 4-3 示教器背部效果图

4.2 示教器控制开关功能说明

示意图	开关名称	功能说明	
	模式切换	控制机器人工作在何种工作模式	TEACH 手动操作模式
			PLAY 自动模式
			REMOTE 远程模式
	急停	紧急停止机器人时使用	
	手动使能	手动操作模式生效，自复位开关需持续按下生效，分为 3 个档位： 静止档位：电机下电，1/2 档位：电机上电，完全按下档位：电机下电	
	负方向点动	手动关节运动模式下，分别控制 1-J1,2-J2,3-J3,4-J4 轴的负方向	
		手动笛卡尔运动模式下，分别控制 1-X,2-Y,3-Z,6-C 轴的负方向	
	正方向点动	手动关节运动模式下，分别控制 1-J1,2-J2,3-J3,4-J4 轴的正方向	
		手动笛卡尔运动模式下，分别控制 1-X,2-Y,3-Z,6-C 轴的正方向	
	自动使能	自动操作模式生效，触发类型开关 电机未上电时按下瞬间电机上电，电机上电后再次按下瞬间电机下电	
	启动/检查	手动模式，按使能后再持续按下，执行当前行程序	
		自动模式，连续或单步模式启动程序	
	暂停	手动、自动模式下暂停程序运行	
	报警清除	清除出现的报警 若报警持续存在，先解除故障，再尝试清除报警	
	加速	手动/自动/远程模式加速	
	减速	手动/自动/远程模式减速	

第五章 软件界面使用介绍

5.1 界面区域介绍

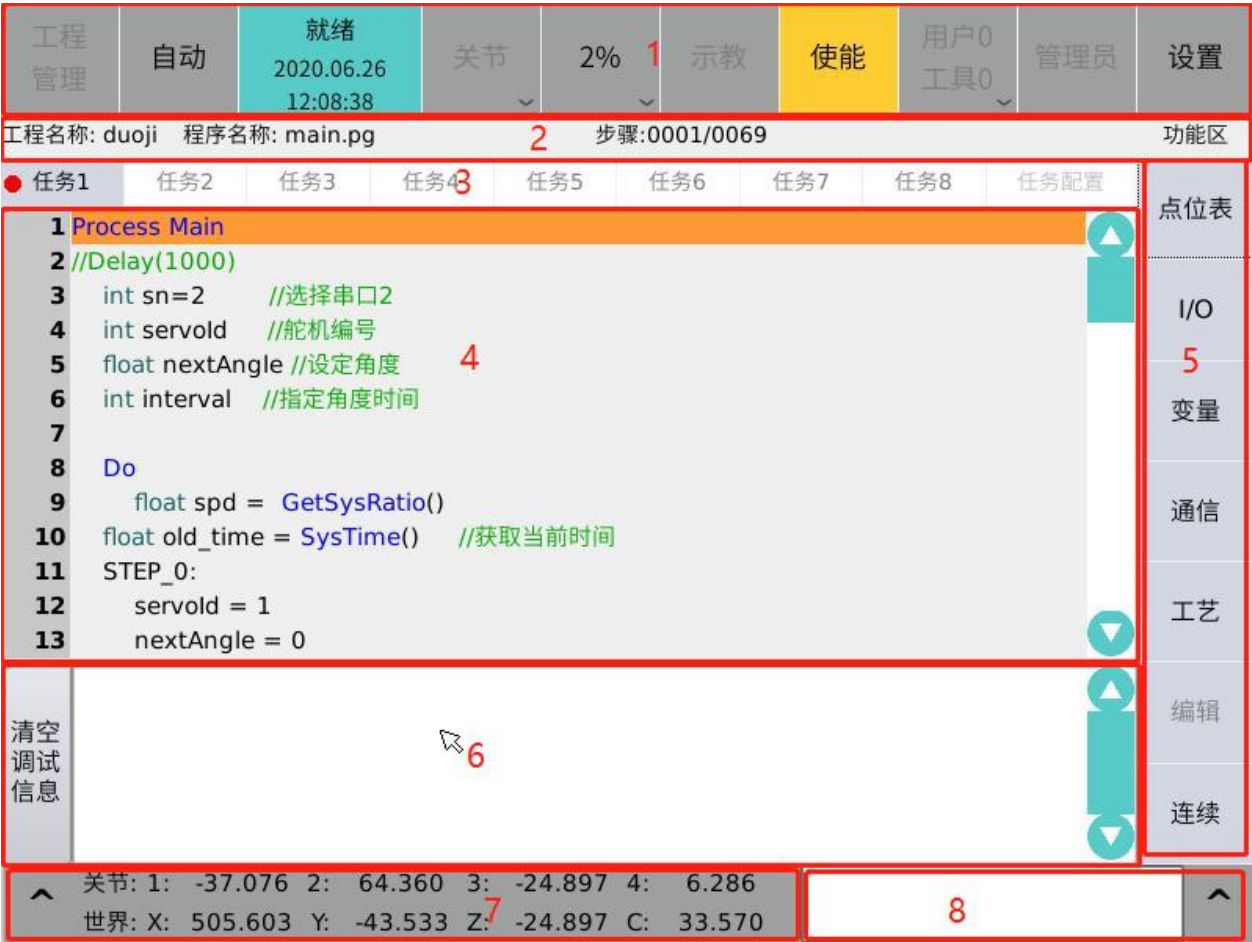


图 5-1 主界面

区域	类别	功能说明			
①	系统菜单	工程管理	按钮，工程管理入口（手动） 切换 新建 删除 改名 导入/出等	手动	状态灯，系统工作模式 手动/自动/远程
		就绪 2020.10.27 10:50:22	状态灯，系统工作状态 就绪 运行 暂停 急停 报警	关节	按钮，手动模式下切换点动模式 关节 世界 用户 工具
		20%	按钮，手动 0.01mm/°~100 自动 1~100	示教	按钮，记录位置到指定点 参看 5.2.5.2 点位修改方法
		未使能	状态灯，电机是否上电 未使能 电机未上电 使能 电机上电	用户0 工具0	按钮，手动用户/工具切换 用户号：0~15 工具号 0~9
		操作员	按钮，切换使用权限 操作员 技术员 管理员	设置	按钮，设置入口（手动） 坐标 参数 升级 备份 等
②	导航	指示当前界面内容或文件信息			

③	并行任务	任务1	点击查看此任务程序内容	任务配置	按钮，多任务配置入口 手动模式才可以点击 任务配置参看 5.2.3 工程任务配置
		停止/暂停 运行 已配置但未启用			
		任务2	白色框表示未配置任务		

④	程序显示	1 Process Main 2 int speed=1 3 point pick=	程序内容：灰色栏行号，橙色条当前执行，绿色字变量类型定义，蓝色字语法		
			手动模式下非编辑状态，只能选行号不能改程序，点右侧编辑按钮可以进入编辑状态		
		自动模式下不能选中行，行号以实际执行刷新			

⑤	工程菜单	点位表	点位管理入口	I/O	I/O 监控入口	
			自动模式下只可查看 参看 5.2.5 点位表		输入 仿真开：选择通断 仿真关：按实际接收 输出 手动：用户选择通断 自动：系统控制	
	变量	Modbus 即时通讯变量查看入口	通信	通讯配置入口 自动模式下只可查看		
	工艺	未开放	编辑	程序编辑入口 参见 5.3.1 程序编辑		
	连续	单步/连续切换				
	⑥	打印输出区域	显示程序 Print()函数打印输出的结果 超过 100 条后自动覆盖		清空调试信息	手动清空打印区域信息

	实时坐标监控	关节:	实时显示机器人关节值 J1-° , J2-° , J3-mm , J4-°	世界:	实时显示机器人世界/用户坐标值 X-mm , Y-mm , Z-mm , C-°
		点击	调出/关闭：轴位置、电机、编码器、I/O 等监控信息		

	人机沟通	通常	即时输出人机沟通信息
		点击	调出/关闭：异常报警记录



提示

图 5-1 中描述的是手动模式界面，自动模式下与手动界面整体一致，但关闭了相应按钮或限制修改仅供查看。

5.2 工程管理

5.2.1 工程文件管理



图 5-2-1 工程管理主页

按钮图标	说明
新建工程	新建工程，根据新建对话框设置工程名称
装载工程	装载工程，根据选中的工程装载并复位工程
复制	将选中的工程复制一份并提示输入复制目标的名称
重命名	更改选中工程名称，正在运行的工程不能更改
删除	删除选中工程，正在运行的工程不能删除
查看内容	打开工程内容界面，详见 5.2.2
本地<->U盘	打开工程导入导出界面，详见 5.2.4
刷新	刷新所有工程
关闭	退出工程管理界面，特别的点击工程管理也可以退出当前界面

5.2.2 工程内容界面



图 5-2-2 工程内容界面

按钮图标	说明
	在当前工程中新建前台或后台程序
	删除选中的内容。某些情况不可以删除，以实际提示为准
	将选中的程序或其他内容复制到【剪切板】
	将【剪切板】中的内容粘贴，这个操作可以跨工程
	将选中的文件重新命名。某些情况不可以重命名，以实际提示为准
	退出当前界面，返回到上一层【工程管理】



注意

在变更工程程序后，需手动点击“加载工程”重新加载一次工程。
此栏目中的一些操作需在任务管理界面下配合操作。

5.2.3 工程任务配置

5.2.3.1 工程任务配置主页（多任务配置器）

多任务配置器

任务号	程序选择	是否启用	启用方式	
1	main.pg	☒	☐ 开机启动 ☒ 普通启动	清空
2		☐	☐ 开机启动 ☐ 普通启动	清空
3		☐	☐ 开机启动 ☐ 普通启动	清空
4		☐	☐ 开机启动 ☐ 普通启动	清空
5		☐	☐ 开机启动 ☐ 普通启动	清空
6		☐	☐ 开机启动 ☐ 普通启动	清空
7		☐	☐ 开机启动 ☐ 普通启动	清空
8		☐	☐ 开机启动 ☐ 普通启动	清空

清除配置 返回

图 5-2-3-1 工程任务配置界面

按钮图标	说明
main.pg	开启程序选择对话框，详细参见 5.2.3.2
☒	是否启用左侧的程序
☐ 开机启动	勾选后，开机进入系统后会立即启动左侧程序；和普通启动方式二选一
☐ 普通启动	勾选后，左侧程序随主任务按键启动；和开机启动方式二选一
清空	清除左侧配置的程序，不会删除程序本身
清除配置	清除所有配置信息
返回	退出并返回到手动界面主页，退出前提示是否保存配置信息，确定后配置立即生效

5.2.3.2 程序选择对话框

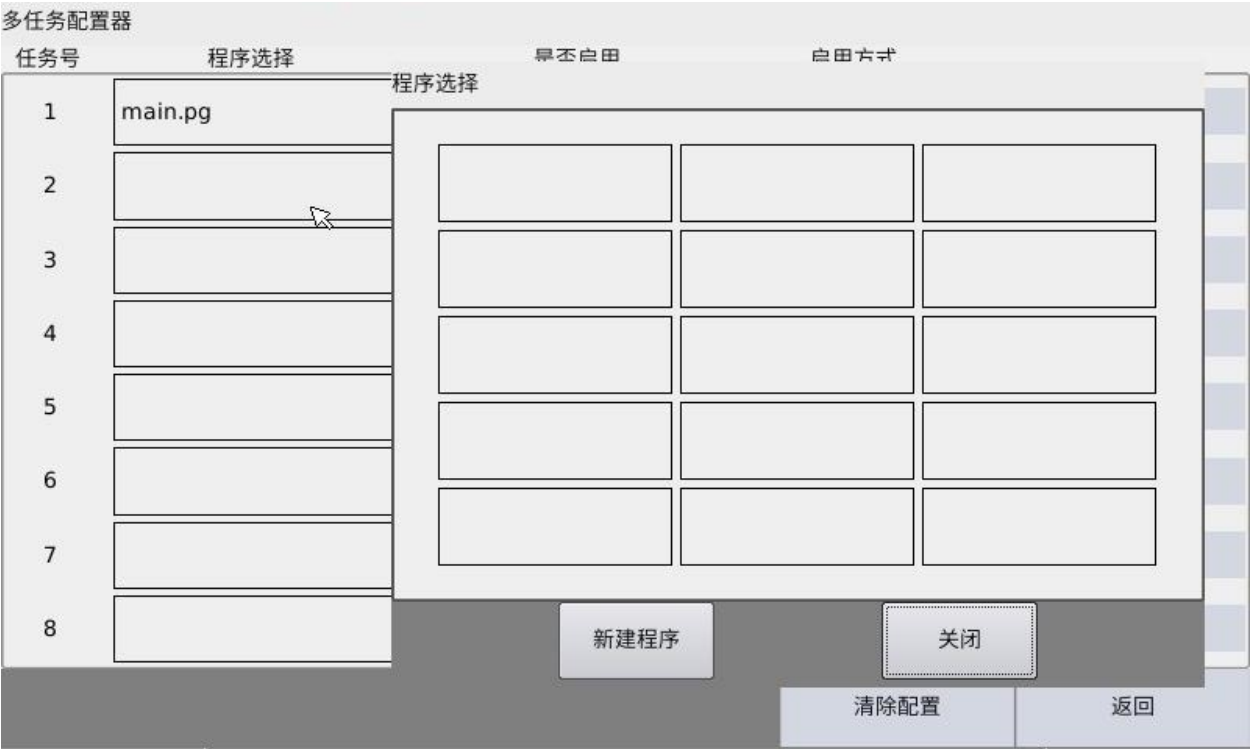


图 5-2-3-2 程序选择对话框

按钮图标	说明
	已有的主程序/后台程序会在列表中对应显示，按对应的名字即可完成选择
	如果事先没有其他后台程序，又想要配置任务时，点击此处
	退出当前界面，返回到上一级

	主程序后缀名是“.pg”,后台程序后缀名是“.pgb” 一个工程文件系统中可能包含很多个主程序和后台程序，但通过配置后最多只能有 1 个主程序和其它 7 个后台程序生效。详情参看 6.1.1 工程文件结构
提示	

5.2.4 程序导入/导出界面（U 盘）



图 5-2-4 工程导入/导出界面

按钮图标	说明
	将 U 盘中选中的工程导入到系统（本地）中
	将系统（本地）中选中的工程导出到 U 盘中
	手动检查 U 盘是否存在，在连接正常情况下，可能需要 1-2 秒
	退出当前界面到工程管理主页

	如外接 U 盘后未检测到，请点击“U 盘检查”按钮，如果还是没有发现 U 盘，请重启示教器或重启系统
提示	系统检测到插入 U 盘后，右下角会有相应提示
	在导入工程时，如果是多任务型的工程，需要重新配置多任务工作，参见 5.2.3
提示	
	请勿尝试将非工程类的文件夹装在 U 盘中作为工程文件导入，可能会引起未知错误，导致文件系统崩溃！
注意	

5.2.5 点位表

5.2.5.1 点位表主页

点位表		备注	(单击此处添加或者删除)		运动方式	MOVJ	限高	40.000	导入.csv	导出.csv
序号	X / J1	Y / J2	Z / J3	C / J4	坐标系	手系	类型	备注		
P00000	0.000	0.000	0.000	0.000	N/A	N/A	关节			
P00001	251.317	-64.645	3.602	-0.017	0	左	笛卡尔			
P00002	251.318	-64.645	3.602	-0.017	0	右	笛卡尔			
P00003										
P00004										
P00005										
P00006										
P00007										
P00008										
P00009										
到此点		保存		清除		上一页		1 / 1000	下一页	
									返回	

图 5-2-5-1 点位表主界面

按钮图标	说明
点位表	显示当前操作的点位编号
P00002	目标点编号，选中后可以【到此点】、【保存】、【清除】程序中引用，可以使用 Pn(2)。（程序中只有读操作，不能修改） 例如 disx=Pn(2).x 或 MOVJ(Pn(2),100,100,100,0)
备注 (单击此处添加或者删除)	展示或修改/删除选中点位的备注名称，选中后在下方展开键盘，允许输入最大 10 个的中文
导入.csv	将 U 盘中的 teachpoint.csv 文件导入到系统工程点位表中
导出.csv	将当前工程的点位表以 teachpoint.csv 名称保存在 U 盘中
MOVJ	MOVJ/MOVL/JUMP 三中运动模式可选
限高 40.000	选中 JUMP 运动方式后生效，每次重新进入点位表将会以 J3 轴上限位作为初始值，运动前可以手动改变，但数值不要低于当前位置

	或目标位置
到此点	以关节插补形式【MOVJ/MOVL/JUMP 可选】运动到达选中目标点，需要手动使能打开状态
保存	将示教或修改后的点位值保存到系统中
清除	清除当前设置的点
上一页	翻到上一页，每页共计 10 个点位
1 /1000	通过输入数值后，快速去到 10 的倍数页面
下一页	翻到下一页，每页共计 10 个点位
返回	退出当前界面到主页

5.2.5.2 点位修改方法

5.2.5.2.1 示教位置

工程管理	手动	就绪 2020.06.26 11:55:38	关节	0.01 mm	示教	未使能	用户0 工具0	管理员	设置
点位表		备注 (单击此处添加或者删除)	运动方式 MOVJ	限高 40.000	导入.csv	导出.csv			
序号	X / J1	Y / J2	Z / J3	C / J4	坐标系	手系	类型	备注	
P00000	0.000	0.000	0.000	0.000	N/A	N/A	关节		
P00001	251.317	-64.645	3.602	-0.017	0	左	笛卡尔		

图 5-2-5-2-1 点位表示教方法

示教方法：

- ① 选中序号，所在行橙色高亮
- ② 点系统菜单中的示教按钮，坐标值会根据当前移动方式、用户坐标系等状态保存数据

当前状态	说明
关节	保存时以各个轴关节数据保存， 类型 下自动显示为关节
世界	保存时以笛卡尔（XYZ...）数据保存， 类型 下自动显示为笛卡尔
工具	保存时以笛卡尔（XYZ...）数据保存， 类型 下自动显示为笛卡尔
用户	保存时以笛卡尔（XYZ...）数据保存， 类型 下自动显示为笛卡尔
用户0 工具0	在用笛卡尔保存数据的时候，当前用户选择的坐标系序号被同时记录在 坐标系 中
序号	范围 0-10000，在程序中用 Pn(数字)引用
手系	根据当前姿态自动生成，笛卡尔型坐标可以通过点击修改左或右

- ③ 选择保存，结束示教工作

 技巧	在示教位置的时候，电机使能没开可能会造成轻微移动，手动使能开启可以锁定位置。
---	--

5.2.5.2.2 修改位置

序号	X / J1	Y / J2	Z / J3	C / J4	坐标系	手系	类型	备注
P00000	0.000	0.000	0.000	0.000	N/A	N/A	关节	
P00001	251.317	7	8	9	-	左	笛卡尔	
P00002	251.318					右	笛卡尔	
P00003		4	5	6	Del			
P00004								
P00005		1	2	3	Esc			
P00006								
P00007		0		FNT				
P00008								

图 5-2-5-2-2 修改位置数据

P00001		324.782	228.543	109.960	180.000	0	1	否
P00002		324.780	-72.986	109.960	180.000	0	1	是
P00003		247.034	-342.511	150.000	0.000	0	1	否

图 5-2-5-2-3 修改坐标类型

	不是所有笛卡尔坐标值都能到达，修改设置了数值后请检查各个关节是否在软限位之内，请充分参考机器人运行范围图。 设置不恰当的数据后，点位将不可到达！
	程序中引用点位的方式是以 Pn(序号)方式引用。 每个工程都有一个点位表文件，且只能访问自己工程的点位表文件。
	P00000 点不可修改，可以从别处到达此点，它代表机器人机械零位。 机械零位表示机器人手臂完全伸直，运动时，请充分考虑周边是否有障碍物。

5.2.6 I/O 监控



图 5-2-6 I/O 监控界面

按钮图标	说明	
	仿真功能关闭（默认）	
	仿真功能开启按钮	仿真只针对输入信号
		在手动/自动模式均可开启
		开启后不再接收外部输入信号，使用 I/O 监控界面中的
	输入信号 按钮型指示灯	在仿真关闭情况下只做指示灯，显示外部输入信号通断状态； 在仿真开启下，点击按钮仿真输入开关，并且在模式切换为自动时，依然处于仿真状态；
	输出信号 按钮型指示灯	手动模式：点击按钮控制输出开关； 并且在模式切换到自动的时候，所点击的状态也不受改变； 如果配置了急停后复位功能，则急停信号触发时复位对应信号；
		自动模式：点击无效，只受系统控制
	选择本地输出组，0~15 或 16~31，扩展 I/O 使用后会显示在此处右侧	
	退出当前界面	

 技巧	仿真输入可以解决没有实际接输入信号，又要检查程序执行逻辑的问题
--------	---------------------------------

5.2.7 变量（界面）

变量简介

此界面所指的变量为系统定义的 Modbus 即时通讯用的系统级全局变量，分别为 int 32 位和 float 32 位类型各 100 个（掉电不保持类型），GInt32 位和 Gfloat32 位各 50 个（掉电保持类型）。

（关于变量分级的详细介绍，请参考 6.1.2 变量种类及作用范围）

全局变量-掉电不保持						
	+0	+1	+2	+3	+4	
I0	0	0	0	0	0	
I5	0	0	0	0	0	
I10	0	0	0	0	0	
I15	0	0	0	0	0	
I20	0	0	0	0	0	
I25	0	0	0	0	0	
I30	0	0	0	0	0	
I35	0	0	0	0	0	
I40	0	0	0	0	0	
I45	0	0	0	0	0	
Int类型	Float类型	GInt类型	GFloat类型		上一页	下一页
					返回	

图 5-2-7 变量监控界面

5.2.7.1 界面说明

上图中+0~4 表示左侧地址的增加个数，举个例子：

如果想查看 I39 的当前值是多少，首先找到 I35，再看+4，这样横竖交叉点的位置表示 I39 的值；手动写入的话，单击对应的交叉点，在弹出的小键盘上输入对应值即可完成立即发送；

5.2.7.2 modbus 自定义变量的读写操作

当用程序指令操作地址读写的时候，参考 6.2.7 通信指令中的 MIRead，MFRead，MIWrite，MFWrite 等指令进行按地址序号的读、写操作，单个的读写可以直接引用 I/F/GI/GF+序号。

5.2.7.2.1 写操作

例 1：

需要将 I39 赋值为 10

I39 = 10 也可以写成：MIWrite(39,10) //掉电不保持

GI0 = 1 也可以写成：GIWrite(0,1) //掉电保持

例 2：

将 I0~I99 复位成 0

```
int i
```

```
For i=0 To 99
```

```
    MIWrite(i, 0)
```

```
Next
```

5.2.7.2.2 读操作

例 1：

等待 I20 的值为 15

```
Function Wait(int addr,int val)//等待 I 变量等于某个值的函数
```

```
    Print("正在等待 I",addr,"=",val)//打印正在 wait 的信息
```

```
    Do While MIRead(addr) != val
```

```
        Delay(5)
```

```
    Loop
```

```
FunctionEnd
```

```
Wait( 20 , 15 ) //调用自定义函数 Wait()
```

例 2：

判断 I20 是否等于 15

```
If I20 == 15 Then //判断此时 I20 是否等于 15
```

```
    ...
```

```
Else
```

```
    ...
```

```
EndIf
```

例 3：

根据主站发送的值做 XYZC 偏移，主站将 XYZC 的偏移值分别发送至 F40~F43 中：

```
MOVJ(standby,speed,acc,dec,cp)
```

```
Break() //打断连续运动，预防提前读取到时机不对的值
```

```
MOVJ(labelingPos+X(F40,0)+Y(F41,0)+Z(F42,0)+C(F43,0),speed,acc,dec,cp)
```



提示

如果用户未使用 Modbus,也可以使用这些地址作为全局变量使用。

5.2.8 通讯



图 5-2-8-1 通讯监控界面

按钮图标	说明
①	串口监控选择，COM1/COM2
②	串口发送区，点击调出键盘，如图 5-2-8-2 通讯发送界面
③	串口收发数据监控
④	网口监控选择
⑤	网口发送区，点击调出键盘
⑥	网口收发数据监控
☐ HEX发送	勾选后以 16 进制格式发送
☐ HEX接收	勾选后以 16 进制格式接收
串口配置	对串口各通道配置
网络配置	对网口各通道配置
返回	退出当前界面



图 5-2-8-2 通讯发送界面

5.2.8.1 串口通讯配置

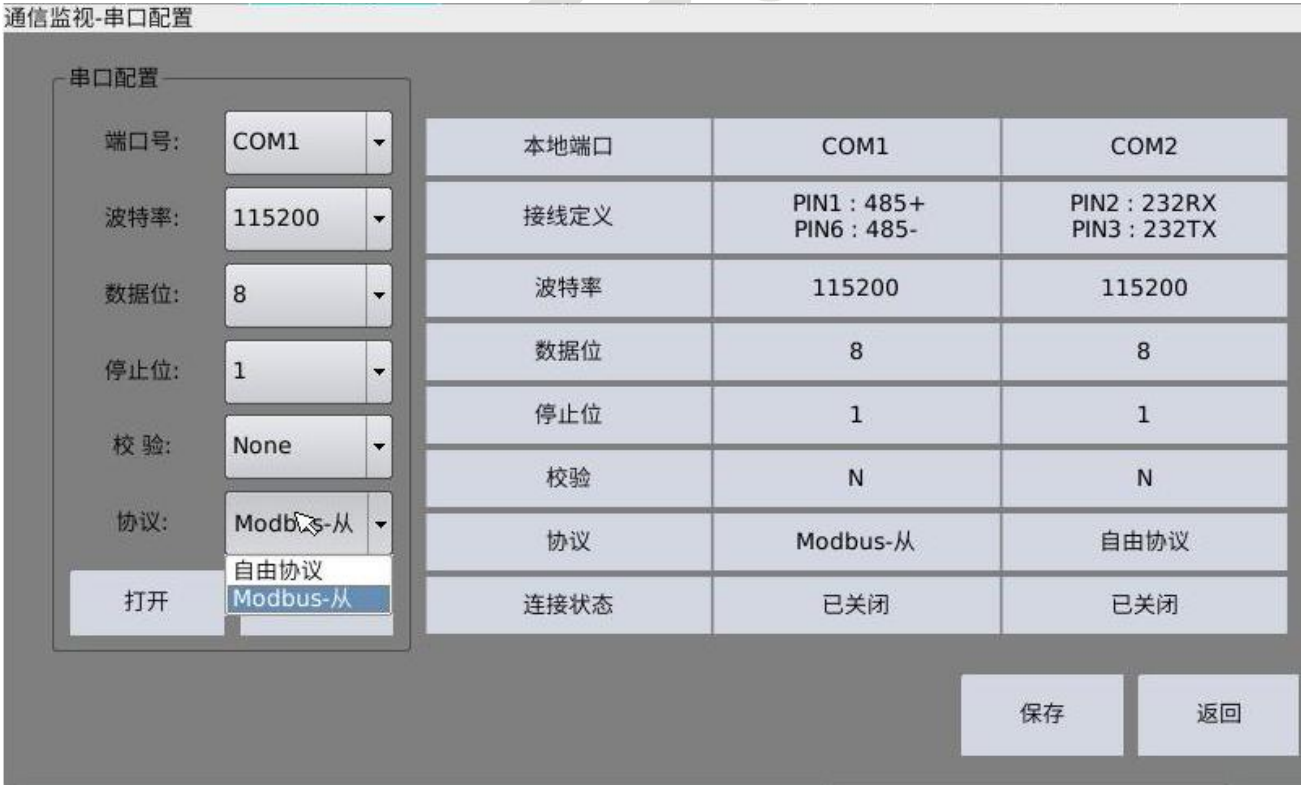


图 5-2-8-1-1 串口通讯配置界面

说明：

目前硬件支持 COM1、COM2 可用；

COM1,在配置为 modbus 协议时，COM1 是**从站**模式，地址默认为 1。

COM2 当前只有自定义通信的功能。

详细参看 **第七章 通讯功能介绍**。

5.2.8.2 网口通讯配置

工程管理	手动	就绪 2020.06.26 11:57:19	关节	0.01 mm	示教	未使能	用户0 工具0	管理员	设置
------	----	------------------------------	----	------------	----	-----	------------	-----	----

通信监控-网口配置

网络配置

IP地址: 192 . 168 . 1 . 80

子网掩码: 255 . 255 . 255 . 0

默认网关: 192 . 168 . 1 . 1

MAC地址: 42-12-39-FF-43-14

通道配置

通道选择: 1

协议模式: 自由协议

本机模式: 客户端

服务端IP地址: 0 . 0

端口: 0

连接

断开

添加

删除

通道号	对方IP地址	端口号	模式	连接状态	协议模式
-----	--------	-----	----	------	------

保存 返回

图 5-2-8-2 网口通讯配置界面

说明：

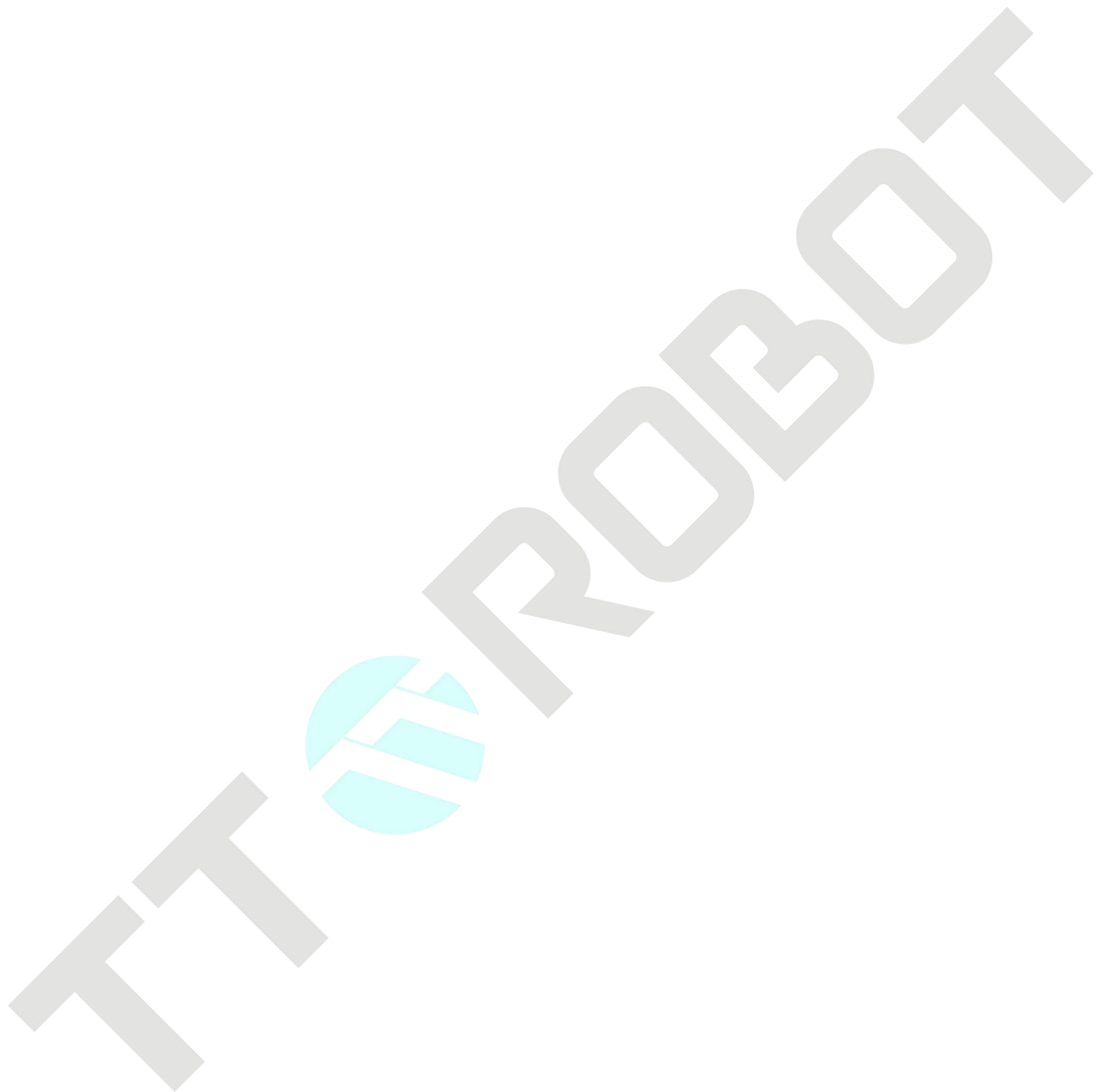
“网络配置”为控制器 EtherNet 网口本地 IP；

通道数量 1-4 有效；

详细参看 第七章 通讯功能介绍。

5.2.9 工艺

暂未开放



5.2.10 程序编辑

5.2.10.1 程序编辑主页面



图 5-2-10-1 程序编辑主界面

区域	类别	功能说明
①	导航	显示当前文件信息和系统时间
②	编辑工具	编辑文件用到的常用工具按钮，如复制、粘贴、删除等
③	编辑窗口	程序：灰色栏行号，橙色条当前执行，浅蓝色字变量类型定义，蓝色字语法
④	全局变量	展开全局变量编辑窗口，详细参看 5.2.10.3 全局变量
⑤	键盘	直接输入字符或调用指令，详细参看 5.2.10.2 指令键盘

 技巧	通过关闭屏幕键盘，可以浏览到更多行程序。
	屏幕键盘上的 Ctrl+X、C、V 也能实现剪切、复制、粘贴。
	选中多行按 Tab 键可以整体缩进，先按住 Shift 再按 Tab 键可以回退缩进

5.2.10.2 指令键盘



图 5-2-10-2-1 开启指令键盘界面

区域	类别	功能说明
①	指令分类	按照指令用途分类
②	具体指令	显示①分类中全部的指令，通过上下翻浏览
③	指令介绍	讲述指令的功能
④	格式选择	同一指令不同参数实现不同的功能
⑤	指令预览	待插入的指令内容预览
⑥	参数修改	通过  按钮，打开修改参数对话框，详细参看 5-2-10-2-2 指令参数修改。 修改完毕后按  完成在指定光标处插入指令
⑦	插入按钮	插入预览窗口中的内容到程序中

5.2.10.2.1 指令参数修改



图 5-2-10-2-2 指令键盘参数修改

区域	类别	功能说明
①	导航	显示当前位置
②	指令介绍	讲述指令的功能
③	参数修改对话框	逐个参数修改
④	键盘	修改参数用到的键盘
⑤	切换参数	切换参数形式，目前仅在运动 指令中的 point_a 切换成 Pn(0)
⑥	修改完成	修改完成，退出当前界面，将变更参数的指令刷新到预览中，如图 5-2-10-2-3



图 5-2-10-2-3 点击修改完成后预览

5.2.10.3 全局变量

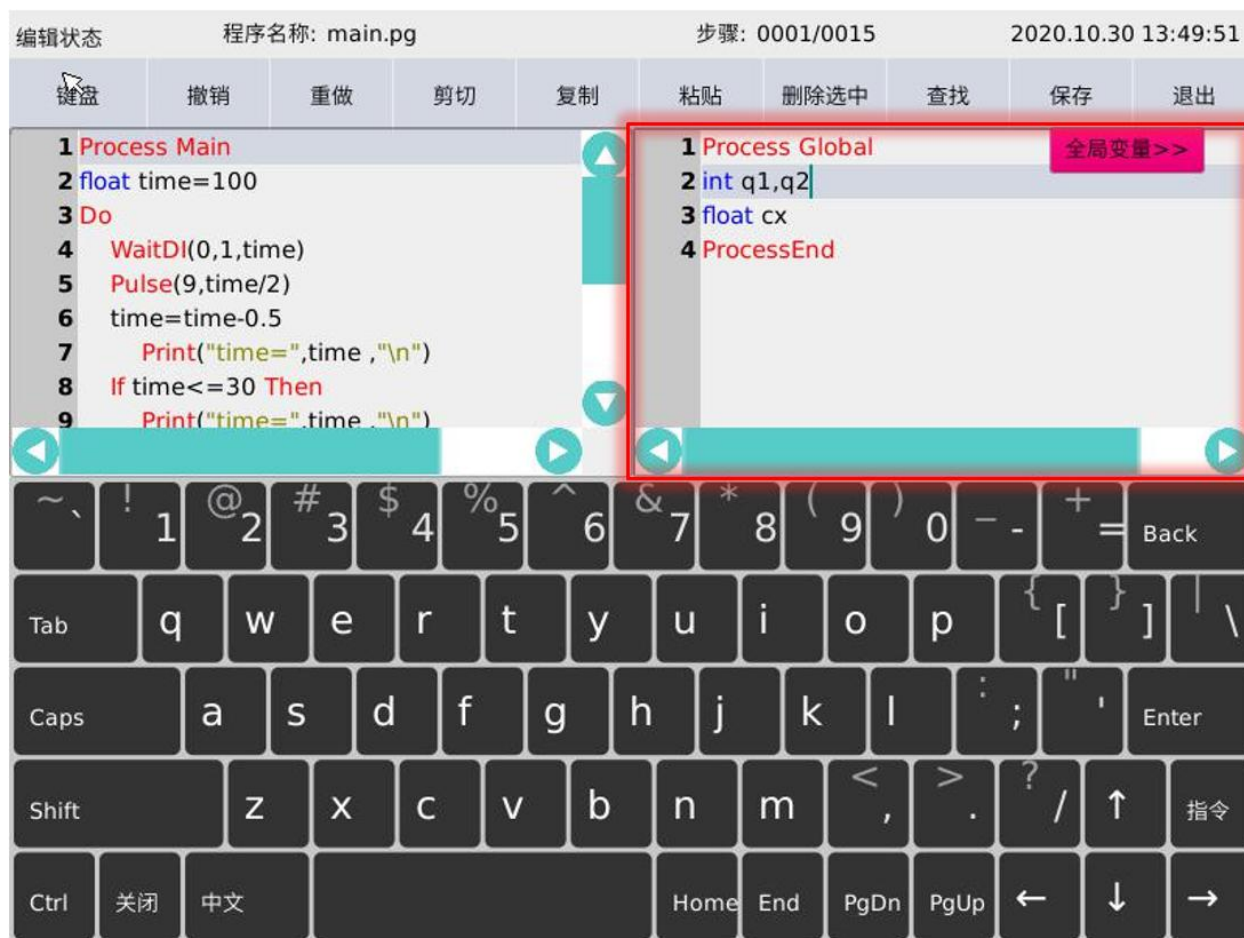


图 5-2-10-3 全局变量

按钮图标	说明
	展开全局变量窗口
	关闭全局变量窗口

	提示 全局变量（文件后缀名“.gl”）总是以 Process Global 关键字开始，ProcessEnd 结束。全局变量声明在这两个之间，变量作用在整个工程范围内
	注意 该窗口在任何一个程序编辑状态中都可以展开，声明变量时注意不要重名。编辑程序的时候注意不要更改开头和结尾的关键字。

5.3 设置

点击[设置]进入设置主页，如图 5-3。



图 5-3 设置主画面

按钮图标	说明
坐标系	用户坐标系、工具坐标系设置
监控区	TCP 末端区域监控：长方体或柱体的障碍区、监控区设置
参数	SCARA 机构参数（DH 参数）、伺服参数、运控参数、高级参数
远程设置	远程指定启动的工程、启停输入输出配置
IO配置	扩展 IO 数量设置，IO 名称配置，导入导出配置
通信配置	串口通信与网络通信配置

系统 设置	系统时间，按键音蜂鸣器，开机画面设定
版本	软件版本的查看（含控制系统与示教器），示教器 HMI、编译前端软件升级
续期	试用时间结束后，解除锁定入口
脱机 模式	示教器脱机按钮

5.3.1 坐标系

5.3.1.1 用户坐标系

用户坐标系定义在工件上，因此也可称作工件坐标系。在机器人动作允许范围内的任意位置，设定任意角度的 X、Y、Z 轴，原点位于机器人抓取的工件上，坐标系的方向根据客户需要任意定义。

1. 用户坐标系查看修改界面

设置-坐标系

用户坐标系

工具坐标系

序号	X	Y	Z	A	B	C
#0	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
#1	452.5637	351.8877	36.4720	-180.0000	0.0000	-49.4372
#2	500.0000	200.0000	0.0000	164.8283	-12.5288	90.0000
#3	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
#4	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
#5	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
#6	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

设置

清除

保存

返回

图 5-3-1-1 用户坐标系

按钮图标	说明
设置	进入设置界面，参看图 5-3-1-2 用户坐标系示教
清除	清除选中序号的内容
保存	保存修改变更
返回	退出当前界面，返回到上一级菜单



注意

序号#0 等同于世界坐标系，不能被修改。

在此界面中可以直接修改坐标系计算值，除非操作者明确知道它的意义，否则使用系统提供的设置方式去修改。

2. 用户坐标系示教设置

设置-坐标系-用户坐标系-用户坐标系示教页面

STEP 1 示教原点

STEP 2 示教X+方向上一点

STEP 3 示教XY平面内一点

ORG X+ X+ X+

X Y Z A B C

示教1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
示教2	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
示教3	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

STEP 4 计算获取结果

计算 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000

返回

图 5-3-1-2 用户坐标系示教

按钮图标	说明
示教1	示教用户坐标系原点
示教2	示教用户坐标系 X+方向上的一点
示教3	示教用户坐标系 XY 平面上 Y+方向任意点
计算	用三点法计算标定结果
返回	退出当前界面，返回到上一级菜单
移动	在示教后出现在图例旁边 通过关节插补方式移动机器人到该位置上，使能接通才可以移动



提示

点击移动按钮之前，需打开手动使能按键。在移动过程中，松开手动使能键或拍下急停按钮可以打断机器人移动。

5.3.1.2 工具坐标系

安装在机器人末端的工具坐标系，原点及方向都是随着末端位置与角度不断变化的，该坐标系实际是将基础坐标系通过旋转及位移变化而来的。工具中心点（TCP）的位置和工具姿态的坐标系。工具坐标系必须事先进行设定。用户可以根据工具的外形、尺寸等建立与工具相对应的工具坐标系。

1. 工具坐标系查看修改界面

设置-坐标系

用户坐标系

工具坐标系

序号	X	Y	Z	A	B	C
#0	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
#1	2.0000	10.5000	0.0000	0.0000	0.0000	0.0000
#2	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
#3	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
#4	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
#5	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
#6	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

设置

清除

保存

返回

图 5-3-1-3 工具坐标系

按钮图标	说明
设置	进入设置界面，参看图 5-3-1-4 工具坐标系示教
清除	清除选中序号的内容
保存	保存修改变更
返回	退出当前界面，返回到上一级菜单



注意

序号#0 为默认不使用任何工具，不能被修改。

在此界面中可以直接修改坐标系计算值，除非操作者明确知道它的意义，否则使用系统提供的方式去设置。

2. 工具坐标系示教

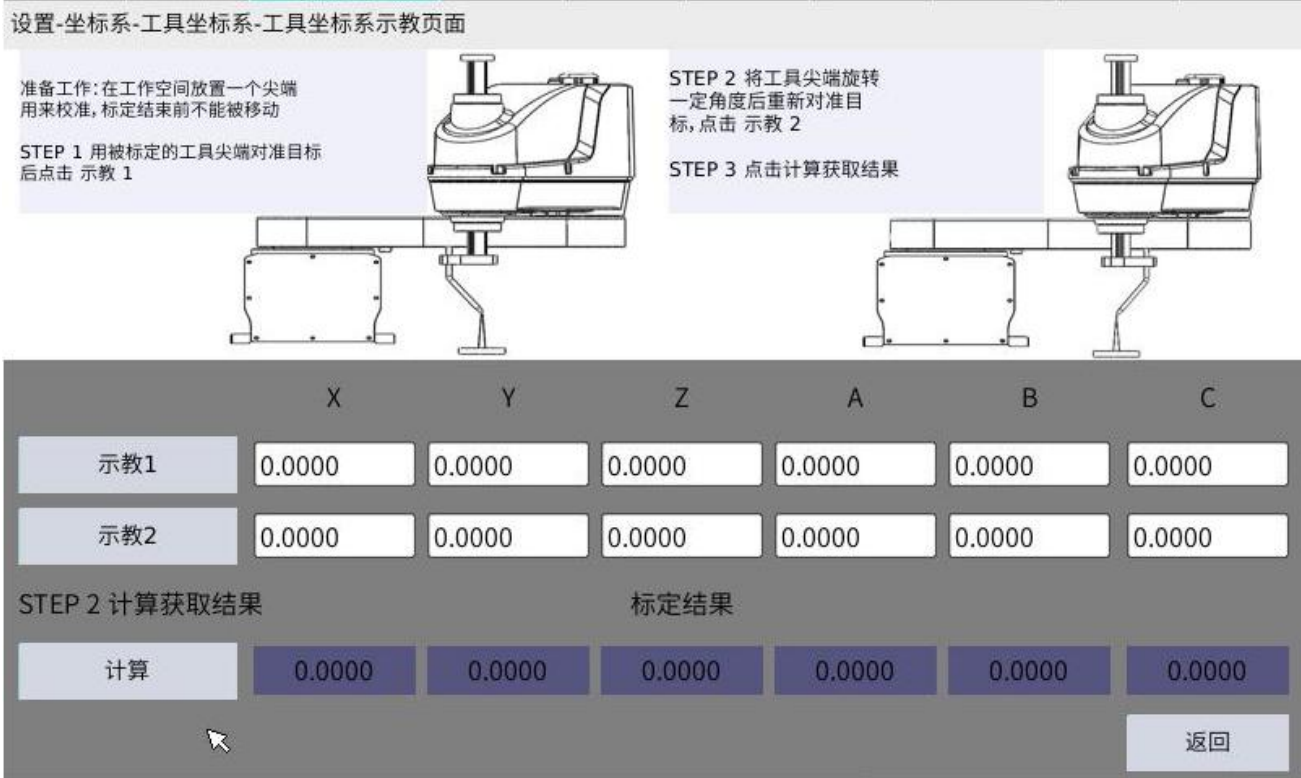


图 5-3-1-4 工具坐标系示教

按钮图标	说明
	示教第一个位置
	让工具尖端示教到第二个位置
	计算标定结果
	退出当前界面，返回到上一级菜单
	移动机器人到该位置上，需要使能按键接通才可以移动

	点击移动按钮之前，需打开手动使能按键。在移动过程中，松开手动使能键或拍下急停按钮可以打断机器人移动。
--	---

5.3.2 监控区

简介	说明
功能介绍	1、实时判断机器人是否在监控区域之内，如果在监控区域之内，按照设置的值输出或者关闭相应的信号与变量； 2、设置障碍区域，使机器人无法进入，可以单独调整减速边界宽度； 3、本机提供最高 8 个监控区域； 4、可以通过单独设置某个区域监控生效或取消生效，避免进入障碍区域而无法移动； 5、监控区域的空间形状可以是长方体或柱体； 6、只做内部使用时，可以使用内部虚拟信号，和外部设备交互时使用 I/O。如需要通信可以将内部虚拟信号通过通信的方式发送出去，但是不建议这样使用，因为通过通信降低了实时性，有可能导致撞机等风险，仅在实时性要求不高的场合下才这样使用；
应用场景	1、机器人与其他设备有公共的操作空间，即机器人与外部设备动作空间有干涉； 2、机器人活动空间与外部物体（固定不动的）有干涉； 3、机器人开机时判断当前位置在哪一个监控区间内，规划不同的返回待机点路径。

5.3.2.1 监控区域设置

1.监控区域查看设置状态



图 5-3-2-1 监控区设置主页

按钮图标	说明
设置	打开监控区设置，如图 5-3-2-2 监控区详细设置
清除	清除对应序号的配置内容
生效	生效：控制器开始监控 TCP 末端是否抵达此目标区域 取消生效：控制器停止监控 TCP 末端是否抵达此目标区域
保存	保存此序号的区域配置信息
返回	退出当前界面，返回到上一次菜单

2. 监控区设置

设置-空间-空间设置

监控类型设置

☒ 长方形

☐ 柱形

监控类型设置

☒ 监控区

☐ 障碍区

监控区域尺寸

长

宽

高

200.00

0.00

0.00

TCP末端点与区域位置关系设定

☒ 中心

☐ 左下角

参考坐标系设置

坐标系序号:

0

TCP末端点坐标值

X

Y

Z

0.00

0.00

0.00

本地I/O:

-1

关

内部变量:

-001

关

内部变量说明:

1000~2000为内部专用IO

保存

返回

图 5-3-2-2 监控区详细设置

按钮图标	说明	
	关：TCP 末端点在监控/障碍区域内输出为假； 开：TCP 末端点在监控/障碍区域内输出为真。	
	打开键盘，输入或修改参数	
	长	参考系 X 轴方向的长度，单位：mm
	宽	参考系 Y 轴方向的长度，单位：mm
	高	参考系 Z 轴方向的长度，单位：mm
	坐标系序号	参考坐标系序号，默认为世界坐标系
	TCP 末端点坐标值	表示在参考坐标系序号下的空间位置，可以通过主界面的【示教】按钮获取当前位置值
	本地 I/O	有效的本地输出端口（包含扩展 IO 的输出）
	内部变量	1000~2000, 可以用 ReadDI 指令查询
	保存设置值	
	退出当前界面，返回到上一次菜单	

43

5.3.3 参数

5.3.3.1 DH 参数

设置-参数设置

DH参数

关节参数

软限位

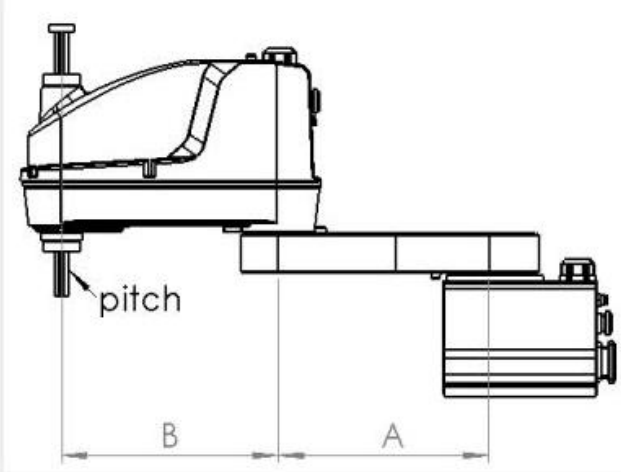
运动参数

高级参数

伺服参数

伺服调试

3、4轴为复合轴



1轴臂长(A)

300.001

mm

2轴臂长(B)

299.580

mm

螺距(pitch)

20.000

mm

零点数据

有效

1轴

-1107412.000

记录

☒

2轴

-231206.000

记录

☒

3轴

121788.000

记录

☒

4轴

1607656.000

记录

☒

保存

返回

图 5-3-3-1 DH 参数设置

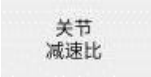
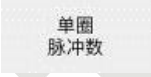
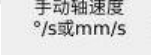
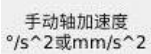
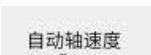
按钮图标	说明
3、4轴为复合轴 ☒	若 3、4 轴是分开情况，取消此处勾选
1轴臂长(A) 300.000 mm 2轴臂长(B) 300.000 mm 螺距(pitch) 20.000 mm	填写对应的参数
零点数据 有效 1轴 10495.000 记录 ☒ 2轴 8372.000 记录 ☒ 3轴 -216733.000 记录 ☒ 4轴 -110886.000 记录 ☒	点记录后，读取编码器的位置值作为当前机械零位值  危险 一定要非常明确的知道此参数的含义，更改不当后可能引起撞机危险！

5.3.3.2 关节参数

设置-参数设置

	DH参数	关节参数	软限位	运动参数	高级参数	伺服参数	伺服调试	
	关节 减速比	单圈 脉冲数	电机 反向	手动轴速度 °/s或mm/s	手动轴加速度 °/s ² 或mm/s ²	自动轴速度 °/s或mm/s	自动轴加速度 °/s ² 或mm/s ²	手动刹车 释放
1轴	80.985	20000.000	1	37.500	375.000	375.000	3750.000	松开
2轴	50.994	20000.000	1	60.000	600.000	600.000	6000.000	松开
3轴	2.000	20000.000	1	83.300	833.000	833.000	8333.000	松开
4轴	40.000	20000.000	-1	75.000	750.000	750.000	7500.000	松开
							保存	返回

图 5-3-3-2 关节参数设置

按钮或图标	说明
	按下松开对应轴的电机抱闸
	此处为出厂匹配好的参数
	
	
	一定要非常明确的知道此参数的含义，更改不当后可能引起撞机危险！ 若要修改，应在技术人员支持下进行！
	影响机器人在轴关节插补运动速度及加速度
	
	
	
	更改不当后可能引起减速机过早损坏或电机烧毁风险！若要修改，应在技术人员支持下进行！

5.3.3.3 软限位设置

设置-参数设置

DH参数

关节参数

软限位

运动参数

高级参数

伺服参数

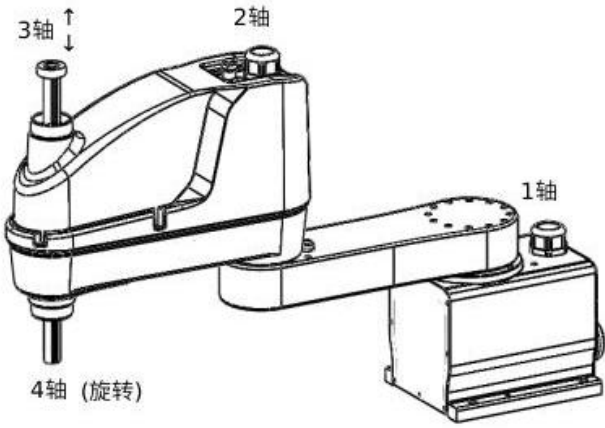
伺服调试

3轴

2轴

1轴

4轴 (旋转)



下限值

上限值

1轴

-120.000 °

120.000 °

2轴

-145.000 °

145.000 °

3轴

-170.000 mm

40.000 mm

4轴

-360.000 °

360.000 °

保存

返回

图 5-3-3-3 软限位参数设置

按钮或图标	说明
下限值	影响机器人最大运行范围 ! 危险 一定要非常明确的知道此参数的含义，更改不当后可能引起撞机危险！ 若要修改，应在技术人员支持下进行！

技巧	关闭伺服使能后打开坐标监控，推动机器人（如 3 轴有刹车需要打开），通过监视窗口观察各个关节最大的运行范围。并在此基础上反方向缩短 3~5，可以有效避免撞机的风险！
--------------------------	--

5.3.3.4 运动参数

DH参数	关节参数	软限位	运动参数	高级参数	伺服参数	伺服调试
手动最大平动速度(mm/s)		400.000		自动最大平动速度(mm/s)		833.000
手动最大平动加速度(mm/s^2)		4000.000		自动最大平动加速度(mm/s^2)		8333.000
手动最大姿态旋转速度(°/s)		180.000		自动最大姿态旋转速度(°/s)		360.000
手动最大姿态角加速度(°/s^2)		1800.000		自动最大姿态角加速度(°/s^2)		3600.000

图 5-3-3-4 运动参数设置

按钮或图标	说明
最大平动速度(mm/s)	影响机器人直线插补速度
最大姿态旋转速度(°/s)	影响机器人姿态插补速度

5.3.3.5 高级参数

设置-参数设置

DH参数	关节参数	软限位	运动参数	高级参数	伺服参数	伺服调试
参数号	参数名	参数值	备注			
0	机器人类型(轴数)		:SCARA;6:6轴; 7:7轴;			
1	关节					
2	关节					
3	关节					
4	关节					
5	关节5方向	1				
6	关节6方向	1				

保存 返回

图 5-3-3-5 高级参数设置

本表包含全部前面提及的参数，修改需谨慎。

厂家密码需要咨询厂家售后人员。



警告

一定要非常明确的知道此参数的含义，更改不当后可能引起减速机过早损坏或电机烧毁风险！

若要修改，应在技术人员支持下进行！

5.3.3.6 伺服参数

设置-参数设置

DH参数	关节参数	软限位	运动参数	高级参数	伺服参数	伺服调试
参数编号	参数名		参数值	参数属性		
0	软件版本		210500002	只读		
1	电机型号		11	只读		
2	指令分倍频分子		8388608	读写		
3	指令分倍频分母		20000	读写		
4	编码器型号		1	读写		
5	磁极零点值		127000	读写		
6	复位绝对值编码器圈数报警		0	读写		
1 轴		2 轴	3 轴	4 轴	保存	返回

图 5-3-3-6 伺服参数设置



警告

一定要非常明确的知道此参数的含义，更改不当后可能引起减速机过早损坏或电机烧毁风险！

若要修改，应在技术人员支持下进行！

5.3.3.7 伺服调试

设置-参数设置

DH参数

关节参数

软限位

运动参数

高级参数

伺服参数

伺服调试

当前电机型号：天太 750W 无刹

初始化选择项：
厂家：

天太

瓦数：

750W 无刹

天太
三协
禾川

初始化

当前编码器零点值：

127000

编码器找零点

停止找零点

1 轴

2 轴

3 轴

4 轴

保存

返回

图 5-3-3-6 伺服参数设置

本页面用于初始化配置伺服参数时使用，非必要情况不要随意设置。



警告

一定要非常明确的知道此参数的含义，更改不当后可能引起减速机过早损坏或电机烧毁风险！

若要修改，应在技术人员支持下进行！

5.3.4 远程配置

5.3.4.1 远程功能简介

远程功能是指：在示教器钥匙转动到远程模式后，不通过示教盒上的相关按键启动机器人，而是通过专用远程 IO 或 M006F-dbus 通信方式控制机器人的运行、暂停、复位、急停等动作，因此也叫外部控制。当输入条件满足要求后会自动打开预设好的指定程序文件开始执行，而不论当前是否打开此预设文件。

5.3.4.2 远程动作时序图

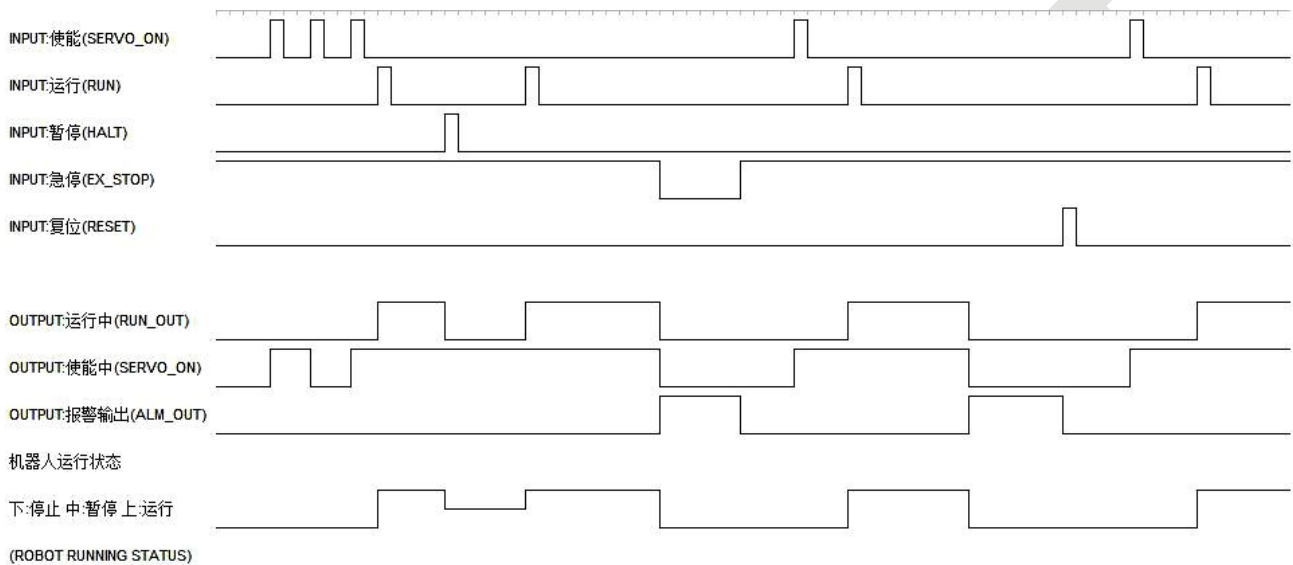


图 5-3-4-2 远程输入输出状态时序图

说明：

- 1 输入信号除急停外均为上升沿有效。持续输入期间只在最开始生效 1 次，后面无效。
- 2 输出信号为持续输出，ON/OFF 状态受输入和输出状态综合控制。
- 3 机器人停止状态，每输入使能信号 1 次，使能状态会取反 1 次。即：未使能和使能状态之间切换。机器人在运行中和暂停中，输入使能信号判定为无效，即：此时输入使能信号不会断使能。
- 4 在未使能状态下输入运行信号机器人不会动作，程序不能运行。但勾选了自动使能选项除外。
- 5 输入暂停/急停信号机器人从运动中到完全停止，耗时约 50~500ms(运动速度不同略有差异)。在此时间内禁止输入启动信号。
- 6 在程序运行期间输入暂停信号后，机器人停止动作和程序并保持处于暂停中，不会断使能。程序指针停止在当前行（此时未运行完成，运动指令表现为中断运动，非运动指令表现为未被执行）。此后再次输入启动信号后，程序继续从当前程序指针所在处开始执行。
- 7 急停生效后机器人会立即断开使能，程序停止工作，光标停止在当前行，同时会输出报警信号。在急停整个生效期间，输入使能信号，机器人不会上使能。急停复位后，报警自动消除。

- 8 急停恢复后，无论是否输入复位信号，程序指针都会复位到第 1 行。
- 9 在机器人程序正常运行期间，如果产生了非急停引起的报警，此时会断开使能，机器人停止运行，输入复位信号解除报警，同时程序指针会复位到第 1 行。

5.3.4.3 远程配置界面

设置-远程设置

远程自动使能: ☐

远程工程选择:

111.pro

清除

远程输入	输入端口(IN)	检测方式	远程输出	输出端口(OUT)
启动	-1	上升沿	运行中	-1
暂停	-1	上升沿	暂停中	-1
使能	-1	上升沿	使能中	-1
复位	-1	上升沿	报警中	-1
急停1	-1	常闭	就绪中	-1
急停2	-1	常闭		

保存

退出

图 5-3-4-3 远程配置界面

功能说明	说明
远程自动使能	勾选并保存后在外部启动信号给出后，自动使能，无需单独给使能信号
工程选择	点框内弹出工程选择画面，设置为远程模式调用的工程； 当未选择时，按当前自动模式下加载的工程启动；
清除	清除已经配置的远程工程
输入/出端口	选择可用的输入端口配置为相应的远程输入功能
上升沿	表示检查信号刚接通时瞬间有效
常闭	表示检查信号闭合有效，断开立即触发急停

备注：急停建议设置为常闭，防止输入意外断开情形下无法急停。

5.3.5 IO 配置

设置-IO配置预览

扩展I/O数量设定

-

0

+

导入

导出

输入	备注(字符串长度<10)	输出	备注(字符串长度<10)	急停后复位
IN0		OUT0		☐
IN1		OUT1		☐
IN2		OUT2		☐
IN3		OUT3		☐
IN4		OUT4		☐
IN5		OUT5		☐
IN6		OUT6		☐
IN7		OUT7		☐

上一页

下一页

清除

保存

返回

图 5-3-4-3 IO 配置界面

功能说明	说明
扩展 I/O 数量设定	设置扩展 IO 的数量，范围 0~4，0 表示不使用扩展，设置完重启后生效
备注	备注 IO 名称，关联手动 IO 界面下的名称。与实际程序变量名没有联系
急停后复位	勾选后制定的输出端口在开始急停的时候停止输出，非持续动作
导出	从控制器中读取 IO 配置文件导出到 U 盘
导入	从 U 盘中读取 IO 配置文件导入到控制器中
清除	清除制定端口的备注名称
保存	保存整个配置文件到控制器中

5.3.6 通信配置

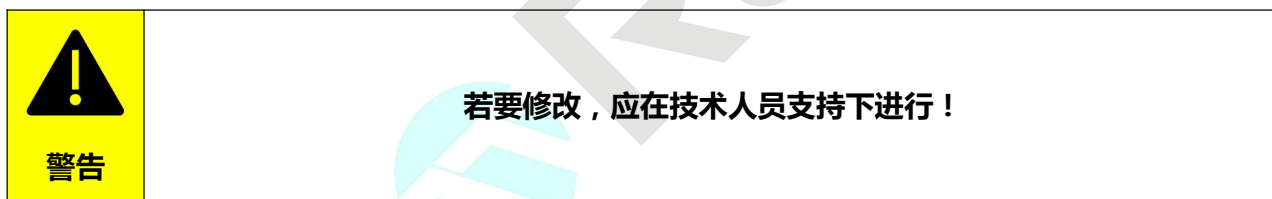
详情查看第七章《通信功能介绍》

5.3.7 版本

5.3.7.1 备份升级



图 5-3-7-1 升级备份界面



5.3.7.2 版本



图 5-3-7-2 版本查看界面

展示当前系统&示教器的软件版本。

5.3.8 系统设置

5.3.8.1 时钟/校准



图 5-3-8-1 系统时间设置界面

设置系统时间日期，并可以选择开启和关闭蜂鸣器。

5.3.8.2 语言

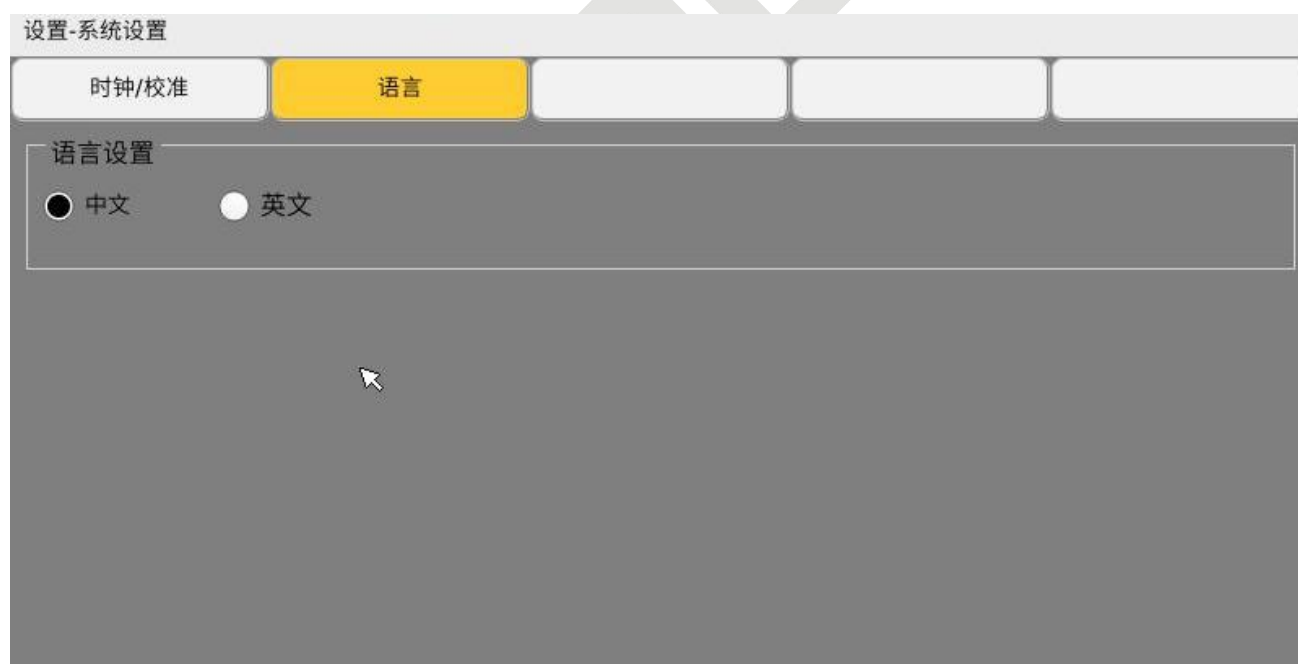


图 5-3-8-2 语言设置界面



提示

切换界面中英文显示，设置完之后需要重启系统。

5.3.9 续期

当用户弹出续期警报后，联系系统厂商获取续期密码后可恢复使用。

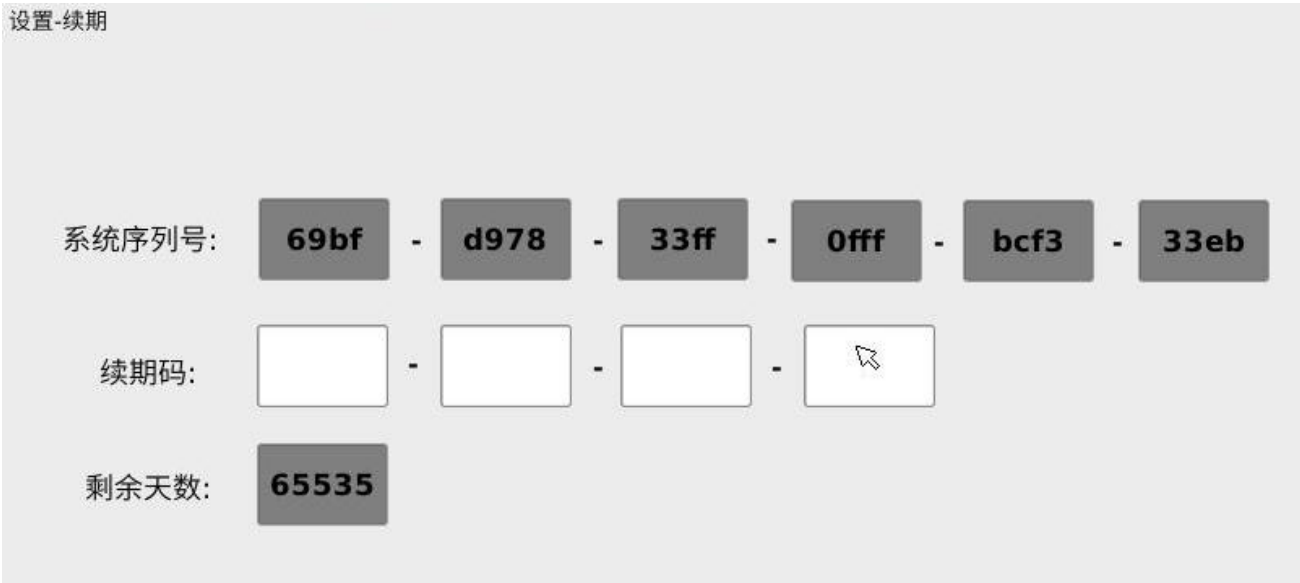


图 5-3-9 续期设置界面

5.3.10 脱机



图 5-3-10 脱机模式画面

控制器处于脱机状态时，可以将示教器拔下或者插上去。这里的脱机表示的是示教器脱开控制器，从而实现热拔插。

 注意	在脱机模式示教器拔出后，再次插入示教器时，本系统预设插头接触时间为 5 秒钟，在此期间接通/断开/再次接通不会有异常，在超过时间之后一旦检测到断开，即判定为急停。
---	--

5.4 实时信息监控

5.4.1 坐标系统监控

坐标系统监控				电机监控				IO监控				编码器			
世界坐标系				用户坐标系				关节坐标系							
X	505.603	A	0.000	X	505.603	A	0.000	J1	-37.076	J4	6.286				
Y	-43.533	B	0.000	Y	-43.533	B	0.000	J2	64.360	J5	0.000				
Z	-24.897	C	33.570	Z	-24.897	C	33.570	J3	-24.897	J6	0.000				

5.4.2 电机监控

坐标系统监控				电机监控				IO监控				编码器			
				负载%				电流A				转速r/m			
J1	0		0	0.0		0.0	0	0.0		0.0	0	0		0	
J2	0		-1	0.0		0.0	0	0.0		0.0	0	0		0	
J3	0		-1	0.0		0.0	0	0.0		0.0	0	0		0	
J4	2		4	0.0		0.0	0	0.0		0.0	0	0		0	

5.4.3 IO 监视

坐标系统监控				电机监控				IO监控				编码器			
IN	0	☐	4	☐	8	☐	12	☐	16	☐	20	☐	OUT	0	☐
	1	☐	5	☐	9	☐	13	☐	17	☐	21	☐		1	☐
	2	☐	6	☐	10	☐	14	☐	18	☐	22	☐		2	☐
	3	☐	7	☐	11	☐	15	☐	19	☐	23	☐		3	☐

5.4.4 机器人电机轴编码器

坐标系统监控	电机监控	IO监控	编码器	
编码器位置				
J1 -01274222 J4 001593687				
J2 -00048875 J5 000000000				
J3 000071296 J6 000000000				

第六章 指令系统介绍

6.1 编程规范

6.1.1 工程文件结构

类别	后缀名	最小数量	最大数量	格式	说明
主任务	pg	1	1	Process Main ... ProcessEnd	在工程中允许有多个.pg 文件存在，但只能配置其中 1 个文件作为主任务运行。1 个工程中必须要有一个主任务。
后台程序	pgb	0	7	Process Sub ... ProcessEnd	在工程中允许有多个存在，但最多只能选择 7 个后台程序文件运行。
全局变量	gl	1	1	Process Global ... ProcessEnd	在工程中储存在 1 个文件中，从任何一个配置的工程（主任务或后台任务）都可以编辑，全局变量文件中的内容只针对本工程有效
点位表	pts	1	1		最多存储 10000 点，每个工程配备一个点位表

6.1.2 变量种类及作用范围

级别	数据类型	作用范围	其他说明
系统级	整型	全部工程中生效	I0~I99（非掉电保持型）/GI0~GI49（掉电保持型）（modbus 即时通讯）
	浮点型	全部工程中生效	F0~F99（非掉电保持型）/GF0Z~GF49（掉电保持型）（modbus 即时通讯）
工程级	系统提供的全部类型	当前工程中生效	允许掉电保存
任务级	系统提供的全部类型	当前任务中生效	单任务常见形式
模块级	系统提供的全部类型	当前模块中生效	不同模块间，变量名可以重复（定义在 Function 自定义函数中的）

6.1.3 自定义变量声明种类

关键字	数据类型	最小取值	最大取值	其他说明
int	整型	-32768	32767	
uint	无符号整形	0	65535	
float	浮点型	N/A	N/A	精度 0.000001
string	字符串型	1	128	长度范围
point	点位型	N/A		类型固定 (float,float,float,float,unit)

6.1.3.1 变量命名规范

1. 必须字母开头，后面可以接字母、数字、下划线。
2. 大小写表示不同的变量，如 int A , a 表示声明 2 个不同的变量。

6.1.3.2 数组

1. 带类型声明后接 () 明确数组容量，最大值 65535。举例：`int arr[20]` , abc
2. 数组下标取值范围：0~容量-1。举例：接上个例子，`abc = arr[2]` //取 arr 数组第 3 个元素赋值给 abc



提示

在全局变量里声明的所有变量都是全局的（当前工程有效），除此之外声明的均为局部变量。

6.1.4 自定义函数

6.1.4.1 格式说明

在程序中使用关键字 **Function** 定义需要的功能函数。

声明格式：

Function 自定义名称(类型定义 参数变量, ...)

自定义的实现方法

(可省略) **Return** 返回值 //执行后直接退出函数

其他操作

FunctionEnd

使用规则：

关于命名：英文字母开头，接下划线、数字、字母，区分大小写；

关于位置：遵循先声明再调用的规则，即：在调用前，提前声明好。

调用规则：

在函数声明的后方调用，引用**自定义名称**+()

详细使用方法参考下面的例子。

最大声明数量：

20 个。



提示

将自定义函数定义在主任务或后台任务中时，相当于局部变量，按照“先定义后使用”的原则，否则会报错。任务和任务之间的函数不能互相访问。

将函数定义在全局变量中，各个任务都可以访问。

6.1.4.2 无运动类型的自定义函数使用举例

例 1：自定义加法函数：

```
Function myadd ( float a , float b )
```

```
    Return a+b
```

```
FunctionEnd
```

在程序中用法如下，比如写在主程序中：

```
Process Main
```

```
    Function myadd ( float a , float b )
```

```
        Return a+b
```

```
    FunctionEnd
```

```
    float i = myadd(100.2 , 2.5)
```

```
    Print("计算结果 i = " , i)
```

```
ProcessEnd
```

```
//调用自定义函数 myadd 计算，i 的返回值为 102.7
```

例 2：自定义位置计算

//4 点法算法定义

```
Function GetposAtRight(point posa,point posb,point posc,point posd,int i,int j,int row,int col)
```

```
    float xa=posa.x , ya=posa.y , za=posa.z , ca=posa.c
```

```
    float xb=posb.x , yb=posb.y , zb=posb.z
```

```
    float xc=posc.x , yc=posc.y , zc=posc.z
```

```
    float xd=posd.x , yd=posd.y , zd=posd.z
```

```
    float resualt_x=((row-i)*(col-j)*xa+i*(col-j)*xb+i*j*xc+(row-i)*j*xd)/(row*col)
```

```
    float resualt_y=((row-i)*(col-j)*ya+i*(col-j)*yb+i*j*yc+(row-i)*j*yd)/(row*col)
```

```
    float resualt_z=((row-i)*(col-j)*za+i*(col-j)*zb+i*j*zc+(row-i)*j*zd)/(row*col)
```

```
    point rtn = BuildPoint(resualt_x,resualt_y,resualt_z,ca,1)//右手系
```

```
    Return rtn
```

```
FunctionEnd
```

6.1.4.3 含运动类型的自定义函数使用举例

例 1：取标程序

```
Function GetLable()//取标程序
```

```
    point feederPos
```

```
    If l80 ==2 Then
```

```
        feederPos = feederPos1
```

EndIf

If I80 == 3 Then

feederPos = feederPos2

EndIf

MOVJ(feederPos+Z(30,0),speed,acc,dec,cp)

Open(vacuum)

MOVJ(feederPos,speed,acc,dec,cp)

Delay(500)

MOVJ(feederPos+Z(30,0),speed,acc,dec,cp)

Break()

FunctionEnd

...

GetLable() //调用取标程序

6.1.5 关键字列表

关键字列表

int	MOVJ	Zlimit	Open	Close	BuildPoint
uint	MOVL	Delay	Pulse	ReadDI	Matrix
float	MOVC	Rand	WaitDI	ReadDO	U2U
string	Dn	And	Or	Not	Where
point	JUMP	SysTime	Print	Pause	X
Do	Exit	While	Util	Loop	Y
If	EndIf	Else	Elseif	Then	Z
For	To	Step	Next	GoTo	C
Switch	EndSwitch	Case	Default		.x
Power	Sin	Cos	Tan	Pn	.y
Abs	Asin	Acos	Atan		.z
OpenCOM	CloseCOM	ReadCOM	WriteCOM		.c
CheckCOM	SetCOM			Tool	.j1
OpenNet	CloseNet	ReadNet	WriteNet	User	.j2
CheckNet	SetNet			SetTool	.j3
StrAscValue	StrAscChar			SetUser	.j4
StrLen	StrLeft	StrRight	StrMid	StrTrim	StrFind
StrReplace	StrSlicer	StrEmpty	StrCat	StrFormat	StrData
StrCmp	StrnCmp	StrToupper	StrTolower	StrToString	StrToValue
Process	Main	Sub	Global	ProcessEnd	Function
FunctionEnd		MIRead	MFRead	MIWrite	MFWrite
		GIRead	GFRead	GIWrite	GFWrite

随系统软件的更新，关键字部分可能会有调整，请以实际提供的指令列表为准。

6.1.6 运算符列表

运算符列表					
运算符	作用	运算符	作用	关系运算符	作用
+	加	//	注释	>	大于
-	减	()	函数	<	小于
*	乘	[]	数组符号	==	等于
/	除	:	跳转标签:	>=	大于等于
%	求余	,	参数分隔	<=	小于等于
=	赋值	.	引用坐标成员	!=	不等于
		"..."	字符串...		
		!...!... !	并行指令		
		\n	换行转义符		
		\$b	2 进制常量		
		\$o	8 进制常量		
		\$x	16 进制常量		
		%X	StrFormat 格式，使用%04X 表示 0x0000 格式		

6.2 指令功能说明

6.2.1 运动类指令

指令	功能
<u>MOVJ</u>	(格式 1) 以点到点关节插补方式从当前位置运动到目标点
<u>MOVJ</u>	(格式 2) 指定某个轴运动到目标位置 (角度或毫米)
<u>MOVL</u>	以直线插补方式从当前位置运动到目标点
<u>MOVC</u>	从当前位置出发,按当前位置、途经位置和目标位置计算以圆弧插补方式到达目标点
<u>JUMP</u>	以门型关节插补方式到达目标点
<u>Dn</u>	设置并行指令同步开始执行处占总行程的百分比数值
<u>Tool</u>	选择工具号立即应用到当前,直至被切换成另一个工具号
<u>SetTool</u>	修改该工具号的内容
<u>User</u>	选择用户号立即应用到当前,直至被切换成另一个工具用户号
<u>SetUser</u>	修改该用户号的内容

MOVJ

MOVJ(point_a, speed, acc, dec, cp)

功能： 以点到点关节插补方式从当前位置运动到目标点

参数：

point_a	目标位置
speed	最高运行速度,设定范围 1~100
acc	加速度设置,设定范围 1~100
dec	减速度设置,设定范围 1~100
cp	轨迹连续设置,设定范围 0：关闭光滑；1：开启光滑

返回： 无

说明： 轨迹不是固定模式，在有干涉场景注意有足够的避让空间

举例： 例 1

```
int speed=100,acc=100,dec=100,cp=0    //定义运动参数

Do

float old_time=SysTime()                //获取当前时间

MOVJ(Pn(1),speed,acc,dec,cp)            //运动到点位表中的 1#点位

MOVJ(Pn(2),speed,acc,dec,cp)            //运动到点位表中的 2#点位

MOVJ(Pn(3),speed,acc,dec,cp)            //运动到点位表中的 3#点位

MOVJ(Pn(2),speed,acc,dec,cp)            //运动到点位表中的 2#点位

// 再次获得当前时间,与之前的时间相
// 减得到两条之间之间的消耗时间

Print("程序运行耗时",(SysTime()-old_time)/1000,"s" )

Loop
```

例 2

```
int speed=100,acc=100,dec=100,cp=0    //定义运动参数

point pt_a=Pn(4)                       //定义点位类型变量 pt_a，用点位表中的
                                        4#点位给该变量赋初值
```

```

point pt_b=Pn(5)                                //定义点位类型变量 pt_b，用点位表
                                                中的 5#点位给该变量赋初值

point pt_c=Pn(6)                                //定义点位类型变量 pt_c，用点位表中的
                                                6#点位给该变量赋初值

Do

float old_time = SysTime()                      //获取当前时间

MOVJ(pt_a, speed, acc, dec, cp)                 //运动到 a 点位，相当于运动到点位表
                                                4#点位

MOVJ(pt_b, speed, acc, dec, cp)                 //运动到 b 点位，相当于运动到点位表
                                                5#点位

MOVJ(pt_c, speed, acc, dec, cp)                 //运动到 c 点位，相当于运动到点位表
                                                6#点位

MOVJ(pt_b, speed, acc, dec, cp)                 //运动到 b 点位，相当于运动到点位表
                                                5#点位

                                                // 再次获得当前时间,与之前的时间相
                                                减得到两条指令之间的消耗时间

Print("程序运行耗时",(SysTime()-old_time)/1000,"s" )

Loop

```

例 3

```

int speed=100,acc=100,dec=100,cp=0            //定义运动参数

point pick=BuildPoint(300,-300,0,0,1)          //定义 pick 点，并赋值

point place=BuildPoint(300,0,0,0,1)            //定义 place 点，并赋值

float h = 25                                    //定义浮点类型变量 h，赋初值 25

Do

float old_time=SysTime()                      //获取当前时间

MOVJ(pick+Z(h,0),speed,acc,dec,cp)             //运动到 pick 上方距离为 h 处点位

MOVJ(pick,speed,acc,dec,cp)                   //运动到 pick 点位

```

MOVJ(pick+Z(h,0),speed,acc,dec,cp)	//运动到 pick 上方距离为 h 处点位
MOVJ(place+Z(h,0),speed,acc,dec,cp)	//运动到 place 上方距离为 h 处点位
MOVJ(place,speed,acc,dec,cp)	//运动到 place 点位
MOVJ(place+Z(h,0),speed,acc,dec, cp)	//运动到 place 上方距离为 h 处点位
	// 再次获得当前时间,与之前的时间相减得到两条指令之间的消耗时间

Print("程序运行耗时",(SysTime()-old_time)/1000,"s")

Loop

参阅： Pn , BuildPoint , Matrix , Where , X , Y , Z , C , Dn ,Break

MOVJ

MOVJ(n, angle_pos, speed, acc, dec, cp)

功能： 指定某个轴运动到目标位置（角度或毫米）

参数：

n	目标轴号(1,2,3,4)
angle	目标绝对位置(1-2-4 轴单位角度,3 轴单位 mm)
speed	最高运行速度,设定范围 1~100
acc	加速度设置,设定范围 1~100
dec	减速度设置,设定范围 1~100
cp	轨迹连续设置,设定范围 0：关闭光滑；1：开启光滑

返回： 无

说明： 要清晰的知道目标轴号运行是否会带来碰撞

举例：

```
int speed=100,acc=100,dec=100,cp=0 //定义运动参数
MOVJ(1,0,speed,acc,dec,cp) //其他轴保持不动，1 轴单独运动到 0
                               度位置
float target=30 //定义 3 轴到达目标，30mm（单位不
                               用写）
MOVJ(3,target,speed,acc,dec,cp) //其他轴保持不动，3 轴单独运动到
                               30mm 位置
```

参阅： Dn , Break

MOVL

MOVL(point_a, speed, acc, dec, cp)

功能： 以直线插补方式从当前位置运动到目标点

参数：

point_a	目标位置
speed	最高运行速度,设定范围 1~100
acc	加速度设置,设定范围 1~100
dec	减速度设置,设定范围 1~100
cp	轨迹连续设置,设定范围 0：关闭光滑；1：开启光滑

返回： 无

说明：

- 1、J2 轴关节角接近 0 度附近是奇异点，在此范围内的直线运动可能引起关节超速报警。选点的时候尽量避免
- 2、当前点和目标点之间的直线如果通过软限位区域，会引起软限位报警，应该尽量避免
- 3、速度、加减速参数为相对值，受参数-运动参数-自动最大平动速度、自动最大平动加速度影响
- 4、实际最大直线运动速度计算方法，自动最大平动速度×speed×倍率

举例：

```

int speed=100,acc=100,dec=100,cp=0           //定义运动参数
MOVL(Pn(1),speed,acc,dec,cp)                 //运动到点位表中设置的点位
point point_a=Pn(2)                          //定义点位并用点位表中 2#点赋值
MOVL(point_a,speed,acc,dec,cp)               //运动到点位 point_a
point_a=BuildPoint( 250,450,-50,20,1)        //更改 point_a 的内容，不会影响到
                                              Pn(2)记录的值
MOVL(point_a,speed,acc,dec,cp)               //运动到更新后的点位
  
```

参阅： Pn , BuildPoint , Matrix , Where , X , Y , Z , C , Dn ,Break

MOVC

MOVC(point1, point2, speed, acc, dec, cp)

功能： 从当前位置出发,按当前位置、途经位置和目标位置计算以圆弧插补方式到达目标点

参数：

point1	途经位置
point2	目标位置
speed	最高运行速度,设定范围 1~100
acc	加速度设置,设定范围 1~100
dec	减速度设置,设定范围 1~100
cp	轨迹连续设置,设定范围 0：关闭光滑；1：开启光滑

返回： 无

说明：

- 1、依据 3 点画圆规则生成轨迹，圆弧段轨迹不要通过软限位
- 2、示教位置的时候需要注意，途经点和目标点的顺序会影响运动轨迹，不具有交换性质

举例：

```
int speed=100,acc=100,dec=100,cp=0      //定义运动参数
MOVJ(Pn(1),speed,acc,dec,cp)             //到达 P1 点
MOVC(Pn(2),Pn(3),speed,acc,dec,cp)       //从 P1 出发,经 P2、P3 的圆弧,最终到达 P3 点
```

参阅： Pn , BuildPoint , Matrix , Where , X , Y , Z , C , Dn ,Break

MOVC(point1, point2, angle, speed, acc, dec, cp)

功能： 按当前位置、途经位置和目标位置计算圆,以当前点-途经点-目标点方向为角度正方向,从当前位置作为角度起点,目标角度为终点做圆弧插补

参数：

point1	途经位置
point2	从当前位置向 Z+ 方向抬升高度,单位 mm
angle	目标角度,相对于当前位置（起点计算）,单位度，取值范围-360~360 内
speed	最高运行速度,设定范围 1~100
acc	加速度设置,设定范围 1~100
dec	减速度设置,设定范围 1~100

cp 轨迹连续设置,设定范围 0 : 关闭光滑 ; 1 : 开启光滑

返回 : 无

说明 :

- 1、依据 3 点画圆规则生成轨迹 , 圆弧段轨迹不要通过软限位
- 2、当角度范围设置超过 ± 360 , 运动到 360 停止 (或-360)
- 3、示教位置的时候需要注意 , 途经点和目标点的顺序会影响运动轨迹 , 不具有交换性质

举例 :

```
int speed=100,acc=100,dec=100,cp=0      //定义运动参数
MOVJ(Pn(1),speed,acc,dec,cp)
MOVC(Pn(2),Pn(3),360,speed,acc,dec,cp)    //从 P1 出发,经 P2、 P3 的圆弧 , 运动
                                           360 度 , 最终回到 P1 点
```

参阅 : Pn , BuildPoint , Matrix , Where , X , Y , Z , C , Dn ,Break

JUMP

JUMP(point_a, h1, h2, h3, speed, acc, dec, cp)

功能： 以门型关节插补方式到达目标点（垂直方向门型,水平方向不定）

参数：

point_a	目标位置
h1	从当前位置向 Z+方向抬升高度,单位 mm
h2	从目标位置向 Z+方向抬升高度,单位 mm
h3	移动过程经过的最高 Z 方向限位(绝对高度),单位 mm
speed	最高运行速度,设定范围 1~100
acc	加速度设置,设定范围 1~100
dec	减速度设置,设定范围 1~100
cp	轨迹连续设置,此处无效,始终运动到目标点位

返回： 无

说明：

- 1、最高限位的设定要同时至少高于当前位置和目标点分别加上抬升高度后的高度；
- 2、当设置的高度低于起始点或目标点增加后的高度，抬升或下降高度将被削减；
- 3、当设置的限制高度稍大于抬升后或者下降前的高度时，可能更有效率；
- 4、最高限位设置值不能超过 J3 轴软限位上限；
- 5、其他用法参考 MOVJ
- 6、当前起始位置高于最高限制高度会造成报警，需要将当前位置降低或者增加最高限制（具体结合实际应用情况分析）

举例： 例 1

```
int speed=100,acc=100,dec=100,cp=0           //定义运动参数
point pick=BuildPoint(300,-300,0,0,1)         //定义 pick 点，并赋值
point place=BuildPoint(300,0,0,0,1)           //定义 place 点，并赋值
point standby=BuildPoint(300,0,0,0,1)         //定义 standby 点，并赋值
float h=0.5,zlim=25                            //定义 h 为提升高度，zlim 为 Z 轴绝对限高
MOVJ(standby,speed,acc,dec,cp)                //运动到点位表中设置的 1#点位
```

Do

float old_time=SysTime() //获取当前时间

JUMP(pick,h,h,zlim,speed,acc,dec,cp) //跳跃运动(门型)到 pick 点位

JUMP(place,h,h,zlim,speed,acc,dec,cp) //跳跃运动(门型)到 place 点位

// 再次获得当前时间,与之前的时间相
减得到两条指令之间的消耗时间

Print("程序运行耗时",(SysTime()-old_time)/1000,"s")

Loop

参阅： Pn , BuildPoint , Matrix , Where , X , Y , Z , C , Dn ,Break

Dn

Dn(per)

功能： 设置并行指令同步开始执行处占总行程的百分比数值

参数： per 0~100 整数，单位为%，不用写单位

返回： 无

说明： 1、不可单独使用

2、用于运动指令之后,格式：!MOVJ(Pn(1),speed,acc,dec,cp)!Dn(20);Open(2)!

3、如果涉及到多个并行操作，在!...!之间用';'隔开

4、JUMP 类型的移动，在抬升和下降阶段不计算总行程

备注：当前版本无 JUMP 运动功能

举例： int speed=100 , acc=100 , dec=100 , cp=0

!MOVJ(Pn(1),speed,acc,dec,cp)!Dn(20);Open(2);Dn(80);Close(2)!

MOVJ(Pn(2),speed,acc,dec,cp)

参阅： MOVJ , Open , Close

Tool

Tool(tool_num)

功能： 选择工具号立即应用到当前,直至被切换成另一个工具号

参数： tool_num 工具号：0~9

返回： 无

说明： 1、特殊情况,在使用的点位是纯关节式的点位,工具号对此不生效
2、Tool(0) //表示没有附加外置工具，它表示所有偏移量都是 0 的默认工具

举例： Tool(1) //选择 1 号工具，直到再次被改变

参阅： SetTool

SetTool

SetTool(tool_num, xpos, ypos, zpos, cpos)

功能： 修改该工具号的内容

参数：

tool_num	工具号：1~9
xpos	工具号 X 的绝对值
ypos	工具号 Y 的绝对值
zpos	工具号 Z 的绝对值
cpo	工具号 C 的绝对值

返回： 无

说明： 0 号工具为法兰中心工具（无任何偏移值），不可以修改

举例： SetTool(1 , 10 ,10 ,0 ,0) //修改 1#工具坐标系，内容为
X=10,Y=10,Z=0,C=0

参阅： Tool

User

User(user_num)

功能： 选择用户号立即应用到当前,直至被切换成另一个用户号

参数： user_num 用户号：0~15

返回： 无

说明： 1、特殊情况,在使用的点位是点位表中引用的点位,用户号对此不生效

User(2)

MOVJ(Pn(1) , speed , acc , dec , cp) //User(2)对此位置无影响

point abc = BuildPoint(23, 45 , 25 , 0 , 0)

MOVJ(abc ,speed , acc , dec , cp) //User (2) 生效下的点位值

2、 User(0)表示世界坐标系，无任何偏移值

User(2)

.....

User(0) //将用户切换回世界坐标系

3、由 BuildPoint 指令生成的点位，在 User()生效后，不同的用户号，最终的位置可能不相同，不同的 Tool()生效后，不同的工具号，最终到达的位置也可能不相同

point abc = BuildPoint(23, 45 , 25 , 0 , 0)

User(2)

Tool(1)

MOVJ(abc ,speed , acc , dec , cp) //User (2) Tool(1)生效下的点位值

Tool(2)

MOVJ(abc ,speed , acc , dec , cp) //User (2) Tool(2)生效下的新点位值

4、想让点位表中的点位跟随不同用户坐标系变化，可以使用 U2U 指令实现

User(2)

MOVJ(Pn(1) , speed , acc , dec , cp) //User(2)对此位置无影响

point abc = U2U(2 , 0 , Pn(1)) //将 Pn(1) 转换为用户 2 下的点位值

MOVJ(abc ,speed , acc , dec , cp)

//User (2) 生效下的点位值

举例： 参考上述说明

参阅： SetUser , Tool , SetTool , U2U

TT ROBOT

SetUser

SetUser(user_num, xpos, ypos, zpos, cpos)

功能： 修改该用户号的内容

参数：

user_num	用户号：0~15
xpos	用户号 X 的绝对值
ypos	用户号 Y 的绝对值
zpos	用户号 Z 的绝对值
cpos	用户号 C 的绝对值

返回： 无

说明： 0 号用户号为机器人 DH 模型对应的世界坐标系,不可以修改

举例： SetUser(1 , 10 ,10 ,0 ,0)

参阅： User

6.2.2 语法指令

指令

功能

[Do...Loop](#)

无条件循环执行 Do 和 Loop 之间的内容

[Do While...Loop](#)

在条件成立时循环执行 Do 和 Loop 之间的内容

[Do Until...Loop](#)

在条件成立时停止循环执行 Do 和 Loop 之间的内容,否则一直循环

[Exit Do](#)

直接退出当前层次的 Do 循环

[Exit For](#)

直接退出当前层次的 For 循环

[If...Then...Else...EndIf](#)

判断条件表达式是否成立,分支条件成立后执行对应的内容

[For...Next](#)

按照指定次数反复执行 For...Next 之间的一系列的语句

[Switch...Case...EndSwitch](#)

通过列举变量值,当检查到 Case 其中之一的数值成立后,选择执行对应后面的指令,执行完毕后面的不再检查

[GoTo label](#)

从当前行无条件跳转到指定的标签号所在行

Do

Do... ..Loop

功能： 无条件循环执行 Do 和 Loop 之间的内容

参数： 无

返回： 无

说明：

- 1.需要退出循环时在内容中用 Exit Do 语句来退出当前层次循环
- 2.建议在容易出现不执行其他指令的死循环之间加 Delay(10) , 防止刷死机

举例： int speed = 100 , acc = 100 , dec = 100,cp = 0

Do

MOVJ(Pn(1), speed, acc, dec, cp)

MOVJ(Pn(2), speed, acc, dec, cp)

Loop

参阅： Exit Do ,Delay

Do

Do While condition... ...Loop

功能： 在条件成立时循环执行 Do 和 Loop 之间的内容

参数： condition 条件表达式：如 sta == 1

返回： 无

说明： 1.需要退出循环时在内容中用 Exit Do 语句来退出当前层次循环
2.建议在容易出现不执行其他指令的死循环之间加 Delay(10)，防止刷死机

举例： int a = 1
int speed=100 , acc=100 , dec=100,cp=0
Do While a == 1
MOVJ(Pn(1), speed, acc, dec, cp)
MOVJ(Pn(2), speed, acc, dec, cp)
Loop

参阅： Exit Do , Delay , While

Do

Do Until condition... ...Loop

功能： 在条件成立时停止循环执行 Do 和 Loop 之间的内容,否则一直循环

参数： condition 条件表达式：如 sta == 1

返回： 无

说明： 1.需要退出循环时在内容中用 Exit Do 语句来退出当前层次循环
2.建议在容易出现不执行其他指令的死循环之间加 Delay(10)，防止刷死机

举例： int sta = 0
Do Until sta==0
Delay(1) //循环执行的内容
Loop

参阅： Exit Do , Delay , Do While , Do Until

Exit

Exit Do

功能： 直接退出当前层次的 Do 循环

参数： 无

返回： 无

说明： 需要退出 Do 循环时在内容中用 Exit Do 语句来退出当前层次循环

举例：

```
int sta = 1
Do While 1
    Delay(1) //循环执行的内容
    If sta == 1 Then
        Exit Do //当 sta=1 的时候，退出当前层次 Do 循环
    EndIf
Loop
```

参阅： Exit Do , Delay , Do While ,Do Until

Exit

Exit For

功能： 直接退出当前层次的 For 循环

参数： 无

返回： 无

说明： 需要退出 Do 循环时在内容中用 Exit Do 语句来退出当前层次循环

举例：

```
int i ,sta = 1
For i=0 To 15 Step 1
    Delay(1)                //循环执行的内容
    If sta ==1 Then
        Exit For            //当 sta=1 的时候，退出当前层次 For
                              循环
    EndIf
Next
```

参阅： For , To , Step , Next , Delay

If

If condition Then... ...Else... ...EndIf**功能：** 判断条件表达式是否成立,分支条件成立后执行对应的内容**参数：** condition 条件表达式：如 sta == 1 , sta > 1 And sta < 3 , a < 5 Or b < 5**返回：** 无

说明：

- 1、在有 Else 的情况下,条件成立时执行 Then 和 Else 之间的指令,不成立时执行 Else 和 EndIf 之间的指令
- 2、在没有 Else 的情况下,条件成立时执行 Then 和 EndIf 之间的指令,不成立时退出条件判断
- 3、Else 不成立分支条件,【缺省】对不成立条件不做处理
- 4、Elseif 附加一个条件,在 If 判断为不成立,进入此判断,【缺省】对不成立条件不做处理
- 5、本指令只检查一次,在需要反复检查的时候,需要嵌入在循环中使用

举例： 例 1

int sta = 1

If sta == 1 Then

//判断 sta 是否等于 1

statement1

//条件成立,执行 statement1 内容

Else

statement2

//条件不成立,执行 statement2 内容

EndIf

//退出判断

例 2

int sta = 2

If sta == 1 Then

//判断 sta 是否等于 1

statement1

//条件成立,执行 statement1 内容

Elseif sta == 2 Then

//判断 sta 是否等于 2

statement2

//条件成立,执行 statement2 内容

Elseif sta == 3 Then

//判断 sta 是否等于 3

```
Statement3                                //条件成立，执行 statement3 内容
Elseif ..... Then
Statement4
Else                                       //以上条件都不满足执行 Statement5
Statement5
EndIf                                     //退出判断
```

参阅： And , Or , Not , Exit , GoTo ,Do , If

For

For i=d1 To d2... ..Next**功能：** 按照指定次数反复执行 For...Next 之间的一系列语句

参数：

i	数值类型变量
d1	起始值
d2	结束值
Step	每次循环变量增减值,[缺省]模式下,i+1

返回： 无

说明： For 语句的下一语句将在达到 Next 命令之后执行。计数器变量（变量名）仅按由增量值（Step 增量值）指定的值进行增量。如果未设置增量值，计数器将每次增加“1”。计数器变量（变量名）与最终数值进行比较。如果计数器变量小于或等于最终数值，将重新执行 For 命令的下一语句。如果计数器变量（变量名）大于最终数值，执行将分支到 For...Next 循环外，继续执行 Next 命令的下一命令。

举例： 例 1

int i

For i = 1 To 10

MOVJ(Pn(i) ,speed , acc , dec , cp) //从当前位置依次运动到点位表的
1~10#位置

Next

例 2

int i

For i = 10 To 1 Step -1

MOVJ(Pn(i) ,speed , acc , dec , cp) //从当前位置依次运动到点位表的
10~1#位置

Next

参阅： Do...Loop

Switch

Switch var...Case d1... ...Default... ...EndSwitch

功能： 通过列举变量值,当检查到 Case 其中之一的数值成立后,选择执行对应后面的指令，执行完毕后面的不再检查

参数：

var	数值类型或字符串类型变量
d1	判断变量是否等于列举值 1
d2	判断变量是否等于列举值 2
...	...
Default	不满足上述列举值的时候,执行此处内容【缺省】不满足上述值不做处理

返回： 无

说明： 同 If 语句一样，本指令只检查一次，在需要反复检查的时候，需要嵌入在循环中使用

举例：

```

int var
Switch var
Case 1                                //var=1
statement1                            //条件成立执行完退出
Case 2                                //var=2
statement2                            //条件成立执行完退出
Case ...
...
Default                                //var 不等于上述所有列举的值
statement_default                    //条件成立执行完退出
EndSwitch
  
```

参阅： If...Then...Else

GoTo

GoTo label

功能： 从当前行无条件跳转到指定的标签号所在行

参数： label 语句跳转目的标签

返回： 无

说明： 1、使用 GoTo 语句前，需要事先声明好标签

2、标签的位置可以在前或后

3、标签可以自定义，但要在后面加上 : 符号

举例： ...

label: //目的标签

...

...

GoTo label //跳转到目的标签

参阅： Do , For , If , Switch

6.2.3 点操作类指令

指令	功能
<u>Pn</u>	引用点位表中的数据
<u>Where</u>	返回当前 TCP 末端所在的位置
<u>Matrix</u>	根据提供的参考点，创建一个平行四边形（包括长方形）的阵列点位集合
<u>Matrix</u>	引用指定阵列编号的某个阵列点
<u>U2U</u>	将源坐标系下的位置点转化成目标坐标系下的坐标点
<u>BuildPoint</u>	创建一个点位类型数据（笛卡尔类型）
<u>BuildJPoint</u>	创建一个点位类型数据(关节类型)
<u>BuildX</u>	根据参考点变更 X 坐标，返回新的位置点
<u>BuildY</u>	根据参考点变更 Y 坐标，返回新的位置点
<u>BuildZ</u>	根据参考点变更 Z 坐标，返回新的位置点
<u>BuildC</u>	根据参考点变更 C 坐标，返回新的位置点
<u>GetPointX</u>	返回目标点所在的位置的 X 坐标分量
<u>GetPointY</u>	返回目标点所在的位置的 Y 坐标分量
<u>GetPointZ</u>	返回目标点所在的位置的 Z 坐标分量
<u>GetPointC</u>	返回目标点所在的位置的 C 坐标分量
<u>GetPointJ1</u>	返回目标点所在的位置的 J1 轴角度分量
<u>GetPointJ2</u>	返回目标点所在的位置的 J2 轴角度分量
<u>GetPointJ3</u>	返回目标点所在的位置的 J3 轴位置分量
<u>GetPointJ4</u>	返回目标点所在的位置的 J4 轴角度分量

Pn

Pn(num)

功能： 引用点位表中的数据

参数： num 点位表中数据引用 范围 0~9999

返回： 点位型数据

说明：

- 1、点位表中的数据分两种形式，关节型或笛卡尔型。
- 2、当引用的关节型数据被运动指令直接使用时，忽略上文中的 User()和 Tool 指令；
- 3、当引用的笛卡尔型数据被运动指令直接使用时，忽略上文中的 User()指令，但会执行 Tool()指令；
- 4、当位置点来自客户 BuildPoint()指令创建的，同时执行 User(),Tool()指定的参考系。
- 5、Pn(0)表示机器人的机械零位，其他的点位号为用户示教的

举例： point pt_new=Pn(1) //声明点位型变量 pt_new,再把点位表
序号 P00001 内容值赋值给 pt new

参阅： MOVJ, MOVL, MOVC, JUMP, JUMPL, User, Tool, BuildPoint, Tool, User, U2U

Where

Where()

功能： 返回当前 TCP 末端所在的位置

参数： 无

返回： 点位数据

说明： 世界坐标系的笛卡尔型

举例： `pt_now=Where()` //把当前位置赋值给 pt_now

参阅： `.x , .y , .z , .c , GetPointX , GetPointY , GetPointZ , GetPointC`

Matrix

Matrix(1, home_pos, dir1_pos, dir2_pos, row, col)

功能： 根据提供的参考点，创建一个平行四边形（包括长方形）的阵列点位集合,总数量等于第 1 方向的点位数量×第 2 方向的点位数量

参数：

1	阵列编号：1~200
home_pos	阵列初始点
dir1_pos	第 1 方向终点
dir2_pos	第 2 方向终点
row	第 1 方向总点位数量≥1，row 和 col 不能同时为 1
col	第 2 方向总点位数量≥1，row 和 col 不能同时为 1

返回： 无

说明： 允许创建为空间四边形，即：点位平面与 XY 平面不一定平行

举例： /*-----把下面 Gi Gj 声明复制到全局变量中-----*/

```
int Gi=0, Gj=0
```

```
-----*/
```

```
int speed=100,acc=100,dec=100,cp=0 //定义运动参数
```

```
int m=4, n=3, i=0, j=0 //设定方向 1 有 m 个，方向 2 有 n 个
```

```
point org=BuildPoint(300.000, 000.000, - //设置 org 点
020.000, 000.000, 0)
```

```
point pt1=BuildPoint(300.000, 300.000, - //设置方向 1 终点
020.000, 000.000, 0)
```

```
point pt2=BuildPoint(400.000, 000.000, - //设置方向 2 终点
020.000, 000.000, 0)
```

```
Matrix(1,org,pt1,pt2,m,n) //根据参考点创建 m×n 编号 1 的阵列
```

```
Do
```

```
For i = Gi To m-1
```

```
Gi = i //保存当前值到全局变量 Gi
```

```
For j = Gj To n-1
```

```
Gj = j                                //保存当前值到全局变量 Gj

float h=5, zlim=10, z=25

/*

MOVJ(Matrix(1,i,j)+Z(z,0),speed,acc,dec,cp)  //依次运动到 1#矩阵中点位+z

MOVJ(Matrix(1,i,j),speed,acc,dec,cp)         //依次运动到 1#矩阵中点位

MOVJ(Matrix(1,i,j)+Z(z,0),speed,acc,dec,cp)  //依次运动到 1#矩阵中点位+z

*/

JUMP(Matrix(1,i,j),h,h,zlim,speed,acc,dec,cp //依次运动到 1#矩阵中点位
)

Break()

Next

Gj = 0                                //循环结束，复位变量至初值

Next

Gi = 0                                //循环结束，复位变量至初值

Loop
```

参阅： BuildX , BuildY , BuildZ , BuildC

Matrix

Matrix(1, i, j)

功能： 引用指定阵列编号的某个阵列点

参数： 1 矩阵编号：1~200
i 第 1 方向序号,从 0 开始
j 第 2 方向序号,从 0 开始

返回： 点位型数据

说明： 序号从 0 开始，直到总数值减 1 结束

举例： 详见 Matrix 格式 1

参阅： BuildX , BuildY , BuildZ , BuildC ,MOVJ , MOVL

U2U

U2U(target_user_num, source_user_num, point_a)

功能： 将源坐标系下的位置点转化成目标坐标系下的坐标点

参数： target_user_num 目标用户号：0~15

source_user_num 源用户号：0~15

point_a 源用户号下的位置点

返回： 点位型数据

说明： 1、由 BuildPoint 创建的点位默认为世界坐标系的点位，对应的 source_user_num 为 0

2、点位表中的关节型点位，对应的 source_user_num 为 0

举例： pt_new=U2U(1,2,point_a) //将 2 坐标系下的 point_a 转成 1 坐标系下的点位值并赋值给 pt_new

参阅： User , BuildPoint , Pn

BuildPoint

BuildPoint(xpos, ypos, zpos, cpos, hand)

功能： 创建一个点位类型数据

参数：

xpos	创建点的 X 坐标
ypos	创建点的 Y 坐标
zpos	创建点的 Z 坐标
cpos	创建点的 C 坐标
hand	手系：左手系 0,右手系 1

返回： 点位型数据

说明： 在程序中创建的点位数据不包含用户坐标系选项，可以通过 User 指令设定，如果未指明 User，默认按照世界坐标系执行，即：User(0)

举例： pt_new=BuildPoint(100,110,120,130,0) //创建点位 xyzc 分别是
100,110,120,130 的 pt_new,左手系,
再把结果值赋值给 pt_new

参阅： MOVJ , MOVL , MOVC , JUMP , JUMPL ,Pn , User , Tool

BuildJPoint

BuildJPoint(j1,j2,j3,j4)

功能： 创建一个点位类型数据(关节类型)

参数： j1 创建点的 j1 坐标

j2 创建点的 j2 坐标

j3 创建点的 j3 坐标

j4 创建点的 j4 坐标

返回： 点位型数据(关节类型)

说明： 该点位不受程序上文的 User、Tool 命令影响。

举例： pt_new=BuildJPoint(-40,50,3,20) //创建关节坐标 J1、J2、J3、J4 分别是-40°，50°，3mm，20°

参阅： MOVJ , MOVL , MOVC , JUMP , JUMPL , Pn , User , Tool

BuildX

BuildX(point_a, 0, dis)

功能： 根据参考点变更 X 坐标，返回新的位置点

参数： point_a 参考点

0 相对关系（1，直接替换）

dis 相对与待转换的点位 X 方向的值,单位 mm

返回： 点位型数据

说明： 该指令与 Point+X (dis , 0/1)起到的效果相同

举例： point_a_new=BuildX(point_a , 0 , 25) //将 point_a 的点位值按照 X 正方向增加 25mm,再把结果赋值给 pt_new

参阅： MOVJ , MOVL , MOVC , JUMP , JUMPL ,Pn , User , Tool

BuildY

BuildY(point_a, 0, dis)

功能： 根据参考点变更 Y 坐标，返回新的位置点

参数： point_a 参考点

0 相对关系（1，直接替换）

dis 相对与待转换的点位 Y 方向的值,单位 mm

返回： 点位型数据

说明： 该指令与 Point+Y (dis , 0/1)起到的效果相同

举例： point_a_new=BuildY(point_a , 0 , 25) //将 point_a 的点位值按照 Y 正方向增加 25mm,再把结果赋值给 pt_new

参阅： MOVJ , MOVL , MOVC , JUMP , JUMPL ,Pn , User , Tool

BuildZ

BuildZ(point_a, 0, dis)

功能： 根据参考点变更 Z 坐标，返回新的位置点

参数： point_a 参考点

0 相对关系（1，直接替换）

dis 相对与待转换的点位 Z 方向的值,单位 mm

返回： 点位型数据

说明： 该指令与 Point+Z(dis , 0/1)起到的效果相同

举例： point_a_new=BuildZ(point_a , 0 , 25) //将 point_a 的点位值按照 Z 正方向增加 25mm,再把结果赋值给 pt_new

参阅： MOVJ , MOVL , MOVC , JUMP , JUMPL ,Pn , User , Tool

BuildC

BuildC(point_a, 0, dis)

功能： 根据参考点变更 C 坐标，返回新的位置点

参数： point_a 参考点

0 相对关系（1，直接替换）

dis 相对与待转换的点位 C 方向的值,单位 mm

返回： 点位型数据

说明： 该指令与 Point+C (dis , 0/1)起到的效果相同

举例： point_a_new=BuildC(point_a , 0 , 25) //将 point_a 的点位值按照 C 正方向增加 25mm,再把结果赋值给 pt_new

参阅： MOVJ , MOVL , MOVC , JUMP , JUMPL ,Pn , User , Tool

GetX

GetPointX(point_a)

功能： 返回目标点所在的位置的 X 坐标分量

参数： point_a 点位值

返回： 点位型数据的 X 坐标分量

说明：

- 1、该指令与 `point_a x` 起到的效果相同
- 2、注意将接收值定义成 `float` 数据类型

举例： `ax=GetPointX(point_a)` //返回 point_a 的 X 分量,把结果赋值给 ax 变量

参阅： `.x` , `.y` , `.z` , `.c` , Where

GetPointY

GetPointY(point_a)

功能： 返回目标点所在的位置的 Y 坐标分量

参数： point_a 点位值

返回： 点位型数据的 Y 坐标分量

说明：

- 1、该指令与 `point_a y` 起到的效果相同
- 2、注意将接收值定义成 `float` 数据类型

举例： `ay=GetPointY(point_a)` `//返回 point_a 的 Y 分量,把结果赋值给 ay 变量`

参阅： .x , .y , .z , .c , Where

GetPointZ

GetPointZ(point_a)

功能： 返回目标点所在的位置的 Z 坐标分量

参数： point_a 点位值

返回： 点位型数据的 Z 坐标分量

说明：

- 1、该指令与 `point_a.z` 起到的效果相同
- 2、注意将接收值定义成 `float` 数据类型

举例： `az=GetPointZ(point_a)` //返回 point_a 的 Z 分量,把结果赋值给 az 变量

参阅： .x , .y , .z , .c , Where

GetPointC

GetPointC(point_a)

功能： 返回目标点所在的位置的 C 坐标分量

参数： point_a 点位值

返回： 点位型数据的 C 坐标分量

说明： 1、该指令与 point_a.c 起到的效果相同

2、注意将接收值定义成 float 数据类型

3、单位为角度制

举例： ac=GetPointC(point_a) //返回 point_a 的 C 分量,把结果赋值
给 ac 变量

参阅： .x , .y , .z , .c , Where

GetPointJ1

GetPointJ1(point_a)

功能： 返回目标点所在的位置的 J1 轴角度分量

参数： point_a 点位值

返回： 点位型数据的 J1 轴角度分量

说明： 1、该指令与 point_a.j1 起到的效果相同

2、注意将接收值定义成 float 数据类型

3、返回值为角度制

举例： jointpos=GetPointJ1(point_a) //返回 point_a 的 J1 轴角度分量,把结果赋值给 jointpos 变量

参阅： .j1 , .j2 , .j3 , .j4 , Where

GetPointJ2

GetPointJ2(point_a)

功能： 返回目标点所在的位置的 J2 轴角度分量

参数： point_a 点位值

返回： 点位型数据的 J2 轴角度分量

说明： 1、该指令与 point_a.j2 起到的效果相同

2、注意将接收值定义成 float 数据类型

3、返回值为角度制

举例： jointpos=GetPointJ2(point_a) //返回 point_a 的 J2 轴角度分量,把结果赋值给 jointpos 变量

参阅： .j1 , .j2 , .j3 , .j4 , Where

GetPointJ3

GetPointJ3(point_a)

功能： 返回目标点所在的位置的 J3 轴位置分量

参数： point_a 点位值

返回： 点位型数据的 J3 轴位置分量

说明： 1、该指令与 point_a.j3 起到的效果相同

2、注意将接收值定义成 float 数据类型

3、返回值单位为 mm

举例： jointpos=GetPointJ3(point_a) //返回 point_a 的 J3 轴位置分量,把结果赋值给 jointpos 变量

参阅： .j1 , .j2 , .j3 , .j4 , Where

GetPointJ4

GetPointJ4(point_a)

功能： 返回目标点所在的位置的 J4 轴角度分量

参数： point_a 点位值

返回： 点位型数据的 J4 轴角度分量

说明： 1、该指令与 point_a.j4 起到的效果相同

2、注意将接收值定义成 float 数据类型

3、返回值为角度制

举例： jointpos=GetPointJ4(point_a) //返回 point_a 的 J4 轴角度分量,把结果赋值给 jointpos 变量

参阅： .j1 , .j2 , .j3 , .j4 , Where

6.2.4 系统类指令

指令	功能
<u>SysTime</u>	返回系统当前时间,单位 ms
<u>Pause</u>	系统暂停
<u>Delay</u>	等待一段时间后再向下执行
<u>Print</u>	打印信息
<u>Break</u>	等待上一条的运动指令执行完毕后再执行 Break 之后的指令,防止该指令下一条非 IO 或 Delay 的指令被提前执行
<u>Power</u>	计算某个数的 n 次方的值
<u>Sqrt</u>	计算某个数的开平方值
<u>Abs</u>	计算某个数的绝对值
<u>Sin</u>	计算某个角度的正弦值
<u>Cos</u>	计算某个角度的余弦值
<u>Tan</u>	计算某个角度的正切值
<u>Asin</u>	计算某个值的反正弦,返回对应的角度值
<u>Acos</u>	计算某个值的反余弦,返回对应的角度值
<u>Atan</u>	计算某个值的反正切,返回对应的角度值
<u>RTA</u>	弧度值转角度值
<u>ATR</u>	角度值转弧度值

SysTime

SysTime()

功能： 返回系统当前时间,单位 ms

参数： 无

返回： 以 ms 为单位的数值

说明： 一般用法为,先后两次获取的时间相减得到消耗的时间差

举例： float old_time = SysTime() //获取当前时间

...

// 再次获得当前时间,与之前的时间相减得到两条之间之间的消耗时间

Print("程序运行耗时",(SysTime()-old_time)/1000,"s")

参阅： Print

Pause

Pause()

功能： 系统暂停

参数： 无

返回： 无

说明： 程序引起暂停后，通过按键或外部命令继续执行

举例： MOVJ(put , speed , acc , dec , cp)

Open (3)

Delay(300)

Close (3)

Pause()

//当程序运行到此处时,暂停执行

MOVJ(put_up , speed , acc , dec , cp)

参阅： MOVJ , MOVL , MOVC , Open , Close , Pulse

Delay

Delay(time)

功能： 等待一段时间后再向下执行

参数： time 需等待的时间长度,单位 ms

返回： 无

说明： 1、该指令接在运动指令后面会打断此处的连续运动
2、在死循环中插入 Delay(0)可以防止程序执行中卡死

举例： Delay(300) //当程序运行到此处时,等待 300ms 后
再向下执行

参阅： MOVJ , MOVL , MOVC , Do...Loop , Break

Print

Print("print_string")

功能： 打印信息

参数： "print_string" 待打印内容

返回： 无

说明： 打印内容包含所有类型，数据与数据之间用"," 隔开

举例： Print("abc=",abc) //打印结果,abc=3

参阅： StrFormat , SysTime

Break

Break()

功能： 等待上一条的运动指令执行完毕后再执行 Break 之后的指令,防止该指令下一条非 IO 或 Delay 的指令被提前执行

参数： 无

返回： 无

说明：

- 1、该指令接在运动指令后面会打断此处的连续运动
- 2、适用在需要完全到达某个位置点后方能执行下面的操作对应的场景
- 3、Delay 或 IO 操作指令前不需要加 Break()

举例： MOVJ(put , speed , acc , dec , cp)

Break ()

gla = gla +1

//gla 为全局型，涉及到后台其他执行，需保证严格到位后才能计算

Open (3)

Delay(300)

Close (3)

参阅： MOVJ , MOVL , MOVC , Open , Close , Pulse

Power

Power(num, pow_n)

功能： 计算某个数的 n 次方的值

参数： num 底数

pow_n 指数

返回： n 次方结果

说明： 注意定义合理的接收值，一般用 float 类型

举例： float rtn = Power(2,3) //返回值为 8.000000

Abs(Power(2 , 0.5)) //返回值为 2 开平方后的绝对值

参阅： Sqrt

Sqrt

Sqrt(num)**功能：** 计算某个数的开平方值**参数：** num 被求数**返回：** 开平方结果，取正值**说明：** 注意定义合理的接收值，一般用 float 类型**举例：** float rtn = Sqrt(4) //返回值为 2.000000**参阅：** Power

Abs

Abs(abs_num)

功能： 计算某个数的绝对值

参数： abs_num 被求数

返回： 求绝对值后结果值

说明： 注意接收单位定义成合适类型 int 或 float

举例： Abs(-1.23) //结果为 1.23

参阅： Power

Sin

Sin(degree)**功能：** 计算某个角度的正弦值**参数：** degree 角度值**返回：** 正弦值 浮点型**说明：** 注意接收单位定义成合适类型一般为 float**举例：** Sin(30) //结果为 0.5**参阅：** Cos , Tan , Asin , Acos , Atan

Cos

Cos(degree)**功能：** 计算某个角度的余弦值**参数：** degree 角度值**返回：** 余弦值 浮点型**说明：** 注意接收单位定义成合适类型一般为 float**举例：** Cos(60) //结果为 0.5**参阅：** Sin , Tan , Asin , Acos , Atan

Tan

Tan(degree)**功能：** 计算某个角度的正切值**参数：** degree 角度单位,浮点型**返回：** 正切值 浮点型**说明：** 注意接收单位定义成合适类型一般为 float**举例：** Tan(45) //结果为 1.0**参阅：** Sin , Cos , Asin , Acos , Atan

Asin

Asin(value)

功能： 计算某个值的反正弦,返回对应的角度值

参数： value 待计算的值

返回： 正弦值对应的角度,角度单位,浮点型

说明： 注意接收单位定义成合适类型一般为 float

举例： Asin(0.5) //结果为 30

参阅： Sin , Cos, Tan , Acos , Atan

Acos

Acos(value)

功能： 计算某个值的反余弦,返回对应的角度值

参数： value 待计算的值

返回： 余弦值对应的角度,角度单位,浮点型

说明： 注意接收单位定义成合适类型一般为 float

举例： Acos(0.5) //结果为 60

参阅： Sin , Cos, Tan , Asin , Atan

Atan

Atan(value)

功能： 计算某个值的反正切,返回对应的角度值

参数： value 待计算的值

返回： 正切值对应的角度,角度单位,浮点型

说明： 注意接收单位定义成合适类型一般为 float

举例： Atan(1) //结果为 45

参阅： Sin , Cos, Tan , Asin , Acos

RTA

RTA(value)

功能： 弧度值转角度值

参数： value 待计算的值(弧度)

返回： 转换后的角度值，浮点型

说明： 注意接收单位定义成合适类型一般为 float

系统默认内部处理均为角度值，与系统外部对接数据是弧度制的情况下用此函数

举例： float rtn=RTA(3.1415926) //结果为 180

参阅： ATR , Sin , Cos, Tan , Asin , Acos , Atan

ATR

ATR(value)**功能：** 角度值转弧度值**参数：** value 待计算的值(角度)**返回：** 转换后的弧度值，浮点型**说明：** 注意接收单位定义成合适类型一般为 float

系统默认内部处理均为角度值，与系统外部对接数据是弧度制的情况下用此函数

举例： float rtn=ATR(180) //结果为 3.141593**参阅：** RTA , Sin , Cos, Tan , Asin , Acos , Atan

6.2.5 I/O 指令

指令

功能

[Open](#)

打开某个端口,保持输出状态直到被关闭

[Close](#)

关闭某个端口输出

[Pulse](#)

使某个端口在指定时间内输出,然后关闭该端口

[ReadDI](#)

读取一定数量的端口号,返回输入值的状态值,按位编码成 10 进制数值

[ReadDO](#)

读取一定数量的端口号,返回输出的状态值,按位编码成 10 进制数值

[WaitDI](#)

读取输入端口信号状态并判断条件是否成立,条件成立后向下执行,否则一直判断该输入信号

[WaitDI](#)

读取输入端口信号状态并判断条件是否在指定时间内成立,条件成立后向下执行,否则等待计时完成后再向下执行,并返回相应的数值

Open

Open(port_num)

功能： 打开某个端口,保持输出状态直到被关闭

参数： port_num 端口号：0~15

返回： 无

说明： Open 某个端口后一段时间再关闭该端口，如果不需要等待可以用 Pulse 替代；
Open 多个端口，可以用“,”间隔，如 Open(1,2,3);目前一次性 Open 最多 16 个端口；

举例： 例 1

```
For i = 0 To 15 //打开本地所有输出端口
```

Open(i)

Next

例 2

```
Open(1) //打开本地1#输出端口
```

```
Delay( 1000 )
```

```
Close( 1 ) //关闭本地 1#输出端口
```

```
Delay( 1000 )
```

参阅： Close , Pulse , WaitDI , ReadDI , ReadDO , Delay

Close

Close(port_num)

功能： 关闭某个端口输出

参数： port_num 端口号：0~15

返回： 无

说明： Open 某个端口后一段时间再关闭（Close）该端口，如果不需要等待可以用 Pulse 替代；

Close 多个端口，可以用“,”间隔，如 Close(1,2,3);目前一次性 Close 最多 16 个端口；

举例： 例 1

```
For i = 0 To 15 //关闭本地所有输出端口
```

Close(i)

Next

例 2

Open(1) //打开本地 1#输出端口

```
Delay( 1000 )
```

```
Close(1) //关闭本地 1#输出端口
```

```
Delay( 1000 )
```

参阅： Open , Pulse , WaitDI , ReadDI , ReadDO , Delay

Pulse

Pulse(port_num, time)

功能： 使某个端口在指定时间内输出,然后关闭该端口

参数： port_num 输出本地端口号 0~15
time 输出保持时间,单位 ms

返回： 无

说明： 本指令输出的时间和后续指令所耗费的时间重叠,直至输出完成
放在最末一句或正在执行中切换模式、暂停等,信号会按设置时间执行完毕

举例： MOVJ(Pn(1) , speed , acc , dec , cp)

Pulse(3 , 500)

MOVJ(Pn(2) , speed , acc , dec , cp)

案例说明, 运动到 Pn(1)结束后, 打开 3#端口 500ms 后再关闭, 同时开启 Pn(2)点位的运动

备注, 上述执行 Pn(1) 和 Pn(2)之间的光滑转接会被打断, 如果需要轨迹光滑, 请使用并行指令

参阅： Open , Close , WaitDI , ReadDI , ReadDO , Delay , Dn

ReadDI

ReadDI(port_num)

功能： 读取一定数量的端口号,返回输入值的状态值,按位编码成 10 进制数值

参数： port_num 数量多的用" , "符号间隔

返回： 数值

说明：

- 1、 编码顺序按照括号内出现顺序编码,先出现的在低位
- 2、 输入端口可以不按照 0,1,2,3...顺序 , 例如可以 3,1,2,0
- 3、 最多支持 20 个输入点编码

举例： int a = ReadDI(0,1,2,3)

Switch a

Case \$b1010 ...

//常数 10 的 2 进制书写格式,也可以写作 Case 10 , 此时对应
IN0,IN1,IN2,IN3 输入状态分别为
OFF,ON,OFF,ON

EndSwitch

If ReadDI(1) == 0 Then

//如果 IN1 输入状态为 OFF

...

EndIf

参阅： Open , Close , Pulse , WaitDI , ReadDO , Delay

ReadDO

ReadDO(port_num)

功能： 读取一定数量的端口号,返回输出的状态值,按位编码成 10 进制数值

参数： port_num 端口号：数量多的用", "符号间隔

返回： 数值

说明：

- 1、编码顺序按照括号内出现顺序编码,先出现的在低位
- 2、输入端口可以不按照 0,1,2,3...顺序，例如可以 3,1,2,0
- 3、最多支持 20 个输出点编码

举例： int a = ReadDO(0,1,2,3)

Switch a

Case \$b1001

//常数 9 的 2 进制书写格式,也可以写作 Case 9，此时

OUT0,OUT1,OUT2,OUT3 输出状态分别为 ON,OFF,OFF,ON

...

EndSwitch

If ReadDO(1) == 0 Then

//如果 OUT1 输出状态为 OFF

...

EndIf

参阅： Open , Close , Pulse , WaitDI , ReadDI , Delay

WaitDI

WaitDI(port_num, 1)

功能： 读取输入端口信号状态并判断条件是否成立,条件成立后向下执行,否则一直判断该输入信号

参数： port_num 输入端口号 0~15
1 开关状态：0,未接通 1,接通

返回： 无

说明： 如果信号状态始终不满足，将会一直等待下去，如果只需要等待一段时间，参考另一种格式

举例： ...

WaitDI(10 , 1) //等待输入端口 10 接通

WaitDI(8 , 0) //等待输入端口 8 断开

参阅： Open , Close , Pulse , ReadDI , ReadDO , Delay

WaitDI

WaitDI(port_num, 1, time)

- 功能：** 读取输入端口信号状态并判断条件是否在指定时间内成立,条件成立后向下执行,否则等待计时完成后再向下执行,并返回相应的数值
- 参数：**
- | | |
|----------|-----------------|
| port_num | 输入端口号 0~15 |
| 1 | 开关状态：0,未接通 1,接通 |
| time | 等待时间 |
- 返回：**
- | | |
|---|-----------|
| 0 | 在时间内收到信号 |
| 1 | 时间到仍未收到信号 |
- 说明：** 增加等待时间，避免程序一直等待
等待时间不要设置过长，及时增加对返回值的处理
- 举例：** `int a = WaitDI(4,1,1000)` //程序执行到此处等待输入端口 4 接通,如立即接通(<1000ms 内),则程序立即向下执行,a=0;如超过 1000ms 仍未接通,则程序也向下执行,a=1
- 参阅：** Open , Close , Pulse , ReadDI , ReadDO , Delay

6.2.6 通信指令

指令	功能
<u>OpenCom</u>	打开串口
<u>CloseCom</u>	关闭串口
<u>ReadCom</u>	返回串口读取的数据
<u>WriteCom</u>	向串口中写入数据
<u>CheckCom</u>	返回串口接收的字符串数量
<u>ClrCom</u>	清除串口接收缓冲区
<u>SetCom</u>	配置串口信息
<u>OpenNet</u>	打开网口
<u>CloseNet</u>	关闭网口
<u>ReadNet</u>	返回网口读取的数据
<u>WriteNet</u>	向网口中写入数据
<u>CheckNet</u>	返回网口接收的字符串数量
<u>ClrNet</u>	清除网口接收缓冲区
<u>SetNet</u>	配置网口信息
<u>MIRead</u>	读取 int 型 modbus 地址对应的数值
<u>MFRead</u>	读取 float 型 modbus 地址对应的数值
<u>MIWrite</u>	写入 int 型 modbus 地址对应的数值
<u>MFWrite</u>	写入 float 型 modbus 地址对应的数值
<u>GIRead</u>	读取 int 型 modbus 地址对应的数值
<u>GFRRead</u>	读取 float 型 modbus 地址对应的数值
<u>GIWrite</u>	写入 int 型 modbus 地址对应的数值
<u>GFWrite</u>	写入 float 型 modbus 地址对应的数值

OpenCom

OpenCom(com_num)**功能：** 打开串口**参数：** com_num 端口号：1-2**返回：** -2 参数错误

1 打开成功

说明： 通讯参数设置参考通信-串口配置

举例：

```

int sn = 1 //com 号范围(1-2)

int BaudRate=115200 //波特率范围
/*115200,57600,38400,19200,9600,
4800,2400,1200*/

int Databit=8 //数据位长度 8

int Stopbit=1 //停止位长度 1

string Parity="N" //校验选择 ("N"-无奇偶校验)

int res=0

string read_data

SetCom( sn, BaudRate, Databit, Stopbit, Parity)

OpenCom(sn)

Do
/*
res = -2 参数错误
res = -1 串口未打开
*/

res = CheckCom(sn)

If res == -2 Then

Print("parameter error")

Elseif res == -1 Then

```

```
Print("no running opencom")
```

```
Elself res > 0 Then
```

```
//如果以 16 进制收发，将下面的 0 改成 2
```

```
read_data = ReadCom(sn , 0)
```

```
//读取串口 sn 的数据并保存  
read_data 变量中
```

```
WriteCom(sn , 0 ,read_data)
```

```
//把变量 read_data 中的数据通过 sn  
通道发送出去
```

```
//Print(read_data)
```

```
EndIf
```

```
Delay(5)
```

```
Loop
```

参阅： CloseCom , SetCom

CloseCom

CloseCom(com_num)

功能： 关闭串口

参数： com_num 端口号：1-2

返回： -2 参数错误

1 关闭成功

说明： 无

举例： CloseCom(1) //关闭 1#串口

使用方法详见 OpenCom

参阅： OpenCom

ReadCom

ReadCom(com_num)

功能：按端口号返回串口读取的 ASCII 数据(字符串)

参数：com_num 端口号：1-2

返回：读取的字符串

说明：无

举例：recvstr = ReadCom(1) //读取 1#串口接收的数据

使用方法详见 OpenCom

参阅：WriteCom

ReadCom(com_num, code)

功能：按端口号以 ACSII 字符串返回串口读取的数据(ASCII,Octal,Hex 格式可选)

参数：com_num 端口号：1-2
code 编码格式：0 表示 ASCII 1 表示 8 进制(Octal) 2 表示 16 进制(Hex)

返回：读取的字符串

说明：

- 1、ReadCom(1,0) 等同于 ReadCom(1)，表示接收 ASCII 数据
- 2、接收 8 进制、16 进制字节数值时，返回的为字符串，如：对方是以 16 进制发送的 0x00,0x1F,0x01，则 recvstr="00 1F 01"
- 3、使用过程中，请确认发送端的发送格式，如果接收格式不匹配会造成显示乱码或收到协议之外的数据

举例：string recvstr = ReadCom(1,2) //读取 1#串口接收 16 进制的数据

string recvstr = ReadCom(2,0) //读取 2#串口接收 ASCII 数据

使用方法详见 OpenCom

参阅：WriteCom

WriteCom

WriteCom(com_num, str_write)

功能： 向串口中写入数据

参数： com_num 端口号：1-2
str_write 需发送的字符串数据

返回： -2 参数错误
0 发送数据成功
1 发送数据失败

说明： 无

举例： string sendstr = "ABCDEFGH" //定义发送内容
WriteCom(1, sendstr) //向 1#串口发送"ABCDEFGH"字符串
使用方法详见 OpenCom

参阅： ReadCom

WriteCom(com_num, code, str_write)

功能： 根据不同的编码格式向串口中写入数据

参数： com_num 端口号：1-2
code 编码格式：0 表示 ASCII 1 表示 8 进制(Octal) 2 表示 16 进制(Hex)
str_write 需发送的字符串数据

返回： -2 参数错误
0 发送数据成功
1 发送数据失败

说明： 1、 WriteCom(1, 0 , str_write)等同于 WriteCom(1,str_write) 表示按照 ASCII 方式向 COM1#端口写入数据

2、在写入 16 进制数据前，应将 16 进制数据转换成字节形式的字符串，再以字符串发送。若格式不正确会报警格式错误

举例： string sendstr = "ABCDEFGH" //定义发送内容

```
WriteCom(1 , 0 , sendstr)           //向 1#串口发送"ABCDEFGH"字符串  
string str_write = "05 1F 3D 42"    //定义字符串内容是 16 进制格式  
WriteCom(1 , 2 , str_write)         //按 16 进制格式写入 0x05 0x1F  
                                     0x3D 0x42 数据到 1#串口
```

使用方法详见 OpenCom

参阅： ReadCom

CheckCom

CheckCom(com_num)

功能： 返回串口接收的字符串长度或字节数量

参数： com_num 端口号：1-2

返回： >0 返回字符串长度或字节数量

-1 串口未打开

-2 端口正在被其他占用

说明： 1、如：对方以 16 进制发送的 0x00,0x1F,0x01，则返回值为 3，表示共有 3 个字节

2、如：对方以 ACSII 编码发送的“http”，则返回值为 4，表示字符串长度为 4

举例： int chk= 0 //定义接收数量
chk = CheckCom(1) //检查接收数量，可以进一步根据接收数量处理对应的内容

使用方法详见 OpenCom

参阅： ReadCom , WriteCom

ClrCom

ClrCom(com_num)**功能：**清除串口接收缓冲区**参数：**com_num 端口号：1-2**返回：**-2 参数错误**说明：**无**举例：**ClrCom(1) //清空 1#通道缓冲区**参阅：**CheckCom , ReadCom

SetCom

SetCom(com_num, baud_rate, data_bit, stop_bit, parity)

功能： 配置串口信息

参数：

com_num	端口号(数值类型)：1~2
baud_rate	波特率(数值类型)： 115200,57600,38400,19200,9600,4800,2400,1200
data_bit	数据位(数值类型)：5 6 7 8
stop_bit	停止位(数值类型)：1 2
parity	奇偶性(字符串类型)：奇数时指定"O",偶数时指定"E",无校验时指定"N"

返回：

-2	参数错误
1	设置成功

说明： 无

举例： SetCom(1 , 115200 , 8 , 1 , "N") //配置 1#通道串口

使用方法详见 OpenCom

参阅： OpenCom , CloseCom

OpenNet

OpenNet(net_num)**功能：** 打开网口**参数：** net_num 通道号：1~4**返回：** -2 参数错误

1 打开成功

说明： 可以在通信界面上配置或手动操作，实际结果以指令执行为准**举例：** 例 1

```

//设置本地 sn 通道为客户端
string Server_Ip="192.168.1.150" //服务器 IP
int Server_port=5000 //服务器端口号
int sn=1 //连接通道号
int res=0
string read_data //读取数据
SetNet(1,sn,Server_Ip,Server_port) //设置通道 sn 为客户端
OpenNet(sn)
Do
/*
res=-5        初使化
res=-4网线断开
res=-3        未执行 opennet(sn)
res=-2        参数错误
res=-1网口已经打开,但与对方未建立连接
res>=0        数据长度
*/
res = CheckNet(sn)
If res == -5 Then

```

```

Print("Ethernet restarting ")

Elseif res == -4 Then

Print("line out")

Elseif res == -3 Then

Print(" no running opennet funciton")

Elseif res == -2 Then

Print("parameter error")

Elseif res == -1 Then

Print("no ESTABLISHED")

Elseif res > 0 Then                                //如果以 16 进制收发，将下面的 0 改
                                                    成 2

read_data=ReadNet(sn,0)                          //数据保存到 read_data 中

Print(read_data)

WriteNet(sn,0,read_data)                          //向通道 sn 发送 read_data 变量保存
                                                    的数据

EndIf

Delay(5)

Loop

```

例 2

```

                                                    //设置 sn 通道为服务端

string Local_Ip="192.168.1.80"                    //本地 IP 地址

int Local_port=5000                               //本地端口号

int sn=1                                           //通道号

int res=0

string read_data                                  //读取数据

SetNet(0,sn,Local_Ip,Local_port)                  //设置通道 sn 为本地服务端

```

OpenNet(sn)

Do

/*

res=-5 初使化

res=-4 网线断开

res=-3 未执行 opennet(sn)

res=-2 参数错误

res=-1 网口已经打开,但与对方未建立连接

res>=0 数据长度

*/

res = CheckNet(sn)

If res == -5 Then

Print("Ethernet restarting ")

Elseif res == -4 Then

Print("line out")

Elseif res == -3 Then

Print(" no running opennet funciton")

Elseif res == -2 Then

Print("parameter error")

Elseif res == -1 Then

Print("no ESTABLISHED")

Elseif res > 0 Then

//如果以 16 进制收发，将下面的 0 改成 2

read_data=ReadNet(sn,0)

//数据保存到 read_data 中

Print(read_data)

WriteNet(sn,0,read_data)

//向通道 sn 发送 read_data 变量保存的数据

EndIf

Delay(5)

Loop

参阅： CloseNet

CloseNet

CloseNet(net_num)

功能： 关闭网口

参数： net_num 通道号：1~4

返回： -2 参数错误

1 关闭成功

说明： 可以在通信界面上配置或手动操作，实际结果以指令执行为准

举例： CloseNet(1) //关闭 1#通道网络

参阅： OpenNet

ReadNet

ReadNet(net_num)

功能：按通道号返回网口读取的 ASCII 数据(字符串)

参数：net_num 通道号：1-4

返回：读取的字符串

说明：可以在通信界面上配置或手动操作，实际结果以指令执行为准

网络未打开不能使用

举例：recvstr = ReadNet(1) //接收 1#通道网络内容保存到字符串
变量 recvstr 中

使用方法详见 OpenNet

参阅：WriteNet

ReadNet(net_num, code)

功能：按通道号以 ACSII 字符串返回网口读取的数据(ASCII,Octal,Hex 格式可选)

参数：net_num 通道号：1-4
code 编码格式：0 表示 ASCII 1 表示 8 进制(Octal) 2 表示 16 进制(Hex)

返回：读取的字符串

说明：1、ReadNet(1,0) 等同于 ReadNet(1)，表示接收 ASCII 数据
2、接收 8 进制、16 进制字节数值时，返回的为字符串，如：对方是以 16 进制发送的 0x00,0x1F,0x01，则 recvstr="00 1F 01"
3、使用过程中，请确认发送端的发送格式，如果接收格式不匹配会造成显示乱码或收到协议之外的数据

举例：string recvstr = ReadNet(1,2) //读取 1#串口接收 16 进制的数据
string recvstr = ReadNet(2,0) //读取 2#串口接收 ASCII 数据

使用方法详见 OpenNet

参阅：WriteNet

WriteNet

WriteNet(net_num, str_write)

功能： 向网口中写入数据

参数： net_num 通道号：1-4
 str_write 需发送的数据

返回： -2 参数错误
 0 发送数据成功（未打开网口）
 1 发送数据失败

说明： 可以在通信界面上配置或手动操作，实际结果以指令执行为准
 网络未打开不能使用

举例： string sendstr = "ABC" //定义发送内容
 WriteNet(1, sendstr) //向 1#网口通道发送字符串"ABC"

参阅： ReadNet

WriteNet(net_num, code, str_write)

功能： 根据不同的编码格式向网口中写入数据

参数： net_num 通道号：1-4
 code 编码格式：0 表示 ASCII 1 表示 8 进制(Octal) 2 表示 16 进制(Hex)
 str_write 需发送的数据

返回： -2 参数错误
 0 发送数据成功（未打开网口）
 1 发送数据失败

说明： 1、WriteNet(1, 0, str_write)等同于 WriteNet(1,str_write) 表示按照 ASCII 方式向 1#连接通道写入数据

2、在写入 16 进制数据前，应将 16 进制数据转换成字节形式的字符串，再以字符串发送。若格式不正确会报警格式错误

举例： string sendstr = "ABCDEFGF" //定义发送内容

```
WriteNet(1 , 0 , sendstr)           //向 1#串口发送"ABCDEFGH"字符串  
string str_write = "05 1F 3D 42"    //定义字符串内容是 16 进制格式  
WriteNet(1 , 2 , str_write)         //按 16 进制格式写入 0x05 0x1F  
                                     0x3D 0x42 数据到 1#连接通道
```

使用方法详见 OpenNet

参阅： ReadNet

CheckNet

CheckNet(net_num)

功能： 返回网口接收的字符串长度或字节数量

参数： net_num 通道号：1-4

返回：

>0	返回字符串长度或字节数量
-1	网口已经打开,但与对方未建立连接
-2	参数错误
-3	未执行 opennet(sn)
-4	网线断开
-5	初始化

说明：

- 1、可以在通信界面上配置或手动操作，实际结果以指令执行为准
- 2、网络未打开不能使用
- 3、如：对方以 16 进制发送的 0x00,0x1F,0x01，则返回值为 3，表示共有 3 个字节
- 4、如：对方以 ACSII 编码发送的“http”，则返回值为 4，表示字符串长度为 4

举例：

```
int chk= 0 //定义接收数量
chk = CheckNet(1) //检查接收数量，可以进一步根据接收数量处理对应的内容
```

使用方法详见 OpenNet

参阅： ReadNet , WriteNet

ClrNet

ClrNet(net_num)**功能：**清除网口接收缓冲区**参数：**net_num 通道号：1-4**返回：**无**说明：**可以在通信界面上配置或手动操作，实际结果以指令执行为准

网络未打开不能使用

举例：ClrNet(1) //清空 1#通道缓冲区**参阅：**ReadNet , WriteNet

SetNet

SetNet(type, net_num, IP, port)

功能：配置网口信息

参数： type 通道类型：0:服务端 1：客户端

net_num 通道号：1-4

IP ip 地址：“192.168.1.150”

port 端口号：1000~65535

返回： -2 参数错误

1 设置成功

说明：

- 1、可以在通信界面上配置或手动操作，实际结果以指令执行为优先；
- 2、例如界面上配置为 A，SetNet 指令设置为 B，在指令执行期间和以后按照 B 配置来执行；
- 3、本机为服务端时，502,8080 端口号不要使用

举例： SetNet(1, 1, "192.168.1.150", 5000) //本机 1#通道设置为客户端，设置对方服务器 IP 和端口号

使用方法详见 OpenNet

参阅： OpenNet, CloseNet

MIRead

MIRead(I_num)

功能： 读取 Int 型 modbus 地址对应的数值

参数： I_num I 变量编号：I0~I99

返回： int 型数值

说明： 可以在变量界面上查看或设置相关数值；

不支持掉电保存；

举例： MIRead(1) //读取 int 型 modbus 编号为 I1 对应的数值，等效于 I1

参阅： MRead

MFRead

MFRead(F_num)

功能： 读取 float 型 modbus 地址对应的数值

参数： F_num F 变量编号：F0~F99

返回： float 型数值

说明： 可以在变量界面上查看或设置相关数值；

不支持掉电保存；

举例： MFRead(1) //读取 float 型 modbus 编号为 I1 对应的数值，等效于 F1

参阅： MRead

MIWrite

MIWrite(I_num, intValue)

功能： 写入 Int 型 modbus 地址对应的数值

参数： I_num I 变量编号：I0~I99

intValue 32 位带符号整数

返回： 无

说明： 可以在变量界面上查看或设置相关数值；

不支持掉电保存；

举例： MIWrite(1,-100) //将 100 写入 int 型 modbus 编号为 I1 对应的地址，等效于 I1 = -100

参阅： MFWrite

MFWrite

MFWrite(F_num, floatValue)

功能： 写入 float 型 modbus 地址对应的数值

参数： F_num F 变量编号：F0~F99

floatValue 32 位浮点数

返回： 无

说明： 可以在变量界面上查看或设置相关数值；

不支持掉电保存；

举例： MFWrite(1,123.456) //将 123.456 写入 float 型 modbus
编号为 F1 对应的地址，等效于 F1 =
123.456

参阅： MIWrite

GIRead

GIRead(I_num)

功能： 读取 Int 型全局变量地址对应的数值

参数： I_num GI 变量编号：GI0~GI49

返回： int 型数值

说明： 可以在变量界面上查看或设置相关数值；

支持掉电保存；

支持 modbus 读写；

举例： GIRead(1) //读取 int 型全局变量编号为 GI1 对应的数值，等效于 GI1

参阅： GFRead

GRead

GRead(F_num)

功能： 读取 float 型全局变量地址对应的数值

参数： F_num GF 变量编号：GF0~GF49

返回： float 型数值

说明： 可以在变量界面上查看或设置相关数值；

支持掉电保存；

支持 modbus 读写；

举例： GRead(1) //读取 float 型全局变量编号为 GF1 对应的数值，等效于 GF1

参阅： MRead

GIWrite

GIWrite(l_num, intValue)

功能： 写入 Int 型全局地址对应的数值

参数： l_num GI 变量编号：GI0~GI49

intValue	32 位带符号整数
----------	-----------

返回： 无

说明： 可以在变量界面上查看或设置相关数值；

支持掉电保存；

支持 modbus 读写；

举例： GIWrite(1,-100) //将 100 写入 GI1 对应的地址，等效于 GI1 = -100

参阅： MFWWrite

GFWrite

GFWrite(F_num, floatValue)

功能： 写入 float 型 modbus 地址对应的数值

参数： F_num GF 变量编号：GF0~GF49

floatValue 32 位浮点数

返回： 无

说明： 可以在变量界面上查看或设置相关数值；

支持掉电保存；

支持 modbus 读写；

举例： GFWrite(1,123.456) //将 123.456 写入 GF1 对应的地址，
等效于 GF1 = 123.456

参阅： MIWrite

6.2.7 字符串指令

指令

功能

[StrLen](#)

统计并返回字符串总字符数量

[StrLeft](#)

返回字符串左侧指定数目的字符串

[StrRight](#)

返回字符串右侧指定数目的字符串

[StrMid](#)

返回字符串中间指定数目的字符串

[StrTrim](#)

去掉字符串中头部和尾部指定内容，返回剩余字符串

[StrFind](#)

查找指定字符串在被查找字符串中的位置，并返回此位置

[StrReplace](#)

先查找指定字符串在被查找字符串中的位置，再替换此位置的字符串

[StrEmpty](#)

清空字符串变量的值

[StrCat](#)

将多个字符串拼接为 1 个字符串

[StrFormat](#)

将多个数值按格式拼接为 1 个字符串

[StrData](#)

将 1 个字符串按格式分割为多个数值

[StrSlicer](#)

将 1 个字符串按指定字符串分割为多个字符串

[StrCmp](#)

比较两个字符串大小

[StrnCmp](#)

比较两个字符串从左侧起指定长度的内容大小

[StrToupper](#)

将字符串中小写字母转为大写

[StrTolower](#)

将字符串中大写字母转为小写

[StrToString](#)

将数字值内容转为字符串

[StrToValue](#)

将字符串内容转为数字值

[StrAscValue](#)

获取字符串首字符的 ASCII 值

[StrAscChar](#)

获取 ASCII 值对应的字符

StrLen

StrLen(str)

功能： 统计并返回字符串总字符数量

参数： str 待统计字符串

返回： 数值

说明： 数值从 1 开始

举例： string str = "abcdefg"

int cnt

cnt = StrLen(str)

Print("str1 包含字符数量为 " , cnt) //打印结果为 7

参阅： StrLeft , StrRight , StrMid , StrFind , StrTrim , Print

StrLeft

StrLeft(str, n)

功能： 返回字符串左侧指定数目的字符串

参数： str 待处理字符串
n 需要返回的字符数量

返回： 字符串

说明： 从左侧第 1 个开始，此操作不改变被操作字符串

举例： string str = "abcdefg" , str1
str1 = StrLeft(str , 3)
Print("str1 = " , str1) //str 的内容为 abc

参阅： StrLen , StrRight , StrMid , StrFind , StrTrim , Print

StrRight

StrRight(str, n)

功能： 返回字符串右侧指定数目的字符串

参数： str 待处理字符串
n 需要返回的字符数量

返回： 字符串

说明： 从右侧第 1 个开始

举例： string str = "abcdefg" , str1
str1 = StrRight(str , 3)
Print("str1 = " , str1) //str 的内容为 efg

参阅： StrLeft , StrLen , StrMid , StrFind , StrTrim , Print

StrMid

StrMid(str, n1, n2)

功能： 返回字符串中间指定位置的字符串

参数：

str	待处理字符串
n1	在字符串中开始的位置
n2	在字符串中结束的位置

返回： 字符串

说明： 从字符串第 1 个字符开始计数为 1

举例：

```
string str = "abcdefg", str1
str1 = StrMid( str , 3 , 4 )
Print( "str1 = ", str1 )           //str 的内容为 cd
```

参阅： StrLeft , StrRight , StrLen , StrFind , StrTrim , Print

StrTrim

StrTrim(str1, str2, str3)

功能： 去掉字符串中头部和尾部指定内容，返回剩余字符串

参数：

str1	待处理字符串
str2	头部需去除的字符串
str3	尾部需去除的字符串

返回： 字符串

说明： 不需要去除的部分用""空值表示

举例：

```
string srcStr="Image[X:123.123;Y:234.234;A:345.345]ImageDone"
string begins="Image"
string ends="ImageDone"
temps = StrTrim ( srcStr , begins , ends ) //temps=[X:123.123;Y:234.234;A:345.345]
```

参阅： StrLeft , StrRight , StrMid , StrFind , StrLen , Print

StrFind

StrFind(str1, str2, n)

功能： 查找指定字符串在被查找字符串中的位置，并返回此位置

参数： str1 待查找字符串

str2 关键字字符串

n 指定第几次出现

返回： 位置值

说明： n 的值从 1 开始，表示从待查找第 1 次出现的位置

举例： charstart = StrFind (temps , "[" , 1)

//temps=[X:123.123;Y:234.234;A:34
5.345],运行后 charstart=1

参阅： StrLeft , StrRight , StrMid , StrLen , StrTrim , Print

StrReplace

StrReplace(str, str1, str2, n)

功能： 先查找指定字符串在被查找字符串中的位置，再替换此位置的字符串

参数： str 被替换的字符串变量，并作为操作后储存的目标

str1 待查找字符串

str2 替换值

n 指定第几次出现的时候替换，[缺省]表示全部替换

返回： 无

说明： n 的值从 1 开始，表示从待查找第 1 次出现的位置

举例： StrReplace (temps , "abc" , "") //temps 的值 123abc456,运行后
temps 的值为 123456

参阅： StrFind , Print

StrReplace

StrReplace(str, str1, str2)

功能： 先查找指定字符串在被查找字符串中的位置，再替换此位置的字符串

参数： str 被替换的字符串变量，并作为操作后储存的目标

str1 待查找字符串

str2 替换值

n 指定第几次出现的时候替换，[缺省]表示全部替换

返回： 无

说明： n 的值从 1 开始，表示从待查找第 1 次出现的位置

举例： StrReplace (temps , "abc" , "") //temps 的值 123abc456,运行后
temps 的值为 123456

参阅： StrFind , Print

StrEmpty

StrEmpty(str)**功能：** 清空字符串变量的值**参数：** str 待处理字符串**返回：** 无**说明：** 此操作无返回值，并改变被操作变量内容**举例：** string str = "abcdefg"

int cnt

StrEmpty(str)

cnt = StrLen(str)

Print("str = " , str , " , cnt = " ,cnt) //str =,cnt=0

参阅： StrReplace , Print

StrCat

StrCat(str1, str2, ...)

功能： 将多个字符串拼接为 1 个字符串

参数： str1 待拼接字符串 1

str2 待拼接字符串 2

... 待拼接字符串...

返回： 字符串

说明： 按照先后出现在参数中的顺序拼接

举例： string str1 = "abc" , str2 = "defg" , str

str = StrCat(str1 , str2)

Print("str = " , str) //str = abcdefg

参阅： StrFormat , Print

StrFormat

StrFormat(format, n1, n2, ...)

功能： 将多个数值按格式拼接为 1 个字符串

参数： format 格式，详见说明

n1 第 1 个数值

n2 第 2 个数值

... 第...个数值

返回： 字符串

说明： 格式中填写想要转出的字符样式，如 string format="X:%f;Y:%f;A:%f"

其中%f 为数值对应的格式

常用的数值格式表示，浮点型为型%f,整数型%d,16 进制%x,2 进制%b,浮点型%s

特别的，当数值类型是浮点型，可以指定显示位数，如%3.3f，表示 xxx.xxx

举例： Print (StrFormat(" charstart= %d, charnum= %d", charstart , charnum), "\n")
//打印内容 charstart= 1, charnum= 31

参阅： StrCat , StrData , StrSlicer , Print

StrData

StrData(str, format, n1, n2, ...)

功能： 将 1 个字符串按格式分割为多个数值

参数： str 待处理字符串

format 格式

n2 第 2 个数值

... 第...个数值

返回： 无

说明： 格式中填写想要转出的字符样式，如 string format="X:%f;Y:%f;A:%f"

其中%f 为数值对应的格式

常用的数值格式表示，浮点型为%f,整数型%d,16 进制%x,2 进制%b,浮点型%s

举例： string format="[X:%f;Y:%f;A:%f]"

StrData(s1, format , x , y , angle) //s1 内容为
[X:123.123;Y:234.234;A:345.345]

Print(x , y , angle) //运行后 x=123.123000
y=234.234000 angle=345.345000

参阅： StrFormat , StrSlicer , Print

StrSlicer

StrSlicer(str, str1, n, str2)

功能： 将 1 个字符串按指定字符串分割为多个字符串

参数：

str	待处理字符串
str1	分隔符，也可以是字符串
n	指定区块，[缺省]全部保存
str2	分割后保存变量

返回： 无

说明： 当分割结果大于 1 的情况，指定字符串数组的下标作为开始储存，可以防止分割后的多数据没有被存储

举例：

```
string temps , s1 , s2 , strtemp_arr[3]
StrSlicer ( s2 , ";" , 1, strtemp_arr[0] )
StrSlicer ( s2 , ";" , 2, strtemp_arr[1] )
StrSlicer ( s2 , ";" , 3, strtemp_arr[2] )
或
string temps , s1 , s2 , strtemp_arr[3]
StrSlicer ( s2 , ";" strtemp_arr[0] )
```

```
//s2 内容为
X:123.123;Y:234.234;A:345.345

//运行后

//strtemp_arr[0] ="123.123"
//strtemp_arr[1] ="234.234"
//strtemp_arr[2] ="345.345"
```

参阅： StrFormat , StrData , Print

StrSlicer

StrSlicer(str, str1, str2)

功能： 将 1 个字符串按指定字符串分割为多个字符串

参数：

str	待处理字符串
str1	分隔符，也可以是字符串
n	指定区块，[缺省]全部保存
str2	分割后保存变量

返回： 无

说明： 当分割结果大于 1 的情况，指定字符串数组的下标作为开始储存，可以防止分割后的多数据没有被存储

举例：

```
string temps , s1 , s2 , strtemp_arr[3]
StrSlicer ( s2 , ";" , 1, strtemp_arr[0] )
StrSlicer ( s2 , ";" , 2, strtemp_arr[1] )
StrSlicer ( s2 , ";" , 3, strtemp_arr[2] )
或
string temps , s1 , s2 , strtemp_arr[3]
StrSlicer ( s2 , ";" strtemp_arr[0] )
```

```
//s2 内容为
X:123.123;Y:234.234;A:345.345

//运行后

//strtemp_arr[0] ="123.123"
//strtemp_arr[1] ="234.234"
//strtemp_arr[2] ="345.345"
```

参阅： StrFormat , StrData , Print

StrCmp

StrCmp(str1, str2)

功能： 比较两个字符串大小

参数： str1 字符串 1

str2 字符串 2

返回： 相差值

0 相等

>0 字符串 1 > 字符串 2

<0 字符串 1 < 字符串 2

说明： 比较的是首字符大小

举例： string temps = "[X:123.123;Y:234.234;A:345.345]"

Do While StrCmp (temps , "") != 0

Delay(50)

Loop

//比较后结果为 91!=0 条件成立进入循环，备注："["字符 ascii 码值为 91

参阅： Do While...Loop , Print

StrnCmp

StrnCmp(str1, str2, n)

功能： 比较两个字符串从左侧起指定长度的内容大小

参数： str1 字符串 1

str2 字符串 2

n 从左侧起指定长度的内容

返回： 相差值

0 相等

>0 字符串 1 > 字符串 2

<0 字符串 1 < 字符串 2

说明： 比较的是多个字符大小

举例： string a="abcde", b="abccdefg"

int i, j

i = StrnCmp(a , b , 3) //abc=abc,结果差值为 0

j = StrnCmp(a , b , 4) //abcd>abcc,结果差值为 1

Print ("i=", i, ", ", "j=", j)

参阅： Print

StrToupper

StrToupper(str)

功能： 将字符串中小写字母转为大写

参数： str 字符串

返回： 字符串

说明： 无

举例： string a="AbcD1e" , s
s = StrToupper(a)
Print(s) //s="ABCD1E"

参阅： Print

StrTolower

StrTolower(str)

功能： 将字符串中大写字母转为小写

参数： str 字符串

返回： 字符串

说明： 无

举例： string a="AbcD1e" , s

s = StrTolower(a)

Print(s) //s="abcd1e"

参阅： Print

StrToString

StrToString(n)

功能： 将数字值内容转为字符串

参数： n 待处理字符串

返回： 字符串

说明： 数值可以为整数型和浮点型

举例： string str

float fx = 345.2

str = StrToString(fx)

Print("str = " , str) //str =345.2

参阅： Print

StrToValue

StrToValue(str)

功能： 将字符串内容转为数字值

参数： str 待处理字符串

返回： 数值

说明： 数值可以为整数型和浮点型

该操作不会改变原字符串内容

举例： string str = "345.256"

float fx

fx = StrToValue(str)

Print("fx = ", fx) //fx=345.256000

如果不知道源字符串是否可以转成数值，可以用以下自定义函数判断

```
Function IsDigit(string s)
```

```
//测试字符串是否可转为数字
```

```
int i,asc,flag=0
```

```
If StrLen(s) < 1 Then
```

```
Return 0                      //无数据
```

```
EndIf
```

```
For i = 1 To StrLen(s)
```

```
asc = StrAscValue(StrMid(s,i,1))
```

```
If asc == 46 Then              //
```

```
flag = flag + 1
```

```
If i == 1 Or flag > 1 Then
```

```
Return 0
```

```
EndIf
```

```
Elseif asc < 48 Or asc > 57 Then
```

```
Return 0
```

Endlf

Next

Return 1

FunctionEnd

参阅： StrToString , StrData , Print

StrAscValue

StrAscValue(str)

功能： 获取字符串首字符的 ASCII 值

参数： str 字符串

返回： 数值

说明： 首字符的 ASCII 值

举例： int a = StrAscValue("12ax")
Print(a) //a = 49 ,"1"的 ASCII 数值 49

参阅： Print

StrAscChar

StrAscChar(n)

功能： 获取 ASCII 值对应的字符

参数： n 数值, 0-128

返回： 字符

说明： 不要将值设置超出边界

举例： string a = StrAscChar(49)
Print(a)//a = "1" , "1"的 ASCII 数值为 49

参阅： Print

6.2.8 其他指令

指令

功能

[int](#)

定义变量类型为整形变量

[float](#)

定义变量类型为浮点型变量

[string](#)

定义变量类型为字符串型变量

[point](#)

定义变量类型为点位型变量

[Function](#)

定义一个方法（功能块/函数），可以在使用时调用

[ServoOn](#)

用于将机器人的电机使能设为 ON

[ServoOff](#)

用于将机器人的电机使能设为 OFF

[GetServoStatus](#)

返回机器人电机使能状态

[PowerOn](#)

启动机器人主程序（任务 1）

[PowerOff](#)

停止机器人主程序（任务 1）

[SysHalt](#)

暂停机器人主程序（任务 1）

[SysReset](#)

复位机器人主程序（任务 1）

[GetSysStatus](#)

返回机器人的工作状态（任务 1）

[GetError](#)

返回机器人的报警序号

[SetSysRatio](#)

设置机器人主程序（任务 1）的运行倍率

[GetSysRatio](#)

获取机器人主程序（任务 1）的运行倍率

int

int

功能： 定义变量类型为整形变量

参数： 无

返回： 无

说明： int 后面需要有空格，如果用多个同样类型的变量，请用“，”隔开

举例： int a = 100 , b = -1 , c

参阅： float , string , point

float

float

功能： 定义变量类型为浮点型变量

参数： 无

返回： 无

说明： float 后面需要有空格，如果用多个同样类型的变量，请用“，”隔开

举例： float af = 1.23 , bf = -1.45 , cf

参阅： int , string , point

string

string

功能： 定义变量类型为字符串型变量

参数： 无

返回： 无

说明： string 后面需要有空格，如果用多个同样类型的变量，请用“，”隔开

举例： float as = "abc", bs = "a:", cs

参阅： int , float , point

point

point

功能： 定义变量类型为点位型变量

参数： 无

返回： 无

说明： point 后面需要有空格，如果用多个同样类型的变量，请用“，”隔开；

一般 point 某变量后可以直接赋值进来；

举例： point posa = Pn(1) , posb = Where() , posc

参阅： int , float , string

Function

Function Name().....FunctionEnd

功能： 定义一个方法（功能块/函数），可以在使用时调用

参数： 根据需要配置

返回： 根据需要返回

说明： 如果要定义的方法带参数的用"()"表示，"(类型 1 变量名 1,类型 2,变量名 2...)",该参数在调用方法时作为可变参数输入；

如果要定义的方法不带参数时，也需要增加括号；

一般 point 某变量后可以直接赋值进来；

举例： Function myName()

...

FunctionEnd

参阅： int , float , string , point

ServoOn

ServoOn()

功能： 用于将机器人的所有轴的电动机设为 ON

参数： 无

返回： 无

说明： ServoOn()用于将电动机电源设为 ON（上使能），并解除所有轴的制动；

举例： If GetServoStatus() == 0 Then

ServoOn() //打开所有轴的使能

EndIf

参阅： GetServoStatus , ServoOff

ServoOff

ServoOff()

功能： 用于将机器人的所有轴的电动机设为 OFF

参数： 无

返回： 无

说明： ServoOff()用于将电动机电源设为 OFF（断使能），并恢复所有轴的制动（如果有）；

举例： ServoOff() //关闭所有轴的使能

参阅： ServoOn

GetServoStatus

GetServoStatus()

功能： 返回机器人的所有轴的电动机使能状态

参数： 无

返回： 0: 当前全部轴未使能
1: 当前全部轴已使能

说明： 用于需要反馈电动机使能状态的情况，例如外部控制设备需要了解机器人使能状态的场景；
或者用于自身需要控制电机使能的情形。

举例： If GetServoStatus() == 0 Then
ServoOn() //打开所有轴的使能
EndIf

参阅： ServoOn , ServoOff

PowerOn

PowerOn()

功能： 启动机器人主程序（任务1）

参数： 无

返回： 无

说明： 用于后台通信任务控制主程序启动；

举例： If StrCmp(sdata[0], "STARTRUN") == 0 Then

//STARTRUN

PowerOn()

EndIf

参阅： PowerOff , SysHalt , SysReset

PowerOff

PowerOff()

功能： 停止机器人主程序（任务 1）

参数： 无

返回： 无

说明： 用于后台通信任务控制主程序停止；

举例： If StrCmp(sdata[0], "STOP") == 0 Then

//STOP

PowerOff()

EndIf

参阅： PowerOn , SysHalt , SysReset

SysHalt

SysHalt()

功能： 暂停机器人主程序（任务1）

参数： 无

返回： 无

说明： 用于后台通信任务控制主程序暂停；

举例： If StrCmp(sdata[0], "HALT") == 0 Then

//HALT

SysHalt()

EndIf

参阅： PowerOn , PowerOff , SysReset

SysReset

SysReset()

功能： 复位机器人主程序（任务1）

参数： 无

返回： 无

说明： 用于后台通信任务控制主程序复位（后台程序不做复位）；

举例： If StrCmp(sdata[0], "RESET") == 0 Then

//RESET

SysReset()

EndIf

参阅： PowerOn , PowerOff , SysHalt

GetSysStatus

GetSysStatus()

功能： 返回机器人的任务 1 的工作状态

参数： 无

返回：

0：	复位中
1：	急停中
2：	报警中
3：	就绪/空闲
4：	运行中
5：	暂停中
6：	单步运行中
7：	单步暂停

说明： 用于需要反馈主任务运行状态的情况，例如外部控制设备需要了解机器人使能状态的场景；
或者用于自身需要控制机器人运行状态的情形。

举例：

```
If getsystem == 1 And GetSysStatus() == 4 Then  
  WriteNet(sn,"STARTRUN,SUCCESS")  
  getsystem = 0  
EndIf
```

参阅： PowerOn , PowerOff , SysHalt , SysReset , GetError

GetError

GetError()

功能： 返回机器人的报警序号

参数： 无

返回： 0： 无报警
 >0: 具体的报警序号 (0~65535 之间的整数)

说明： 用于查询报警状态时具体的报警序号。

举例： If GetError() > 0 Then
 If SysTime()-timer1 > 2000 Then
 WriteNet(sn,StrCat("ERROR,",StrToString(GetError())))
 timer1 = SysTime()
 EndIf
 EndIf

参阅： GetServoStatus

SetSysRatio

SetSysRatio(ratio)

功能： 设置机器人主程序（任务1）的运行倍率

参数： ratio 范围：1~100

返回： 无

说明： 系统会按照就近原则选择靠近的档位；通常用于 socket 通信控制。

举例：

```
If IsDigit(sdata[1]) == 1 Then           //IsDigit()为自定义功能函数，检查是否
                                         是否为数值型字符串
                                         //用户设置的倍率 Ratio
    SetSysRatio(StrToValue(sdata[1]))
Else
    IO = 1000                           //1000 上位机发送数值出错
EndIf
```

参阅： GetSysRatio

GetSysRatio

GetSysRatio()

功能： 获取机器人主程序（任务1）的运行倍率

参数： 无

返回： 范围：1~100

说明： 系统会按照就近原则靠近的档位；通常用于 socket 通信控制。

举例：

```
If StrCmp( sdata[0], "GETSYSRATIO") == 0 Then

    WriteNet(sn,StrCat("GETSYSRATIO,",StrToString(GetSysRatio()),",SUCCESS"
))

EndIf
```

参阅： SetSysRatio

第七章 通讯功能介绍

7.1 通讯功能简介

MDCS-510 系统提供串口 (RS-485) 和以太网 (Ethernet) 两种物理接口方式通讯。详细使用功能如下表：

接口	说明					
RS-485	端口位置	系统接口面板上标识为 RS-485 的 DB-9				
	接线定义	编号	类型	引脚 A	引脚 B	引脚 C
		COM1	RS485	1#：485+	6#：485-	-
		COM2	RS232	2#：232RX	3#：232TX	5#GND·
	功能模式	串口通讯	自由协议		端口	COM1，COM2
			Modbus-RTU		端口	COM1
		远程控制	Modbus-RTU		端口	COM1
	Modbus-RTU 配置	功能	串口通讯或远程控制：		I/O 扩展	
		工作方式	从站(slave) 系统预设，用户无需配置		主站(master) 暂无主站功能	
		站地址	系统地址号：1 内置，通过高级参数 159#配置		暂无主站功能	
出厂参数	波特率：115200，数据位：8，停止位：1，校验：无					
Ethernet	端口数量	1 个，备注：多路连接需交换机设备				
	连接数量	4 路（自由协议）+2 路（Modbus 协议）				
	功能模式	网口通信	自由协议		4 路	
			Modbus-TCP		2 路（内置）	
		远程控制	Modbus-TCP		2 路内置，与网口通信共用	
	Modbus-TCP 配置	工作方式：从站(slave) 站地址：1（默认），可以通过 159#高级参数更改，范围：1~250				
	出厂参数	IP：192.168.1.80		从站地址：1		

7.2 串口 (RS-485) /EXTIO 使用方法

7.2.1 接线

MDCS-510 提供如下串口：

一路（COM1）为 RS-485 两线制接线方式，最高支持 115200 波特率的传输速率。

一路（COM2）为 RS232 模式，用户需要将 RX,TX,GND 按端口配置接好。

用户在使用前先按照<7.1 通讯功能简介>中提供的接线方式和所需求的功能模式，将设备与系统的通讯线接在一起。



注意

接线时务必确认好 485+和 485-/232RX 和 232TX 的线序，否则可能引起通信故障！

7.2.2 自由协议模式（COM1/COM2）

1) 打开通讯界面-串口配置；

通信监视-串口配置

本地端口	波特率	数据位	停止位	校验	协议	连接状态
COM1	115200	8	1	None	自由协议	已断开
COM2	115200	8	1	None	Modbus	已连接
COM3	115200	8	1	None	自由协议	已断开
COM4	115200	8	1	None	自由协议	已断开
COM5	115200	8	1	None	自由协议	已断开
COM6	115200	8	1	None	自由协议	已断开

图 7-2-2-1 串口通信设置界面

2) 通信参数设定

包含波特率，数据位，停止位，校验方式等设置值，如果有不一样的地方，修改任何一方的参数均可，使双方参数一致，COM1 设置为自由协议（默认）。

3) 保存配置

点击保存立即生效。

用户在点保存之前，可以手动打开，测试收发数据；也可以不打开，由程序命令控制是否打开；

4) 串口通信的测试方法

手动测试可以使用通信监视窗口，如图 7-2-2-2；用指令运行时（自动）测试通讯结果，可用打印指令配合打印窗口显示；



图 7-2-2-2 通信监视窗口

7.2.3 Modbus-RTU 模式 (COM1)

- 1) 新版本已经默认开启 Modbus 功能，用户可以跳过此步骤（主控版本号 V1.28+ ）。
- 2) 进入[通讯]-[串口配置]-选择[COM1]
设置通讯[波特率]、[数据位]、[停止位]、[校验]，[协议]→[Modbus 从]；
- 3) 点击→[打开]，开始生效；
备注：打开后立即生效，外部主站可与系统立即通讯，并且在重新开机后依然生效，直到用户按下→[关闭]按钮；
- 4) 点击→[保存]；

 提示	务必将通信双方的通信参数设置成相同的数值，否则可能引发数据发送、接收异常！
 提示	在进行任何配置后，请保存参数，以防重新开机后不生效; 有关本机通讯地址信息参看附录 B Modbus 通信地址表

7.2.4 扩展 I/O 的设置 (EXTIO)

1) 设置 IO 配置参数

选择设置→IO 配置，在扩展 I/O 数量设定下侧的方框中选择需要扩展的 I/O 板数量，范围从 0~4，其中 0 表示不使用；

设置-IO配置预览

扩展I/O数量设定

-

0

+

导入

导出

输入	备注(字符串长度<10)	输出	备注(字符串长度<10)	急停后复位
IN0		OUT0		☐
IN1		OUT1		☐
IN2		OUT2		☐
IN3		OUT3		☐
IN4		OUT4		☐
IN5		OUT5		☐
IN6		OUT6		☐
IN7		OUT7		☐

上一页

下一页

清除

保存

返回

点保存键退出，系统在保存后会自动设置完成，重新启动操作系统后生效。

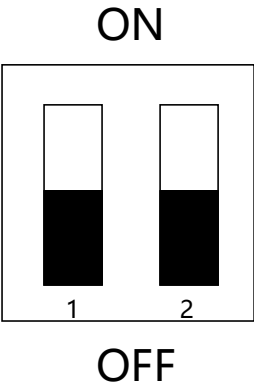
i

提示

MDCS-510 的扩展 I/O 接口从 DCS-415 的 COM1 口转移到 I/O 专用扩展接口 EXTIO 上，接线方法详见 3.4 EXTIO 端口接线定义

2) 扩展模块站号介绍

设置扩展模块的拨码（从站号），如果只使用 1 块扩展板，拨码分别调整到 OFF 位，表示当前设置的站号是 1，如果有多块扩展板对应的，2.3.4 站按照如下设置。



拨码 1	拨码 2	站号	I/O 序号范围(IN/OUT)
OFF	OFF	1	16~31
OFF	ON	2	32~47
ON	OFF	3	48~63
ON	ON	4	64~79

备注：I/O 编号计算方法： $16 * \text{站号} + 0 \sim 15$ （扩展板内编号）

拨码设置完毕后，扩展模块板需断电重启之后才生效。

3) 拨码开关所在位置

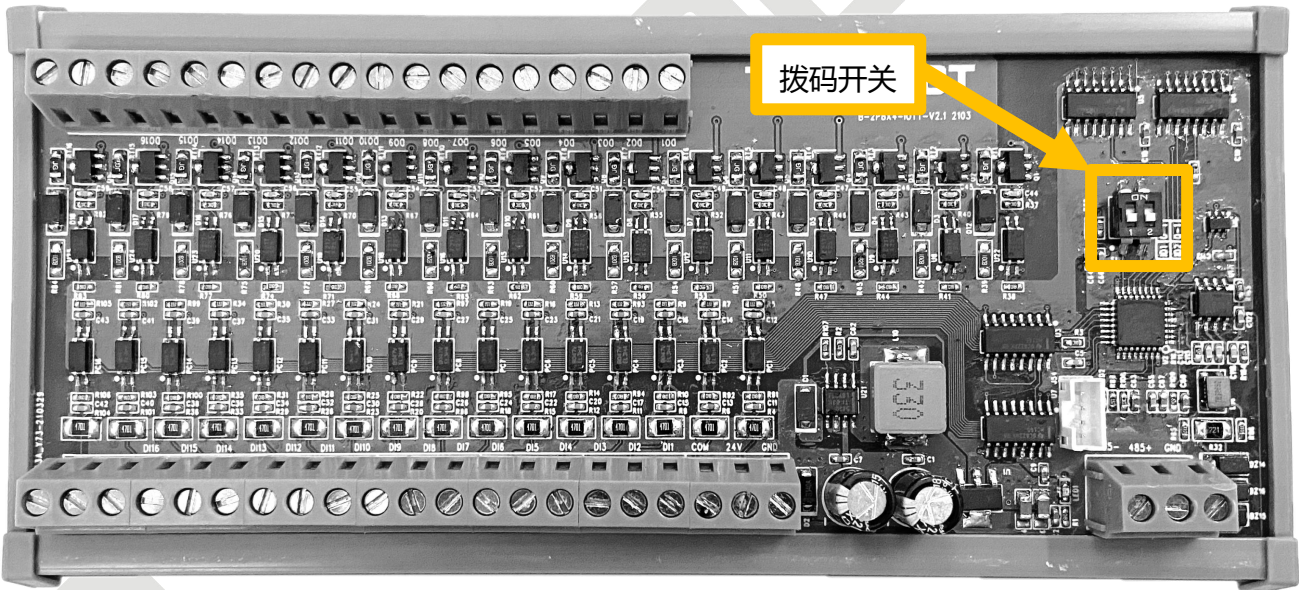


图 7-2-2-3 I/O 扩展板平面图

4) 扩展 I/O 板用户接线

4.1) 扩展 I/O 板与控制系统通信接线

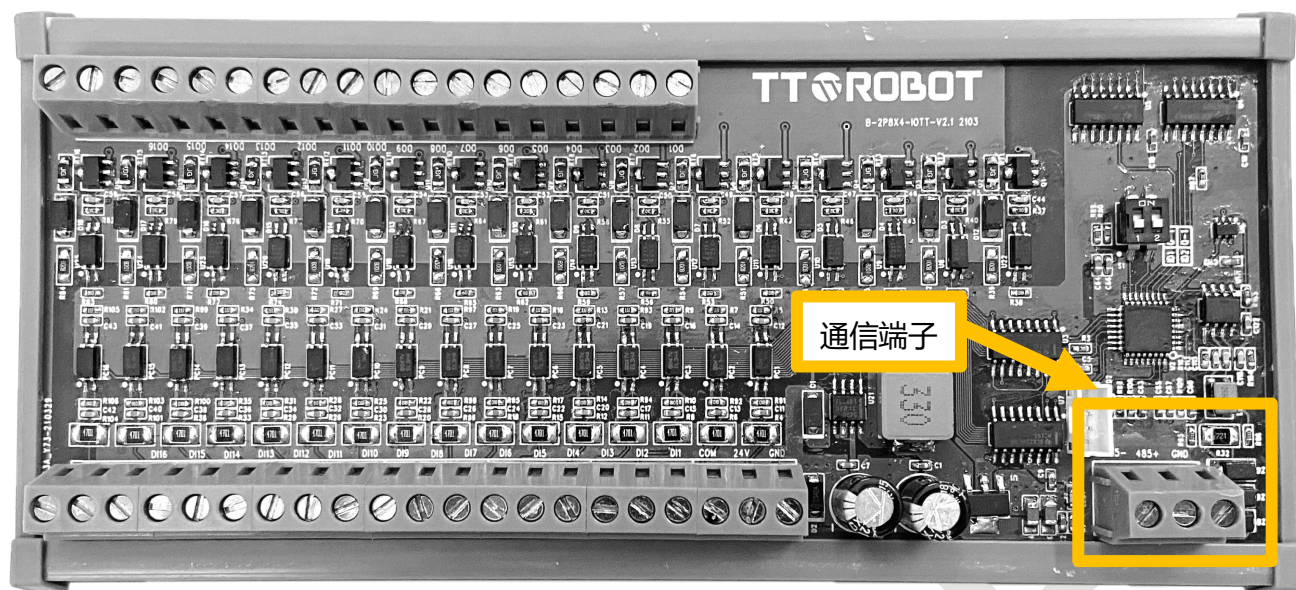


图 7-2-2-4 I/O 扩展板平面图

如上图所示，用户需准备 2 根电线，线径 0.2mm^2 或更大，最好是双绞线，也可以用直连线替代，长度不超过 2m；

一端连接控制箱 COM1 端口，需要焊接随机附送的 DB9（公头），另一端 485+ 接中间（标识 485+/A），485- 接左边（标识 485-/B）；

4.2) 接用户线

与系统自带 I/O 接线方法类似，详情参考 3.5 扩展 I/O 板接线定义（选件）。

4.3) 使用说明

举个例子，用 OUT16----输出单线圈型电磁阀，将 24V 接到电磁阀正极，将电磁阀 0V 接到输出侧第 1 个端子（标识为 0 号的引脚）。

接好之后，打开示教器 I/O—扩展 I/O—手动输出 OUT16，观察电磁阀是否动作；

4.4) 指示灯说明：

V2.2 版的扩展 I/O 板内带指示灯，可以直观反映各通道信号输入或输出的状态；

其他的指示灯说明如下：

LED1：电源指示灯，有电供给时常亮；

LED2：扩展 I/O 板心跳灯，交替亮或灭 1 秒钟，表示 I/O 板内程序已启动；

RX：表示从控制器收到消息，正常工作时快速闪烁，未收到消息不亮；

TX：表示发给控制器消息，正常工作时快速闪烁，未发送消息不亮。

7.2.5 串口通信范例(自由协议)

Process Main

```

int sn = 2 //com 号范围 (1-2)
int BaudRate=115200 //波特率范围
//115200,57600,38400,19200,9600,4800,2400,1200*/
int Databit=8 //数据位长度 8
int Stopbit=1 //停止位长度 1
string Parity="N" //校验选择 ("N"-无奇偶校验)
int res=0
string read_data
SetCom( sn, BaudRate, Databit, Stopbit, Parity)
OpenCom(sn)
Do
    /*
        res = -2 参数错误
        res = -1 串口未打开
    */
    res = CheckCom(sn)
    If res == -2 Then
        Print("parameter error")
    ElseIf res == -1 Then
        Print("no running opencom")
    ElseIf res > 0 Then
        read_data = ReadCom(sn) //读取串口 sn 的数据并保存 read_data 变量中
        Print(read_data)
        WriteCom(sn,read_data) //把变量 read_data 中的数据通过 sn 通道发送出去
    EndIf
    Delay(5)
Loop
ProcessEnd

```



提示

如果使用的是 Modbus-RTU 协议，用户无需编写收发程序，使用方法参考
5.2.7.2 modbus 自定义变量的读写操作

7.3 网口 (Ethernet) 使用方法

7.3.1 协议模式说明

MDCS-510 提供的外部网络同时最高支持连接 6 路，其中 2 路为系统固定占用处理 Modbus-TCP 或电脑上位机软件连接通道；其余 4 路为用户提供自由协议的连接通道（在界面中显示最高 4 路）；

7.3.2 自由协议模式(TCP/IP)

1) 配置网络通讯界面

a. 工程师进入通信→网口配置，如图 7-3-2-1 所示；

通信监控-网口配置

网络配置

IP地址: 192.168.2.23

子网掩码: 255.255.255.0

默认网关: 192.168.2.1

MAC地址: 00-08-DC-11-11-11

通道配置

通道选择: 1

协议模式: 自由协议

本机模式: 服务器

端口: 4294967063

添加

删除

开启

关闭

通道号	服务器IP地址	端口号	模式	连接状态	协议模式
1	192.168.2.23	4294967063	服务器	打开	自由

保存 返回

图 7-3-2-1 网口通讯配置界面

- 在网络配置组中设置 IP，子网掩码，和默认网关；
- 在通道配置中选择某 1 个通道，将本机模式设置为服务器/客户端；
其中可能会配置到 IP 地址和端口，根据选择的模式而不同；
- 通道配置完毕后点→添加，完成通道配置；
- 点击→保存，以便下次开机依然生效。

2) 参考下文通信范例，选择机器人做服务器或客户端模式

7.3.3 Modbus-TCP 模式配置

7.3.3.1 配置网口信息

将 IP 地址设置成 192.168.1.80 , 255.255.255.0 192.168.1.1

设置完成后保存；

7.3.3.2 配置 modbus 功能启用

系统开机后默认开启 modbus 功能；

主控版本在 1.27 以下的控制器软件需要设置 159#为 1 开启 modbus 功能；

7.3.3.3 将模式钥匙切换到远程模式

只有在此模式下，modbus 外部控制方才生效。

在此模式下，示教器上的按键仅响应急停、V+、V-、使能按键以及模式开关键

7.3.3 网口通信范例（自由协议机器人做客户端）

Process Main

```
//设置本地 sn 通道为客户端
string serverIp="192.168.1.150"      //服务器端 IP
int server_port=5000                //服务器端口号
int sn=1                            //通道号
int res=0
string read_data
SetNet(1,sn, serverIp, server_port) //设置通道 sn 为客户端
OpenNet(sn)
Do
    /*
        res=-5 初使化
        res=-4 网线断开
        res=-3 未执行 opennet(sn)
        res=-2 参数错误
        res=-1 网口已经打开,但与对方未建立连接
        res>=0 数据长度
    */
    res = CheckNet(sn)
    If res == -5 Then
        Print("Ethernet restarting ")
    ElseIf res == -4 Then
        Print("line out")
    ElseIf res == -3 Then
        Print(" no running opennet funciton")
    ElseIf res == -2 Then
        Print("parameter error")
    ElseIf res == -1 Then
        Print("no ESTABLISHED")
    ElseIf res > 0 Then
        read_data=ReadNet(sn)      //数据保存到 read_data 中
        Print(read_data)
        WriteNet(sn,read_data)    //向通道 sn 发送 read_data 变量保存的数据
    EndIf
    Delay(5)
Loop
ProcessEnd
```

7.3.4 网口通信范例（自由协议机器人做服务器）

Process Main

```
//设置 sn 通道为服务端
string ServerIp="192.168.1.66"           //服务端 IP
int Server_port=6098                     //服务端口号
int sn=1                                 //通道号
int res=0
string read_data
SetNet(0,sn,ServerIp,Server_port)       //设置通道 sn 为服务端
OpenNet(sn)
Do
    /*
        res=-5 初使化
        res=-4 网线断开
        res=-3 未执行 opennet(sn)
        res=-2 参数错误
        res=-1 网口已经打开,但与对方未建立连接
        res>=0 数据长度
    */
    res = CheckNet(sn)
    If res ==-5 Then
        Print("Ethernet restarting ")
    ElseIf res ==-4 Then
        Print("line out")
    ElseIf res ==-3 Then
        Print(" no running opennet funciton")
    ElseIf res == -2 Then
        Print("parameter error")
    ElseIf res ==-1 Then
        Print("no ESTABLISHED")
    ElseIf res > 0 Then
        read_data=ReadNet(sn)           //数据保存到 read_data 中
        Print(read_data)
        WriteNet(sn,read_data)          //向通道 sn 发送 read_data 变量保存的数据
    EndIf
    Delay(5)
Loop
ProcessEnd
```

第八章 附录

附录 A 常见报警

错误码	错误信息
1	不能识别的符号
2	数值输入格式错误
3	加工文件不存在
4	不支持的语法格式
10	以默认设置初始化参数
11	参数文件打开失败
12	参数文件数据读出异常
13	参数文件创建失败
14	参数文件写入失败
15	参数文件 CRC 校验错误
16	存储的参数值错误
17	参数项变更
18	新增了参数项
30	以默认设置初始坐标系
31	坐标系文件打开失败
32	坐标系文件数据读出异常
33	坐标系文件创建失败
34	坐标系文件写入失败
35	坐标系文件 CRC 校验错误
40	以默认设置初始化 IO 配置
41	IO 配置文件打开失败
42	IO 配置文件读出异常
43	IO 配置文件创建失败
44	IO 配置文件写入失败
45	IO 配置文件 CRC 校验错误
50	以默认设置初始化区域监控数据
51	区域监控配置文件打开失败
52	区域监控配置文件读取失败
53	区域监控配置文件创建失败
54	区域监控配置文件写入失败
55	区域监控配置文件 CRC 校验错
60	区域监控参考点录入失败
70	以默认设置初始化示教点

71	示教点文件打开失败
72	示教点文件数据读出异常
73	示教点文件创建失败
74	示教点存储空间申请失败
75	示教点文件写入失败
76	示教点文件 CRC 校验错误
80	示教点关节坐标转换笛卡尔坐标失败
81	示教点笛卡尔坐标转换关节坐标失败
82	示教点用户坐标系转换失败
83	示教点无效
84	示教点录入失败
100	初始化工程配置信息错误
101	读取当前工程信息错误
102	当前未加载工程
103	工程装载失败
104	未找到远程模式工程
200	第 1 轴 TT-LINK 实时通信总线异常，请重启或检查硬件
201	第 2 轴 TT-LINK 实时通信总线异常，请重启或检查硬件
202	第 3 轴 TT-LINK 实时通信总线异常，请重启或检查硬件
203	第 4 轴 TT-LINK 实时通信总线异常，请重启或检查硬件
1000	不能识别的符号
1001	数值输入格式错误
1002	加工文件不存在
1003	不支持的语法格式
1004	任务内容为空
1005	中间码文件保存失败
1006	文件开始标志和任务级别不匹配
1007	输入参数错误
1008	modbus 全局变量不能被声明类型
2000	构建语法树错误
2001	变量重复声明
2002	变量声明初始化类型错误
2003	变量未声明使用
2004	操作数类型错误
2005	错误的赋值类型
2006	赋值操作左值类型错误
2007	不支持的操作符类型

2008	数组未声明的引用
2009	数组名变量重复声明
2010	point 类型数组初始化行数大于数组元素个数
2011	point 类型数组初始化行数值不等于 6
2012	数组声明初始化值类型错误
2013	取点位的数据类型错误
2014	标签重复定义
2015	GoTo 标签没有声明
2016	标签注册失败
2017	表达式语句格式错误
2018	case 表达式仅支持常数
2019	Do 嵌套层数(20)超出上限
2020	If 嵌套层数(20)超出上限
2021	Switch 嵌套层数(20)超出上限
2022	For 嵌套层数(20)超出上限
2023	仅支持全局变量声明
2024	数组初始化数目与数组大小不符
2025	变量声明出错
2026	数组声明出错
2027	语句没有与之相匹配的结束符
2028	函数嵌套层数超出上限(10)
2100	不支持的函数调用
2101	函数参数数目不对
2102	函数参数类型错误
2103	函数参数超出范围
2104	子任务中不允许有运动指令
2105	错误的函数使用格式
2106	自定义函数注册失败
2107	函数赋值对象错误
2108	自定义函数返回值数量超出上限
2109	自定义函数返回值数量和左值列表数不匹配
2110	指令函数不支持多重赋值
2111	自定义函数重复声明
2112	不支持的并行执行函数
2113	并行执行期间不支持自定义函数
2114	Dn 指令仅支持常数参数
2115	主任务中执行了错误的指令(PowerEn\GetError)

2200	数组初始化值类型错误
2201	数组声明中元素数量必须为正整数
2202	数组初始化数值和数组大小不一致
2203	数组名重复声明
2204	数组名未声明使用
2205	数组引用中元素必须为正整数
2300	标签重复定义
5000	除数不能为 0
5001	临时变量下标>上限(10)
5002	获取非 Point 数值错误
5003	获取 Point 数值错误
5004	数值类型错误
5005	变量下标>上限值
5006	保存数值出错
5007	跳转目标出错
5008	操作码类型错误
5009	字符串标号>上限值
5010	从指定行位置启动出错
5011	数组元素类型错误
5012	获取数组元素出错
5013	switch 选择条件类型错误
5014	case 判断条件类型错误
5015	非 point 变量赋值出错
5016	point 变量赋值出错
5017	获取临时变量数值错误
5018	保存非 point 类型临时变量值出错
5019	保存 point 类型临时变量值错误
5020	坐标转换错误
5021	笛卡尔坐标点位偏移错误
6000	函数解析出错
6001	函数调用参数数量超出上限(10)
6002	函数调用参数数量匹配错误
6003	示教点不存在
6004	不支持的函数引用
6005	不支持的打印类型
6006	函数参数类型错误
6007	函数返回赋值类型错误

6008	字符串长度小于截取长度
6009	字符串于数值不匹配
6010	获取阵列点位出错
6011	保存 point 类型临时变量错误
6012	获取函数参数类型错误
6013	获取函数参数内容出错
6014	获取用户坐标值错误
6015	参数轴号不在范围内
6016	阵列方向上点数量必须 ≥ 2
6017	自定义函数自嵌套深度超限(> 10)
6018	自定义函数申请栈空间出错
6019	自定义函数多重赋值返回值和赋值对象数目不匹配
6020	字符串格式不匹配
6021	格式化字符串使用错误
6022	函数参数值超出限定范围
6023	创建阵列的点位坐标系不一致
6024	保存字符串错误
7000	申请内存空间出错
7001	当前段数据为空
7100	识别中间码文件出错
7101	加载全局变量文件出错
8000	未定义的报警
10000	系统急停
10001	伺服未使能，请先使能伺服后操作
10002	无效操作
10003	系统未就绪，不可使能伺服
10004	伺服使能关闭响应超时，强制关闭
10005	急停响应超时，强制急停
10006	全局变量任务启动超时
10007	全局变量任务编译后端响应超时
10008	主任务启动方式异常，请检查工程文件
10010	系统超过试用期，自动功能禁用，请联系厂商解锁/续期
12000	运控响应超时
12001	编译前端响应超时
12010	任务 1 编译响应超时
12011	任务 2 编译响应超时
12012	任务 3 编译响应超时

12013	任务 4 编译响应超时
12014	任务 5 编译响应超时
12015	任务 6 编译响应超时
12016	任务 7 编译响应超时
12017	任务 8 编译响应超时
12020	任务 1 逻辑响应超时
12021	任务 2 逻辑响应超时
12022	任务 3 逻辑响应超时
12023	任务 4 逻辑响应超时
12024	任务 5 逻辑响应超时
12025	任务 6 逻辑响应超时
12026	任务 7 逻辑响应超时
12027	任务 8 逻辑响应超时
18000	TCP 进入障碍区
20001	机器人类型错误
20002	路径类型错误
20003	机器人初始化失败
20004	禁止形态转换
20005	关节超软限位
20006	第 1 轴超软限位
20007	第 2 轴超软限位
20008	第 3 轴超软限位
20009	第 4 轴超软限位
20010	第 5 轴超软限位
20011	第 6 轴超软限位
20012	第 1 轴超速
20013	第 2 轴超速
20014	第 3 轴超速
20015	第 4 轴超速
20016	第 5 轴超速
20017	第 6 轴超速
20018	脉冲过大
20019	运动时间为 0
20020	匀速段速度计算失败
20021	加速时间为 0
20022	匀速时间为 0
20023	减速时间为负

20024	位置过冲
20025	时间过冲
20026	轨迹类型错误
20027	1 维方向错误
20028	形态类型错误
20029	运控参数错误
20030	方向错误
20031	运控数据帧缓存区已满
20032	运控数据帧缓存区已空
20033	等待运控数据帧超时
20034	圆心角过小
20035	工具坐标系标定失败
20036	发送脉冲失败
20037	获取数据帧失败
20038	求根失败
20039	向量单位化失败
20040	矩阵求逆失败
20041	向量维度为 0
20042	向量维度过高
20043	速度参数错误
20044	关节超软限位
20045	关节 1 超软限位下限
20046	关节 2 超软限位下限
20047	关节 3 超软限位下限
20048	关节 4 超软限位下限
20049	关节 5 超软限位下限
20050	关节 6 超软限位下限
20051	关节运动指令格式错误
20052	轴号映射失败
20053	耦合参数错误
20054	等待伺服到位时间超时
20055	电机编码器值通讯失败
20056	计算圆心失败
20057	获取工具坐标系失败
20058	获取用户坐标系失败
20059	关节 1 伺服到位超时
20060	关节 2 伺服到位超时

20061	关节 3 伺服到位超时
20062	关节 4 伺服到位超时
20063	关节 5 伺服到位超时
20064	关节 6 伺服到位超时
20065	等待读取插补数据超时
20066	工具坐标系参数 A 为非 0 值或 B 为非 0 值
20067	运控完成度百分比错误
20068	目标点类型错误
20069	用户坐标系 Z 轴与世界坐标系 Z 轴不平行
20070	速度参数错误
20071	加速度参数错误
20072	运控数据帧命令号顺序错乱
20073	平滑参数错误
20074	向量夹角参数错误
20075	第 4 轴圈数错误
20076	计算第 4 轴圈数错误
20077	运控缓存区数据帧数目错误
20078	切换工具坐系失败
20079	加速度过大
20080	运动时间为负
20081	门形运动参数错误
20082	迭代失败
20083	门形运动目标点序号错误
20084	计算姿态角 C 值失败
20085	用户坐标系参数错误
20086	时间错误
20087	编译端运控数据帧参数错误
20088	暂不支持 CP 平滑
20089	插补数据缓存区为空
20090	当前 Z 高度大于门形运动限制 Z 高度
25701	笛卡尔位移为 0
25702	关节位移为 0
25703	速度值忽略
25704	空闲状态执行暂停命令
25705	空闲状态执行急停命令
25706	运行状态执行运行命令
25707	运行状态执行复位命令

25708	急停状态执行运行命令
25709	急停状态执行暂停命令
25710	急停状态执行急停命令
25711	暂停状态执行暂停命令
25712	暂停状态执行急停命令
30000	第 1 轴伺服驱动报警
30001	第 1 轴伺服 VCE 报警< AL001 >
30002	第 1 轴伺服过压< AL002 >
30003	第 1 轴伺服欠压< AL003 >
30004	第 1 轴伺服电流过流< AL004 >
30006	第 1 轴伺服电流采样通道异常< AL006 >
30007	第 1 轴伺服超速< AL007 >
30008	第 1 轴伺服位置超差< AL008 >
30009	第 1 轴伺服用户转矩过载< AL009 >
30016	第 1 轴伺服制动异常< AL010 >
30017	第 1 轴伺服驱动器使用超时< AL011 >
30020	第 1 轴伺服过热< AL014 >
30021	第 1 轴伺服过载< AL015 >
30022	第 1 轴伺服驱动过负载< AL016 >
30023	第 1 轴伺服电机过负载< AL017 >
30032	第 1 轴伺服省线式初始化错/UVW 初始态错< AL020 >
30033	第 1 轴伺服绝对值编码器通讯错误< AL021 >
30034	第 1 轴伺服绝对值编码器 CRC 校验错误< AL022 >
30035	第 1 轴伺服编码器电池报警< AL023 >
30036	第 1 轴伺服绝对值编码器圈数溢出< AL024 >
30037	第 1 轴伺服绝对值编码器报警< AL025 >
30038	第 1 轴绝对型伺服坐标遗失警告，坐标建立后自动清除< AL026 >
30039	第 1 轴绝对值编码器清圈数失败警告，自动清除< AL027 >
30040	第 1 轴绝对伺服系统
30048	第 1 轴伺服紧急停止< AL030 >
30049	第 1 轴伺服正向限位触发< AL031 >
30050	第 1 轴伺服反向限位触发< AL032 >
30053	第 1 轴伺服不同的 DI 重复分配了相同功能< AL035 >
30056	第 1 轴伺服位置指令方向信号异常< AL038 >
30064	第 1 轴伺服初始化参数失败(EEP)< AL040 >
30065	第 1 轴获取 EEPROM 中故障记录失败< AL041 >
30080	第 1 轴串行通信故障< AL050 >

30081	第 1 轴串行通信超时故障<AL051 >
31000	第 2 轴伺服驱动报警
31001	第 2 轴伺服 VCE 报警< AL001 >
31002	第 2 轴伺服过压< AL002 >
31003	第 2 轴伺服欠压< AL003 >
31004	第 2 轴伺服电流过流< AL004 >
31006	第 2 轴伺服电流采样通道异常< AL006 >
31007	第 2 轴伺服超速< AL007 >
31008	第 2 轴伺服位置超差< AL008 >
31009	第 2 轴伺服用户转矩过载< AL009 >
31016	第 2 轴伺服制动异常< AL010 >
31017	第 2 轴伺服驱动器使用超时< AL011 >
31020	第 2 轴伺服过热< AL014 >
31021	第 2 轴伺服过载< AL015 >
31022	第 2 轴伺服驱动过负载< AL016 >
31023	第 2 轴伺服电机过负载< AL017 >
31032	第 2 轴伺服省线式初始化错/UVW 初始态错< AL020 >
31033	第 2 轴伺服绝对值编码器通讯错误< AL021 >
31034	第 2 轴伺服绝对值编码器 CRC 校验错误< AL022 >
31035	第 2 轴伺服编码器电池报警< AL023 >
31036	第 2 轴伺服绝对值编码器圈数溢出< AL024 >
31037	第 2 轴伺服绝对值编码器报警< AL025 >
31038	第 2 轴绝对型伺服坐标遗失警告，坐标建立后自动清除< AL026 >
31039	第 2 轴绝对值编码器清圈数失败警告，自动清除< AL027 >
31040	第 2 轴绝对伺服系统
31048	第 2 轴伺服紧急停止< AL030 >
31049	第 2 轴伺服正向限位触发< AL031 >
31050	第 2 轴伺服反向限位触发< AL032 >
31053	第 2 轴伺服不同的 DI 重复分配了相同功能< AL035 >
31056	第 2 轴伺服位置指令方向信号异常< AL038 >
31064	第 2 轴伺服初始化参数失败(EEP)< AL040 >
31065	第 2 轴获取 EEPROM 中故障记录失败< AL041 >
31080	第 2 轴串行通信故障< AL050 >
31081	第 2 轴串行通信超时故障<AL051 >
32000	第 3 轴伺服驱动报警
32001	第 3 轴伺服 VCE 报警< AL001 >
32002	第 3 轴伺服过压< AL002 >

32003	第 3 轴伺服欠压< AL003 >
32004	第 3 轴伺服电流过流< AL004 >
32006	第 3 轴伺服电流采样通道异常< AL006 >
32007	第 3 轴伺服超速< AL007 >
32008	第 3 轴伺服位置超差< AL008 >
32009	第 3 轴伺服用户转矩过载< AL009 >
32016	第 3 轴伺服制动异常< AL010 >
32017	第 3 轴伺服驱动器使用超时< AL011 >
32020	第 3 轴伺服过热< AL014 >
32021	第 3 轴伺服过载< AL015 >
32022	第 3 轴伺服驱动过负载< AL016 >
32023	第 3 轴伺服电机过负载< AL017 >
32032	第 3 轴伺服省线式初始化错/UVW 初始态错< AL020 >
32033	第 3 轴伺服绝对值编码器通讯错误< AL021 >
32034	第 3 轴伺服绝对值编码器 CRC 校验错误< AL022 >
32035	第 3 轴伺服编码器电池报警< AL023 >
32036	第 3 轴伺服绝对值编码器圈数溢出< AL024 >
32037	第 3 轴伺服绝对值编码器报警< AL025 >
32038	第 3 轴绝对型伺服坐标遗失警告, 坐标建立后自动清除< AL026 >
32039	第 3 轴绝对值编码器清圈数失败警告, 自动清除< AL027 >
32040	第 3 轴绝对伺服系统
32048	第 3 轴伺服紧急停止< AL030 >
32049	第 3 轴伺服正向限位触发< AL031 >
32050	第 3 轴伺服反向限位触发< AL032 >
32053	第 3 轴伺服不同的 DI 重复分配了相同功能< AL035 >
32056	第 3 轴伺服位置指令方向信号异常< AL038 >
32064	第 3 轴伺服初始化参数失败(EEP)< AL040 >
32065	第 3 轴获取 EEPROM 中故障记录失败< AL041 >
32080	第 3 轴串行通信故障< AL050 >
32081	第 3 轴串行通信超时故障<AL051 >
33000	第 4 轴伺服驱动报警
33001	第 4 轴伺服 VCE 报警< AL001 >
33002	第 4 轴伺服过压< AL002 >
33003	第 4 轴伺服欠压< AL003 >
33004	第 4 轴伺服电流过流< AL004 >
33006	第 4 轴伺服电流采样通道异常< AL006 >
33007	第 4 轴伺服超速< AL007 >

33008	第 4 轴伺服位置超差 < AL008 >
33009	第 4 轴伺服用户转矩过载 < AL009 >
33016	第 4 轴伺服制动异常 < AL010 >
33017	第 4 轴伺服驱动器使用超时 < AL011 >
33020	第 4 轴伺服过热 < AL014 >
33021	第 4 轴伺服过载 < AL015 >
33022	第 4 轴伺服驱动过负载 < AL016 >
33023	第 4 轴伺服电机过负载 < AL017 >
33032	第 4 轴伺服省线式初始化错/UVW 初始态错 < AL020 >
33033	第 4 轴伺服绝对值编码器通讯错误 < AL021 >
33034	第 4 轴伺服绝对值编码器 CRC 校验错误 < AL022 >
33035	第 4 轴伺服编码器电池报警 < AL023 >
33036	第 4 轴伺服绝对值编码器圈数溢出 < AL024 >
33037	第 4 轴伺服绝对值编码器报警 < AL025 >
33038	第 4 轴绝对型伺服坐标遗失警告，坐标建立后自动清除 < AL026 >
33039	第 4 轴绝对值编码器清圈数失败警告，自动清除 < AL027 >
33040	第 4 轴绝对伺服系统
33048	第 4 轴伺服紧急停止 < AL030 >
33049	第 4 轴伺服正向限位触发 < AL031 >
33050	第 4 轴伺服反向限位触发 < AL032 >
33053	第 4 轴伺服不同的 DI 重复分配了相同功能 < AL035 >
33056	第 4 轴伺服位置指令方向信号异常 < AL038 >
33064	第 4 轴伺服初始化参数失败(EEP) < AL040 >
33065	第 4 轴获取 EEPROM 中故障记录失败 < AL041 >
33080	第 4 轴串行通信故障 < AL050 >
33081	第 4 轴串行通信超时故障 < AL051 >

附录 B Modbus 通讯地址表

分组	定义	地址类型	数值	读写 -系统侧	起始地址 -10 进制	起始地址- 16 进制	长 度
系统 状态	系统工作 模式	unsigned short(16b)	0x00 : 初始化 ; 0x01 : 远程自动 ; 0x02 : 远程手动 ; 0x03 : 远程 DNC ;	只读	24576	0x6000	1
系统 状态	系统状态	unsigned short(16b)	0x00 : 初始化/复位 ; 0x01 : 急停 ; 0x02 : 报警 ; 0x03 : 就绪 ; 0x04 : 运行中 ; 0x05 : 暂停中 ;	只读	24577	0x6001	1
系统 状态	伺服状态	unsigned short(16b)	0x00 : 初始化/伺服使能关 ; 0x01 : 伺服使能开 ;	只读	24578	0x6002	1
系统 状态	当前工程 编号	unsigned short(16b)	0 : 当前无工程 ; 1~99 : 当前使用工程 ;	只读	24579	0x6003	1
系统 状态	当前报警 数量	unsigned short(16b)	0~99	只读	24580	0x6004	1
系统 状态	当前报警 号	unsigned short(16b) ×10	报警号	只读	24581	0x6005	10
系统 控制	系统控制 命令	unsigned short(16b)	0x01 : 复位 ; 0x02 : 启动 ; 0x03 : 暂停 ; 0x04 : 报警读取/清除报警 ;	读/写	24592	0x6010	1
系统 控制	伺服使能 命令	unsigned short(16b)	0x01 : 伺服使能 ON <-> OFF;	读/写	24593	0x6011	1
系统 控制	伺服监控 清零命令	unsigned short(16b)	0x01 : 伺服监控数据记录清零	读/写	24594	0x6012	1
系统 控制	系统模式 切换命令	unsigned short(16b)	0x01 : 切换到自动模式 ; 0x02 : 切换到手动模式 ;	读/写	24595	0x6013	1
系统 控制	倍率修改 命令	unsigned short(16b)	0x01 : 倍率+ ; 0x02 : 倍率-	读/写	24596	0x6014	1
系统 控制	自动倍率	unsigned short(16b)	0~16 对应 : 1% , 2% , 5% , 10% , 15%,20%,25% , 30%,35%,40%,45%,50%,60%,7 0%,80%,90% , 100%共计 17 档 ;	只读	24597	0x6015	1
系统 控制	手动倍率	unsigned short(16b)	0~19 对应 : 0.01mm , 0.1mm , 1mm , 1% , 2% , 5% , 10% , 15%,20%,25% , 30%,35%,40%,45%,50%,60%,7 0%,80%,90% , 100%共计 20 档 ;	只读	24598	0x6016	1

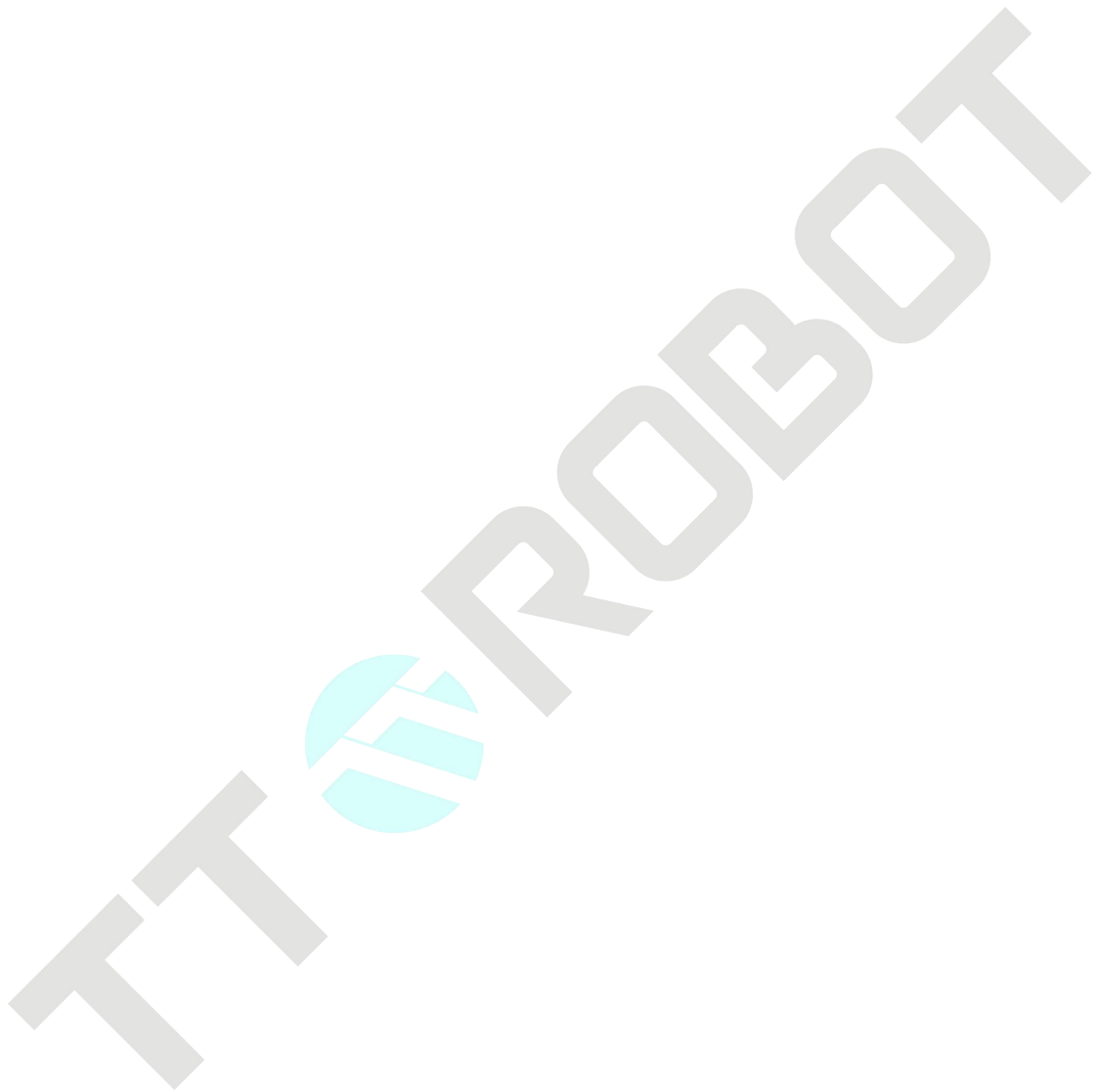
系统控制	手动模式	unsigned short(16b)	0x00 : 关节 ; 0x01 : 直角 ; 0x02 : 工具 ; 0x03 : 用户 ;	读/写	24599	0x6017	1
系统控制	工程操作指令	unsigned short(16b)	1 : 装载编号指定的工程 ;	读/写	24600	0x6018	1
系统控制	目标工程编号	unsigned short(16b)	1~99	读/写	24601	0x6019	1
运行数据	当前用户坐标系编号	unsigned short(16b)	0~15	读/写	24624	0x6030	1
运行数据	当前工具坐标系编号	unsigned short(16b)	0~8	读/写	24625	0x6031	1
运行数据	当前运行行号	unsigned short(16b)	0~49999	只读	24626	0x6032	1
运行数据	当前运控指令执行进度	unsigned short(16b)	0~100	只读	24627	0x6033	1
运行数据	实时世界坐标	float(32b)×6	X,Y,Z,A,B,C	只读	24640	0x6040	2*6
运行数据	实时当前用户坐标	float(32b)×6	UX,UY,UZ,UA,UB,UC	只读	24656	0x6050	2*6
运行数据	实时关节坐标	float(32b)×6	J1,J2,J3,J4,J5,J6	只读	24672	0x6060	2*6
伺服监控	实时转速	float(32b)	1 轴实时转速 r/pm	只读	24832	0x6100	2
伺服监控	最大转速	float(32b)	1 轴最大转速 r/pm	只读	24834	0x6102	2
伺服监控	实时负载	float(32b)	1 轴实时负载 %	只读	24836	0x6104	2
伺服监控	最大负载	float(32b)	1 轴最大负载 %	只读	24838	0x6106	2
伺服监控	实时电流	float(32b)	1 轴实时电流 A	只读	24840	0x6108	2
伺服监控	最大电流	float(32b)	1 轴最大电流 A	只读	24842	0x610a	2
伺服监控	电机实时绝对位置	int(32b)	1 轴电机实时 pulse	只读	24844	0x610c	2
伺服监控	伺服指令位置	int(32b)	1 轴指令实时 pulse	只读	24846	0x610e	2
伺服监控	2 轴实时数据	同 1 轴	同 1 轴	只读	24848	0x6110	2
伺服监控	3 轴实时数据	同 1 轴	同 1 轴	只读	24864	0x6120	2
伺服监控	4 轴实时数据	同 1 轴	同 1 轴	只读	24880	0x6130	2
伺服	5 轴实时	同 1 轴	同 1 轴	只读	24896	0x6140	2

监控	数据						
伺服 监控	6 轴实时 数据	同 1 轴	同 1 轴	只读	24912	0x6150	2
示教 点操 作	目标示教 点编号	unsigned short(16b)	0~999	读/写	25088	0x6200	1
示教 点操 作	示教点操 作命令	unsigned short(16b)	0x01 : 当前坐标录入示教点 ; 0x02 : 移动到示教点位置 ; 0x03 : 示教点数据修改更新 ; 0x04 : 示教点文件存储 ; 0x05 : 删除示教点 ;	读/写	25089	0x6201	1
示教 点操 作	移动到示 教点方式	unsigned short(16b)	0x01 : MOVJ ; 0x02 : MOVL ; 0x03 : JUMP ;	读/写	25857	0x6501	1
示教 点操 作	JUMP 移 动方式限 制高度	float(32b)	Z 限高数据 mm	读/写	25878	0x6516	2
示教 点数 据	当前操作 示教点编 号	unsigned short(16b)	0~999	读/写	25090	0x6202	1
示教 点数 据	p[n.0]有 效性	unsigned short(16b)	1 : 有效 ; 0 : 无效	读/写	25091	0x6203	1
示教 点数 据	p[n.0]录 入形态	short(16b)	0 : 左手 ; 1 : 右手 ; -1 : N/A	读/写	25092	0x6204	1
示教 点数 据	p[n.0]坐 标类型	unsigned short(16b)	0 : 关节 ; 1 : 笛卡尔	读/写	25093	0x6205	1
示教 点数 据	p[n.0]用 户坐标序 号	short(16b)	0~15 ; -1 : N/A	读/写	25094	0x6206	1
示教 点数 据	预留	short(16b)			25095	0x6207	1
示教 点数 据	p[n.0]坐 标值	float(32b)	X/J1	读/写	25096	0x6208	2
示教 点数 据	p[n.0]坐 标值	float(32b)	Y/J2	读/写	25098	0x620a	2
示教 点数 据	p[n.0]坐 标值	float(32b)	Z/J3	读/写	25100	0x620c	2
示教 点数 据	p[n.0]坐 标值	float(32b)	A/J4	读/写	25102	0x620e	2
示教	p[n.0]坐	float(32b)	B/J5	读/写	25104	0x6210	2

点数据	标值						
示教点数据	p[n.0]坐标值	float(32b)	C/J6	读/写	25106	0x6212	2
坐标系操作	当前坐标系类型	unsigned short(16b)	0x00：用户坐标系； 0x01：工具坐标系	读/写	25344	0x6300	1
坐标系操作	当前操作坐标系编号	unsigned short(16b)	0~15：对应用户坐标系的0~15； 0~8：对应工具坐标系0~8；	读/写	25345	0x6301	1
坐标系操作	当前操作标定点序号	unsigned short(16b)	1~6：对应第1~第6标定点；	读/写	25346	0x6302	1
坐标系操作	坐标系命令	unsigned short(16b)	0x01：清空坐标； 0x02：标定点录入； 0x03：移动到标定点； 0x04：标定计算； 0x05：标定点数据存储； 0x06：坐标系数据存储； 0x07：取消保存； 0x08：标定点数据更新； 0x09：坐标系数据更新；	读/写	25347	0x6303	1
坐标系系数数据	标定点临时数据	float(32b)	X/J1	读/写	25348	0x6304	2
坐标系系数数据	标定点临时数据	float(32b)	Y/J2	读/写	25350	0x6306	2
坐标系系数数据	标定点临时数据	float(32b)	Z/J3	读/写	25352	0x6308	2
坐标系系数数据	标定点临时数据	float(32b)	A/J4	读/写	25354	0x630a	2
坐标系系数数据	标定点临时数据	float(32b)	B/J5	读/写	25356	0x630c	2
坐标系系数数据	标定点临时数据	float(32b)	C/J6	读/写	25358	0x630e	2
坐标系系数数据	坐标系临时数值	float(32b)	X/J1	读/写	25360	0x6310	2
坐标系系数数据	坐标系临时数值	float(32b)	Y/J2	读/写	25362	0x6312	2
坐标系	坐标系临时数值	float(32b)	Z/J3	读/写	25364	0x6314	2

系数数据	时数值						
坐标系系数数据	坐标系临时数值	float(32b)	A/J4	读/写	25366	0x6316	2
坐标系系数数据	坐标系临时数值	float(32b)	B/J5	读/写	25368	0x6318	2
坐标系系数数据	坐标系临时数值	float(32b)	C/J6	读/写	25370	0x631a	2
输入点	输入线圈	bit(coil)	0 : OFF ; 1 : ON ;	读/写	16384	0x4000	1
输出点	输出线圈	bit(coil)	0 : OFF ; 1 : ON ;	读/写	16640	0x4100	1
内部信号量	线圈信号量	bit(coil)		读/写	16896	0x4200	1
手动指令	正向运动命令	bit(coil)	手动 X/J1+命令 (1-move ; 0-N/A)	读/写	17408	0x4400	1
手动指令	正向运动命令	bit(coil)	手动 Y/J2+命令 (1-move ; 0-N/A)	读/写	17409	0x4401	1
手动指令	正向运动命令	bit(coil)	手动 Z/J3+命令 (1-move ; 0-N/A)	读/写	17410	0x4402	1
手动指令	正向运动命令	bit(coil)	手动 A/J4+命令 (1-move ; 0-N/A)	读/写	17411	0x4403	1
手动指令	正向运动命令	bit(coil)	手动 B/J5+命令 (1-move ; 0-N/A)	读/写	17412	0x4404	1
手动指令	正向运动命令	bit(coil)	手动 C/J6+命令 (1-move ; 0-N/A)	读/写	17413	0x4405	1
手动指令	负向运动命令	bit(coil)	手动 X/J1-命令 (1-move ; 0-N/A)	读/写	17424	0x4410	1
手动指令	负向运动命令	bit(coil)	手动 Y/J2-命令 (1-move ; 0-N/A)	读/写	17425	0x4411	1
手动指令	负向运动命令	bit(coil)	手动 Z/J3-命令 (1-move ; 0-N/A)	读/写	17426	0x4412	1
手动指令	负向运动命令	bit(coil)	手动 A/J4-命令 (1-move ; 0-N/A)	读/写	17427	0x4413	1
手动指令	负向运动命令	bit(coil)	手动 B/J5-命令 (1-move ; 0-N/A)	读/写	17428	0x4414	1
手动指令	负向运动命令	bit(coil)	手动 C/J6-命令 (1-move ; 0-N/A)	读/写	17429	0x4415	1
自定义功能	数据 I0	signed int(32b)	整数型	读/写	40960	0xa000	2
自定义功能	数据 I1	signed int(32b)	整数型	读/写	40962	0xa002	2

自定义功能	...	signed int(32b)	整数型	读/写		...	
自定义功能	数据 I99	signed int(32b)	整数型	读/写	41158	0xa0c6	2
自定义功能	数据 F0	float(32b)	浮点型	读/写	41160	0xa0c8	2
自定义功能	数据 F1	float(32b)	浮点型	读/写	41162	0xa0ca	2
自定义功能	...	float(32b)	浮点型	读/写		...	
自定义功能	数据 F99	float(32b)	浮点型	读/写	41358	0xa18e	2
自定义功能	数据 GI0	signed int(32b)	整数型	读/写	41984	0xa400	2
自定义功能	数据 GI1	signed int(32b)	整数型	读/写	41986	0xa402	2
自定义功能	...	signed int(32b)	整数型	读/写		...	
自定义功能	数据 GI99	signed int(32b)	整数型	读/写	42082	0xa462	2
自定义功能	数据 GF0	float(32b)	浮点型	读/写	42240	0xa500	2
自定义功能	数据 GF1	float(32b)	浮点型	读/写	42242	0xa502	2
自定义功能	...	float(32b)	浮点型	读/写		...	
自定义功能	数据 GF99	float(32b)	浮点型	读/写	42338	0xa562	2



广东天太机器人有限公司

地址：广东省佛山市顺德区 G105 国道大良红岗段 23 号

全国服务热线：400-800-1807

官网：www.gdtiantai.com