



机器人指令手册

Robot Instruction Manual

版本：v0.0.1

日期：2026.01.22

目录

| | |
|--------------------------------------|----|
| OS 指令一览 | 9 |
| 通用语句 | 9 |
| 数据类型 | 9 |
| 运动指令 | 10 |
| 独立轴指令 | 12 |
| 点位、I/O、编码器等备注指令 | 12 |
| I/O 指令 | 13 |
| 寄存器读写指令 | 14 |
| 通讯指令 | 14 |
| 系统指令 | 15 |
| 码垛指令 | 16 |
| 传送带跟踪指令 | 16 |
| 数学运算指令 | 17 |
| 字符串运算指令 | 18 |
| 点位运算指令 | 19 |
| 通用语句详细说明 | 21 |
| Call | 21 |
| Do.. Loop | 22 |
| Else/Elseif | 23 |
| Exit | 24 |
| For.. To.. Next | 25 |
| Function.. Fend | 26 |
| Global | 28 |
| GoSub.. Return | 29 |
| GoTo | 30 |
| If.. Then.. Else.. EndIf | 31 |
| Select.. Case.. Default.. Send | 32 |
| 数据类型详细说明 | 33 |
| Boolean | 33 |
| Byte | 34 |
| Double | 35 |
| Int32 | 36 |
| Integer | 37 |
| Long | 38 |
| Real | 39 |
| Short | 40 |
| String | 41 |
| UByte | 42 |
| UInt32 | 43 |
| UShort | 44 |
| Preserve | 45 |
| 运动指令详细说明 | 46 |
| 指令手册 | 46 |

| | |
|------------------|----|
| !...! | 46 |
| Motor | 47 |
| ServoOn、ServoOff | 48 |
| Power | 49 |
| SpeedFactor | 50 |
| Speed | 51 |
| Accel | 52 |
| Go | 53 |
| BGo | 54 |
| TGo | 55 |
| Speeds | 56 |
| AccelS | 57 |
| Move | 58 |
| BMove | 59 |
| TMove | 60 |
| Jump | 61 |
| Arc、Arc3 | 62 |
| CP | 63 |
| CPZone | 64 |
| R180 | 65 |
| Here | 66 |
| Home | 67 |
| Local | 68 |
| TLSet | 70 |
| Tool | 71 |
| TCalib | 72 |
| Till | 73 |
| Agl | 74 |
| PAgl | 75 |
| AglToPls | 76 |
| Arch | 77 |
| Find | 78 |
| FindPos | 79 |
| InPos | 80 |
| JA | 81 |
| Joint | 81 |
| LimZ | 82 |
| Pass | 82 |
| Pls | 83 |
| PPls | 84 |
| JS | 85 |
| PosFound | 86 |
| Pulse | 87 |
| RobLoad | 88 |
| Sense | 89 |
| TillOn | 90 |
| WaitPos | 91 |
| XY | 92 |

| | |
|----------------------------------|------------|
| CX、CY、CZ、CU、CV、CW、CR、CS、CT | 93 |
| TargetOK | 94 |
| ColSup | 95 |
| ZeroPls | 96 |
| Hand | 97 |
| Elbow | 98 |
| Wrist | 99 |
| 独立轴指令详细说明 | 100 |
| IndPower | 100 |
| IndHalt | 101 |
| IndMovDone | 102 |
| IndMovAbs | 103 |
| IndMovRel | 104 |
| IndMovVel | 105 |
| IndReadPos | 106 |
| IndReadVel | 107 |
| IndReadStatus | 108 |
| IndReset | 109 |
| 点位、I/O、编码器等备注指令详细说明 | 110 |
| PDef | 110 |
| PDel | 111 |
| PLabel | 112 |
| PLabel\$ | 113 |
| PNumber | 114 |
| PList | 115 |
| PLocal | 116 |
| PDescription | 117 |
| PDescription\$ | 118 |
| GetPointFiles | 119 |
| DIDescription | 120 |
| DIDescription\$ | 121 |
| DODescription | 122 |
| DODescription\$ | 123 |
| AIDescription | 124 |
| AIDescription\$ | 125 |
| AODescription | 126 |
| AODescription\$ | 127 |
| EncDescription | 128 |
| EncDescription\$ | 129 |
| I/O 指令详细说明 | 130 |
| Wait | 130 |
| Mask | 131 |
| On | 132 |
| Off | 133 |
| Oport | 134 |

| | |
|---------------------|-----|
| Sw | 135 |
| SetSw | 136 |
| In | 137 |
| SetIn | 138 |
| Out | 139 |
| InW | 140 |
| SetInW | 141 |
| OutW | 142 |
| InBCD | 143 |
| OpBCD | 144 |
| InReal | 145 |
| OutReal | 146 |
| AIO_InW | 147 |
| AIO_OutW | 148 |
| 寄存器读写指令详细说明 | 149 |
| MemOn | 149 |
| MemOff | 150 |
| MemSw | 151 |
| MemIn | 152 |
| MemOut | 153 |
| MemInW | 154 |
| MemOutW | 155 |
| MemIn32 | 156 |
| MemOut32 | 157 |
| MemInReal | 158 |
| MemOutReal | 159 |
| SysMemOn | 160 |
| SysMemOff | 161 |
| SysMemSw | 162 |
| SysMemIn | 163 |
| SysMemOut | 164 |
| SysMemInW | 165 |
| SysMemOutW | 166 |
| SysMemIn32 | 167 |
| SysMemOut32 | 168 |
| SysMemInReal | 169 |
| SysMemOutReal | 170 |
| 通讯指令详细说明 | 171 |
| SetNet | 171 |
| OpenNet | 172 |
| WaitNet | 173 |
| ChkNet | 174 |
| ClrNet | 175 |
| CloseNet | 176 |
| SetCom | 177 |
| OpenCom | 178 |

| | |
|--------------------------|------------|
| ChkCom | 179 |
| ClrCom | 179 |
| CloseCom | 180 |
| Read | 180 |
| ReadBin | 181 |
| Write | 181 |
| WriteBin | 182 |
| Print # | 183 |
| Input # | 184 |
| Line Input # | 185 |
| 系统指令详细说明 | 186 |
| Print | 186 |
| Xqt | 187 |
| Pause | 188 |
| Halt | 189 |
| Quit | 190 |
| StartMain | 191 |
| Resume | 192 |
| Reset | 193 |
| MyTask | 194 |
| TaskDone | 195 |
| TaskState | 196 |
| TaskWait | 197 |
| SyncLock | 198 |
| SyncUnLock | 199 |
| Signal | 200 |
| WaitSig | 201 |
| TW | 202 |
| ElapsedTime | 203 |
| ResetElapsedTime | 204 |
| Tmr | 205 |
| TmReset | 206 |
| EVal | 207 |
| Time | 208 |
| Time\$ | 209 |
| Date | 210 |
| Date\$ | 211 |
| 码垛语句详细说明 | 212 |
| Pallet | 212 |
| PalletClr | 214 |
| 传送带跟踪语句详细说明 | 215 |
| SyStart | 215 |
| SyEnd | 216 |
| SyReset | 217 |
| SyGetPoint | 218 |

| | |
|---|------------|
| SyGetData | 219 |
| SyGetPose | 220 |
| SyGetNum | 221 |
| SyGetQuePose | 222 |
| SyMove | 223 |
| SyArc | 224 |
| SyTrig | 225 |
| SyPoint | 226 |
| SyAddVisTarget | 227 |
| SyTimeOffset | 228 |
| SyRejectDis | 229 |
| SyStartDis | 230 |
| SyEndDis | 231 |
| SySave | 232 |
| 数学运算指令详细说明 | 233 |
| +, -, *, /, **, Mod, And, Or, Xor, Not, >, >=, <, <=, <>, = | 233 |
| DegToRad | 234 |
| RadToDeg | 235 |
| Sin, Cos, Tan, Asin, Acos, Atan, Atan2 | 236 |
| HexToFloat | 237 |
| FloatToHex | 237 |
| Abs | 238 |
| Sqr | 238 |
| Sgn | 239 |
| LShift | 239 |
| RShift | 240 |
| BClr | 240 |
| BSet | 241 |
| BTst | 241 |
| Ubound | 242 |
| Dist | 243 |
| 字符串运算指令详细说明 | 244 |
| ParseStr | 244 |
| + | 245 |
| <>, = | 245 |
| Val | 246 |
| Str\$ | 246 |
| Len | 247 |
| Asc | 247 |
| Chr\$ | 248 |
| Left\$ | 248 |
| Mid\$ | 249 |
| Right\$ | 249 |
| LSet\$ | 250 |
| RSet\$ | 250 |
| Space\$ | 251 |

| | |
|-------------------------------|-----|
| LCase\$ | 251 |
| UCase\$ | 252 |
| LTrim\$ | 252 |
| RTrim\$ | 253 |
| Trim\$ | 253 |
| InStr | 254 |
| Tab\$ | 254 |
| Hex\$ | 255 |
| 点位运算指令详细说明 | 256 |
| +、- | 256 |
| / | 257 |
| : | 257 |
| = | 258 |
| @ | 258 |
| X、Y、Z、U、V、W | 259 |
| RX、RY、RZ | 259 |
| TLX、TLY、TLZ、TLU、TLV、TLW | 260 |
| SavePoints | 260 |
| LoadPoints | 261 |
| ClearPoints | 262 |

0S 指令一览

通用语句

| | |
|-----------------------------|---------------------------------------|
| Call | 调用函数 |
| Do..Loop | 循环语句，根据条件在 Do..Loop 之间反复执行 |
| Else/Elseif | 搭配判断语句使用，If 不成立则执行 Else...EndIf 之间的代码 |
| Exit | 强制结束循环或函数 |
| For..To..Next | 反复执行一定次数 For...Next 之间的代码 |
| Function..Fend | 函数定义 |
| Global | 声明全局变量 |
| GoSub..Return | 跳转到子程序 |
| GoTo | 跳转到标签处 |
| If..Then..Else..EndIf | 判断语句，判断 If 成立则执行 Then...EndIf 之间的代码 |
| Select..Case..Default..Send | 根据 Case 结果执行对应语句 |

数据类型

| | |
|----------|----------------------------------|
| Boolean | 布尔类型，True/False |
| Byte | 字节类型，-128~127 |
| Double | 双精度浮点数 |
| Int32 | 32 位整型，-2147483648~2147483647 |
| Integer | 16 位整形，-32768~32767 |
| Long | 长整型，-2147483648~2147483647 |
| Real | 单精度浮点数 |
| Short | 短整型，-32768~32767 |
| String | 字符串，最多包含 255 个字符 |
| UByte | 无符号字节，0~255 |
| UInt32 | 无符号 32 位整形，0~4294967295 |
| UShort | 无符号短整形，0~65535 |
| Preserve | 将全局变量定义为掉电保持状态，停止程序或重启控制器保持变量记忆。 |

运动指令

| | |
|-------------|---|
| !...! | 并行处理运动过程中 I/O 的输入输出 |
| Motor | 整体上下使能 |
| ServoOff | 单轴下使能 |
| ServoOn | 单轴上使能 |
| Power | 设置高低功率模式 |
| SpeedFactor | 用于设置/显示全局运行速度，会影响 Speed 和 SpeedS |
| Speed | 用于设置/显示 Go、Jump、Pulse 命令等 PTP 运动速度 |
| Accel | 设置/显示 PTP 运动的加减速度 |
| Go | 用于在当前位置和指定位置之间进行 PTP 运动 |
| BGo | 用于在已选择的用户坐标系上执行偏移 PTP 运动 |
| TGo | 用于在已选择的工具坐标系上执行偏移 PTP 运动 |
| SpeedS | 用于设置/显示 Move、Arc、Arc3、Jump3、Jump3CP 等 CP 运动速度 |
| AccelS | 设置/显示直线插补运动的加减速度 |
| Move | 用于在当前位置和指定位置之间进行直线插补运动 |
| BMove | 用于在已选择的用户坐标系上执行偏移直线插补运动 |
| TMove | 用于在已选择的工具坐标系上执行偏移直线插补运动 |
| Jump | PTP 门型运动 |
| Arc | 用于在 XY 平面上进行圆弧插补运动 |
| Arc3 | 用于在立体空间上进行圆弧插补运动 |
| CP | 用于设置/显示运动平滑 |
| CPZone | 用于设置/显示连续运动平滑参数 |
| R180 | 用于约束 U 轴正负角度至 ±180 度 |
| Here | 用于示教/返回机器人当前坐标 |
| Home | 返回 Home 点 |
| Local | 用于设置/显示用户坐标系 |
| TLSet | 用于设置/显示工具坐标系 |
| Tool | 用于选择/显示指定编号的工具坐标 |
| TCalib | 用于三点定义工具坐标系 |
| Till | 用于在运动过程中判断条件成立并停止运动 |
| Agl | 用于返回指定关节的角度或位置 |

| | |
|----------|--|
| PAgl | 用于返回指定点位的关节角度 |
| AglToPls | 用于将机器人各关节角度转换为脉冲 |
| Arch | 用于设置/显示 Jump、Jump3、Jump3CP 命令的 Arch 参数 |
| Find | 用于设置/显示在动作命令中保存坐标的条件 |
| FindPos | 用于在执行动作命令过程中通过 Find 返回保存的坐标 |
| InPos | 返回机器人是否到位 |
| JA | 以指定关节角度返回机器人坐标 |
| Joint | 用于在关节坐标系显示机器人当前坐标 |
| LimZ | 用于设置/显示 Jump 命令时第 3 关节高度(Z 坐标值)初始值 |
| Pass | 用于进行穿过指定点附近且不停止的 PTP 运动 |
| Pls | 用于返回当前位置各关节的脉冲值 |
| PPls | 用于指定坐标点的关节脉冲值 |
| JS | 用于返回 Sense 输入是否成立 |
| PosFound | 用于返回 Find 命令时执行的状态 |
| Pulse | 以 PTP 运动到各关节脉冲指定位置处, 或根据脉冲值返回点位 |
| RobLoad | 用于选择/显示指定编号的负载 |
| Sense | 用于设置/显示 Jump、Jump3、Jump3CP 停在目标坐标上方的条件 |
| TillOn | 用于返回 Till 的状态 |
| WaitPos | 等待机器人运动停止 |
| XY | 以指定数值返回点位 |
| CX | 用于设置/获取点位的 X 坐标 |
| CY | 用于设置/获取点位的 Y 坐标 |
| CZ | 用于设置/获取点位的 Z 坐标 |
| CU | 用于设置/获取点位的 U 坐标 |
| CV | 用于设置/获取点位的 V 坐标 |
| CW | 用于设置/获取点位的 W 坐标 |
| CR | 用于设置/获取点位的 R 坐标 |
| CS | 用于设置/获取点位的 S 坐标 |
| CT | 用于设置/获取点位的 T 坐标 |
| TargetOK | 用于检测指定点位坐标是否可达 |
| ColSup | 用于设置/获取碰撞检测系数 |

| | |
|---------|--------------|
| ZeroPls | 用于设置/获取机器人零位 |
| Hand | 用于设置/获取当前手系 |
| Elbow | 用于设置/获取肘部姿态 |
| Wrist | 用于设置/获取腕部姿态 |

独立轴指令

| | |
|---------------|-------------------|
| IndPower | 用于设置独立轴使能状态 |
| IndHalt | 用于控制独立轴停止运动 |
| IndMovDone | 用于等待独立轴运动完成 |
| IndMovAbs | 用于控制独立轴运动至指定的绝对位置 |
| IndMoveRel | 用于控制独立轴运动至相对运动 |
| IndMovVel | 用于控制独立轴保持恒速速度运动 |
| IndReadPos | 用于读取独立轴当前位置 |
| IndReadVel | 用于读取独立轴当前运动速度 |
| IndReadStatus | 用于读取独立轴当前状态 |
| IndReset | 用于清除独立轴错误 |

点位、I/O、编码器等备注指令

| | |
|-----------------|--------------|
| PDef | 用于获取点位是否存在定义 |
| PDel | 用于删除点位 |
| PLabel | 用于设置点位数据的标签 |
| PLabel\$ | 用于获取点位标签 |
| PNumber | 用于获取点编号 |
| PList | 用于获取点位数据信息 |
| PLocal | 用于返回点位的用户坐标 |
| PDescription | 用于设置点位的描述 |
| PDescription\$ | 用于返回点的描述 |
| GetPointFiles | 用于获取点位文件列表 |
| DIDescription | 设置数字量输入数据的描述 |
| DIDescription\$ | 返回数字量输入数据的描述 |
| DODescription | 设置数字量输出的数据描述 |

| | |
|------------------|--------------|
| DODescription\$ | 返回数字量输出的数据描述 |
| AIDescription | 设置模拟量输入的数据描述 |
| AIDescription\$ | 返回模拟量输入的数据描述 |
| AODescription | 设置模拟量输出的数据描述 |
| AODescription\$ | 返回模拟量输出的数据描述 |
| EncDescription | 设置编码器数据的描述 |
| EncDescription\$ | 返回编码器数据的描述 |

I/O 指令

| | |
|----------|------------------------------------|
| Wait | 延时指令，或在一定时间内等待条件成立 |
| Mask | 以位为单位，用于对表示 Wait 输入条件的值进行位 And 运算。 |
| On | 将 DO 置为 ON，或将 DO 置 ON 一定时间后置 OFF |
| Off | 将 DO 置为 OFF，或将 DO 置 ON 一定时间后置 ON |
| Oport | 用于返回指定 DO 的状态 |
| Sw | 用于返回指定 DI 的状态 |
| SetSw | 用于将指定 DI 置 ON 或置 OFF |
| In | 根据 8421 返回 8 位 DI 的状态，0~255 |
| SetIn | 根据 8421 设置 8 位 DI 的状态，0~255 |
| Out | 根据 8421 设置/返回 8 位 DO 的状态，0~255 |
| InW | 根据 8421 返回 16 位 DI 的状态，0~65535 |
| SetInW | 根据 8421 设置 16 位 DI 的状态，0~65535 |
| OutW | 根据 8421 设置/返回 16 位 DO 的状态，0~65535 |
| InBCD | 根据 8421 返回 8 位 DI 的状态，0~99 |
| OpBCD | 根据 8421 设置 8 位 DO 的状态，0~99 |
| InReal | 以 32 位浮点数返回 DI 值 |
| OutReal | 以 32 位浮点数设置/返回 DO 值 |
| AIO_InW | 返回模拟量输入的值 |
| AIO_OutW | 设置/返回模拟量输出的值 |

寄存器读写指令

| | |
|---------------|-------------------------|
| MemOn | 设置用户寄存器 1 位为 1 |
| MemOff | 设置用户寄存器 1 位为 0 |
| MemSw | 返回用户寄存器 1 位的值 |
| MemIn | 返回用户寄存器 8 位的值 |
| MemOut | 设置用户寄存器 8 位的值 |
| MemInW | 返回用户寄存器 16 位的值 |
| MemOutW | 设置用户寄存器 16 位的值 |
| MemIn32 | 返回用户寄存器 32 位的值 |
| MemOut32 | 设置用户寄存器 32 位的值 |
| MemInReal | 以 32 位浮点数返回用户寄存器 32 位的值 |
| MemOutReal | 以 32 位浮点数设置用户寄存器 32 位的值 |
| SysMemOn | 设置系统寄存器 1 位为 1 |
| SysMemOff | 设置系统寄存器 1 位为 0 |
| SysMemSw | 返回系统寄存器 1 位的值 |
| SysMemIn | 返回系统寄存器 8 位的值 |
| SysMemOut | 设置系统寄存器 8 位的值 |
| SysMemInW | 返回系统寄存器 16 位的值 |
| SysMemOutW | 设置系统寄存器 16 位的值 |
| SysMemIn32 | 返回系统寄存器 32 位的值 |
| SysMemOut32 | 设置系统寄存器 32 位的值 |
| SysMemInReal | 以 32 位浮点数返回系统寄存器 32 位的值 |
| SysMemOutReal | 以 32 位浮点数设置系统寄存器 32 位的值 |

通讯指令

| | |
|---------|--------------------|
| SetNet | 用于设置 TCP/IP 端口参数 |
| OpenNet | 用于打开 TCP/IP 端口 |
| WaitNet | 用于等待 TCP/IP 端口建立连接 |

| | |
|--------------|----------------------------|
| ChkNet | 用于返回网路端口的接收缓冲器内的字节数 |
| ClrNet | 用于清空网路端口接收缓冲器 |
| CloseNet | 用于关闭 OpenNet 打开的 TCP/IP 端口 |
| SetCom | 用于设置串口通讯参数 |
| OpenCom | 用于打开串口通讯参数 |
| ChkCom | 用于串口端口的接收缓存器内的字节数 |
| ClrCom | 用于清空串口端口接收缓冲器 |
| CloseCom | 用于关闭 OpenCom 打开的串口端口 |
| Read | 用于从通讯端口读取指定的字节数 |
| ReadBin | 用于从通讯端口读取二进制数据 |
| Write | 将字符串写入到通讯端口中 |
| WriteBin | 将二进制数据写入到通讯端口中 |
| Print # | 将数据输出到指定的通讯端口中 |
| Input # | 用于从通讯端口中读取字符串或数据 |
| Line Input # | 用于从通讯端口中读取 1 行数据 |

系统指令

| | |
|------------|---------------------------|
| Print | 打印字符串或数据 |
| Xqt | 启动多线程 |
| Pause | 暂停所有线程 |
| Halt | 暂停指定线程 |
| Quit | 停止所有或指定线程 |
| Resume | 继续所有或指定线程 |
| Reset | 将控制器重置为初始状态 |
| MyTask | 返回当前线程编号 |
| TaskDone | 用于确认线程是否结束 |
| TaskState | 用于返回指定线程的状态 |
| TaskWait | 用于等待指定线程结束 |
| SyncLock | 用于相互排他锁定，使多个线程同步 |
| SyncUnLock | 用于解锁 SyncLock 锁定的信号 |
| Signal | 用于向正在执行 WaitSig 命令的任务发送信号 |

| | |
|------------------|---------------------------------|
| WaitSig | 用于等待其他任务的 Signal 命令发出的同步信号 |
| TW | 用于返回 Wait、WaitNet、WaitSig 命令的状态 |
| ElapsedTime | 以秒为单位返回计算节拍时间计时器开始计时之后经过的时间 |
| ResetElapsedTime | 重置 ElapsedTime 计时器 |
| Tmr | 以秒为单位返回计时器开始计时之后经过的时间 |
| TmReset | 重置 Tmr 计时器 |
| EVal | 用于执行命令窗口的语句。 |
| Time | 用于打印/返回控制器时间 |
| Time\$ | 用于以指定格式 hh:mm:ss 格式返回控制器系统时间 |
| Date | 用于打印/返回控制器系统日期 |
| Date\$ | 用于以指定格式 yyyy/MM/dd 格式返回控制器系统日期 |

码垛指令

| | |
|-----------|------------------|
| Pallet | 定义阵列、获取阵列点位、打印阵列 |
| PalletClr | 清空阵列 |

传送带跟踪指令

| | |
|----------------|---------------------|
| SyStart | 开始传送带跟踪功能 |
| SyEnd | 结束传送带跟踪功能 |
| SyReset | 清空当前传送带跟踪队列数据 |
| SyGetPoint | 获取可进行传送带跟踪的点位数据 |
| SyGetUserData | 获取当前跟踪目标的附加信息 |
| SyGetPose | 获取当前跟踪目标在传送带上的位置 |
| SyGetQuePose | 获取跟踪队列中点位当前在传送带上的位置 |
| SyGetNum | 获取传送带跟踪队列中的目标个数 |
| SyMove | 传送带跟踪直线插补运动 |
| SyArc | 传送带跟踪圆弧插补运动 |
| SyTrig | 记录指定传动带编码器脉冲数值 |
| SyPoint | 根据当前传送带追踪生成追踪点位 |
| SyAddVisTarget | 将视觉坐标注册到传送带跟踪队列中 |

| | |
|--------------|------------------|
| SyRejectDis | 设置/获取传送带跟踪功能滤重间距 |
| SyTimeOffset | 设置/获取跟踪时间偏移 |
| SyStartDis | 设置/获取跟踪开始距离 |
| SyEndDis | 设置/获取跟踪结束距离 |
| SySave | 保存程序中对传送带跟踪功能的修改 |

数学运算指令

| | |
|----------|---------|
| + | 加 |
| - | 减 |
| * | 乘 |
| / | 除 |
| ** | 乘方 |
| Mod | 整数取模 |
| And | 与 |
| Or | 或 |
| Xor | 异或 |
| Not | 非 |
| > | 大于 |
| >= | 大于等于 |
| < | 小于 |
| <= | 小于等于 |
| <> | 不等于 |
| = | 等于、赋值 |
| DegToRad | 角度转弧度 |
| RadToDeg | 弧度转角度 |
| Sin | 正弦函数 |
| Cos | 余弦函数 |
| Tan | 正切函数 |
| Asin | 反正弦函数 |
| Acos | 反余弦函数 |
| Atan | 反正切函数 |
| Atan2 | 反正切函数 2 |

| | |
|------------|-----------------------|
| HexToFloat | 十六进制浮点数转单精度浮点数 |
| FloatToHex | 单精度浮点数转十六进制浮点数 |
| Abs | 绝对值 |
| Sqr | 平方根 |
| Sgn | 返回数值的符号 |
| LShift | 左移 |
| RShift | 右移 |
| BClr | 将数值指定的 Bit 置 0 并返回该数值 |
| BSet | 将数值指定的 Bit 置 1 并返回该数值 |
| BTst | 返回数值指定 Bit 的值 |
| Ubound | 获取数组长度 |
| Dist | 返回两点之间的直线距离 |

字符串运算指令

| | |
|----------|--------------------|
| ParseStr | 字符串分割 |
| + | 字符串拼接 |
| <> | 不等于 |
| = | 等于、赋值 |
| Val | 字符串转数值 |
| Str\$ | 数值转字符串 |
| Len | 获取字符串长度 |
| Asc | 字符转 ASCII 码 |
| Chr\$ | ASCII 码转字符 |
| Left\$ | 从字符串左侧开始提取指定的字符 |
| Mid\$ | 在字符串中提取指定范围的字符或字符串 |
| Right\$ | 从字符串右侧开始提取指定的字符 |
| LSet\$ | 从字符串左侧开始获取指定长度的字符串 |
| RSet\$ | 从字符串右侧开始获取指定长度的字符串 |
| Space\$ | 返回指定长度的空格字符串 |
| LCase\$ | 返回小写字母的字符串 |
| UCase\$ | 返回大写字母的字符串 |
| LTrim\$ | 删除字符串左侧的空格并返回 |

| | |
|---------|-------------------|
| RTrim\$ | 删除字符串右侧的空格并返回 |
| Trim\$ | 删除字符串两侧的空格并返回 |
| InStr | 从字符串中检索指定字符的位置并返回 |
| Tab\$ | 返回指定数量的制表符字符串 |
| Hex\$ | 将十六进制数值转换为字符串并返回 |

点位运算指令

| | |
|-------------|------------------|
| + | 相对坐标增量运动 |
| - | 相对坐标负增量运动 |
| / | 修改手系、修改用户坐标系 |
| : | 绝对坐标运动 |
| = | 赋值 |
| @ | 转换到指定用户坐标系 |
| X | X 方向分量运算 |
| Y | Y 方向分量运算 |
| Z | Z 方向分量运算 |
| U | U 方向分量运算 |
| V | V 方向分量运算 |
| W | W 方向分量运算 |
| RX | 用户坐标系统 X 轴旋转分量运算 |
| RY | 用户坐标系统 Y 轴旋转分量运算 |
| RZ | 用户坐标系统 Z 轴旋转分量运算 |
| TLX | 工具坐标系统 X 轴旋转分量运算 |
| TLY | 工具坐标系统 Y 轴旋转分量运算 |
| TLZ | 工具坐标系统 Z 轴旋转分量运算 |
| TLU | 工具坐标系统 U 轴旋转分量运算 |
| TLV | 工具坐标系统 V 轴旋转分量运算 |
| TLW | 工具坐标系统 W 轴旋转分量运算 |
| SavePoints | 保存点位文件 |
| LoadPoints | 加载点位文件 |
| ClearPoints | 清空点位 |

文本符号声明：

1. {参数} 大括号指大括号中的参数是可省略不写的，并不是强制的语法要求。
2. [参数] 中括号指中括号中的参数必须任选其一，必须要填入，属于语法强制要求。

通用语句详细说明

Call

[功能]

调用函数。

[语法]

Call 函数名

[参数说明]

函数名 需要调用的 Function 名。

[使用案例]

```
Function main
  Call Task2
  Task2
  Task2()
Fend
```

Do..Loop

[功能]

根据条件是否成立，在 Do..Loop 之间反复执行代码。

[语法]

```
Do {While / Until 条件表达式}
```

```
...
```

```
Loop
```

或

```
Do
```

```
...
```

```
Loop {While/Until 条件表达式}
```

[参数说明]

While/Until 可在 Do 或 Loop 后面追加关键字来使用条件表达式。

条件表达式 根据条件表达式是否成立来跳出 Do..Loop 循环。

[使用案例]

```
Do                            '一直循环
```

```
  Go P1  
  Wait 1  
  Go P2  
  Wait 2
```

```
Loop
```

```
  Do Until Sw(1) <> 1 '当 DI1 不等于 1 时进入循环
```

```
  Go P1  
  Wait 1  
  Go P2  
  Wait 2
```

```
Loop
```

```
Do                            '一直循环
```

```
  Go P1  
  Wait 1  
  Go P2  
  Wait 2
```

```
Loop Until ChkNet(201) < 0 '当 TCP/IP 201 端口通讯异常时跳出循环
```

Else/ElseIf

[功能]

搭配判断语句使用，当 If 条件不成立时，执行 Else/ElseIf 内的代码。

[语法]

If 条件表达式 Then

...

ElseIf 条件表达式 Then

...

Else

...

EndIf

[参数说明]

条件表达式 根据条件表达式是否成立来执行 Else/ElseIf 内代码。

[使用案例]

```
If Sw(1) = 1 Then  
    Print "DI1 为 On"
```

```
ElseIf Sw(1) = 0 Then  
    Print "DI1 为 Off"
```

```
Else  
    Print "DI1 异常"
```

```
EndIf
```

Exit

[功能]

用于强制结束循环或函数。

[语法]

```
Exit [Do/For/Function]
```

[参数说明]

Exit Do 退出 Do..Loop 循环。如存在多层 Do..Loop 嵌套，Exit Do 只会退出一级循环。

Exit For 退出 For 循环。如存在多层 For 嵌套，Exit For 只会退出一级循环。

Exit Function 退出 Function。如存在多层 Function 嵌套，Exit Function 只会退出至上一级 Function。

[使用案例]

```
Do                                '一直循环
  Go P1
  Go P2
  If Sw(1) = 1 Then '当 DI1 等于 1 时进入判断
    Exit Do        '退出 Do 循环
  EndIf
Loop

Integer var

For var = 1 To 5 '循环 5 次
  Go P1
  Go P2
  If Sw(1) = 1 Then '当 DI1 等于 1 时进入判断
    Exit For      '退出 For 循环
  EndIf
Next

Function main
  Exit Function  '退出函数
Fend
```

For..To..Next

[功能]

反复执行一定次数 For..Next 之间的代码。

[语法]

For 变量 = 起始值 To 结束值 {Step 增量值}

...

Next

[参数说明]

| | |
|-----|-----------------------------|
| 变量 | 用于在 For 循环中迭代的变量，该变量需要提前定义。 |
| 起始值 | 变量在 For 循环开始时的起始值。 |
| 结束值 | 变量在 For 循环结束时的结束值。 |
| 增量值 | 每完成一次循环后的叠加的数值，默认为叠加 1 步。 |

[使用案例]

```
Integer var  
For var = 1 To 5 Step 1 '循环 5 次  
    Go P1  
    Go P2  
    Print var  
Next
```

Function..Fend

[功能]

全局函数定义。

[语法]

Function 函数名 {(自变量 **As** 数据类型, 自变量 数据类型, ...)} {**As** 数据类型}

...

Fend

[参数说明]

函数名 必须定义的 Function 名。

自变量列表 需要在 Function 中传递数值或数值运算的形式参数，多个变量时用逗号进行分隔。

ByRef 调用形参为数组时需配合该指令使用，可修改自变量索引值。

ByVal 调用形参为数组时需配合该指令使用，不可修改自变量索引值。

自变量 As 数据类型 必要的参数。请声明自变量的类型。

函数 As 数据类型 将函数名生成为变量并存储返回值。请声明函数名所属的变量类型。

[使用案例]

Function aa '普通的 Function

```
Print "123"
Print "456"
Print "789"
```

Fend

aa '调用

Function Trig_CCD(IO_Idx **As** Integer, Integer_Time **As** Double) '带自变量的 Function

```
Do
    On IO_Idx
    Wait Integer_Time
    Off IO_Idx
```

Loop

Fend

Function Add(Num1 **As** Integer, Num2 **As** Integer) **As** Integer '带自变量且带返回值的 Function

```
Integer result
result = Num1 + Num2
Add = result
```

Fend

```
Function main                                     ‘调用 Return_Array 函数改变数组状态值
    Integer AAA(2)
    AAA(0) = 10
    AAA(1) = 20
    test(ByRef AAA())
    Print AAA(0), AAA(1)                           ‘打印结果为 20 20
Fend
```

```
Function Return_Array(ByRef BBB() As Integer) ‘通过给进的数组值并改写
    BBB(0) = BBB(0)*2
Fend
```

Global

[功能]

全局变量定义。

[语法]

Global 数据类型 变量名

[参数说明]

变量名 必须定义的变量名称。

注意事项：

1. 全局变量只能在函数外定义，不能在函数内定义。
2. 全局变量总数不得超过 100000 个。String 类型不得超过 10000 个。

[使用案例]

| | | |
|----------------|----------------------|-----------|
| Global String | PointList\$(300, 24) | '字符串二维数组 |
| Global Boolean | P_Mode | '布尔类型 |
| Global String | CCD_Dev_X\$ | '字符串类型 |
| Global Double | SafeHight | '双精度浮点数类型 |
| Global Integer | ABCD | '整形 |
| Global Integer | AABB(10) | '整形一维数组 |

GoSub . Return

[功能]

GoSub 到标签处，将程序控制权移交给子程序，子程序通过 Return 将程序返回至原来 GoSub 的位置。

[语法]

GoSub [标签]

...

[标签]:

...

Return

[参数说明]

标签 必须定义的标签名称。

[使用案例]

```
Function main
    GoSub checkin          ' 跳转至标签处
        On 1
        On 2
    Exit Function

Checkin:                  ' 标签
    If In(0) = 1 And In(1) = 1 Then
        On 1
    Else
        Off 1
    EndIf
    Return                 ' 返回 GoSub 处
Fend
```

GoTo

[功能]

跳转到标签处继续执行程序。

[语法]

GoTo [标签]

...

[标签]:

...

[参数说明]

标签 必须定义的标签名称。

注意事项:

1. 与 GoSub 的区别在于，GoTo 没有 Return 操作。
2. 标签名称不可与函数名称、变量名称、点位编号、点标签相同。
3. GoTo 指令不允许跨函数跳转。

[使用案例]

```
Function main
    If Sw(1) = Off Then
        GoTo mainAbort
    EndIf
    Print "DI1 为 On"

Exit Function

mainAbort:
    Print "DI1 为 Off"
Fend
```

If..Then..Else..EndIf

[功能]

判断语句，根据条件表达式执行对应的程序。

[语法]

```
If 条件表达式 Then
```

```
...
```

```
Elseif 条件表达式 Then
```

```
...
```

```
Else
```

```
...
```

```
EndIf
```

[参数说明]

条件表达式 根据条件表达式是否成立来执行对应的程序。

[使用案例]

```
Function main
    String AA$
    Integer num

    Input #201, AA$

    Num = val(AA$)                    '接收内容转换成数值并赋值给 Num

    If num > 1 Then
        Print    "num 大于 1"
    ElseIf num < 1 Then
        Print    "num 小于 1"
    Else
        Print    "num 等于 1"
    EndIf
EndFunction
```

Select..Case..Default..Send

[功能]

选择语句，根据值来决定执行那个段落的程序。

[语法]

Select 变量

Case 值 1

...

{Case 值 2

...}

Default

...

Send

[参数说明]

变量 用来作为条件判断的变量名。

值 1、值 2 判断变量是否等于该值，进入该程序段。

Default 当所有 Case 不成立时，执行 Default 的程序段。

[使用案例]

```
Function main
```

```
    Integer I
```

```
    For I = 0 To 10
```

```
        Select I
```

```
            Case 0
```

```
                Print "I = 1"
```

```
                Off 1; On 2; Jump P1
```

```
            Case 3
```

```
                Print "I = 3"
```

```
                On 1; Off 2
```

```
                Jump P2; Move P3; On 3
```

```
            Case 7
```

```
                Print "I = 3"
```

```
                On 4
```

```
            Default
```

```
                On 7
```

```
        Send
```

```
    Next
```

```
End
```

数据类型详细说明

Boolean

[功能]

布尔类型，2 字节。

[语法]

Boolean 变量{(数组元素个数)}

Global Boolean 变量{(数组元素个数)}

[参数说明]

变量 必须定义的变量名。

取值范围 True/False。

注意事项：

1. 局部变量最大数量：2000
2. 全局变量最大数量：100000

[使用案例]

```
Function main
```

```
    Boolean AA
```

```
    If AA = True Then
```

```
        Print    "AA 为真"
```

```
    ElseIf AA = False Then
```

```
        Print    "AA 为假"
```

```
    EndIf
```

```
Fend
```

Byte

[功能]

字节类型，2 字节。

[语法]

Byte 变量 {(数组元素个数)}

Global Byte 变量 {(数组元素个数)}

[参数说明]

变量 必须定义的变量名。

取值范围 -128~127。

注意事项：

3. 局部变量最大数量：2000
4. 全局变量最大数量：100000

[使用案例]

```
Function main
    Byte A1(10)           'Byte 类型的一维数组
    Byte B1(10, 10)      'Byte 类型的二维数组
    Byte CC(5, 5, 5)     'Byte 类型的三维数组
    Byte Test_ok

    Test_ok = 15

    Test_ok = (test_ok And 8)

    If test_ok <> 8 Then
        Print "高位 bit 为 ON"
    Else
        Print "高位 bit 为 OFF"
    EndIf
Fend
```

Double

[功能]

双精度浮点数，8 字节。

[语法]

`Double` 变量{(数组元素个数)}

`Global Double` 变量{(数组元素个数)}

[参数说明]

变量 必须定义的变量名。

取值范围 可取值到小数点后 14 位。

注意事项：

5. 局部变量最大数量：2000
6. 全局变量最大数量：100000

[使用案例]

```
Function main
    Double var1
    Double A1(10)           'Double 类型的一维数组
    Double B1(5,5)         'Double 类型的二维数组
    Double CC(5,5,5)       'Double 类型的三维数组
    Double arrayvar(10)
    Integer i

    For i = 1 To 5
        Print arrayvar(i)
    Next
End
```

Int32

[功能]

32 位整型，4 字节。

[语法]

`Int32` 变量{(数组元素个数)}

`Global Int32` 变量{(数组元素个数)}

[参数说明]

变量 必须定义的变量名。

取值范围 -2147483648~2147483647。

注意事项：

7. 局部变量最大数量：2000
8. 全局变量最大数量：100000

[使用案例]

`Function` main

`Int32` A1 (10) ' Int32 类型的一维数组

`Int32` B1 (5, 5) ' Int32 类型的二维数组

`Int32` CC (5, 5, 5) ' Int32 类型的三维数组

`Int32` var1, arrayvar (10)

`End`

Integer

[功能]

整型类型，2 字节。

[语法]

`Integer` 变量{(数组元素个数)}

`Global Integer` 变量{(数组元素个数)}

[参数说明]

变量 必须定义的变量名。

取值范围 -32768~32767。

注意事项：

9. 局部变量最大数量：2000
10. 全局变量最大数量：100000

[使用案例]

```
Function main
    Integer A1(10)           ' Integer 类型的一维数组
    Integer B1(5,5)         ' Integer 类型的二维数组
    Integer CC(5,5,5)       ' Integer 类型的三维数组
    Integer var1,arrayvar(10)
Fend
```

Long

[功能]

长整型类型，4 字节。

[语法]

Long 变量{(数组元素个数)}

Global Long 变量{(数组元素个数)}

[参数说明]

变量 必须定义的变量名。

取值范围 -2147483648~2147483647。

注意事项：

11. 局部变量最大数量：2000
12. 全局变量最大数量：100000

[使用案例]

```
Function main
    Long A1(10)                 'Long 类型的一维数组
    Long B1(5,5)                'Long 类型的二维数组
    Long CC(5,5,5)              'Long 类型的三维数组
    Long var1,arrayvar(10)
Fend
```

Real

[功能]

单精度浮点数，4 字节。

[语法]

Real 变量{(数组元素个数)}

Global Real 变量{(数组元素个数)}

[参数说明]

变量 必须定义的变量名。

取值范围 小数点后六位。

注意事项：

13. 局部变量最大数量：2000
14. 全局变量最大数量：100000

[使用案例]

```
Function main
    Real A1(10)            'Real 类型的一维数组
    Real B1(5,5)          'Real 类型的二维数组
    Real CC(5,5,5)        'Real 类型的三维数组
    Real var1,arrayvar(10)
End
```

Short

[功能]

短整型，2 字节。

[语法]

Short 变量{(数组元素个数)}

Global Short 变量{(数组元素个数)}

[参数说明]

变量 必须定义的变量名。

取值范围 -32768~32767。

注意事项：

15. 局部变量最大数量：2000
16. 全局变量最大数量：100000

[使用案例]

```
Function main
    Short A1(10)           'Short 类型的一维数组
    Short B1(5,5)         'Short 类型的二维数组
    Short CC(5,5,5)       'Short 类型的三维数组
    Short var1,arrayvar(10)
Fend
```

String

[功能]

字符串类型。

[语法]

`String` 变量\${(数组元素个数)}

`Global String` 变量\${(数组元素个数)}

[参数说明]

变量\$ 必须定义的变量名。

取值范围 最大 255 个字符。

注意事项：

17. 局部变量最大数量：2000
18. 全局变量最大数量：10000

[使用案例]

```
Function main
    String A1$(10)           'String 类型的一维数组
    String B1$(5,5)         'String 类型的二维数组
    String CC$(1,2,3)       'String 类型的三维数组
    String var1$,arrayvar$(10)
Fend
```

UByte

[功能]

无符号字节类型，2 字节。

[语法]

UByte 变量 {(数组元素个数)}

Global UByte 变量 {(数组元素个数)}

[参数说明]

变量 必须定义的变量名。

取值范围 0~255。

注意事项：

19. 局部变量最大数量：2000
20. 全局变量最大数量：100000

[使用案例]

```
Function main
    UByte A1 (10)           'UByte 类型的一维数组
    UByte B1 (5, 5)        'UByte 类型的二维数组
    UByte CC (1, 2, 3)     'UByte 类型的三维数组
    UByte var1, arrayvar (10)
Fend
```

UInt32

[功能]

无符号 32 位整型，4 字节。

[语法]

`UInt32` 变量{(数组元素个数)}

`Global UInt32` 变量{(数组元素个数)}

[参数说明]

变量 必须定义的变量名。

取值范围 0~4294967295。

注意事项：

21. 局部变量最大数量：2000
22. 全局变量最大数量：100000

[使用案例]

`Function` main

`UInt32` A1(10) 'UInt32 类型的一维数组

`UInt32` B1(5, 5) 'UInt32 类型的二维数组

`UInt32` CC(1, 2, 3) 'UInt32 类型的三维数组

`UInt32` var1, arrayvar(10)

`End`

UShort

[功能]

无符号短整型，2 字节。

[语法]

UShort 变量{(数组元素个数)}

Global UShort 变量{(数组元素个数)}

[参数说明]

变量 必须定义的变量名。

取值范围 0~65535。

注意事项：

23. 局部变量最大数量：2000
24. 全局变量最大数量：100000

[使用案例]

```
Function main
    UShort A1(10)           'UShort 类型的一维数组
    UShort B1(5, 5)        'UShort 类型的二维数组
    UShort CC(1, 2, 3)     'UShort 类型的三维数组
    UShort var1, arrayvar(10)
Fend
```

Preserve

[功能]

将全局变量定义为掉电保持状态，停止程序或重启控制器保持变量记忆。

[语法]

`Global Preserve Integer` 变量{(数组元素个数)}

[参数说明]

变量 必须定义的变量名。

注意事项：

25. 掉电保持最大数量：4000

[使用案例]

`Global Preserve Integer Hi`

`Global Preserve Integer OS`

运动指令详细说明

!...!

[功能]

并行处理运动过程中 I/O 等的输入输出。

[语法]

运动指令 !处理语句!

[参数说明]

运动指令 Go、Move、Arc、Jump 等相关的运动指令。

处理语句 如下。

1. D: 百分比处理语句。
2. On/Off、MemOff、Out、OutW、MemOut、Signal、Wait、Print、Print#等 I/O 指令。详情请查阅 I/O 指令详细说明。

[使用案例]

Function main

Go P1 !D50; **On** 1!

'PTP 运动至 P1 点过程中, 运动至 50%路径时 D01 置 ON

Jump P2 !D30; **On** 1; D70; **Off** 1!

'门型运动至 P2 点过程中, 运动至 30%路程时 D01 置 ON, 70%时 D01 置 OFF

Move P3 !D10; **On** 5; **Wait** 0.5; **Off** 5!

'直线运动至 P3 过程中, 运动至 10%路程时 D05 置 ON, 并在 0.5 秒后 D05 置 OFF

Fend

Motor

[功能]

机器人整体上/下使能、或返回机器人使能状态。

[语法]

Motor {On/Off}

[参数说明]

On/Off On 为整体上使能，Off 为整体下使能。省略不填时则返回机器人使能状态。

[使用案例]

```
Function main
  If Motor = Off Then
    Motor On
  EndIf
Fend
```

[回传说明]

回传数据 1 0 / 1 , 0 为 Off, 1 为 On。

ServoOn 、 ServoOff

[功能]

单轴上/下使能、或返回单轴上/下使能状态。

[语法]

上下使能：

```
ServoOn 关节编号 1 {, 关节编号 2...}
```

```
ServoOff 关节编号 1 {, 关节编号 2...}
```

获取使能状态：

```
ServoOn(关节编号)
```

```
ServoOff(关节编号)
```

[参数说明]

关节编号 以 1、2、3、4、5、6 来指定需要上下使能的关节。或需要返回使能状态的关节。

[使用案例]

```
Function main
  If ServoOff = True Then     '判断 1 关节使能状态
    ServoOn 1                 '1 关节上使能
  EndIf
Fend
```

[回传说明]

回传数据 1 TRUE / FALSE ， TRUE 为真，FALSE 为假。

Power

[功能]

切换高低功率模式、或返回当前功率模式。

[语法]

```
Power {High/Low}
```

[参数说明]

High/Low High 为高功率、Low 为低功率。省略不填则返回当前功率状态。

[使用案例]

```
Function main
    If Power = Low Then     '判断当前是否处于低功率
        Power High         '切换高功率
    EndIf
Fend
```

[回传说明]

回传数据 1 0 / 1, 0 为低功率模式, 1 为高功率模式。

SpeedFactor

[功能]

设置全局运动速度，或返回当前全局速度。

[语法]

SpeedFactor {百分比}

[参数说明]

百分比 以 1~100%来设定当前的全局速度。省略不填则返回当前的全局速度。

[使用案例]

Function main

Motor On

Power High

Speed 100 'PTP 速度

Accel 100, 100

SpeedS 2000 'Line 速度

AccelS 100, 100

SpeedFactor 80 '全局速度

Print SpeedFactor '打印当前全局速度

End

[回传说明]

回传数据 1 百分比数值 1 -100 %，数据类型默认为浮点数。

Speed

[功能]

设置 PTP 运动的运行速度，或返回当前 PTP 运动的运行速度。

[语法]

`Speed` {百分比[, 转速速度百分比, 接近速度百分比]}

[参数说明]

| | |
|---------|----------------------------------|
| 百分比 | 以 1~100%来设定当前的 PTP 运动速度。 |
| 转速速度百分比 | 以 1~100%来设定 Jump 命令时的转移动作速度，可省略。 |
| 接近速度百分比 | 以 1~100%来设定 Jump 命令时的转移动作速度，可省略。 |

注意事项：若程序中未使用该指令定义 PTP 运动速度时，将使用默认速度：20%

[使用案例]

```
Function main
  Motor On
  Power High

  Speed 100      'PTP 速度
  Accel 100,100
  SpeedS 2000   'Line 速度
  AccelS 100,100

  SpeedFactor 80 '全局速度

  Print SpeedFactor '打印当前全局速度
End
```

[回传说明]

回传数据 1 百分比数值 1 - 100 %，数据类型默认为浮点数。

Accel

[功能]

设置 PTP 运动的运行加减速速度，或返回当前 PTP 运动的运行加减速速度。

[语法]

`Accel` {加速度百分比, 减速度百分比, {转速加速度百分比, 转速减速度百分比, 接近加速度百分比, 接近减速度百分比}}

[参数说明]

| | |
|----------|-----------------------------------|
| 加速度百分比 | 以 1~100%来设定当前的 PTP 运动加速度。 |
| 减速度百分比 | 以 1~100%来设定当前的 PTP 运动减速度。 |
| 转速加速度百分比 | 以 1~100%来设定 Jump 命令时的转移动作加速度，可省略。 |
| 转速减速度百分比 | 以 1~100%来设定 Jump 命令时的转移动作减速度，可省略。 |
| 接近加速度百分比 | 以 1~100%来设定 Jump 命令时的接近动作加速度，可省略。 |
| 接近减速度百分比 | 以 1~100%来设定 Jump 命令时的接近动作减速度，可省略。 |

[使用案例]

```
Function main
    Accel 100, 100, 50, 50, 50, 50 '设置 PTP 运动加减速速度

    Print Accel (1)                '设置 PTP 运动加速度
    Print Accel (2)                '设置 PTP 运动减速度
    Print Accel (3)                '设置 JUMP 运动转移动作加速度
    Print Accel (4)                '设置 JUMP 运动转移动作减速度
    Print Accel (5)                '设置 JUMP 运动接近动作加速度
    Print Accel (6)                '设置 JUMP 运动接近动作减速度

Fend
```

[回传说明]

回传数据 百分比数值 1 -100 %，数据类型默认为浮点数。

Go

[功能]

用于在当前位置到指定位置之间执行 PTP 运动。

[语法]

Go 目标坐标 [CP] {Till/Find} {!...!}

[参数说明]

目标坐标 以点数据指定目标位置。

CP 设置运动平滑，可省略。

Till/Find Till 表达式 = On/Off。Find 表达式 = On/Off。可省略。详情请参考 Till/Find 的详细说明。

!...! 在运动过程中并行处理 I/O 等输入输出。可省略。详情请参考并行处理的详细说明。

[使用案例]

Function main

Go P1

Go P2 CP

Go P1+X(20)

'运动 P1 点且 X 方向相对偏移 20mm

Go P2:X(200)

'运动 P2 点且 X 坐标改变为 200mm 处

Go P3 Till Sw(1) = On And Sw(5) = Off

'运动 P3 点过程中，当 D11 为 On，D15 为 Off 时停止运动

Go P4 Till Sw(1) = On !D50; On 5!

'运动 P4 点过程中，当 D11 为 On 时停止运动，且运动到 50%路径时将 D05 置 On

Go P5 /R '以右手系运动至 P5 点

Go XY(X 坐标 , Y 坐标 , Z 坐标 , U 坐标) '移动至指定坐标

Fend

BGo

[功能]

用于在已选择的用户坐标系上执行偏移 PTP 运动。

[语法]

BGo 目标坐标 {CP} {Till/Find} {!...!}

[参数说明]

| | |
|-----------|--|
| 目标坐标 | 以点数据指定目标位置。 |
| CP | 设置运动平滑，可省略。 |
| Till/Find | Till 表达式 = On/Off。Find 表达式 = On/Off。可省略。详情请参考 Till/Find 的详细说明。 |
| !...! | 在运动过程中并行处理 I/O 等输入输出。可省略。详情请参考并行处理的详细说明。 |

[使用案例]

Function main

```
BGo XY(100, 0, 0, 0)           '在已选择用户坐标上，向 X 方向移动 100mm
```

```
P1 = XY(300, 300, -20, 0)
```

```
P2 = XY(300, 300, -20, 0) /L
```

```
Local 1, XY(0, 0, 0, 45)
```

```
Go P1
```

```
Print Here
```

```
'X:300.000 Y:300.000 Z:-20.000 U:0.000 V:0.000 W:0.000 /R /0
```

```
BGo XY(0, 50, 0, 0)
```

```
Print Here
```

```
'X:300.000 Y:350.000 Z:-20.000 U:0.000 V:0.000 W:0.000 /R /0
```

```
Go P2
```

```
Print Here
```

```
'X:300.000 Y:300.000 Z:-20.000 U:0.000 V:0.000 W:0.000 /L /0
```

```
BGo XY(0, 50, 0, 0)
```

```
Print Here
```

```
'X:300.000 Y:350.000 Z:-20.000 U:0.000 V:0.000 W:0.000 /L /0
```

```
BGo XY(0, 50, 0, 0) /L
```

```
Print Here
```

```
'X:264.645 Y:385.355 Z:-20.000 U:0.000 V:0.000 W:0.000 /L /0
```

Fend

TGo

[功能]

用于在当前工具坐标系上执行偏移 PTP 动作。

[语法]

TGo 目标坐标 {CP} {Till/Find} {!...!}

[参数说明]

| | |
|-----------|--|
| 目标坐标 | 以点数据指定目标位置。 |
| CP | 设置运动平滑，可省略。 |
| Till/Find | Till 表达式 = On/Off。Find 表达式 = On/Off。可省略。详情请参考 Till/Find 的详细说明。 |
| !...! | 在运动过程中并行处理 I/O 等输入输出。可省略。详情请参考并行处理的详细说明。 |

[使用案例]

Function main

TGo XY(100, 0, 0, 0) '在已选择工具坐标系上，向 X 方向移动 100mm

Tool 0

P1 = XY(300, 300, -20, 0)

P2 = XY(300, 300, -20, 0) /L

Go P1

Print Here

'X:300.000 Y:300.000 Z:-20.000 U:0.000 V:0.000 W:0.000 /R /0

TGo XY(0, 0, -30, 0)

Print Here

'X:300.000 Y:350.000 Z:-50.000 U:0.000 V:0.000 W:0.000 /R /0

Go P2

Print Here

'X:300.000 Y:300.000 Z:-20.000 U:0.000 V:0.000 W:0.000 /L /0

TGo XY(0, 0, 30, 0)

Print Here

'X:300.000 Y:350.000 Z:-50.000 U:0.000 V:0.000 W:0.000 /L /0

Fend

Speeds

[功能]

设置直线插补运动的运行速度，或返回当前直线插补运动的运行速度。

[语法]

`SpeedS` {速度[, 转速速度, 接近速度]}

[参数说明]

速度 以 0.1~2000mm/s 来设定当前的直线插补运动速度。

转移速度 以 0.1~2000mm/s 来设定 Jump3 命令时的转移动作速度，可省略。

接近速度 以 0.1~2000mm/s 来设定 Jump3 命令时的转移动作速度，可省略。

[使用案例]

Function main

 Motor On

 Power High

 Speed 100 'PTP 速度

 Accel 100, 100

 SpeedS 2000 'Line 速度

 AccelS 100, 100

 SpeedFactor 80 '全局速度

 Print SpeedFactor '打印当前全局速度

Fend

[回传说明]

回传数据 1 百分比数值 1 -100 %，数据类型默认为浮点数。

AccelS

[功能]

设置直线插补运动的运行加减速速度，或返回当前直线插补运动的运行加减速速度。

[语法]

AccelS {加速度百分比, 减速度百分比, {转速加速度百分比, 转速减速度百分比, 接近加速度百分比, 接近减速度百分比}}

[参数说明]

| | |
|----------|---|
| 加速度百分比 | 以 1~100%来设定当前的直线插补运动加速度。 |
| 减速度百分比 | 以 1~100%来设定当前的直线插补运动减速度。 |
| 转移加速度百分比 | 以 1~100%来设定 Jump3 和 Jum3CP 命令时的转移动作加速度，可省略。 |
| 转移减速度百分比 | 以 1~100%来设定 Jump3 和 Jum3CP 命令时的转移动作减速度，可省略。 |
| 接近加速度百分比 | 以 1~100%来设定 Jump3 和 Jum3CP 命令时的接近动作加速度，可省略。 |
| 接近减速度百分比 | 以 1~100%来设定 Jump3 和 Jum3CP 命令时的接近动作减速度，可省略。 |

[使用案例]

```
Function main
  Motor On
  Power High

  Speed 100      'PTP 速度
  Accel 100, 100
  SpeedS 2000   'Line 速度
  AccelS 100, 100

  SpeedFactor 80 '全局速度

  Print SpeedFactor '打印当前全局速度
Fend
```

Move

[功能]

用于在当前位置到指定位置之间以直线插补运动。

[语法]

Move 目标坐标 {**CP**} {**Till/Find**} {!...!}

[参数说明]

目标坐标 以点数据指定目标位置。

CP 设置运动平滑，可省略。

Till/Find **Till** 表达式 = 0n/0ff。**Find** 表达式 = 0n/0ff。可省略。详情请参考 Till/Find 的详细说明。

!...! 在运动过程中并行处理 I/O 等输入输出。可省略。详情请参考并行处理的详细说明。

[使用案例]

Function main

Move P1 ' 直线运动到 P1 点

Move P2 **CP** ' 直线运动到 P2 点，且在 P2 点处平滑过渡

Move P3 **Till** Sw(1) = 0n **And** Sw(5) = 0ff

' 运动 P3 点过程中，当 D11 为 0n，Di5 为 0ff 时停止运动

Move P4 **Till** Sw(1) = 0n !D50; 0n 5!

' 运动 P4 点过程中，当 D11 为 0n 时停止运动，且运动到 50%路径时将 D05 置为 0n

Fend

BMove

[功能]

用于在已选择的用户坐标系上执行偏移直线插补运动。

[语法]

BMove 目标坐标 {CP} {Till/Find} {!...!}

[参数说明]

| | |
|-----------|--|
| 目标坐标 | 以点数据指定目标位置。 |
| CP | 设置运动平滑，可省略。 |
| Till/Find | Till 表达式 = On/Off。Find 表达式 = On/Off。可省略。详情请参考 Till/Find 的详细说明。 |
| !...! | 在运动过程中并行处理 I/O 等输入输出。可省略。详情请参考并行处理的详细说明。 |

[使用案例]

Function main

BMove XY(100 , 0 , 0 , 0) '在已选择用户坐标上，向 X 方向移动 100mm

P1 = **XY**(300 , 300 , -20 , 0)

P2 = **XY**(300 , 300 , -20 , 0) /L

Local 1 , **XY**(0 , 0 , 0 , 45)

Go P1

Print Here

'X:300.000 Y:300.000 Z:-20.000 U:0.000 V:0.000 W:0.000

BMove XY(0 , 50 , 0 , 0)

Print Here

'X:300.000 Y:350.000 Z:-20.000 U:0.000 V:0.000 W:0.000

Go P2

Print Here

'X:300.000 Y:300.000 Z:-20.000 U:0.000 V:0.000 W:0.000

BMove XY(0 , 50 , 0 , 0)

Print Here

'X:300.000 Y:350.000 Z:-20.000 U:0.000 V:0.000 W:0.000

Fend

TMove

[功能]

用于在当前工具坐标系上执行偏移直线动作。

[语法]

TMove 目标坐标 {CP} {Till/Find} {!...!}

[参数说明]

| | |
|-----------|--|
| 目标坐标 | 以点数据指定目标位置。 |
| CP | 设置运动平滑，可省略。 |
| Till/Find | Till 表达式 = On/Off。Find 表达式 = On/Off。可省略。详情请参考 Till/Find 的详细说明。 |
| !...! | 在运动过程中并行处理 I/O 等输入输出。可省略。详情请参考并行处理的详细说明。 |

[使用案例]

Function main

```
TMove XY(100 , 0 , 0 , 0) '在已选择工具坐标上，向 X 方向移动 100mm
```

```
Tool 0 '选择工具 0
```

```
P1 = XY(300 , 300 , -20 , 0)
```

```
P2 = XY(300 , 300 , -20 , 0) /L
```

```
Go P1
```

```
Print Here
```

```
'X:300.000 Y:300.000 Z:-20.000 U:0.000 V:0.000 W:0.000
```

```
TMove XY(0 , 0 , -30 , 0)
```

```
Print Here
```

```
'X:300.000 Y:350.000 Z:-50.000 U:0.000 V:0.000 W:0.000
```

```
Go P2
```

```
Print Here
```

```
'X:300.000 Y:300.000 Z:-20.000 U:0.000 V:0.000 W:0.000
```

```
TMove XY(0 , 0 , -30 , 0)
```

```
Print Here
```

```
'X:300.000 Y:350.000 Z:-50.000 U:0.000 V:0.000 W:0.000
```

Fend

Jump

[功能]

用于在当前位置到指定位置之间以门型 PTP 运动。先上升，再平移，再下降的门型运动。

[语法]

Jump 目标坐标 {C Arch 编号} {LimZ 坐标值} {CP} {Till/Find/Sense} {!...!}

[参数说明]

- 目标坐标** 以点数据指定目标位置。
- C Arch 编号** 以 C0~C7 来选定不同的 Arch。详情请参考 Arch 详细说明。
- LimZ 坐标值** 设定 Jump 运动过程中，第 3 关节可移动的最大值(限制值)。详情请参考 LimZ 详细说明，可省略。
- CP** 设置运动平滑，可省略。
- Till/Find/Sense** Till 表达式 = On/Off。Find 表达式 = On/Off。Sense 表达式 = On/Off。可省略。详情请参考 Till/Find/Sense 的详细说明。
- !...!** 在运动过程中并行处理 I/O 等输入输出。可省略。详情请参考并行处理的详细说明。

[使用案例]

Function main

Go P1

Jump P2:Z(-50) C0 LimZ -50 CP

Go P3:Z(0) CP

Jump P4 C0 LimZ 0

Jump P1

Jump P2 CP

Jump P3 Till Sw(1) = On And Sw(5) = Off

'运动 P3 点过程中，当 DI1 为 ON，DI5 为 OFF 时停止运动

Jump P4 Till Sw(1) = On !D50; On 5!

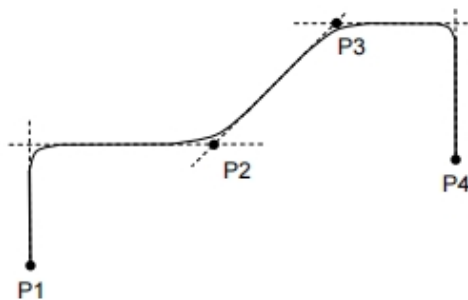
'运动 P4 点过程中，当 DI1 为 On 时停止运动，且运动到 50%路径时将 D05 置 On

Jump P5 /R C1 LimZ-10 Sense Sw(2) = On

'选择 Arch1 为接近参数，以右手系运动 P5 点过程中，Z 轴最大只能-10MM

'当 DI2 为 On 时停在目标点上方，且运动到 50%路径时将 D05 置 On

Fend



Arc、Arc3

[功能]

Arc 用于在 XY 平面上以圆弧插补动作将机械臂从当前位置移至指定位置。

Arc3 用于在三维空间上以圆弧插补动作将机械臂从当前位置移至指定位置。

[语法]

Arc 经过坐标, 目标坐标 {CP} {Till/Find} {!...!}

Arc3 经过坐标, 目标坐标 {CP} {Till/Find} {!...!}

[参数说明]

经过坐标 以点数据指定指定圆弧会经过的位置。

目标坐标 以点数据指定目标位置。

CP 设置运动平滑, 可省略。

Till/Find Till 表达式 = On/Off。Find 表达式 = On/Off。详情请参考 Till/Find/Sense 的详细说明。

!...! 在运动过程中并行处理 I/O 等输入输出。可省略。详情请参考并行处理的详细说明。

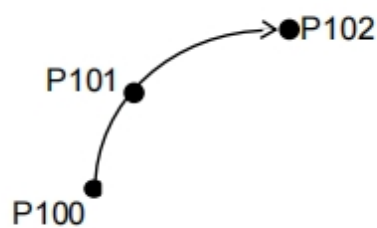
[使用案例]

Function main

Go P100

Arc P101 , P102

Fend



CP

[功能]

设置运动平滑。

[语法]

CP {On/Off}

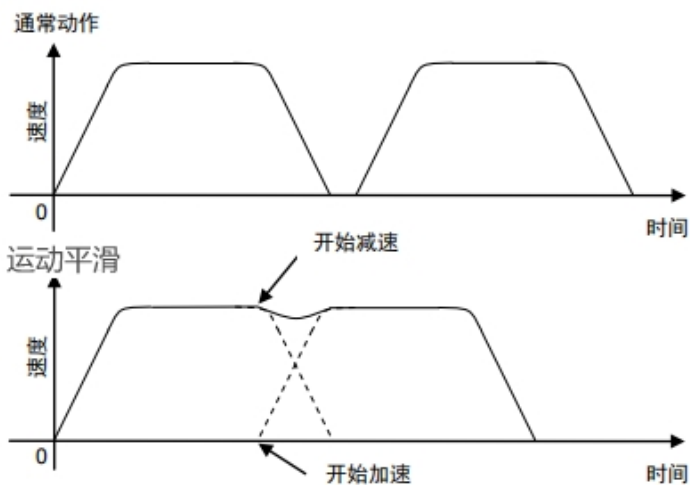
运动指令 目标坐标 CP

[参数说明]

On/Off On 为打开运动平滑，Off 为关闭。

运动指令 Go、Move、Arc、Jump 等运动指令。

[使用案例]



Function main

Go P1 CP

Move P2 CP

CP On 'CP On 以后的运动指令默认平滑过渡

Move P3

Move P4

CP Off

If CP = Off Then

 CP On

EndIf

Fend

CPZone

[功能]

设置运动平滑的比例程度。

[语法]

`CPZone {0n/0ff} {PTP 运动平滑比例, 直线插补平滑比例}`

[参数说明]

0n/0ff 0n 为开启运动平滑比例, 0ff 为关闭平滑比例。0ff 后则采用系统默认比例。

PTP 运动平滑比例 以 1~200 mm 来设定 PTP 运动平滑的融合程度。

直线插补平滑比例 以 1~200 mm 来设定直线插补运动平滑的融合程度。

[使用案例]

Function main

```
CPZone 0n , 50 , 50
```

Fend

R180

[功能]

用于约束 U 轴正负角度至±180 度。

[语法]

R180 {On/Off}

Go 目标坐标 R180

Print R180

[参数说明]

On/Off On 为开启约束，Off 为关闭约束。

[使用案例]

Function main

 If R180 = Off Then

 R180 On

 EndIf

 Go P1 R180

 Go P2 R180

 Go P3

 R180 Off

 Print R180

Fend

Here

[功能]

用于将机器人当作坐标示教至点位中，或获取机器人当前坐标。

[语法]

Here {点编号/点名称}

[参数说明]

点编号/点名称 将当前坐标示教至该点位。

[使用案例]

Function main

```
Here P1                    '将当前坐标示教到 P1 点  
Go Here + X(10)          '在当前位置向 X 方向偏移 10mm
```

Fend

Home

[功能]

将机器人移动至用户定义的 Home 点处。

[语法]

Home

[参数说明]

[使用案例]

```
Function main
```

```
    Home    '回到 Home 点
```

```
Fend
```

Local

[功能]

用于定义和显示用户坐标系。

[语法]

一点示教: **Local** 用户坐标编号, 点位

两点示教: **Local** 用户坐标编号, 点位 1, 点位 2, {X/Y}

三点示教: **Local** 用户坐标编号, 原点, X 轴点, Y 轴点

[参数说明]

用户坐标编号 以 1~64 指定需要示教的用户坐标系。

原点、点位 1... 以点变量指定用户坐标的点数据。

L/R 将用户坐标原点对准左(1号)或右(2号), 可省略。

X/Y 将连接原点与 X 轴或 Y 轴指定的直线定义为本地坐标系的 X 轴或 Y 轴, 可省略, 默认 X。

[使用案例]

一点示教用法:

Local 用户坐标编号, 点位

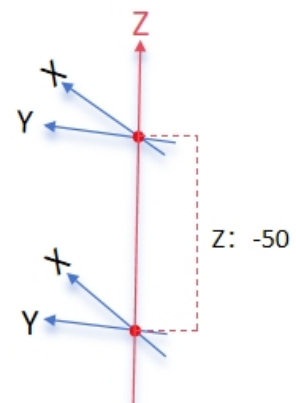
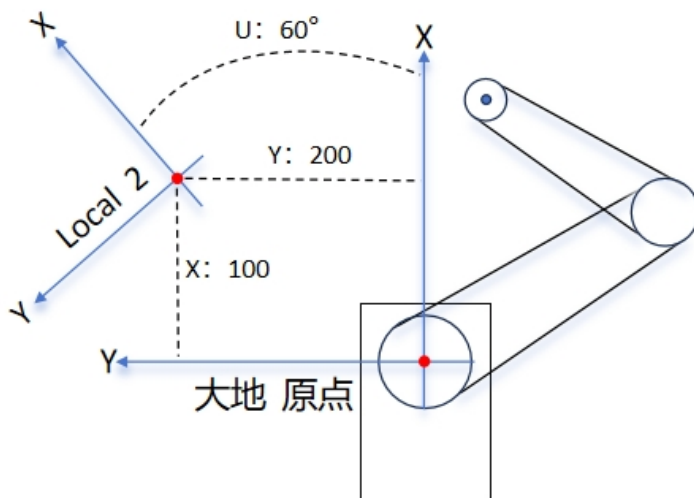
写法1:

例: P1 = XY(100, 200, -50, 60)

例: **Local** 2, P1

写法2:

例: **Local** 2, XY(100, 200, -50, 60)

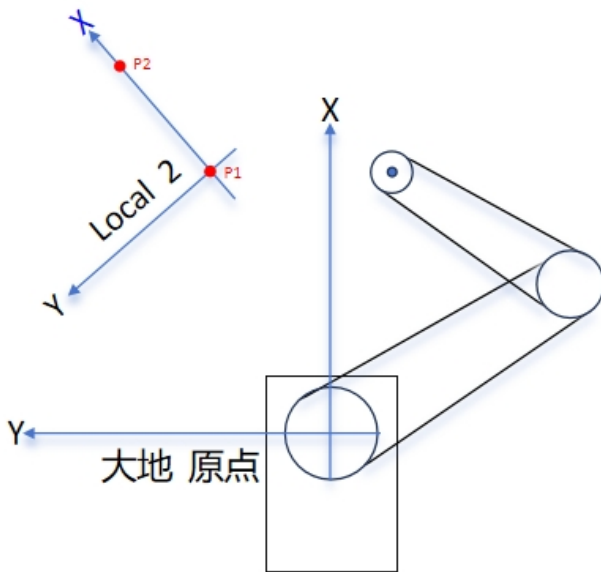


两点示教用法:

Local 用户坐标编号, 点位1, 点位2, {X/Y}

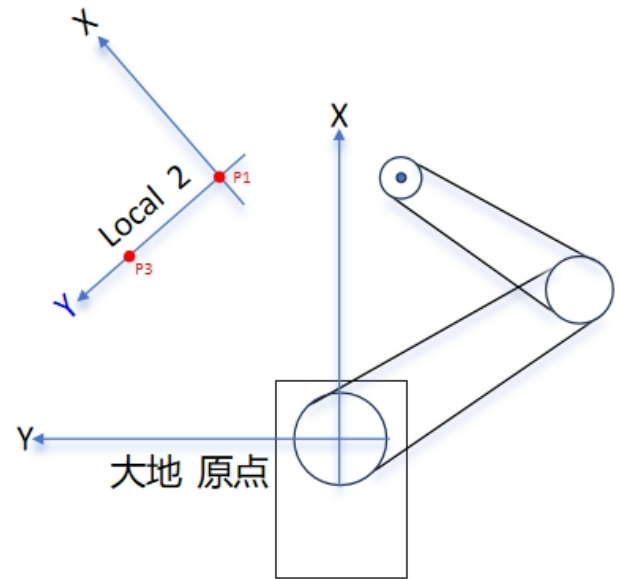
写法1:

例: Local 2, P1, P2, X



写法2:

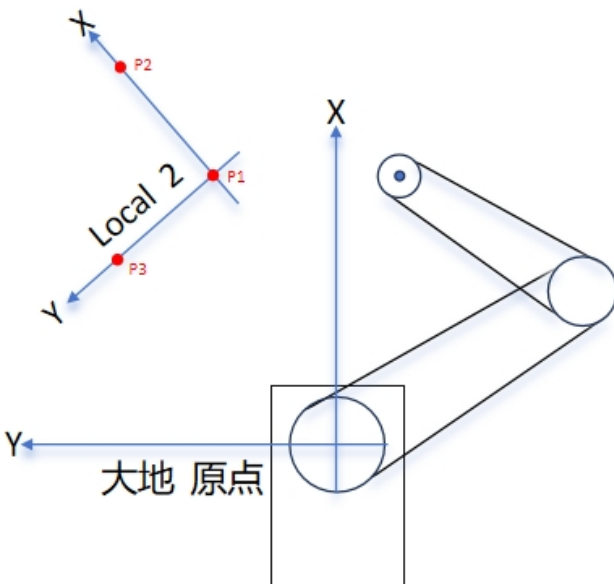
例: Local 2, P1, P3, Y



三点示教用法:

Local 用户坐标编号, 原点, X轴点, Y轴点

例: Local 2, P1, P2, P3



TLSet

[功能]

用于定义和显示工具坐标系。

[语法]

TLSet {工具坐标编号, {工具设置数据}}

[参数说明]

工具坐标编号 以 1~64 指定需要示教的工具坐标系。

工具设置数据 以点编号、点名称、点变量来设置工具坐标系的原点和方向。

注意事项： 省略所有参数，则显示所有 TLSet 的设置。

[使用案例]

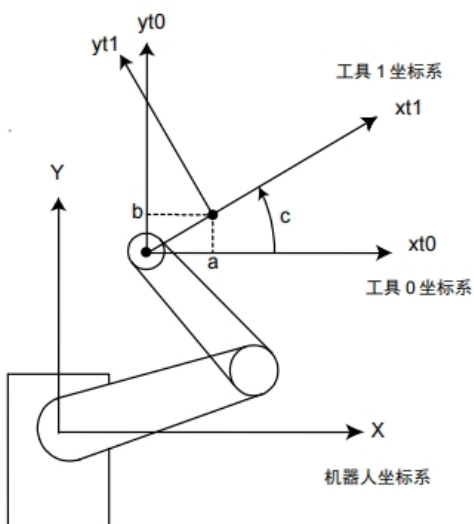
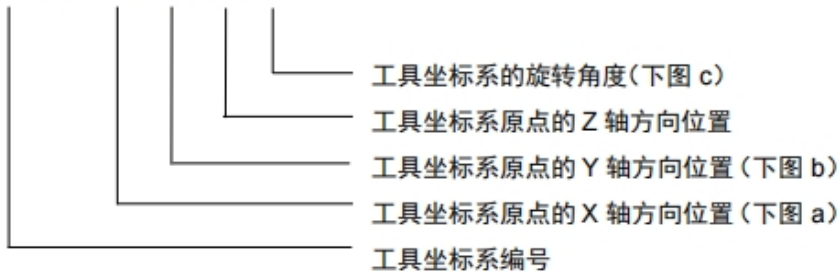
Function main

```
TLSet 1, XY(50, 100, -20, 30)
```

```
TLSet 2, P10 +X(20)
```

End

```
TLSET 1, XY(100, 60, -20, 30)
```



Tool

[功能]

用于选择工具坐标系，或返回当前已选择的工具坐标系。

[语法]

`Tool {工具坐标编号}`

[参数说明]

工具坐标编号 以 1~64 指定选择的工具坐标系。

[使用案例]

`Function main`

```
    Tool 1                                 '选择 1 号工具坐标系
```

```
    If Tool = 0 Then
```

```
        Tool 2
```

```
    EndIf
```

```
End
```

TCalib

[功能]

使用三点法定义工具坐标系。

[语法]

TCalib 工具坐标编号, 工具辅助点 1, 工具辅助点 2, 工具辅助点 3

[参数说明]

| | |
|---------|--|
| 工具坐标编号 | 以 1~64 指定选择的工具坐标系。 |
| 工具辅助点 1 | 以安装在机械手丝杆的治具末端中心对准示教标识物。 |
| 工具辅助点 2 | 在工具辅助点 1 的基础上 U 轴顺时针或逆时针旋转 $60 \sim 120^\circ$ ，并使用寸动功能将治具末端中心再次对准示教标识物。 |
| 工具辅助点 3 | 在工具辅助点 2 的基础上 U 轴向同一旋转方向再次旋转 $60 \sim 120^\circ$ ，并使用寸动功能将治具末端中心再次对准示教标识物。 |

[使用案例]

Function main

```
TCalib 1, P0, P1, P2 '以 P0、P1、P2 作为辅助点示教 1 号工具坐标
```

Fend

Till

[功能]

用于设置 Jump、Go、Move 等运动指令，运动过程中停止运动的条件。

[语法]

Till {条件表达式}

[参数说明]

条件表达式 运动过程中停止运动的条件。可使用各种 I/O 指令、运算符等。

[使用案例]

Function main

```
Till Sw(1) = Off        '设置 Till 条件 (DI1 为 OFF)
Go P1 Till             '满足前一行的条件时停止
```

```
Till Sw(1) = On And Sw(1) = On '设置新的 Till 条件
Move P2 Till                    '满足前一行的条件时停止
```

```
Move P5 Till Sw(10) = On       '满足该行条件时停止
```

End

Agl

[功能]

用于返回指定关节的角度或位置的函数。

[语法]

Agl (关节编号)

[参数说明]

关节编号 以整数指定关节编号。Scara 机器人范围 1~4，六轴 1~6。如有拓展轴则根据拓展轴增加编号。

[使用案例]

```
Function main
    Print PAgl (1) , Agl (2)
Fend
```

PAgl

[功能]

用于返回指定点位的关节角度。

[语法]

PAgl (点编号, 关节编号)

[参数说明]

点编号 以点位文件中的排序号做索引。

关节编号 以整数指定关节编号。Scara 机器人范围 1~4，六轴 1~6。如有拓展轴则根据拓展轴增加编号。

[使用案例]

Function main

```
Print PAgl (P1, 1)
```

```
Print PAgl (P1, 2)
```

```
Print PAgl (P1, 3)
```

```
Print PAgl (P1, 4)
```

End

AgItoPIs

[功能]

用于将机器人各关节角度转换为脉冲值。

[语法]

AgItoPIs(关节位置 1, 关节位置 2, 关节位置 3, 关节位置 4, {关节位置 5, 关节位置 6}....)

[参数说明]

关节位置 根据顺序，依次指定各个关节的关节角度。可根据机型和拓展轴酌情增加形参数量。

指令配合 做为 Go 参数时执行绝对关节运动，作为 Print 参数时输出脉冲值，其余用法均返回空间点位。

[使用案例]

Function main

```
P1 = AgItoPIs(0 , 0 , 0 , 90 , 0 , 0)
```

```
Go P1
```

```
Move AgItoPIs(0 , 0 , 0 , 90 , 0 , 0)
```

```
Go AgItoPIs(0 , 0 , 0 , 90 , 0 , 0)
```

Fend

Arch

[功能]

用于设置/显示 Jump、Jump3、Jump3CP 命令的 Arch 参数。

[语法]

Arch {Arch 编号 [, 转移距离, 接近距离]}

Arch(Arch 编号, 参数编号)

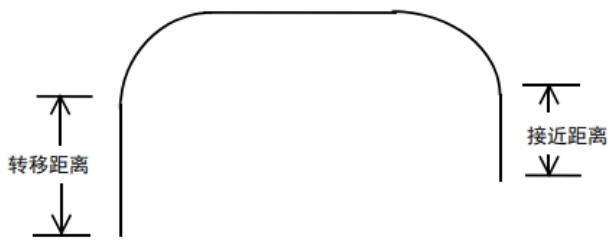
[参数说明]

Arch 编号 以整数 0~6 指定 Arch 编号。不指定的情况下，系统有默认值，详情看下表格。

转移距离 从出发点算起的垂直距离，单位 mm。

转移距离 从目标点算起的垂直距离，单位 mm。

参数编号 1:转移距离。2:接近距离。



Arch 表格默认值

| Arch 编号 | 转移距离 | 接近距离 |
|---------|------|------|
| 0 | 30 | 30 |
| 1 | 40 | 40 |
| 2 | 50 | 50 |
| 3 | 60 | 60 |
| 4 | 70 | 70 |
| 5 | 80 | 80 |
| 6 | 90 | 90 |

[使用案例]

Function main

```
Arch 0 , 20 , 20 '给 Arch 0 设置两个距离参数
```

```
Jump P1 C0 '在 Jump 中启用 Arch0
```

```
If Arch(0,1) = 30 And Arch(0,2) = 30 Then
```

```
    '判断 Arch0 的两个距离参数是否符合条件
```

```
    Arch 0 , 25 , 25
```

```
EndIf
```

Fend

Find

[功能]

用于设置在运动过程中根据条件表达式成立保存坐标。

[语法]

Find 条件表达式

[参数说明]

条件表达式 运动过程中保存坐标的条件。可使用各种 I/O 指令、运算符等。

[使用案例]

Function main

```
Go P10 Find Sw(5) = On                    '运动过程中 DI5 置为 On 时保存坐标

If PosFound Then
  Go FindPos                                '通过 FindPos 获取 Find 保存的坐标
Else
  Print "运动过程中 DI5 未置为 ON"
EndIf
```

Fend

FindPos

[功能]

用于返回 Find 保存的坐标。

[语法]

```
FindPos
```

```
Go FindPos
```

[参数说明]

[使用案例]

```
Function main
```

```
    Go P10 Find Sw(5) = On           '运动过程中 DI5 置为 On 时保存坐标
```

```
    If PosFound Then
```

```
        Go FindPos                   '通过 FindPos 获取 Find 保存的坐标
```

```
    Else
```

```
        Print "运动过程中 DI5 未置为 ON"
```

```
    EndIf
```

```
Fend
```

InPos

[功能]

返回机器人的到位状态。

[语法]

InPos

[参数说明]

注意事项:

1. 完成定位时返回 True。
2. 机器人运动中返回 False。

[使用案例]

Function main

```
P0 = XY(0 , -100 , 0 , 0)
```

```
P1 = XY(0 , 100 , 0 , 0)
```

```
Xqt MonitorPosition
```

```
Do
```

```
    Jump P0
```

```
    Wait 0.5
```

```
    Jump P1
```

```
    Wait 0.5
```

```
Loop
```

```
Fend
```

Function MonitorPosition

```
    Boolean oldInPos , Pos
```

```
Do
```

```
    Pos = InPos
```

```
    If Pos <> oldInPos Then
```

```
        Print "InPos = " , Pos
```

```
    EndIf
```

```
    oldInPos = Pos
```

```
Loop
```

```
Fend
```

JA

[功能]

用于根据关节角度返回机器人坐标。

[语法]

JA(关节角度 1, 关节角度 2, 关节角度 3, 关节角度 4, {关节角度 5, 关节角度 6}....)

[参数说明]

关节角度 根据顺序依次指定各个关节的关节角度。单位：旋转关节 deg。移动关节 mm。
可根据机型和拓展轴酌情增加形参数量。

[使用案例]

Function main

```
P10 = JA(60 , 30 , -50 , 45)
```

```
Go JA(135 , 90 , -50 , 90)
```

```
P3 = JA(0 , 0 , 0 , 0 , 0 , 0)
```

Fend

Joint

[功能]

用于在关节坐标系中显示当前机器人位置。

[语法]

Joint

[参数说明]

[使用案例]

Function main

```
Joint
```

```
'打印信息:Joint: 1:-11.880deg 2:30.432deg 3: -21.902mm 4:-18.552deg
```

Fend

LimZ

[功能]

用于设置/显示 Jump 运动过程中第 3 关节高度 (Z 坐标) 初始值。

[语法]

LimZ {Z 坐标值}

[参数说明]

Z 坐标值 指定第 3 关节在动作范围内的坐标值。

[使用案例]

```
Function main
  LimZ -10           '设置 Limz 的默认值
  Jump P1           '执行 JUMP 时水平移动 Z 轴高度保持为-10
  Jump P2 LimZ -20  '执行 JUMP 时水平移动 Z 轴高度保持为-20
  Jump P3           '执行 JUMP 时水平移动 Z 轴高度保持为-10
  If LimZ > -10 Then '当设置值大于-10 时
    LimZ -30        '将 Limz 默认值设置为-30
  EndIf
Fend
```

Pass

[功能]

用于进行穿过指定点附近(不停止)的 PTP 运动。

[语法]

Pass 目标坐标 1 {, 目标坐标 2, 目标坐标 3. . . }

[参数说明]

目标坐标 以点编号、点名称设定需要移动到的位置。可根据需求后面酌情增加点位形参。

[使用案例]

```
Function main
  Pass P1 , P2 , P3 '从 P1 → P2 → P3 连续平滑过渡运动
Fend
```

Pls

[功能]

用于返回当前位置指定关节的脉冲值。

[语法]

Pls(关节编号)

[参数说明]

关节编号 以整数 1~关节数量指定需要获取脉冲值的关节。

[使用案例]

```
Function main
    Real Axle_1 , Axle_2 , Axle_3 , Axle_4
    Axle_1 = pls(1)
    Axle_2 = pls(2)
    Axle_3 = pls(3)
    Axle_4 = pls(4)
    Print "当前位置一轴脉冲：", Axle_1
    Print "当前位置二轴脉冲：", Axle_2
    Print "当前位置三轴脉冲：", Axle_3
    Print "当前位置四轴脉冲：", Axle_4
End
```

PPIs

[功能]

用于指定坐标点的关节脉冲值。

[语法]

PPIs(点编号, 关节编号)

[参数说明]

点编号 以点位文件中的排序号做索引。

关节编号 以整数 1~关节数量指定需要获取脉冲值的关节。

[使用案例]

```
Function main
```

```
  Print "P1 点一轴脉冲位置: ", PPIs(P1, 1)
```

```
  Print "P1 点二轴脉冲位置: ", PPIs(P1, 2)
```

```
  Print "P1 点三轴脉冲位置: ", PPIs(P1, 3)
```

```
  Print "P1 点四轴脉冲位置: ", PPIs(P1, 4)
```

```
End
```

JS

[功能]

返回 Jump、Jump3、Jump3CP 运动过程中 Sense 条件是否成立。

[语法]

JS

[参数说明]

注意事项：

1. Sense 条件成立停在目标点上方时，返回 True。
2. 正常完成了 Jump 动作并到达目标点，返回 False。

[使用案例]

```
Function main
```

```
    Print searchSensor
```

```
End
```

```
Function searchSensor As Boolean
```

```
    Sense Sw(5) = On
```

```
    Jump P0
```

```
    Jump P1 Sense
```

```
    If JS = True Then
```

```
        Print "Sense 条件成立"
```

```
        searchSensor = True
```

```
    EndIf
```

```
End
```

PosFound

[功能]

用于返回 Find 命令时执行的状态。

[语法]

PosFound

[参数说明]

注意事项:

1. 运动过程中 Find 成立则返回 True, 否则返回 False。

[使用案例]

```
Function main
```

```
    Go P10 Find Sw(5) = On '运动过程中 DI5 置 On 时保存坐标
```

```
    If PosFound Then
```

```
        Go FindPos          '通过 FindPos 获取 Find 保存的坐标
```

```
    Else
```

```
        Print "运动过程中 DI5 未置为 ON"
```

```
    EndIf
```

```
End
```

Pulse

[功能]

将机械臂以 PTP 移动到指定的各关节脉冲值的位置。或返回脉冲值对应的点位。

[语法]

`Pulse {关节 1 脉冲值, 关节 2 脉冲值, 关节 3 脉冲值, 关节 4 脉冲值[, 关节 5 脉冲值,]}`

`Pulse(关节 1 脉冲值, 关节 2 脉冲值, 关节 3 脉冲值, 关节 4 脉冲值[, 关节 5 脉冲值,])`

[参数说明]

关节脉冲值 根据顺序依次填入各关节的脉冲值，结合机器人实际轴数可酌情增加形参。

[使用案例]

`Function main`

```
  Pulse 123456 , 222333 , 333444 , 111111
```

```
  '移动到指定脉冲位置
```

```
  P1 = Pulse(123456 , 222333 , 333444 , 111111)
```

```
  '将指定脉冲的点位保存到 P1
```

```
  Pulse '打印当前所有轴脉冲值
```

`Fend`

RobLoad

[功能]

用于选择指定的负载参数，或返回指定的负载参数。

[语法]

RobLoad {负载编号}

[参数说明]

负载编号 以整数 1~64 指定需要选择的负载。

[使用案例]

```
Function main
  RobLoad 3
  Go P1

  If RobLoad = 0 Then
    RobLoad 1
  EndIf
Fend
```

Sense

[功能]

用于设置/显示 Jump、Jump3、Jump3CP 停在目标点上方的条件。

[语法]

`Sense` {条件表达式}

[参数说明]

条件表达式 运动过程中停在目标点上方的条件。可使用各种 I/O 指令、运算符等。

[使用案例]

```
Function main
  Sense Sw(5) = Off
  Jump P1 C2 Sense
  '当 D15 为 Off 时停在目标点上方

  If JS = True Then
    Print "机器人已停在目标点上方"
  EndIf
  On 1; Wait 0.2; Off 1
Fend
```

TillOn

[功能]

用于返回 Till 的状态。

[语法]

TillOn

[参数说明]

注意事项：

1. 运动过程中 Till 条件成立则返回 True。否则返回 False。

[使用案例]

```
Function main
  Go P1 Till Sw(1) = On

  If TillOn Then
    Print "因为DI1 置为 0n, 已停止运动"
  EndIf
Fend
```

WaitPos

[功能]

用于在运动平滑指令后等待机器人进行减速停止后再将程序移交给后续程序。

[语法]

WaitPos

[参数说明]

注意事项:

1. 通常情况下，在运动平滑有效的情况下 (CP On 或运动指令带 CP 时)，动作命令会在开始减速的同时将控制提前移交给后续的程序。如果不想提前移交，可以在后面插入 WaitPos，则会完成减速动作之后才移交。

[使用案例]

Function main

Off 1

CP On

Move P1

Move P2

WaitPos '等待机器人减速

On 1

CP Off

End

XY

[功能]

根据指定的数据返回点位。

[语法]

XY(X 坐标值, Y 坐标值, Z 坐标值, U 坐标值 [, V 坐标值, W 坐标值])

[参数说明]

坐标值 根据顺序填入各个轴的坐标。根据实际机器人型号酌情增加 V 轴和 W 轴的坐标值。
单位 mm。

[使用案例]

Function main

```
P10 = XY(60 , 30 , -50 , 45) + P20
```

'索引 P20 点位的数据加上指定偏差坐标生成 P10 点位

```
P10 = XY(60 , 30 , -50 , 45) /L
```

'将指定的坐标以及手系生成为 P10 点位

```
Go XY(10 , 20 , 30 , 40)
```

'圆弧移动至指定坐标位置

Fend

CX、CY、CZ、CU、CV、CW、CR、CS、CT

[功能]

用于设置(修改)点的坐标值。

[语法]

CX(点位) [= 坐标值]

CY(点位) [= 坐标值]

CZ(点位) [= 坐标值]

.....

[参数说明]

点位 点编号、点名称。

坐标值 设置的坐标值，单位 mm。

[使用案例]

```
Function main
    Double New_Y

    CX(P1) = 123           '将 P1 的 X 坐标改为 123mm
    CU(P1) = 321           '将 P1 的 U 坐标改为 321deg

    If CY(P1) > 200 Then
        CY(P1) = 200     '当 P1 的 Y 坐标大于 200，强制等于 200
    EndIf

    P1 = Here             '获取当前位置并赋值 P1 点
    Print CX(P1)         '打印当前位置 X 坐标
    New_Y = CY(P1)       '将当前 Y 坐标赋值给 New_Y 变量
End
```

TargetOK

[功能]

用于检测指定点位坐标是否可达。

[语法]

TargetOK(点位)

.....

[参数说明]

点位 点编号、点名称。

[使用案例]

```
Function main
  If TargetOK(P1) = False Then
    Print "该点位无法到达"
  ElseIf TargetOK(P1) = True Then
    Go P1
  EndIf
Fend
```

CoISup

[功能]

用于设置(修改)碰撞检测系数。

[语法]

```
CoISup On/Off [, level]
```

```
CoISup
```

```
....
```

[参数说明]

On/Off 是否需要设定碰撞检测系数。

level 需要设定的碰撞检测系数值，取值范围：1 - 300。

[使用案例]

```
Function main
    CoISup On, 50      '设置碰撞检测系数
    CoISup Off        '关闭碰撞检测设置
    Print CoISup      '打印当前碰撞检测系数
Fend
```

ZeroPls

[功能]

用于设置(修改)机器人零位。

[语法]

```
ZeroPls 脉冲 1 , 脉冲 2 , 脉冲 3 [ , 脉冲 4 , 脉冲 5 , 脉冲 6]
```

```
ZeroPls
```

```
.....
```

[参数说明]

脉冲 1 , 脉冲 2 , 脉冲 3 [, 脉冲 4 , 脉冲 5 , 脉冲 6]

各成员轴需要设置零位的脉冲值。

[使用案例]

```
Function main
```

```
ZeroPls 0 , 0 , 0 , 0
```

```
'将 XYZU 四轴零点位置设置为脉冲值为 0 处
```

```
Print ZeroPls
```

```
'打印当前的零位
```

```
Fend
```

Hand

[功能]

用于设置(修改)手系。

[语法]

Hand [点位, 手系]

Hand

....

[参数说明]

点位 点编号、点名称。

手系 修改点位手系: Lefty 为左手、Righty 为右手。当设定为空时则返回点位手系。

[使用案例]

Function main

Print Hand(P0) '打印 P0 点手系

Hand P0, Righty '设置 P0 点为右手系

Print Hand '打印当前手系

End

Elbow

[功能]

用于设置(修改)肘姿态。

[语法]

Elbow [点, 姿态]

Elbow

.....

[参数说明]

点 点编号、点名称。

姿态 点位所属得肘部姿态: Above 为肘部朝上、Below 为肘部朝下。

[使用案例]

Function main

Print Elbow(P0) '打印 P0 点姿态

Elbow P0 , Above '设置 P0 点姿态, 肘部朝上

Print Elbow '打印当前姿态

Fend

Wrist

[功能]

用于设置(修改)手腕姿态。

[语法]

`Wrist` [点, 姿态]

`Wrist`

.....

[参数说明]

点 点编号、点名称。

姿态 点位所属得手腕姿态: Flip 为手腕翻转、NoFlip 为手腕不翻转。

[使用案例]

`Function` main

`Print Wrist`(P0) '打印 P0 点手腕姿态

`Wrist` P0 , `NoFlip` '设置 P0 点姿态, 肘部朝上

`Print Wrist` '打印当前姿态

`End`

独立轴指令详细说明

IndPower

[功能]

用于设置独立轴使能状态。

[语法]

`IndPower` 独立轴 ID , `On` / `Off`

[参数说明]

独立轴 ID 指定独立轴编号(范围: 1 - 16)。

On / Off 指定独立轴使能状态(On: 上使能 / Off: 下使能)。

[使用案例]

`Function` main

```
IndPower 1 , On           '1 号独立轴上使能
```

```
Wait 1
```

```
IndPower 1 , Off        '1 号独立轴下使能
```

`Fend`

IndHalt

[功能]

用于控制独立轴停止运动。

[语法]

`IndHalt` 独立轴 ID , 加减速

[参数说明]

| | |
|--------|----------------------|
| 独立轴 ID | 指定独立轴编号(范围: 1 - 16)。 |
| 加减速 | 指定独立轴的加减速, 写 0 为停止。 |

[使用案例]

Function main

```
IndPower 1 , On           '1 号独立轴上使能
IndMovAbs 1 , 1000, 30, 100 '指定 1 号独立轴运动至 1000 (mm/deg/Pulse) 处
Wait 1
IndHalt 1 , 0             '1 号独立轴停止运动
```

Fend

IndMovAbs

[功能]

用于控制控制独立轴运动至指定的绝对位置。

[语法]

`IndMovAbs` 独立轴 ID , 绝对位置 , 速度 , 加减速 [, 速度锁定]

[参数说明]

| | |
|--------|-------------------------|
| 独立轴 ID | 指定独立轴编号(范围: 1 - 16)。 |
| 绝对位置 | 指定独立轴的绝对位置, 单位(Unit)。 |
| 速度 | 指定独立轴的最大运动速度(Unit / S)。 |
| 加减速 | 指定独立轴的加减速时间(1-1000 ms)。 |
| 速度锁定 | 指定独立轴运动速度不被全局速度百分比所影响。 |

注意事项:

1. 与独立轴相关的指令不会等待命令执行完成, 即指令生效后向下流走。
2. Unit 的计算方法: 毫米/角度 = N 个脉冲 = 1 个 Unit

[使用案例]

Function main

```

IndPower 1 , On           '1 号独立轴上使能
IndMovAbs 1 , 1000, 30, 100 '指定 1 号独立轴运动至 1000 (mm/deg/Pulse) 处
IndMovDone 1             '等待 1 号独立轴运动完成

```

Fend

IndMovRel

[功能]

用于控制独立轴运动至相对位置。

[语法]

`IndMovRel` 独立轴 ID , 相对位置 , 速度 , 加减速 [, 速度锁定]

[参数说明]

| | |
|--------|-------------------------|
| 独立轴 ID | 指定独立轴编号(范围: 1 - 16)。 |
| 相对位置 | 指定独立轴的相对位置, 单位(Unit)。 |
| 速度 | 指定独立轴的最大运动速度(Unit / S)。 |
| 加减速 | 指定独立轴的加减速时间(1-1000 ms)。 |
| 速度锁定 | 指定独立轴运动速度不被全局速度百分比所影响。 |

注意事项:

1. 与独立轴相关的指令不会等待命令执行完成, 即指令生效后向下流走。
2. Unit 的计算方法: 毫米/角度 = N 个脉冲 = 1 个 Unit

[使用案例]

Function main

```
IndPower 1 , On           '1 号独立轴上使能
IndMovRel 1 , 1000, 30, 100 '指定 1 号独立轴运动至 1000 (mm/deg/Pulse) 处
IndMovDone 1             '等待 1 号独立轴运动完成
```

Fend

IndMovVel

[功能]

用于控制独立轴保持恒定速度运动。

[语法]

`IndMovVel` 独立轴 ID , 速度 , 加减速 [, 速度锁定]

[参数说明]

| | |
|--------|-------------------------|
| 独立轴 ID | 指定独立轴编号(范围: 1 - 16)。 |
| 速度 | 指定独立轴的恒定运动速度(Unit / S)。 |
| 加减速 | 指定独立轴的加减速时间(1-1000 ms)。 |
| 速度锁定 | 指定独立轴运动速度不被全局速度百分比所影响。 |

注意事项:

1. 与独立轴相关的指令不会等待命令执行完成, 即指令生效后向下流走。

[使用案例]

Function main

```
IndPower 1 , On           '1 号独立轴上使能
IndMovVel 1 , 30, 100     '指定 1 号独立轴保持恒速运动
IndMovDone 1             '等待 1 号独立轴运动完成
```

Fend

点位、IO、编码器等备注指令详细说明

PDef

[功能]

用于返回点是否已定义。

[语法]

PDef (点编号)

PDef (P 点编号)

PDef (P (点编号))

PDef (标签名称)

[参数说明]

| | |
|--------|-----------------------------|
| 点编号 | 以点位文件中的排序号做索引。 |
| P 点编号 | 以点位文件中的排序号做索引。 |
| P(点编号) | 以点位文件中的排序号做索引。 |
| 标签名称 | 赋予标签变量后，依据变量值在点位文件中的排序号做索引。 |

[使用案例]

Function main

```
Print "0:" , PDef(0)           '打印 P0 号点是否存在定义
Print "P1:" , PDef(P1)       '打印 P1 号点是否存在定义

Print "P(1):" , PDef(P(1))   '打印 P1 号点是否存在定义

Integer Point_Num
Point_Num = 0
Print "Point_Num:" , PDef(Point_Num) '打印对应点是否存在定义
```

Fend

[回传说明]

回传数据 1 TRUE / FALSE ， TRUE 为点位存在定义，FALSE 为点位未定义。

PDeI

[功能]

用于删除点位。

[语法]

PDeI 点编号

PDeI 起始点编号 , 结束点编号

[参数说明]

| | |
|-------|----------------|
| 点编号 | 以点位文件中的排序号做索引。 |
| 起始点编号 | 从指定的编号开始删除点位。 |
| 结束点编号 | 从指定的编号结束删除点位。 |

[使用案例]

Function main

```
PDeI 1          '删除 P1 号点位信息  
  
PDeI 1 , 5     '删除 P1 - P5 号点位信息  
  
SavePoints "robot1" '保存至点位表
```

Fend

PLabel

[功能]

用于设置指定点数据的标签。

[语法]

PLabel 点编号, 新标签

[参数说明]

点编号 以点位文件中的排序号做索引。

新标签 为指定的点编号赋予新的标签(暂不支持中文)。

[使用案例]

Function main

```
PLabel 1 , "New_Point"        '为 P1 号点赋予新标签
```

```
Go P(New_Point)
```

```
SavePoints "robot1"        '保存至点位文件
```

End

PLabel\$

[功能]

用于返回指定点数据的标签。

[语法]

PLabel\$(点编号)

PLabel\$(P点编号)

PLabel\$(P(点编号))

[参数说明]

| | |
|--------|----------------|
| 点编号 | 以点位文件中的排序号做索引。 |
| P点编号 | 以点位文件中的排序号做索引。 |
| P(点编号) | 以点位文件中的排序号做索引。 |

注意事项:

1. 若使用该指令索引空点位或索引的点位不存在标签时，不做任何输出。

[使用案例]

Function main

```
Print PLabel$(1)           '打印 1 号点位标签

Print PLabel$(P1)         '打印 1 号点位标签

Print PLabel$(P(1))       '打印 1 号点位标签

If PLabel$(2) = "" Then   '判断 2 号点标签等于空
    Print "该点位不存在标签"
EndIf

Fend
```

PNumber

[功能]

用于返回指定标签名称的点编号。

[语法]

`PNumber` (标签名称)

`PNumber` (“标签名称”)

[参数说明]

标签名称 点位数据中的其中一项，标签。

[使用案例]

Function main

```
Print PNumber(EA)                      '打印点位标签为 EA 的点编号
```

```
Print PNumber("AA")                  '打印点位标签为 AA 的点编号
```

Fend

PList

[功能]

用于显示指定点数据。

[语法]

PList

PList 点编号

PList 起始点编号 , 结束点编号

[参数说明]

| | |
|--------|----------------|
| 点编号 | 以点位文件中的排序号做索引。 |
| P 点编号 | 以点位文件中的排序号做索引。 |
| P(点编号) | 以点位文件中的排序号做索引。 |

[使用案例]

Function main

PList '打印所有点位数据

PList 1 '打印 1 号点位数据

PList 2 , 10 '打印 2 号点至 10 号点数据

Fend

PLocal

[功能]

用于指定 / 获取点位的用户坐标信息。

[语法]

PLocal (点编号)

PLocal (点编号) = Value

[参数说明]

点编号 以点位文件中的排序号做索引。

Value 指定的用户坐标系。

[使用案例]

Function main

```
Print PLocal (1)                '打印 1 号点所属的用户坐标
```

```
PLocal (1) = 1                 '设置 1 号点所属的用户坐标
```

Fend

PDescription

[功能]

用于设置指定点的描述。

[语法]

`PDescription` 点编号 , 新描述

[参数说明]

点编号 以点位文件中的排序号做索引。

新描述 显示于点位文件中的描述数据。

[使用案例]

`Function` main

```
    PDescription 1 , "新的描述"                      '将 P1 号点的描述内容改为 "新的描述"
```

```
    Print PDescription$(1)                      '打印 P1 号点的描述
```

`End`

PDescription\$

[功能]

用于返回指定点的描述。

[语法]

`PDescription$` (点编号)

`PDescription$` (P 点编号)

`PDescription$` (P (点编号))

`PDescription$` (标签名称)

[参数说明]

| | |
|--------|-----------------------------|
| 点编号 | 以点位文件中的排序号做索引。 |
| P 点编号 | 以点位文件中的排序号做索引。 |
| P(点编号) | 以点位文件中的排序号做索引。 |
| 标签名称 | 赋予标签变量后，依据变量值在点位文件中的排序号做索引。 |

[使用案例]

`Function` main

```
PDescription 1 , "新的描述" '将 P1 号点的描述内容改为 "新的描述"
```

```
Print PDescription$(1) '打印 P1 号点的描述
```

```
Print PDescription$(P1) '打印 P1 号点的描述
```

```
Print PDescription$(P(1)) '打印 P1 号点的描述
```

`End`

GetPointFiles

[功能]

用于获取点位文件列表。

[语法]

`GetPointFiles` 字符串数组\$()

[参数说明]

字符串数组 将点位表的名称存放与变量数组中。

[使用案例]

```
Function main
    Integer i
    String names$(0)

    GetPointFiles names$() '将点位表名称放入数组中

    For i = 0 To UBound(names$())
        Print names$(i)
    Next
End
```

DI Description

[功能]

用于设置数字量输入数据的描述。

[语法]

`DI Description` 编号 , 描述

[参数说明]

编号 以 0 - 1023 指定需要修改描述的数字量输入编号。

描述 以字符串格式提供需要写入的描述。

[使用案例]

`Function main`

```
DI Description 0 , "新的描述"     '修改 DI-0 的描述为“新的描述”
```

`End`

DI>Description\$

[功能]

用于返回数字量输入数据的描述。

[语法]

`DI>Description$(编号)`

[参数说明]

编号 以 0 - 1023 指定需要返回描述的 DI 编号。

[使用案例]

```
Function main
```

```
    String AA$  
    AA$ = DI>Description$(1)  
    Print AA$                      '打印 DI-1 的描述  
  
    Print DI>Description$(1) '打印 DI-1 的描述
```

```
End
```

D0Description

[功能]

用于设置数字量输出数据的描述。

[语法]

`D0Description` 编号 , 描述

[参数说明]

编号 以 0 - 1023 指定需要修改描述的数字量编号。

描述 以字符串格式提供需要写入的描述。

[使用案例]

`Function` main

```
    D0Description 0 , "新的描述"        '修改 D0-0 的描述为“新的描述”
```

`End`

D0Description\$

[功能]

用于返回数字量输出数据的描述。

[语法]

`D0Description$(编号)`

[参数说明]

编号 以 0 - 1023 指定需要返回描述的 D0 编号。

[使用案例]

```
Function main
```

```
    String AA$  
    AA$ = D0Description$(1)  
    Print AA$                      '打印 D0-1 的描述  
  
    Print D0Description$(1) '打印 D0-1 的描述
```

```
End
```

AI Description

[功能]

用于设置模拟量输入的描述。

[语法]

`AI Description` 编号 , 描述

[参数说明]

编号 以 0 - 255 指定需要修改描述 AI 编号。

描述 以字符串格式提供需要写入的描述。

[使用案例]

`Function main`

```
AI Description 0 , "新的描述"        '修改 AI-0 的描述为“新的描述”
```

`Fend`

AI>Description\$

[功能]

用于返回模拟量输入数据的描述。

[语法]

AI>Description\$(编号)

[参数说明]

编号 以 0 - 255 指定需要返回描述的 AI 编号。

[使用案例]

```
Function main
```

```
    String AA$  
    AA$ = AI>Description$(1)  
    Print AA$                      '打印 AI-1 的描述  
  
    Print AI>Description$(1) '打印 AI-1 的描述
```

```
End
```

A0Description

[功能]

用于设置模拟量输出的描述。

[语法]

`A0Description` 编号 , 描述

[参数说明]

编号 以 0 - 255 指定需要修改描述的 A0 编号。

描述 以字符串格式提供需要写入的描述。

[使用案例]

`Function` main

```
    A0Description 0 , "新的描述"        '修改 A0-0 的描述为“新的描述”
```

`End`

A0Description\$

[功能]

用于返回模拟量输出数据的描述。

[语法]

`A0Description$(编号)`

[参数说明]

编号 以 0 - 255 指定需要返回描述的 A0 编号。

[使用案例]

```
Function main
```

```
    String AA$  
    AA$ = A0Description$(1)  
    Print AA$                      '打印 A0-1 的描述  
  
    Print A0Description$(1) '打印 A0-1 的描述
```

```
End
```

EncDescription

[功能]

用于设置编码器数据的描述。

[语法]

`EncDescription` 编号 , 描述

[参数说明]

编号 以 0 - 7 指定需要修改描述的 Enc 编号。

描述 以字符串格式提供需要写入的描述。

[使用案例]

`Function` main

```
    EncDescription 0 , "新的描述"    '修改 Enc-0 的描述为 "新的描述"
```

`Fend`

EncDescription\$

[功能]

用于返回编码器数据的描述。

[语法]

`EncDescription$(编号)`

[参数说明]

编号 以 0 - 7 指定需要返回描述的 Enc 编号。

[使用案例]

Function main

```
String AA$  
AA$ = EncDescription$(0)  
Print AA$                      '打印 Enc-0 的描述  
  
Print EncDescription$(0)      '打印 Enc-0 的描述
```

End

I/O 指令详细说明

Wait

[功能]

延时指令、或在一定时间内等待条件成立。

[语法]

Wait 时间

Wait 条件表达式

Wait 条件表达式, 时间

[参数说明]

时间 以 0~2147483 之间的数指定延时的时间。单位：秒。最小延时 0.01 秒。

条件表达式 运动过程中停在目标点上方的条件。可使用各种 I/O 指令、运算符等。

[使用案例]

Function main

'等待输入 0 变为 On 状态

```
Wait Sw(0) = On
```

'等待 60.5 秒后继续执行

```
Wait 60.5
```

'等待输入 0 变为 OFF、输入 1 变为 On 状态

```
Wait Sw(0) = Off And Sw(1) = On
```

'等待寄存器 bit 位 0 变为 On 或寄存器 bit 位 1 变为 On

```
Wait MemSw(0) = On Or MemSw(1) = On
```

'等待 1 秒钟, 然后将输出 1 设为 On

```
Wait 1; On 1
```

Fend

Mask

[功能]

以位为单位，用于对表示 Wait 输入条件的值进行位 And 运算。

[语法]

Wait 条件表达式 Mask Bit 位最大值 = 条件返回值

[参数说明]

| | |
|----------|--------------------------------|
| Wait | 延时指令、或在一定时间内等待条件成立。 |
| 条件表达式 | 指定的 DIO bit 位或组别、寄存器地址或 bit 位。 |
| Mask | 用于对表示 Wait 输入条件的值进行位 And 运算。 |
| Bit 位最大值 | 索引的 Bit 位最大值。 |
| 条件达成值 | 满足当前条件表达式的目标值。 |

[使用案例]

```
Function main
  Do
    Wait InW(0) Mask 1 = 0
    '等待 16 位 DI 组 0(前 1bit 位) 返回值为 0 时满足跳出条件

    Wait InW(0) Mask 3 = 2
    '等待 16 位 DI 组 0(前 2bit 位) 返回值为 2 时满足跳出条件

    Wait InW(0) Mask 7 = 4
    '等待 16 位 DI 组 0(前 3bit 位) 返回值为 4 时满足跳出条件

    Wait InW(0) Mask 15 = 8
    '等待 16 位 DI 组 0(前 4bit 位) 返回值为 8 时满足跳出条件

    Wait InW(0) Mask 31 = 16
    '等待 16 位 DI 组 0(前 5bit 位) 返回值为 16 时满足跳出条件

    Wait InW(0) Mask 63 = 32
    '等待 16 位 DI 组 0(前 6bit 位) 返回值为 32 时满足跳出条件

  Loop
Fend
```

On

[功能]

控制 D0 置 On, 或置 On 一定时间后置 Off。

[语法]

On D0 编号 {, 时间, 同步处理}

[参数说明]

| | |
|--------------|--|
| D0 编号 | 以整数指定 D0 编号, 或使用 D0 标签指定。 |
| 时间 | 以秒指定 D0 置 On 的时长。达到该时长后自动置 Off。 |
| 同步处理 | 指定了 On 的时间后, 可设置同步处理的方法来指定后续指令的执行时序, 可省略 0 - 将 D0 设为 On 的同时执行后续命令, 此时的最大时间设置为 10 秒。 1 - 默认设置: 将 D0 设为 On, 等待指定的时间后设为 Off, 才允许后续指令执行。 |

[使用案例]

Function main

```
On 1           'Do1 至为 on  
On 3           'Do3 至为 on  
  
On 5 , 1      'Do5 至为 on, 1 秒后置 off
```

Fend

Off

[功能]

控制 D0 置 Off, 或置 Off 一定时间后置 On。

[语法]

Off D0 编号 {, 时间 , 同步处理}

[参数说明]

D0 编号/D0 标签 以整数指定 D0 编号, 或使用 D0 标签指定。

时间 以秒指定 D0 置 Off 的时长。达到该时长后自动置 On。

同步处理 指定了 Off 的时间后, 可设置同步处理的方法来指定后续指令的执行时序, 可省略

0 - 将 D0 设为 Off 的同时执行后续命令, 此时的最大时间设置为 10 秒。

1 - 默认设置 : 将 D0 设为 Off, 等待指定的时间后设为 On, 才允许后续指令执行。

[使用案例]

Function main

```
Off 1           'Do1 至为 off
Off 3           'Do3 至为 off

Off 5 , 1      'Do5 至为 off, 1 秒后置 On
```

End

Oport

[功能]

用于返回指定 D0 状态。

[语法]

Oport (D0 编号)

[参数说明]

D0 编号 以整数指定需要获取状态的 D0。

注意事项：

1. D0 为 On 返回 1。
2. D0 为 Off 返回 0。

[使用案例]

Function main

```
    If Oport(1) = 0 Then  
        On 1  
    EndIf
```

```
    Wait Oport(3)
```

Fend

Sw

[功能]

用于返回指定 DI 状态。

[语法]

Sw(DI 编号)

[参数说明]

DI 编号 以整数指定需要获取状态的 DI。

注意事项：

3. DI 为 On 返回 1。
4. DI 为 Off 返回 0。

[使用案例]

Function main

```
If Sw(1) = 0 Then  
    SetSw 1 , On  
EndIf
```

```
Wait Sw(1)
```

Fend

SetSw

[功能]

用于将指定 DI 置 On 或置 Off。

[语法]

```
SetSw DI 编号, 状态
```

[参数说明]

DI 编号 以整数指定需要控制的 DI。

状态 0 为 Off, 1 为 On。填入 Off 或 On 也能生效。

[使用案例]

```
Function main
```

```
    SetSw 1 , On
```

```
    SetSw 4 , Off
```

```
Fend
```

In

[功能]

根据 8421 返回 8 位 DI 的状态, 0~255。

[语法]

In(DI 组编号)

[参数说明]

DI 组编号 以整数指定需要读取状态的一组 DI。

注意事项:

1. DI 组编号取 0, 返回 DI0~DI7 的状态。输入 1, 返回 DI8~DI15 的状态。以此类推。
2. 8421 原理示意图如下:

| 返回值 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-----|-----|-----|-----|-----|-----|-----|-----|----|
| 1 | Off | Off | Off | Off | Off | Off | Off | On |
| 5 | Off | Off | Off | Off | Off | On | Off | On |
| 15 | Off | Off | Off | Off | On | On | On | On |
| 255 | On | On | On | On | On | On | On | On |

[使用案例]

Function main

```
Print In(0)
```

```
Integer A1
```

```
A1 = In(1)
```

```
Select A1
```

```
Case 0
```

```
Go P1
```

```
Case 8
```

```
Go P2
```

```
Default
```

```
Go P3
```

```
Send
```

```
Fend
```

SetIn

[功能]

根据 8421 设定 8 位 DI 的状态，0~255。

[语法]

SetIn DI 组编号, 状态值

[参数说明]

DI 组编号 以整数指定需要设定状态的一组 DI。

状态值 根据 8421 设定该组 DI 的状态。

注意事项：

1. DI 组编号取 0，是 DI0~DI7。输入 1，是 DI8~DI15。以此类推。
2. 8421 原理示意图如下：

| 返回值 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-----|-----|-----|-----|-----|-----|-----|-----|----|
| 1 | Off | Off | Off | Off | Off | Off | Off | On |
| 5 | Off | Off | Off | Off | Off | On | Off | On |
| 15 | Off | Off | Off | Off | On | On | On | On |
| 255 | On | On | On | On | On | On | On | On |

[使用案例]

```
Function main
    SetIn 0 ,85           '将 DI0、2、4、6 置 On
Fend
```

Out

[功能]

根据 8421 设置/返回 8 位 D0 的状态, 0~255。

[语法]

Out D0 组编号, 状态值

Out (D0 组编号)

[参数说明]

D0 组编号 以整数指定需要设置或读取状态的一组 D0。

状态值 根据 8421 设定该组 D0 的状态。

注意事项:

1. D0 组编号取 0, 是 D00~D07。输入 1, 是 D08~D015。以此类推。
2. 8421 原理示意图如下:

| 返回值 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-----|-----|-----|-----|-----|-----|-----|-----|----|
| 1 | Off | Off | Off | Off | Off | Off | Off | On |
| 5 | Off | Off | Off | Off | Off | On | Off | On |
| 15 | Off | Off | Off | Off | On | On | On | On |
| 255 | On | On | On | On | On | On | On | On |

[使用案例]

```
Function main
    Out 0 ,85                '将 D00、2、4、6 置 On

    If Out(0) = 0 Then      '读取 D00~D07 的状态
        Out 0 ,85
    EndIf
Fend
```

InW

[功能]

根据 8421 返回 16 位 DI 的状态，0~65535。

[语法]

InW(DI 组编号)

[参数说明]

DI 组编号 以整数指定需要读取状态的一组 DI。

注意事项：

- DI 组编号取 0，是 DI0~DI15 的状态。输入 1，是 DI16~DI31 的状态。以此类推。
- 8421 原理示意图如下：

| 返回值 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-----|-----|-----|-----|-----|-----|-----|-----|----|
| 1 | Off | Off | Off | Off | Off | Off | Off | On |
| 5 | Off | Off | Off | Off | Off | On | Off | On |
| 15 | Off | Off | Off | Off | On | On | On | On |
| 255 | On | On | On | On | On | On | On | On |

[使用案例]

```
Function main
```

```
  Print InW(0)
```

```
  Integer A1
```

```
  A1 = InW(1)
```

```
  Select A1
```

```
  Case 0
```

```
    Go P1
```

```
  Case 8
```

```
    Go P2
```

```
  Default
```

```
    Go P3
```

```
  Send
```

```
Fend
```

SetInW

[功能]

根据 8421 设定 16 位 DI 的状态, 0~65535。

[语法]

SetInW DI 组编号 , 状态值

[参数说明]

DI 组编号 以整数指定需要设定状态的一组 DI。

状态值 根据 8421 设定该组 DI 的状态。

注意事项:

1. DI 组编号取 0, 是 DI0~DI15。输入 1, 是 DI16~DI31。以此类推。
2. 8421 原理示意图如下:

| 返回值 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-----|-----|-----|-----|-----|-----|-----|-----|----|
| 1 | Off | Off | Off | Off | Off | Off | Off | On |
| 5 | Off | Off | Off | Off | Off | On | Off | On |
| 15 | Off | Off | Off | Off | On | On | On | On |
| 255 | On | On | On | On | On | On | On | On |

[使用案例]

```
Function main
    SetInW 0 ,85          '将 DI0、2、4、6 置 On
End
```

OutW

[功能]

根据 8421 设置/返回 16 位 D0 的状态, 0~65535。

[语法]

OutW D0 组编号, 状态值

OutW(D0 组编号)

[参数说明]

D0 组编号 以整数指定需要设置或读取状态的一组 D0。

状态值 根据 8421 设定该组 D0 的状态。

注意事项:

1. D0 组编号取 0, 是 D00~D015。输入 1, 是 D016~D031。以此类推。
2. 8421 原理示意图如下:

| 返回值 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-----|-----|-----|-----|-----|-----|-----|-----|----|
| 1 | Off | Off | Off | Off | Off | Off | Off | On |
| 5 | Off | Off | Off | Off | Off | On | Off | On |
| 15 | Off | Off | Off | Off | On | On | On | On |
| 255 | On | On | On | On | On | On | On | On |

[使用案例]

```
Function main
    OutW 0 ,85                '将 D00、2、4、6 置 On

    If OutW(0) = 0 Then      '读取 D00~D015 的状态
        OutW 0 ,85
    EndIf
Fend
```

InBCD

[功能]

根据 8421 返回 8 位 DI 的状态，0~99。

[语法]

InBCD (DI 组编号)

[参数说明]

DI 组编号 以整数指定需要读取状态的一组 DI。

注意事项：

1. DI 组编号取 0，返回 DI0~DI7 的状态。输入 1，返回 DI8~DI15 的状态。以此类推。
2. 该函数与 In 的区别在于取值范围不同。In 取值范围是 0~255。
3. 8421 原理示意图如下：

| 返回值 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-----|-----|-----|-----|-----|-----|-----|-----|----|
| 1 | Off | Off | Off | Off | Off | Off | Off | On |
| 5 | Off | Off | Off | Off | Off | On | Off | On |
| 15 | Off | Off | Off | Off | On | On | On | On |
| 255 | On | On | On | On | On | On | On | On |

[使用案例]

```
Function main
```

```
  Print InBCD(0)
```

```
  Integer A1
```

```
  A1 = InBCD(1)
```

```
  Select A1
```

```
  Case 0
```

```
    Go P1
```

```
  Case 8
```

```
    Go P2
```

```
  Default
```

```
    Go P3
```

```
  Send
```

```
Fend
```

OpBCD

[功能]

根据 8421 设置 8 位 D0 的状态, 0~99。

[语法]

OpBCD D0 组编号, 状态值

[参数说明]

D0 组编号 以整数指定需要设置或读取状态的一组 D0。

状态值 根据 8421 设定该组 D0 的状态。

注意事项:

1. D0 组编号取 0, 是 D00~D07。输入 1, 是 D08~D015。以此类推。
2. 该函数与 Out 的区别在于取值范围不同。Out 取值范围是 0~255。
3. 8421 原理示意图如下:

| 返回值 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-----|-----|-----|-----|-----|-----|-----|-----|----|
| 1 | Off | Off | Off | Off | Off | Off | Off | On |
| 5 | Off | Off | Off | Off | Off | On | Off | On |
| 15 | Off | Off | Off | Off | On | On | On | On |
| 255 | On | On | On | On | On | On | On | On |

[使用案例]

```
Function main
    OpBCD 0 ,85          '将 D00、2、4、6 置 On
Fend
```

InReal

[功能]

以 32 位浮点数返回 DI 状态。

[语法]

`InReal` (DI 组编号)

[参数说明]

DI 组编号 以整数指定需要设置或读取状态的一组 D0。

注意事项：

1. DI 组编号取 0，是 DI0~DI31。输入 1，是 DI32~DI63。以此类推。

[使用案例]

```
Function main
    Real realVal

    realVal = InReal (32)
End
```

OutReal

[功能]

以 32 位浮点数设置/返回 D0 状态。

[语法]

`OutReal` D0 组编号, 状态值

`OutReal` (D0 组编号)

[参数说明]

D0 组编号 以整数指定需要设置或读取状态的一组 D0。

状态值 根据 32 位浮点数设定该组 D0 的状态。

注意事项:

1. D0 组编号取 0, 是 D00~D031。输入 1, 是 D032~D063。以此类推。

[使用案例]

```
Function main
    OutReal 32 , 2.543
End
```

AIO_InW

[功能]

读取模拟量 AI 的值。

[语法]

`AIO_InW(AI 编号, {Signed/Unsigned})`

[参数说明]

AI 编号 以整数指定需要读取状态的模拟量 AI。

Signed/Unsigned 读取有符号数值或无符号数值。默认无符号。

[使用案例]

```
Function main
  If AIO_InW(1) > 123456 Then
    Print "模拟量大于 123456"
  EndIf
Fend
```

AIO_OutW

[功能]

设置/读取模拟量 A0 的值。

[语法]

`AIO_OutW AI 编号, 模拟量值`

`AIO_OutW(AI 编号)`

[参数说明]

AI 编号 以整数指定需要设置或读取状态的模拟量 A0。

模拟量值 为该 A0 设定值。

[使用案例]

```
Function main
  If AIO_OutW(1) > 123456 Then
    AIO_OutW 1, 0                'A01 模拟量大于 123456 清零
  EndIf
Fend
```

寄存器读写指令详细说明

MemOn

[功能]

指定用户寄存器中的一位 bit 为 1。

[语法]

MemOn bit 位

[参数说明]

bit 位 指定第几个 bit 为 1。

注意事项：

1. 一个用户寄存器是 16 位，因此 0~15bit 是第 0 个寄存器。16~31bit 是第 1 个寄存器，以此类推。

[使用案例]

Function main

 MemOn 3 '将第一个寄存器的第 3 个 Bit 置 1

 MemOn 16 '将第二个寄存器的第 0 个 Bit 置 1

End

MemOff

[功能]

指定用户寄存器中的一位 bit 为 0。

[语法]

```
MemOff bit 位
```

[参数说明]

bit 位 指定第几个 bit 为 0。

注意事项：

1. 一个用户寄存器是 16 位，因此 0~15bit 是第 0 个寄存器。16~31bit 是第 1 个寄存器，以此类推。

[使用案例]

```
Function main
```

```
    MemOff 3            '将第一个寄存器的第 3 个 Bit 置 0
```

```
    MemOff 16          '将第二个寄存器的第 0 个 Bit 置 0
```

```
Fend
```

MemSw

[功能]

获取用户寄存器中的一位 bit 的值。

[语法]

MemSw (bit 位)

[参数说明]

bit 位 指定第几个 bit。

注意事项：

1. 一个用户寄存器是 16 位，因此 0~15bit 是第 0 个寄存器。16~31bit 是第 1 个寄存器，以此类推。

[使用案例]

Function main

```
    If MemSw (16) = 1 Then                    '获取第二个寄存器第 0 个 Bit 值
        Go P1
    Else
        Go P2
    EndIf
```

Fend

MemIn

[功能]

获取用户寄存器中 8 位的值。

[语法]

`MemIn`(bit 编号组, {Signed/Unsigned})

[参数说明]

bit 编号组 指定第几组 bit。

Signed/Unsigned 读取有符号数值或无符号数值。默认无符号。

注意事项：

1. 一个用户寄存器是 16 位，因此 0~1 分别是第 0 个寄存器的低八位和高八位。2~3 是第 1 个寄存器的低八位和高八位。以此类推。

| Bit 编号组 | 对应地址 |
|---------|------------------------|
| 0 | 用户寄存器 0 (Bit0 - Bit7) |
| 1 | 用户寄存器 0 (Bit8 - Bit15) |
| 2 | 用户寄存器 1 (Bit0 - Bit7) |
| 3 | 用户寄存器 1 (Bit8 - Bit15) |
| 4 | 用户寄存器 2 (Bit0 - Bit7) |
| 5 | 用户寄存器 2 (Bit8 - Bit15) |

[使用案例]

`Function main`

```
Print MemIn(2)
```

```
'获取第一个寄存器的低八位
```

```
Print MemIn(1, Signed)
```

```
'以有符号数值的形式获取第 0 个寄存器的高八位
```

`End`

MemOut

[功能]

设置用户寄存器中 8 位的值。

[语法]

`MemOut bit 编号组, 值`

[参数说明]

bit 编号组 指定第几组 bit。

值 给寄存器设定的值。

注意事项：

1. 一个用户寄存器是 16 位，因此 0~1 分别是第 0 个寄存器的低八位和高八位。2~3 是第 1 个寄存器的低八位和高八位。以此类推。

| Bit 编号组 | 对应地址 |
|---------|------------------------|
| 0 | 用户寄存器 0 (Bit0 - Bit7) |
| 1 | 用户寄存器 0 (Bit8 - Bit15) |
| 2 | 用户寄存器 1 (Bit0 - Bit7) |
| 3 | 用户寄存器 1 (Bit8 - Bit15) |
| 4 | 用户寄存器 2 (Bit0 - Bit7) |
| 5 | 用户寄存器 2 (Bit8 - Bit15) |

[使用案例]

```
Function main
```

```
    MemOut 1 , 123
```

```
    '给第 0 个寄存器的高八位设定为 123
```

```
End
```

MemInW

[功能]

获取用户寄存器中 16 位的值。

[语法]

`MemInW`(寄存器编号, {Signed/Unsigned})

[参数说明]

寄存器编号 指定第几个寄存器。

Signed/Unsigned 读取有符号数值或无符号数值。默认无符号。

注意事项：

1. 一个用户寄存器是 16 位，因此 0 是第 0 个寄存器。1 是第 1 个寄存器。以此类推。

[使用案例]

`Function` main

```
Print MemInW(1)
```

```
' 获取第一个寄存器的值
```

```
Print MemInW(2, Signal)
```

```
' 以有符号数值的形式获取第 2 个寄存器的值
```

`End`

MemOutW

[功能]

设置用户寄存器中 16 位的值。

[语法]

`MemOutW` 寄存器编号, 值

[参数说明]

寄存器编号 指定第几个寄存器。

值 给寄存器设定的值。

注意事项:

1. 一个用户寄存器是 16 位，因此 0 是第 0 个寄存器。1 是第 1 个寄存器。以此类推。

[使用案例]

Function main

```
MemOutW 5 , 1234
```

```
'给第 5 个寄存器的值设定为 1234
```

End

MemIn32

[功能]

获取用户寄存器中 32 位的值。

[语法]

MemIn32(寄存器组, {Signed/Unsigned})

[参数说明]

寄存器组 指定第几组寄存器。

值 给寄存器设定的值。

注意事项：

1. 一个用户寄存器是 16 位，因此 0 是第 0 个寄存器+第 1 个寄存器合并的 32 位数值。1 是第 2 个寄存器+第 3 个寄存器合并的 32 位数值。以此类推。

| | |
|------|---------------------|
| 寄存器组 | 对应地址 |
| 0 | 用户寄存器 0 + 用户寄存器 1 |
| 1 | 用户寄存器 2 + 用户寄存器 3 |
| 2 | 用户寄存器 4 + 用户寄存器 5 |
| 3 | 用户寄存器 6 + 用户寄存器 7 |
| 4 | 用户寄存器 8 + 用户寄存器 9 |
| 5 | 用户寄存器 10 + 用户寄存器 11 |

[使用案例]

Function main

```
Print MemIn32(0)
```

'获取第 0 个寄存器和第 1 个寄存器合并的 32 位数值

```
Print MemIn32(1, Signal)
```

'以有符号数值的形式，获取第 2 个寄存器和第 3 个寄存器合并的 32 位数值

Fend

MemOut32

[功能]

设置用户寄存器中 32 位的值。

[语法]

MemOut32 寄存器组, 值

[参数说明]

寄存器组 指定第几组寄存器。

值 给寄存器设定的值。

注意事项：

1. 一个用户寄存器是 16 位，因此 0 是第 0 个寄存器+第 1 个寄存器合并的 32 位数值。1 是第 2 个寄存器+第 3 个寄存器合并的 32 位数值。以此类推。

| 寄存器组 | 对应地址 |
|------|---------------------|
| 0 | 用户寄存器 0 + 用户寄存器 1 |
| 1 | 用户寄存器 2 + 用户寄存器 3 |
| 2 | 用户寄存器 4 + 用户寄存器 5 |
| 3 | 用户寄存器 6 + 用户寄存器 7 |
| 4 | 用户寄存器 8 + 用户寄存器 9 |
| 5 | 用户寄存器 10 + 用户寄存器 11 |

[使用案例]

```
Function main
```

```
    MemOut32 1 , 1234
```

```
    '给第 2 个寄存器和第 3 个寄存器合并的 32 位数值设置为 1234
```

```
End
```

MemInReal

[功能]

以 32 位浮点数形式获取用户寄存器中 32 位的值。

[语法]

`MemInReal` (寄存器组)

[参数说明]

寄存器组 指定第几组寄存器。

注意事项：

1. 一个用户寄存器是 16 位，因此 0 是第 0 个寄存器+第 1 个寄存器合并的 32 位数值。1 是第 2 个寄存器+第 3 个寄存器合并的 32 位数值。以此类推。

| 寄存器组 | 对应地址 |
|------|---------------------|
| 0 | 用户寄存器 0 + 用户寄存器 1 |
| 1 | 用户寄存器 2 + 用户寄存器 3 |
| 2 | 用户寄存器 4 + 用户寄存器 5 |
| 3 | 用户寄存器 6 + 用户寄存器 7 |
| 4 | 用户寄存器 8 + 用户寄存器 9 |
| 5 | 用户寄存器 10 + 用户寄存器 11 |

[使用案例]

`Function main`

```
Print MemInReal(0)
```

```
'以 32 位浮点数获取第 0 个寄存器和第 1 个寄存器合并的 32 位数值'
```

`Fend`

MemOutReal

[功能]

设置用户寄存器中 32 位的值。

[语法]

MemOutReal 寄存器组, 值

[参数说明]

寄存器组 指定第几组寄存器。

值 给寄存器设定的值。

| 寄存器组 | 对应地址 |
|------|---------------------|
| 0 | 用户寄存器 0 + 用户寄存器 1 |
| 1 | 用户寄存器 2 + 用户寄存器 3 |
| 2 | 用户寄存器 4 + 用户寄存器 5 |
| 3 | 用户寄存器 6 + 用户寄存器 7 |
| 4 | 用户寄存器 8 + 用户寄存器 9 |
| 5 | 用户寄存器 10 + 用户寄存器 11 |

[注意事项:]

1. 一个用户寄存器是 16 位，因此 0 是第 0 个寄存器+第 1 个寄存器合并的 32 位数值。1 是第 2 个寄存器+第 3 个寄存器合并的 32 位数值。以此类推。

[使用案例]

Function main

 MemOutReal 1 , 123.321

 '给第 2 个寄存器和第 3 个寄存器合并的 32 位数值设置为 123.321

End

SysMemOn

[功能]

指定系统寄存器中的一位 bit 为 1。

[语法]

`SysMemOn` bit 编号

[参数说明]

bit 编号 指定第几个 bit 为 1。

注意事项：

1. 一个系统寄存器是 16 位，因此 0~15bit 是第 0 个寄存器。16~31bit 是第 1 个寄存器，以此类推。

[使用案例]

`Function` main

```
    SysMemOn 0           '将第 0 个系统寄存器的第 1 个 Bit 置 On
    SysMemOn 15          '将第 0 个系统寄存器的第 16 个 Bit 置 On
    SysMemOn 16          '将第 1 个系统寄存器的第 1 个 Bit 置 On
```

`Fend`

SysMemOff

[功能]

指定系统寄存器中的一位 bit 为 0。

[语法]

`SysMemOff bit 编号`

[参数说明]

bit 编号 指定第几个 bit 为 0。

注意事项：

1. 一个系统寄存器是 16 位，因此 0~15bit 是第 0 个寄存器。16~31bit 是第 1 个寄存器，以此类推。

[使用案例]

`Function main`

```
    SysMemOff 0                    '将第 0 个系统寄存器的第 1 个 Bit 置 Off
```

```
    SysMemOff 15                  '将第 0 个系统寄存器的第 16 个 Bit 置 Off
```

```
    SysMemOff 16                  '将第 1 个系统寄存器的第 1 个 Bit 置 Off
```

`Fend`

SysMemSw

[功能]

获取系统寄存器中的一位 bit 的值。

[语法]

SysMemSw bit 编号

[参数说明]

bit 编号 指定第几组 bit。

注意事项：

1. 一个系统寄存器是 16 位，因此 0~15bit 是第 0 个寄存器。16~31bit 是第 1 个寄存器，以此类推。

[使用案例]

Function main

```
    If SysMemSw(0) = On Then                    '第 0 个系统寄存器的第一位 bit 值为 on 时成立
        Go P1

    ElseIf SysMemSw(15) = On Then            '第 0 个系统寄存器的第十六位 bit 值为 on 时成立
        Go P2
    EndIf
```

Fend

SysMemIn

[功能]

获取系统寄存器中 8 位的值。

[语法]

`SysMemIn`(bit 编号组, {Signed/Unsigned})

[参数说明]

bit 编号组 指定第几组 bit。

Signed/Unsigned 读取有符号数值或无符号数值。默认无符号。

注意事项：

1. 一个系统寄存器是 16 位，因此 0~1 分别是第 0 个寄存器的低八位和高八位。2~3 是第 1 个寄存器的低八位和高八位。以此类推。

| Bit 编号组 | 对应地址 |
|---------|------------------------|
| 0 | 用户寄存器 0 (Bit0 - Bit7) |
| 1 | 用户寄存器 0 (Bit8 - Bit15) |
| 2 | 用户寄存器 1 (Bit0 - Bit7) |
| 3 | 用户寄存器 1 (Bit8 - Bit15) |
| 4 | 用户寄存器 2 (Bit0 - Bit7) |
| 5 | 用户寄存器 2 (Bit8 - Bit15) |

[使用案例]

`Function` main

```
Print SysMemIn(0)
```

'打印第 0 个寄存器的低八位'

```
Print SysMemIn(1, Unsigned)
```

'以无符号数值的形式'打印第 0 个寄存器的高八位'

`End`

SysMemOut

[功能]

设置系统寄存器中 8 位的值。

[语法]

`SysMemOut` bit 编号组, 值

[参数说明]

bit 编号组 指定第几组 bit。

值 给寄存器设定的值。

注意事项：

1. 一个系统寄存器是 16 位，因此 0~1 分别是第 0 个寄存器的低八位和高八位。2~3 是第 1 个寄存器的低八位和高八位。以此类推。

| Bit 编号组 | 对应地址 |
|---------|------------------------|
| 0 | 用户寄存器 0 (Bit0 - Bit7) |
| 1 | 用户寄存器 0 (Bit8 - Bit15) |
| 2 | 用户寄存器 1 (Bit0 - Bit7) |
| 3 | 用户寄存器 1 (Bit8 - Bit15) |
| 4 | 用户寄存器 2 (Bit0 - Bit7) |
| 5 | 用户寄存器 2 (Bit8 - Bit15) |

[使用案例]

```
Function main
```

```
    SysMemOut 1 , 123
```

```
    '设置第 0 个寄存器高八位为 123
```

```
End
```

SysMemInW

[功能]

获取系统寄存器中 16 位的值。

[语法]

`SysMemInW` 寄存器编号, {Signed/Unsigned})

[参数说明]

寄存器编号 指定第几个寄存器。

Signed/Unsigned 读取有符号数值或无符号数值。默认无符号。

注意事项:

1. 一个系统寄存器是 16 位，因此 0 是第 0 个寄存器。1 是第 1 个寄存器。以此类推。

[使用案例]

`Function` main

```
Print SysMemInW(0, Unsigned)
```

'以无符号整形格式打印 0 号系统寄存器'

```
Print SysMemInW(1, Signed)
```

'以有符号数值的形式打印 1 号系统寄存器'

`End`

SysMemOutW

[功能]

设置系统寄存器中 16 位的值。

[语法]

`SysMemOutW` 寄存器编号, 值

[参数说明]

寄存器编号 指定第几个寄存器。

值 给寄存器设定的值。

注意事项:

1. 一个系统寄存器是 16 位，因此 0 是第 0 个寄存器。1 是第 1 个寄存器。以此类推。

[使用案例]

Function main

```
    SysMemOutW 0 , 123
```

```
    '将第 0 个寄存器地址写入值 123
```

End

SysMemIn32

[功能]

获取系统寄存器中 32 位的值。

[语法]

`SysMemIn32` (寄存器组, {Signed/Unsigned})

[参数说明]

寄存器组 指定第几组寄存器。

Signed/Unsigned 读取有符号数值或无符号数值。默认无符号。

注意事项：

1. 一个系统寄存器是 16 位，因此 0 是第 0 个寄存器+第 1 个寄存器合并的 32 位数值。1 是第 2 个寄存器+第 3 个寄存器合并的 32 位数值。以此类推。

| 寄存器组 | 对应地址 |
|------|---------------------|
| 0 | 用户寄存器 0 + 用户寄存器 1 |
| 1 | 用户寄存器 2 + 用户寄存器 3 |
| 2 | 用户寄存器 4 + 用户寄存器 5 |
| 3 | 用户寄存器 6 + 用户寄存器 7 |
| 4 | 用户寄存器 8 + 用户寄存器 9 |
| 5 | 用户寄存器 10 + 用户寄存器 11 |

[使用案例]

Function main

```
Print SysMemIn32(0)
```

'获取第 0 个寄存器和第 1 个寄存器合并的 32 位数值'

```
Print SysMemIn32(1, Unsigned)
```

'以有符号数值的形式，获取第 2 个寄存器和第 3 个寄存器合并的 32 位数值'

Fend

SysMemOut32

[功能]

设置系统寄存器中 32 位的值。

[语法]

`SysMemOut32` 寄存器组, 值

[参数说明]

寄存器组 指定第几组寄存器。

值 给寄存器设定的值。

注意事项:

1. 一个系统寄存器是 16 位，因此 0 是第 0 个寄存器+第 1 个寄存器合并的 32 位数值。1 是第 2 个寄存器+第 3 个寄存器合并的 32 位数值。以此类推。

| 寄存器组 | 对应地址 |
|------|---------------------|
| 0 | 用户寄存器 0 + 用户寄存器 1 |
| 1 | 用户寄存器 2 + 用户寄存器 3 |
| 2 | 用户寄存器 4 + 用户寄存器 5 |
| 3 | 用户寄存器 6 + 用户寄存器 7 |
| 4 | 用户寄存器 8 + 用户寄存器 9 |
| 5 | 用户寄存器 10 + 用户寄存器 11 |

[使用案例]

```
Function main
```

```
    SysMemOut32 1, 1234
```

```
    '给第 2 个系统寄存器和第 3 个系统寄存器合并的 32 位数值设置为 1234
```

```
End
```

SysMemInReal

[功能]

以 32 位浮点数形式获取系统寄存器中 32 位的值。

[语法]

`SysMemInReal` (寄存器组)

[参数说明]

寄存器组 指定第几组寄存器。

注意事项:

1. 一个系统寄存器是 16 位，因此 0 是第 0 个寄存器+第 1 个寄存器合并的 32 位数值。1 是第 2 个寄存器+第 3 个寄存器合并的 32 位数值。以此类推。

| 寄存器组 | 对应地址 |
|------|---------------------|
| 0 | 用户寄存器 0 + 用户寄存器 1 |
| 1 | 用户寄存器 2 + 用户寄存器 3 |
| 2 | 用户寄存器 4 + 用户寄存器 5 |
| 3 | 用户寄存器 6 + 用户寄存器 7 |
| 4 | 用户寄存器 8 + 用户寄存器 9 |
| 5 | 用户寄存器 10 + 用户寄存器 11 |

[使用案例]

```
Function main
```

```
    Print SysMemInReal (0)
```

```
    '以 32 位浮点数获取第 0 个寄存器和第 1 个寄存器合并的 32 位数值
```

```
Fend
```

SysMemOutReal

[功能]

以 32 位浮点数形式设置系统寄存器中 32 位的值。

[语法]

`SysMemOutReal` 寄存器组, 值

[参数说明]

寄存器组 指定第几组寄存器。

值 给寄存器设定的值。

注意事项:

1. 一个系统寄存器是 16 位，因此 0 是第 0 个寄存器+第 1 个寄存器合并的 32 位数值。1 是第 2 个寄存器+第 3 个寄存器合并的 32 位数值。以此类推。

| | |
|------|---------------------|
| 寄存器组 | 对应地址 |
| 0 | 用户寄存器 0 + 用户寄存器 1 |
| 1 | 用户寄存器 2 + 用户寄存器 3 |
| 2 | 用户寄存器 4 + 用户寄存器 5 |
| 3 | 用户寄存器 6 + 用户寄存器 7 |
| 4 | 用户寄存器 8 + 用户寄存器 9 |
| 5 | 用户寄存器 10 + 用户寄存器 11 |

[使用案例]

```
Function main
```

```
  SysMemOutReal 1 , 123.321
```

```
  '给第 2 个寄存器和第 3 个寄存器合并的 32 位数值设置为 123.321
```

```
End
```

通讯指令详细说明

SetNet

[功能]

用于设置 TCP/IP 的通讯参数。

[语法]

`SetNet #通讯端口编号, IP 地址, 端口号 [, 结束符, 流控制, 超时时间, 通讯协议]`

[参数说明]

| | |
|----------------|---|
| #通讯端口编号 | 指定需要修改通讯参数的端口编号。范围：#201~216。 |
| IP 地址 | 用于 TCP/IP 通讯的 IP 地址。例如：192.168.1.1。 |
| 端口号 | 用于 TCP/IP 通讯的端口号。例如：8090，范围：0~65535。 |
| 结束符 | 通讯数据的末尾字符。可以设定 CR、LF、CRLF (回车、换行、回车换行)，可省略。 |
| 流控制 | 指软件流控制。设定为 NONE，可省略。 |
| 超时时间 | 以秒设定收发的最长时间。0 为永不超时，可省略。 |
| 通讯协议 | 当前只支持 TCP 协议，可省略。 |

[使用案例]

```
Function main
  ReConnect:
  CloseNet #201
  SetNet #201 , "192.168.1.220" , 9000 ,CRLF
  OpenNet #201 As Client
  WaitNet #201, 5
  If ChkNet(201) = -1 Then
    Print "视觉 TCP 通讯错误, 端口已打开, 但是未建立通讯"
    Wait 1
    GoTo ReConnect
  ElseIf ChkNet(201) = -2 Then
    Print "视觉 TCP 通讯错误, 其他任务正在使用端口"
    Wait 1
    GoTo ReConnect
  ElseIf ChkNet(201) = -3 Then
    Print "视觉 TCP 通讯错误, 未打开端口"
    Wait 1
    GoTo ReConnect
  EndIf
Fend
```

OpenNet

[功能]

用于开启 TCP/IP 通讯，或获取 OpenNet 的线程编号。

[语法]

```
OpenNet #通讯端口编号 As [Client/Server]
```

```
OpenNet (通讯端口编号)
```

[参数说明]

#通讯端口编号 指定需要修改通讯参数的端口编号，范围：#201~216。

Client/Server Client 为客户端。Server 为服务器。

注意事项：

1. 使用 OpenNet 作为客户端时，SetNet 指令参数 (IP 地址) 填写服务器 IP 地址。
2. 使用 OpenNet 作为服务器时，SetNet 指令参数 (IP 地址) 填写控制器 IP 地址。

[使用案例]

```
Function main
  ReConnect:
  CloseNet #201
  SetNet #201 , "192.168.1.220" , 9000 ,CRLF
  OpenNet #201 As Client
  WaitNet #201, 5
  If ChkNet(201) = -1 Then
    Print "视觉 TCP 通讯错误，端口已打开，但是未建立通讯"
    Wait 1
    GoTo ReConnect
  ElseIf ChkNet(201) = -2 Then
    Print "视觉 TCP 通讯错误，其他任务正在使用端口"
    Wait 1
    GoTo ReConnect
  ElseIf ChkNet(201) = -3 Then
    Print "视觉 TCP 通讯错误，未打开端口"
    Wait 1
    GoTo ReConnect
  EndIf
Fend
```

WaitNet

[功能]

用于等待 TCP/IP 通讯端口建立连接。

[语法]

```
WaitNet #通讯端口编号 {, 超时时间}
```

[参数说明]

#通讯端口编号 指定需要修改通讯参数的端口编号，范围：#201~216。

超时时间 以秒设定等待时间。省略则无限等待。

注意事项：

1. 超时时间非省略时，程序在该语句行等待指定的超时时间，或满足建立连接条件。

[使用案例]

```
Function main
  ReConnect:
  CloseNet #201
  SetNet #201 , "192.168.1.220" , 9000 ,CRLF
  OpenNet #201 As Client
  WaitNet #201, 5
  If ChkNet(201) = -1 Then
    Print "视觉 TCP 通讯错误，端口已打开，但是未建立通讯"
    Wait 1
    GoTo ReConnect
  ElseIf ChkNet(201) = -2 Then
    Print "视觉 TCP 通讯错误，其他任务正在使用端口"
    Wait 1
    GoTo ReConnect
  ElseIf ChkNet(201) = -3 Then
    Print "视觉 TCP 通讯错误，未打开端口"
    Wait 1
    GoTo ReConnect
  EndIf
Fend
```

ChkNet

[功能]

用于返回通讯端口接收缓存器内的字节数。

[语法]

ChkNet (通讯端口编号)

[参数说明]

通讯端口编号 指定需要修改通讯参数的端口编号，范围：201~216。

注意事项：

1. 以整数返回字节数。如果数据不存在，以负值返回通讯端口状态。
 - 1 为控制器端口已开启，但未找到目标对象。（检查通讯目标是否在同一局域网内）
 - 2 为其他线程(函数)正在使用该通讯端口。（仅允许在同一线程内调用端口）
 - 3 为控制器端口未能正确启用。（检查 SetNet 指令设定的 IP 地址是否正确）

[使用案例]

```
Function main
  ReConnect:
  CloseNet #201
  SetNet #201 , "192.168.1.220" , 9000 , CRLF
  OpenNet #201 As Client
  WaitNet #201, 5
  If ChkNet(201) = -1 Then
    Print "视觉 TCP 通讯错误，端口已打开，但是未建立通讯"
    Wait 1
    GoTo ReConnect
  ElseIf ChkNet(201) = -2 Then
    Print "视觉 TCP 通讯错误，其他任务正在使用端口"
    Wait 1
    GoTo ReConnect
  ElseIf ChkNet(201) = -3 Then
    Print "视觉 TCP 通讯错误，未打开端口"
    Wait 1
    GoTo ReConnect
  EndIf
Fend
```

ClrNet

[功能]

用于清空 TCP/IP 通讯缓存。

[语法]

`ClrNet` #通讯端口编号

[参数说明]

通讯端口编号 指定需要修改通讯参数的端口编号，范围：201~216。

[使用案例]

```
Function main
  ReConnect:
  CloseNet #201
  SetNet #201 , "192.168.1.220" , 9000 ,CRLF
  OpenNet #201 As Client
  WaitNet #201, 5
  If ChkNet(201) = -1 Then
    Print "视觉 TCP 通讯错误, 端口已打开, 但是未建立通讯"
    Wait 1
    GoTo ReConnect
  ElseIf ChkNet(201) = -2 Then
    Print "视觉 TCP 通讯错误, 其他任务正在使用端口"
    Wait 1
    GoTo ReConnect
  ElseIf ChkNet(201) = -3 Then
    Print "视觉 TCP 通讯错误, 未打开端口"
    Wait 1
    GoTo ReConnect
  EndIf

  Print #201 , "EXW,1"
  Input #201 , CCD_X$, CCD_Y$, CCD_U$
  ClrNet #201

Fend
```

CloseNet

[功能]

用于关闭 OpenNet 打开的 TCP/IP 通讯端口。

[语法]

```
CloseNet [#通讯端口编号/All]
```

[参数说明]

#通讯端口编号 指定需要修改通讯参数的端口编号。范围：#201~216。All 为关闭所有通讯端口。

注意事项：

1. 触发程序停止时，会导致通讯关闭。

[使用案例]

```
Function main
  ReConnect:
  CloseNet #201
  SetNet #201 , "192.168.1.220" , 9000 ,CRLF
  OpenNet #201 As Client
  WaitNet #201, 5
  If ChkNet(201) = -1 Then
    Print "视觉 TCP 通讯错误, 端口已打开, 但是未建立通讯"
    Wait 1
    GoTo ReConnect
  ElseIf ChkNet(201) = -2 Then
    Print "视觉 TCP 通讯错误, 其他任务正在使用端口"
    Wait 1
    GoTo ReConnect
  ElseIf ChkNet(201) = -3 Then
    Print "视觉 TCP 通讯错误, 未打开端口"
    Wait 1
    GoTo ReConnect
  EndIf
Fend
```

SetCom

[功能]

用于设置 RS232/485 串口通讯参数。

[语法]

SetCom #通讯端口编号 {, 波特率, 数据位长度, 结束位长度, 奇偶性, 结束符, H/W 流控制, S/W 流控制, 超时时间}

[参数说明]

| | |
|----------------|--|
| #通讯端口编号 | 指定需要修改通讯参数的端口编号，范围：#1~16。 |
| 波特率 | 可指定 4800、9600、14400、19200、38400、56000、57600、115200、128000、230400、256000。省略时默认 19200。 |
| 数据位长度 | 以 7 或 8 指定每个字符的数据位长度，可省略。 |
| 结束位长度 | 以 1 或 2 指定每个字符的结束位长度，可省略。 |
| 奇偶性 | 以 E、O、N 指定通讯的校验方式。 |
| 结束符 | 通讯数据的末尾字符。可以设定 CR、LF、CRLF(回车、换行、回车换行)，可省略。 |
| H/W 流控制 | 硬件控制有效时指定 RTS，无效时指定 NONE，可省略。 |
| S/W 流控制 | 软件控制有效时指定 XON，无效时指定 NONE，可省略。 |
| 超时时间 | 以秒设定收发的最长时间。0 为永不超时，可省略。 |

[使用案例]

```
Function main
    Integer AA
    ReConnect:
    SetCom #2, 38400, 8, 2, N, CRLF, NONE, NONE, 0
    OpenCom #2
    Do
        Print #2, "OK"
        Input #2, AA
        Print AA
    Loop
Fend
```

OpenCom

[功能]

用于开启 RS232/485 串口通讯，或获取 RS232/485 串口的线程编号。

[语法]

OpenCom #通讯端口编号

OpenCom(通讯端口编号)

[参数说明]

#通讯端口编号 指定需要修改通讯参数的端口编号，范围：#1~16。

[使用案例]

```
Function main
    Integer AA
    ReConnect:
    SetCom #2, 38400, 8, 2, N, CRLF, NONE, NONE, 0
    OpenCom #2
    Do
        Print #2, "OK"
        Input #2, AA
        Print AA
    Loop
Fend
```

ChkCom

[功能]

用于返回串口通讯端口接收缓存器内的字节数。

[语法]

`ChkCom`(通讯端口编号)

[参数说明]

通讯端口编号 指定需要修改通讯参数的端口编号，范围：1~16。

注意事项：

1. 以整数返回字节数。如果数据不存在，以负值返回通讯端口状态。-2 为其他线程正在使用该通讯端口。-3 为通讯端口未开启。

[使用案例]

```
Function main
    Integer numChars
    numChars = ChkCom(1)
Fend
```

ClrCom

[功能]

用于清空通讯缓存。

[语法]

`ClrCom` #通讯端口编号

[参数说明]

#通讯端口编号 指定需要修改通讯参数的端口编号，范围：#1~16。

[使用案例]

```
Function main
    ClrCom #1
Fend
```

CloseCom

[功能]

用于关闭 OpenCom 打开的串口通讯端口。

[语法]

```
CloseCom [#通讯端口编号/All]
```

[参数说明]

#通讯端口编号 指定需要修改通讯参数的端口编号。范围：#1~16。All 为关闭所有。

[使用案例]

```
Function main
    CloseCom #1
Fend
```

Read

[功能]

用于从通讯端口读取指定的字符数，不包含结束符号。

[语法]

```
Read #通讯端口编号, 字符串变量$, 字符数
```

[参数说明]

#通讯端口编号 指定需要修改通讯参数的端口编号。TCP/IP 范围：#201~216。串口范围：#1~16。

字符串变量\$ 用于接收端口内字符的字符串变量。

字符数 读取字符的数量。

[使用案例]

```
Function main
    String strBin$
    Read #201,strBin$,4 '从#201 端口读取四个字符
Fend
```

ReadBin

[功能]

用于从 TCP/IP 通讯端口读取二进制数据。

[语法]

`ReadBin` #通讯端口编号, 变量名

`ReadBin` #通讯端口编号, 数组变量名(), 字节数

[参数说明]

#通讯端口编号 指定需要修改通讯参数的端口编号。TCP/IP 范围：#201~216。串口范围：#1~16。

变量名 用于接收端口内二进制数据的变量。

数组变量名() 用于接收端口内二进制数据的数组变量。

字节数 读取字节的数量。

[使用案例]

```
Function main
    Integer num
    ReadBin #201,num
    Integer arr(5)
    ReadBin #201,arr(),5
```

Fend

Write

[功能]

将字符串写入到通讯端口中，不包含结束符号。

[语法]

`Write` #通讯端口编号, 字符串

[参数说明]

#通讯端口编号 指定需要修改通讯参数的端口编号。TCP/IP 范围：#201~216。串口范围：#1~16。

字符串 需要写入通讯端口的字符串。

[使用案例]

```
Function main
    Write #201 , "abcd"
```

Fend

WriteBin

[功能]

将二进制数据写入通讯端口，不包含结束符号。

[语法]

`WriteBin` #通讯端口编号, 数据

`WriteBin` #通讯端口编号, 数组变量名(), 字节数

[参数说明]

| | |
|----------------|--|
| #通讯端口编号 | 指定需要修改通讯参数的端口编号。TCP/IP 范围：#201~216。串口范围：#1~16。 |
| 变量名 | 用于接收端口内二进制数据的变量。 |
| 数组变量名() | 用于接收端口内二进制数据的数组变量。 |
| 字节数 | 读取字节的数量。 |

[使用案例]

Function main

```
WriteBin #201 , 123456
```

```
Integer arr(5)
```

```
arr(0) = 22
```

```
arr(1) = 43
```

```
arr(2) = 67
```

```
arr(3) = 45
```

```
arr(4) = 33
```

```
WriteBin #201 , arr() , 5
```

Fend

Print

[功能]

将数据或字符串输出到指定通讯端口中，包含结束符号。

[语法]

`Print #`通讯端口编号, 数据 1 {, 数据 2...}

[参数说明]

#通讯端口编号 指定需要修改通讯参数的端口编号。TCP/IP 范围：#201~216。串口范围：#1~16。

数据 可用变量、数值、字符串等来输出想输出的内容。可根据需求酌情增加数据量。

[使用案例]

```
Function main
    String CCD_X$, CCD_Y$, CCD_U$
    CloseNet #201
    SetNet #201 , "192.168.1.220" , 9000 , CRLF
    OpenNet #201 As Client
    WaitNet #201, 5
    Print #201 , "EXW,1"
    Input #201 , CCD_X$, CCD_Y$, CCD_U$
    Print CCD_X$, CCD_Y$, CCD_U$
Fend
```

Input

[功能]

用于从通讯端口读取数据或字符串，包含结束符号。

[语法]

`Input #` 通讯端口编号, 变量 1 {, 变量 2...}

[参数说明]

#通讯端口编号 指定需要修改通讯参数的端口编号。TCP/IP 范围：#201~216。串口范围：#1~16。

变量 用于接收数据或字符串的变量名。可根据数据长度酌情增加变量。

注意事项：

1. `Input #`默认字符串的分割为“,”（逗号）或“ ”（空白）。其余分隔符请使用令牌字符串指令。

[使用案例]

Function main

```
String CCD_X$, CCD_Y$, CCD_U$
CloseNet #201
SetNet #201 , "192.168.1.220" , 9000 , CRLF
OpenNet #201 As Client
WaitNet #201, 5
Print #201 , "EXW,1"
Input #201 , CCD_X$, CCD_Y$, CCD_U$
Print CCD_X$, CCD_Y$, CCD_U$
```

Fend

Line Input

[功能]

用于从通讯端口读取一行数据或字符串，包含结束符号。

[语法]

`Line Input` #通讯端口编号, 变量

[参数说明]

#通讯端口编号 指定需要修改通讯参数的端口编号。TCP/IP 范围：#201~216。串口范围：#1~16。

变量 用于接收数据或字符串的变量名。

[使用案例]

Function main

```
String CCD_TxT$, strArr$(0)
CloseNet #201
SetNet #201, "192.168.1.220", 9000, CRLF
OpenNet #201 As Client
WaitNet #201, 5
Print #201, "EXW,1"
Line Input #201, CCD_TxT$
Print CCD_TxT$
```

`ParseStr CCD_TxT$, strArr$(), "$"` '以\$为分割符，将字符串的内容分割处理后存放在数组中

```
For i = 0 To UBound(strArr$())
Print "Array content = ", strArr$(i)
Next
```

Fend

系统指令详细说明

Print

[功能]

打印指令，打印字符串、数值、变量。

[语法]

`Print` 内容 1 {, 内容 2. }

[参数说明]

内容 想要打印的字符串、数值或变量。可根据需求酌情增加变量。

[使用案例]

```
Function main
  ReConnect:
  CloseNet #201
  SetNet #201 , "192.168.1.220" , 9000 ,CRLF
  OpenNet #201 As Client
  WaitNet #201, 5
  If ChkNet(201) = -1 Then
    Print "视觉 TCP 通讯错误, 端口已打开, 但是未建立通讯"
    Wait 1
    GoTo ReConnect
  ElseIf ChkNet(201) = -2 Then
    Print "视觉 TCP 通讯错误, 其他任务正在使用端口"
    Wait 1
    GoTo ReConnect
  ElseIf ChkNet(201) = -3 Then
    Print "视觉 TCP 通讯错误, 未打开端口"
    Wait 1
    GoTo ReConnect
  EndIf

  Print ChkNet(201)

Fend
```

Xqt

[功能]

启动多线程。

[语法]

`Xqt` {任务编号, } 函数名(形参 1...) {, `Normal`/`NoPause`/`NoEmgAbort`}

[参数说明]

| | |
|-------------------|--|
| 任务编号 | 以整数指定多线程的编号，可省略。 |
| 函数名 | 要执行多线程的 Function 名。如果 Function 有形参则需要填上 () 并补上形参，如没有则直接填函数名即可。 |
| Normal | 普通的多线程。三者省略不填，默认 Normal。 |
| NoPause | 发生 Pause、或有外部暂停信号、及安全门打开的状态下不会暂停的多线程。 |
| NoEmgAbort | 紧急停止、或发生错误时仍能继续处理的多线程。 |

注意事项：

1. 前台多线程，任务编号 1~16。后台多线程，任务编号 66~80。

[使用案例]

```
Function main
  Xqt A1
Fend

Function A1
  Print "Runing....."
Fend
```

Pause

[功能]

用于暂停可暂停的所有多线程。

[语法]

Pause

[参数说明]

注意事项：

1. 前台多线程，任务编号 1~16。后台多线程，任务编号 66~80。

[使用案例]

```
Function main
```

```
    Xqt A1
```

```
    Wait 1
```

```
    Pauser           ' 暂停程序
```

```
Fend
```

```
Function A1
```

```
    Print "Runing....."
```

```
Fend
```

Halt

[功能]

用于暂停指定的正在执行的多线程。

[语法]

`Halt` [函数名/任务编号]

[参数说明]

函数名 正在执行的多线程 Function 名。

任务编号 正在执行的多线程任务编号。

注意事项:

1. 前台多线程，任务编号 1~16。后台多线程，任务编号 66~80。

[使用案例]

```
Function main
  Xqt A1
  Wait 1
  Halt A1      ' 暂停 A1 线程
Fend

Function A1
  Do
    Print "Runing....."
  Loop
Fend
```

Quit

[功能]

停止所有或指定的线程。

[语法]

`Quit` [函数名/任务编号/All]

[参数说明]

函数名 正在执行的多线程 Function 名。

任务编号 正在执行的多线程任务编号。

All 停止所有线程。

注意事项：

1. 前台多线程，任务编号 1~16。后台多线程，任务编号 66~80。

[使用案例]

```
Function main
  Xqt A1
  Wait 1
  Quit A1      '停止 A1 线程
Fend
```

```
Function A1
  Do
    Print "Runing....."
  Loop
Fend
```

StartMain

[功能]

用于启用主任务。

[语法]

`StartMain` [主任务名称]

[参数说明]

主任务名称 `main - main63`。

[使用案例]

```
Function bgmain                    '后台任务
  If MemInW(0) = 1 Then
    StartMain main
    Wait MemInW(0) <> 1            '等待信号消失，防止重复触发
  EndIf
Fend
```

Resume

[功能]

用于继续运行被 Halt 命令而暂停的多线。

[语法]

`Resume` [函数名/任务编号/All]

[参数说明]

函数名 正在执行的多线程 Function 名。

任务编号 正在执行的多线程任务编号。

All 继续所有线程。

注意事项：

1. 前台多线程，任务编号 1~16。后台多线程，任务编号 66~80。

[使用案例]

```
Function main
  Xqt A1
  Wait 1
  Halt A1       ' 暂停 A1 线程
  Wait Sw(1) = 1
  Resume A1     ' 继续运行 A1 线程
Fend
```

```
Function A1
  Do
    Print "Runing....."
  Loop
Fend
```

Reset

[功能]

用于重置控制器异警状态。

[语法]

Reset [Error]

[参数说明]

Error 指所有错误将被指令清除。

注意事项：

1. 调用指令时填写 Error 则清除所有报警，若未填写 Error 则只清除急停状态。

[使用案例]

Function main

 Reset '清除急停状态

End

Function A1

 Reset Error '清除异警状态

End

MyTask

[功能]

用于返回当前线程的任务编号。

[语法]

MyTask

[参数说明]

注意事项：

1. 前台多线程，任务编号 1~16。后台多线程，任务编号 66~80。

[使用案例]

```
Function main
```

```
    Xqt 2 , A1
```

```
    Xqt 3 , A1
```

```
    Xqt 4 , A1
```

```
Fend
```

```
Function A1
```

```
    On MyTask      '将编号与当前任务编号相同的 IO 端口设为 On
```

```
    Wait 1
```

```
    Off MyTask     '将编号与当前任务编号相同的 IO 端口设为 Off
```

```
Fend
```

TaskDone

[功能]

确认线程是否结束。

[语法]

`TaskDone` (函数名/任务编号)

[参数说明]

函数名 多线程 Function 名。

任务编号 多线程任务编号。

注意事项：

1. 前台多线程，任务编号 1~16。后台多线程，任务编号 66~80。
2. 线程结束返回 True。否则返回 False。

[使用案例]

```
Function main
  Xqt A1
  Do
    Print "A1"
  Loop Until TaskDone(A1)
Fend

Function A1
  Go P1
Fend
```

TaskState

[功能]

返回线程当前的状态。

[语法]

`TaskState` (函数名/任务编号)

[参数说明]

函数名 多线程 Function 名。

任务编号 多线程任务编号。

注意事项：

1. 前台多线程，任务编号 1~16。后台多线程，任务编号 66~80。
2. 0 为未执行。4 为待机。5 为启动中。6 为运行中。7 为错误状态。8 为暂停状态。9 为恢复中状态。
10 为停止中状态。

[使用案例]

```
Function main
  If TaskState(A1) = 0 Then
    Xqt 2 , A1
  EndIf
Fend
```

```
Function A1
  Go P1
Fend
```

TaskWait

[功能]

用于等待指定线程结束。

[语法]

`TaskWait` (函数名/任务编号)

[参数说明]

函数名 多线程 Function 名。

任务编号 多线程任务编号。

注意事项：

1. 前台多线程，任务编号 1~16。后台多线程，任务编号 66~80。

[使用案例]

```
Function main
  Xqt 2 , A1
  TaskWait A1
Fend
```

```
Function A1
  Go P1
Fend
```

SyncLock

[功能]

用于使用相互排他锁定，使多个线程同步。

[语法]

`SyncLock` 信号编号 {, 超时时间}

[参数说明]

信号编号 以表达式或数值指定要接收的信号编号，范围 0~63。

超时时间 以表达式或数值指定锁定之前等待的时间，可省略。

[使用案例]

'下例所示为使用 `SyncLock` 和 `SyncUnlock` 设为 1 次只有 1 个任务将信息写入到记录文件中。

```
Function Main
    Xqt Func1
    Xqt Func2
Fend

Function Func1
    Long count
    Do
        Wait 0.5
        count = count+1
        LogMsg("Msg from Func1,"+ Str$(count))
    Loop
Fend

Function Func2
    Long count
    Do
        Wait 0.5
        count = count+1
        LogMsg("Msg from FuncmsgSCloseCom2,"+ Str$(count))
    Loop
Fend

Function LogMsg(msg$ As String)
    SyncLock 1
    Print msg$
    Wait 1
    SyncUnlock 1
Fend
```

SyncUnlock

[功能]

用于解除 SyncLock 锁定的信号编号。

[语法]

SyncUnlock 信号编号

[参数说明]

信号编号 以表达式或数值指定要接收的信号编号，范围 0~63。

[使用案例]

' 下例所示为使用 SyncLock 和 SyncUnlock 设为 1 次只有 1 个任务将信息写入到记录文件中。

```
Function Main
    Xqt Func1
    Xqt Func2
Fend

Function Func1
    Long count
    Do
        Wait 0.5
        count = count+1
        LogMsg("Msg from Func1, "+ Str$(count))
    Loop
Fend

Function Func2
    Long count
    Do
        Wait 0.5
        count = count+1
        LogMsg("Msg from FuncmsgSCloseCom2, "+ Str$(count))
    Loop
Fend

Function LogMsg(msg$ As String)
    SyncLock 1
    Print msg$
    Wait 1
    SyncUnlock 1
Fend
```

Signal

[功能]

用于向正在执行 WaitSig 命令的任务发送信号。

[语法]

Signal 信号编号

[参数说明]

信号编号 以表达式或数值指定要接收的信号编号，范围 0~63。

[使用案例]

```
Function main
  Xqt 2 , A1
  Signal 1
Fend
```

```
Function A1
  WaitSig 1
Fend
```

WaitSig

[功能]

用于等待其他任务的 Signal 命令发送的同步信号。

[语法]

`WaitSig` 信号编号 {, 超时时间}

[参数说明]

信号编号 以表达式或数值指定要接收的信号编号，范围 0~63。

超时时间 以表达式或数值指定最长等待的时间，可省略。

[使用案例]

```
Function main
```

```
    Xqt 2 , A1
```

```
    Signal 1
```

```
Fend
```

```
Function A1
```

```
    WaitSig 1
```

```
Fend
```

TW

[功能]

用于返回 Wait、WaitNet、WaitSig 指令的状态。

[语法]

TW

[参数说明]

注意事项:

1. 在指定时间内 Wait、WaitNet、WaitSig 条件成立范围 False，超时返回 True。

[使用案例]

```
Function main
    Wait Sw(0) = On , 5
    If TW = True Then
        '输入 DIO 变为 ON 状态之前待机 5 秒
        Print "DIO 并没有置为 On"
    EndIf
Fend
```

ElapsedTime

[功能]

以秒为单位返回计算节拍时间计时器开始计时之后经过的时间，不包含程序暂停时间。

[语法]

ElapsedTime

[参数说明]

注意事项：

1. 计时器计量范围 $0 \sim 1.7E+31$ 。最小单位是 0.001 秒。

[使用案例]

```
Function Main
    Integer i
    ResetElapsedTime           '重置计算节拍时间用的计时器
    For i = 1 To 10
        GoSub Cycle
        Wait 0.1
    Next
    Cycle:
    Print "次数: ", i, ElapsedTime / 10 '计算并打印节拍时间
    If i < 10 Then
        Return
    EndIf
Fend
```

ResetElapsedTime

[功能]

用于重置由 ElapsedTime 使用的计算节拍时间用的计时器。

[语法]

ResetElapsedTime

[参数说明]

[使用案例]

```
Function Main
    Integer i
    ResetElapsedTime           '重置计算节拍时间用的计时器
    For i = 1 To 10
        GoSub Cycle
        Wait 0.1
    Next
    Cycle:
    Print "次数: ", i, ElapsedTime / 10 '计算并打印节拍时间
    If i < 10 Then
        Return
    EndIf
Fend
```

Tmr

[功能]

以秒为单位返回计时器开始计时之后经过的时间，包含程序暂停时间。

[语法]

Tmr (计时器编号)

[参数说明]

计时器编号 以表达式或数值指定返回那个计时器的时间，范围 0~63。

[使用案例]

```
Function Main
    Integer i
    TmReset 0                    '重置计算节拍时间用的计时器
    For i = 1 To 10
        GoSub Cycle
        Wait 0.1
    Next
    Cycle:
    Print "次数: ", i, Tmr(0) / 10    '计算并打印节拍时间
    If i < 10 Then
        Return
    EndIf
Fend
```

TmReset

[功能]

用于重置 Tmr 计时器。

[语法]

`TmReset` 计时器编号

[参数说明]

计时器编号 以表达式或数值指定返回那个计时器的时间，范围 0~63。

[使用案例]

```
Function Main
    Integer i
    TmReset 0 '重置计算节拍时间用的计时器
    For i = 1 To 10
        GoSub Cycle
        Wait 0.1
    Next
    Cycle:
    Print "次数: ", i, Tmr(0) / 10 '计算并打印节拍时间
    If i < 10 Then
        Return
    EndIf
Fend
```

EVal

[功能]

用于执行命令窗口的语句。

[语法]

EVal 命令

[参数说明]

命令 以字符串指定要执行的命令。

返回值 返回通过执行命令返回的错误代码。 当命令正常结束时，返回 0。

[使用案例]

Function Main

Integer errCMD

String strBin\$

Do

 strBin\$ = "Motor On" '字符串赋值

 EVal(strBin\$) '执行字符串

 Print EVal("Motor On") '执行伺服上电并打印 EVal 的状态码

 errCMD = EVal("Motor On") '执行伺服上电并将状态码赋值给 ErrCMD 变量

Loop

Fend

Time

[功能]

用于打印/返回控制器系统时间。

[语法]

Time

Print Time(编号)

[参数说明]

编号 以 0 - 2 指定获取控制器系统时间 时/分/秒。

[使用案例]

Function Main

```
Time                    '打印控制器系统时间
```

```
Integer New_Time
```

```
New_Time = Time(0) '返回控制器系统时数
```

```
Print New_Time
```

```
New_Time = Time(1) '返回控制器系统分数
```

```
Print New_Time
```

```
New_Time = Time(2) '返回控制器系统秒数
```

```
Print New_Time
```

End

Time\$

[功能]

用于以指定格式 hh:mm:ss 格式返回控制器系统时间。

[语法]

```
Print Time$
```

[参数说明]

Time\$ 以时/分/秒格式返回控制器系统时间。

[使用案例]

```
Function Main
```

```
    Print Time$        '打印控制器系统时间
```

```
End
```

Date

[功能]

用于打印/返回控制器系统日期。

[语法]

Date

Print Date (编号)

[参数说明]

编号 以 0 - 2 指定获取控制器系统日期 年/月/日。

[使用案例]

Function Main

```
Date                    '打印控制器系统日期
```

```
Integer New_Date
```

```
New_Date = Date(0)    '返回控制器系统年份
```

```
Print New_Date
```

```
New_Date = Date(1)    '返回控制器系统月数
```

```
Print New_Date
```

```
New_Date = Date(2)    '返回控制器系统日期
```

```
Print New_Date
```

Fend

Date\$

[功能]

用于以指定格式 yyyy:MM:dd 格式返回控制器系统日期。

[语法]

Print Date\$

[参数说明]

Date\$ 以年/月/日格式返回控制器系统日期。

[使用案例]

Function Main

```
Print Date$            '打印控制器系统日期
```

Fend

码垛语句详细说明

Pallet

[功能]

用于定义/显示托盘。

[语法]

定义托盘:

`Pallet {OutSide,} 托盘编号, 点位 1, 点位 2, 点位 3{, 点位 4,} 个数 1, 个数 2`

打印单个托盘:

`Pallet 托盘编号`

打印所有托盘:

`Pallet`

获取托盘点位:

`Pallet (托盘编号, 托盘位置编号)`

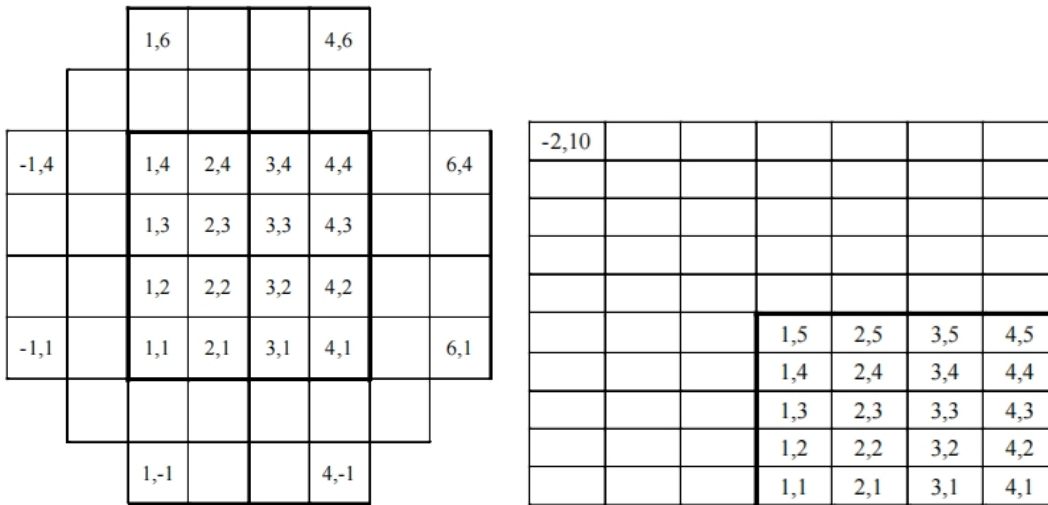
`Pallet (托盘编号, 行坐标, 列坐标)`

[参数说明]

| | |
|----------------|--------------------------------------|
| OutSide | 在指定的行和列的范围以外生成额外的托盘，可省略。 |
| 托盘编号 | 以表达式或数值指定托盘的编号，范围 0~15。 |
| 点位 1~3 | 标准 3 点法定义托盘。 |
| 点位 4 | 可使用 4 点法定义托盘进一步提升托盘点位的精度。 |
| 个数 1 | 以整数指定位点 1 与点位 2 之间有多少个点位，范围 1~32767。 |
| 个数 2 | 以整数指定位点 1 与点位 3 之间有多少个点位，范围 1~32767。 |
| 行坐标 | 以整数指定托盘的第几行。 |
| 列坐标 | 以整数指定托盘的第几列。 |

注意事项:

1. OutSide 的效果如下图：在原有托盘的基础上拓展托盘外的点位。


[使用案例]

Function main

Integer var

'定义 5*3 的托盘

Pallet 1 , P1 , P2 , P3 , 5 , 3

'打印 2 号托盘

Pallet 2

'打印所有托盘

Pallet

'for 循环遍历托盘点

For var = 1 To 15

Go Pallet(1 , var)

Next

Go Pallet(1,2,3) '移动到 1 号托盘第二行第三列的位置

Fend

PalletClr

[功能]

用于清除已设置的 Pallet。

[语法]

PalletClr 托盘编号

[参数说明]

托盘编号 以数值指定需要清除的托盘编号，范围 0~15。

[使用案例]

```
Function main
```

```
    Integer var
```

```
    '定义 5*3 的托盘
```

```
    Pallet 1 , P1 , P2 , P3 , 5 , 3
```

```
    '清除 1 号托盘
```

```
    PalletClr 1
```

```
End
```

传送带跟踪语句详细说明

SyStart

[功能]

开始传送带跟踪功能。

[语法]

`SyStart` 传送带编号

[参数说明]

传送带编号 以表达式或数值指定传动带编号，范围 1~4。

注意事项：

1. 当机器人需要开始传送带跟踪跟踪物料时使用该指令。

[使用案例]

Recrt:

```
SyStart Cvt_One '开启皮带跟踪
```

```
If SyGetPose(Cvt_One) > 700 Then
```

```
Print "物料不够加工距离"
```

```
SyEnd Cvt_One '关闭皮带跟踪
```

```
GoTo Recry
```

```
EndIf
```

```
SyStart Cvt_One        '开启皮带跟踪进行生产流程
```

```
.
```

```
.
```

```
.
```

```
.
```

```
SyEnd Cvt_One        '结束皮带跟踪
```

SyEnd

[功能]

结束传送带跟踪功能。

[语法]

SyEnd 传送带编号

[参数说明]

传送带编号 以表达式或数值指定传动带编号。范围 1~4。

注意事项：

1. 当机器人传送带跟踪完毕，不需要再跟踪时使用该指令。

[使用案例]

Recrt:

```
SyStart Cvt_One '开启皮带跟踪
```

```
If SyGetPose(Cvt_One) > 700 Then
```

```
  Print "物料不够加工距离"
```

```
  SyEnd Cvt_One '关闭皮带跟踪
```

```
  GoTo Recry
```

```
EndIf
```

```
SyStart Cvt_One        '开启皮带跟踪进行生产流程
```

```
.
```

```
.
```

```
.
```

```
.
```

```
SyEnd Cvt_One        '结束皮带跟踪
```

SyReset

[功能]

清除当前传送带跟踪目标队列，复位传送带跟踪。

[语法]

`SyReset` 传送带编号

[参数说明]

传送带编号 以表达式或数值指定传动带编号，范围 1~4。

[使用案例]

```
SyReset Cvt_One        '初始化皮带跟踪
```

```
Recrt:
```

```
    SyStart Cvt_One '开启皮带跟踪
```

```
If SyGetPose(Cvt_One) > 700 Then
```

```
    Print "物料不够加工距离"
```

```
    SyEnd Cvt_One '关闭皮带跟踪
```

```
    GoTo Recry
```

```
EndIf
```

```
SyStart Cvt_One        '开启皮带跟踪进行生产流程
```

```
.
```

```
.
```

```
.
```

```
.
```

```
SyEnd Cvt_One        '结束皮带跟踪
```

SyGetPoint

[功能]

获取指定传送带目标队列中的点位信息。

[语法]

`SyGetPoint` (传送带编号)

[参数说明]

传送带编号 以表达式或数值指定传动带编号，范围 1~4。

[使用案例]

```
SyMove SyGetPoint (Cvt_One) :X(X1) :Y(Y1) :Z(Z1) :U(RZ1) CP
```

```
SyMove SyGetPoint (Cvt_One) :X(X1) :Y(Y1) :Z(Z1) :U(RZ1)
```

```
On 0
```

```
Wait 1
```

```
Off 0
```

```
Wait 1
```

SyGetUserData

[功能]

获取指定传送带目标队列中点位的附加信息。

[语法]

`SyGetUserData` (传送带编号)

[参数说明]

传送带编号 以表达式或数值指定传动带编号，范围 1~4。

注意事项：

1. 该信息由 `SyAddVisTarget` 指令添加。

[使用案例]

```
Print SyGetUserData(1)
```

SyGetPose

[功能]

获取指定传送带目标当前在传送带上的位置，单位 mm。

[语法]

`SyGetPose` (传送带编号)

[参数说明]

传送带编号 以表达式或数值指定传动带编号，范围 1~4。

[使用案例]

Recrt:

```
SyStart Cvt_One '开启皮带跟踪
```

```
If SyGetPose(Cvt_One) > 700 Then
```

```
Print "物料不够加工距离"
```

```
SyEnd Cvt_One '关闭皮带跟踪
```

```
GoTo Recry
```

```
EndIf
```

```
SyStart Cvt_One        '开启皮带跟踪进行生产流程
```

```
.
```

```
.
```

```
.
```

```
.
```

```
SyEnd Cvt_One        '结束皮带跟踪
```

SyGetNum

[功能]

获取指定传送带的队列中目标的个数。

[语法]

`SyGetNum`(传送带编号)

[参数说明]

传送带编号 以表达式或数值指定传动带编号，范围 1~4。

[使用案例]

```
If SyGetNum(1) > 1 Then
    Print "跟踪队列中存在目标"
EndIf
```

SyGetQuePose

[功能]

获取指定传送带的队列中点位在传送带的位置。

[语法]

`SyGetQuePose` (传送带编号 , 队列编号)

[参数说明]

传送带编号 以表达式或数值指定传动带编号，范围 1~4。

队列编号 以整数 1 - 64 指定传送带跟踪队列中的编号

[使用案例]

```
If SyGetQuePose(1, 1) > 600 Then
    Print "超出跟踪范围"
    Print SyGetQuePose(1, 1) '打印目标距离位置
EndIf
```

[回传说明]

回传数据 1 距离值，单位毫米

SyMove

[功能]

传送带跟踪直线插补运动。

[语法]

`SyMove` 目标坐标 {`CP`} {`Till/Find`} {`!...!`}

[参数说明]

| | |
|------------------------|--|
| 目标坐标 | 以传送带跟踪指令获取的点数据指定目标位置。 |
| <code>CP</code> | 设置运动平滑，可省略。 |
| <code>Till/Find</code> | <code>Till</code> 表达式 = On/Off。 <code>Find</code> 表达式 = On/Off。可省略。详情请参考 <code>Till/Find</code> 的详细说明。 |
| <code>!...!</code> | 在运动过程中并行处理 I/O 等输入输出。可省略。详情请参考并行处理的详细说明。 |

[使用案例]

```
SyMove SyGetPoint(Cvt_One):X(X1) :Y(Y1) :Z(Z1) :U(RZ1) CP
```

```
SyMove SyGetPoint(Cvt_One):X(X1) :Y(Y1) :Z(Z1) :U(RZ1)
```

```
On 0
```

```
Wait 1
```

```
Off 0
```

```
Wait 1
```

SyArc

[功能]

传送带跟踪圆弧插补运动。

[语法]

SyArc 经过坐标, 目标坐标 {CP} {Till/Find} {!...!}

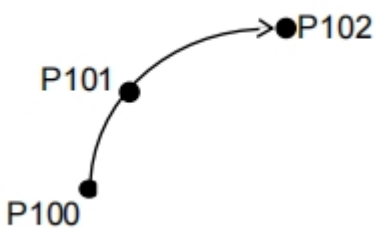
[参数说明]

| | |
|-----------|--|
| 经过坐标 | 以传送带跟踪指令获取的点数据指定圆弧会经过的位置。 |
| 目标坐标 | 以传送带跟踪指令获取的点数据指定目标位置。 |
| CP | 设置运动平滑, 可省略。 |
| Till/Find | Till 表达式 = On/Off。Find 表达式 = On/Off。可省略。详情请参考 Till/Find 的详细说明。 |
| !...! | 在运动过程中并行处理 I/O 等输入输出。可省略。详情请参考并行处理的详细说明。 |

[使用案例]

```
On 0  
Wait 1  
SyArc SyGetPoint (Cvt_One) :X(X1) :Y(Y1) :Z(Z1) :U(RZ1)
```

```
On 0  
Wait 1
```



SyTrig

[功能]

记录当前传动带的编码器脉冲值。

[语法]

SyTrig 传送带编号

[参数说明]

传送带编号 以表达式或数值指定传动带编号，范围 1~4。

注意事项：

1. 后续的 SyAddVisTarget 将会使用此指令记录的编码器脉冲值。

[使用案例]

SyTrig 1

SyPoint

[功能]

将普通坐标或点位转换为传送带跟踪坐标或点位。

[语法]

`SyPoint` (传送带编号, 坐标 X, 坐标 Y, 坐标 U)

`SyPoint` (传送带编号, 点位)

[参数说明]

传送带编号 以表达式或数值指定传动带编号，范围 1~4。

坐标 以 mm 为单位，指定需要转换为传送带跟踪的坐标。

点位 以表达式、点名称、点编号指定需要转换为传送带跟踪的坐标。

[使用案例]

`SyAddVisTarget 1 , SyPoint(1 , 10 , 20 , 30)`

`SyMove SyPoint(1 , 5 , 5 , 5)`

SyAddVisTarget

[功能]

将视觉拍照结果注册进传送带跟踪队列中。

[语法]

`SyAddVisTarget` 传送带编号, 点位[, 附加信息]

[参数说明]

- 传送带编号** 以表达式或数值指定传动带编号, 范围 1~4。
- 点位** 以表达式、点名称、点编号指定需要注册的点位。
- 附加信息** 以整数指定附加信息。

[使用案例]

Do

```
Input #TCP_Id , temp1$(0) , temp1$(1) , temp1$(2) , temp1$(3)
Print temp1$(0) , ",", temp1$(1) , ",", temp1$(2) , ",", temp1$(3)
CCD_X$ = temp1$(1)
CCD_Y$ = temp1$(2)
CCD_U$ = temp1$(3)
If Val(temp1$(0)) = 1 Then
    Print "注册点位"
    SyAddVisTarget Cvt_One , SyPoint(Cvt_One , Val(CCD_X$) , Val(CCD_Y$) , Val(CCD_U$))
EndIf
```

Loop Until ChkNet(TCP_Id) < 0

SyTimeOffset

[功能]

设置/获取跟踪时间偏移。

[语法]

`SyRejectDis` 传送带编号, 偏移时间

`SyRejectDis` (传送带编号)

[参数说明]

传送带编号 以表达式或数值指定传动带编号。范围 1~4。

偏移时间 以毫秒为单位设置跟踪偏移时间。

注意事项:

1. 请勿在跟踪过程中使用该函数。

[使用案例]

Recrt:

```
SyTimeOffset Cvt_One , 20  
SySave
```

```
SyStart Cvt_One '开启皮带跟踪
```

```
If SyGetPose(Cvt_One) > 700 Then
```

```
  Print "物料不够加工距离"
```

```
  SyEnd Cvt_One '关闭皮带跟踪
```

```
  GoTo Recry
```

```
EndIf
```

```
SyStart Cvt_One            '开启皮带跟踪进行生产流程
```

```
·
```

```
·
```

```
·
```

```
·
```

```
SyEnd Cvt_One            '结束皮带跟踪
```

SyRejectDis

[功能]

设置/获取传送带跟踪功能滤重间距。

[语法]

`SyRejectDis` 传送带编号 , 距离

`SyRejectDis` (传送带编号)

[参数说明]

传送带编号 以表达式或数值指定传动带编号。范围 1~4。

距离 以毫米为单位区分物料间距。

注意事项:

1. 请勿在跟踪过程中使用该函数。

[使用案例]

Recrt:

```
SyRejectDis Cvt_One , 20  
SySave
```

```
SyRejectDis Cvt_One
```

```
SyStart Cvt_One '开启皮带跟踪
```

```
If SyGetPose(Cvt_One) > 700 Then
```

```
Print "物料不够加工距离"
```

```
SyEnd Cvt_One '关闭皮带跟踪
```

```
GoTo Recry
```

```
EndIf
```

```
SyStart Cvt_One        '开启皮带跟踪进行生产流程
```

```
.  
. .  
. .  
. .
```

```
SyEnd Cvt_One        '结束皮带跟踪
```

SyStartDis

[功能]

设置/获取跟踪开始距离。

[语法]

`SyStartDis` 传送带编号 , 开始距离

`SyStartDis` (传送带编号)

[参数说明]

传送带编号 以表达式或数值指定传动带编号。范围 1~4。

开始距离 以毫米为单位设置相机至开始跟踪点的距离。

注意事项:

1. 请勿在跟踪过程中使用该函数。

[使用案例]

Recrt:

```
SyStartDis Cvt_One , 200
```

```
SyEndDis Cvt_One , 550
```

```
SySave
```

```
SyStart Cvt_One '开启皮带跟踪
```

```
If SyGetPose(Cvt_One) > 700 Then
```

```
Print "物料不够加工距离"
```

```
SyEnd Cvt_One '关闭皮带跟踪
```

```
GoTo Recry
```

```
EndIf
```

```
SyStart Cvt_One            '开启皮带跟踪进行生产流程
```

```
.
```

```
.
```

```
.
```

```
.
```

```
SyEnd Cvt_One            '结束皮带跟踪
```

SyEndDis

[功能]

设置/获取跟踪时间偏移。

[语法]

`SyEndDis` 传送带编号 , 结束距离

`SyEndDis` (传送带编号)

[参数说明]

传送带编号 以表达式或数值指定传动带编号。范围 1~4。

结束距离 以毫米为单位设置跟踪结束跟踪点的距离。

注意事项:

1. 请勿在跟踪过程中使用该函数。

[使用案例]

Recrt:

```
SyStartDis Cvt_One , 200
```

```
SyEndDis Cvt_One , 550
```

```
SySave
```

```
SyStart Cvt_One '开启皮带跟踪
```

```
If SyGetPose(Cvt_One) > 700 Then
```

```
Print "物料不够加工距离"
```

```
SyEnd Cvt_One '关闭皮带跟踪
```

```
GoTo Recry
```

```
EndIf
```

```
SyStart Cvt_One            '开启皮带跟踪进行生产流程
```

```
.
```

```
.
```

```
.
```

```
.
```

```
SyEnd Cvt_One            '结束皮带跟踪
```

SySave

[功能]

保存程序中对传送带跟踪功能的修改。

[语法]

SySave

[参数说明]

注意事项：

1. 所有在程序中对传送带跟踪功能的参数修改都需要使用该指令进行保存。

[使用案例]

Recrt:

```
SyRejectDis Cvt_One , 20  
SySave
```

```
SyRejectDis Cvt_One
```

```
SyStart Cvt_One '开启皮带跟踪
```

```
If SyGetPose(Cvt_One) > 700 Then  
  Print "物料不够加工距离"  
  SyEnd Cvt_One '关闭皮带跟踪  
  GoTo Recry
```

```
EndIf
```

```
SyStart Cvt_One '开启皮带跟踪进行生产流程  
.  
.  
.  
.  
SyEnd Cvt_One '结束皮带跟踪
```

数学运算指令详细说明

+、-、*、/、**、Mod、And、Or、Xor、Not、>、>=、<、<=、<>、=

[功能]

加、减、乘、除、乘方、取模、与、或、异或、非、大于、大于等于、小于、小于等于、不等于、等于。

[语法]

| 运算符 | 格式示例 | 说明 |
|-----|---------|----------|
| + | A+B | 加法 |
| - | A-B | 减法 |
| * | A*B | 乘法 |
| / | A/B | 除法 |
| ** | A**B | 乘方 |
| = | A=B | A 等于 B |
| > | A>B | A 大于 B |
| < | A<B | A 小于 B |
| >= | A>=B | A 大于等于 B |
| <= | A<=B | A 小于等于 B |
| <> | A<>B | A 不等于 B |
| And | A And B | 逻辑与 |
| Mod | A Mod B | 整数的取模 |
| Not | Not A | 非 |
| Or | A Or B | 逻辑或 |
| Xor | A Xor B | 逻辑异或 |

[参数说明]

[使用案例]

DegToRad

[功能]

角度转弧度。

[语法]

`DegToRad` (角度)

[参数说明]

角度 以数值指定需要转换为弧度的角度，单位 deg。

[使用案例]

```
Function main
```

```
    Double rad_num , cos_num
```

```
    rad_num = DegToRad(30)
```

```
    cos_num = Cos(rad_num)
```

```
Fend
```

RadToDeg

[功能]

弧度转角度。

[语法]

RadToDeg (弧度)

[参数说明]

弧度 以数值指定需要转换为角度的弧度。

[使用案例]

```
Function main
```

```
    Double deg_num
```

```
    deg_num = RadToDeg(1.21)
```

```
End
```

Sin、Cos、Tan、Asin、Acos、Atan、Atan2

[功能]

正弦函数、余弦函数、正切函数、反正弦函数、反余弦函数、反正切函数、反正切函数 2。

[语法]

Sin(弧度)

Cos(弧度)

Tan(弧度)

Asin(数值)

Acos(数值)

Atan(数值)

Atan2(y, x)

[参数说明]

弧度 以整数或浮点数指定进行运算的弧度。

数值 以整数或浮点数指定进行运算的数值。

[使用案例]

Function main

Double num

num = Sin(DegToRad(30))

num = Cos(DegToRad(30))

num = Tan(DegToRad(30))

num = Asin(0.5)

num = Acos(0.5)

num = Atan(0.5)

num = Atan2(1, 2)

Fend

HexToFloat

[功能]

十六进制转单精度浮点数，符合 IEEE754。

[语法]

HexToFloat(数值)

[参数说明]

数值 可以十进制整数或十六进制数(&H 数值)指定。

[使用案例]

Function main

```
Print HexToFloat(&H40490FDA) '输出 3.1415926
```

End

FloatToHex

[功能]

单精度浮点数转十六进制数，符合 IEEE754。

[语法]

FloatToHex(数值)

[参数说明]

数值 可以十进制整数或单精度浮点数指定。

[使用案例]

Function main

```
Print Hex$(FloatToHex(3.1415926)) '输出 40490FDA
```

End

Abs

[功能]

取绝对值。

[语法]

Abs(数值)

[参数说明]

数值 可以十进制整数或浮点数指定。

[使用案例]

```
Function main
    Abs(-1.234)
Fend
```

Sqr

[功能]

取平方根。

[语法]

Sqr(数值)

[参数说明]

数值 可以十进制整数或浮点数指定。

[使用案例]

```
Function main
    Sqr(4)
Fend
```

Sgn

[功能]

返回数值的符号。

[语法]

Sgn(数值)

[参数说明]

数值 可以十进制整数或浮点数指定。

[使用案例]

```
Function main
```

```
    Sgn(4)
```

```
End
```

LShift

[功能]

左移。

[语法]

LShift(数值, 位数)

[参数说明]

数值 可以十进制整数或浮点数指定。

位数 以整数指定左移几位。

[使用案例]

```
Function main
```

```
    LShift(4433 , 3)
```

```
End
```

RShift

[功能]

右移。

[语法]

`RShift` (数值, 位数)

[参数说明]

数值 可以十进制整数或浮点数指定。

位数 以整数指定左移几位。

[使用案例]

```
Function main
```

```
    RShift(4433 , 3)
```

```
Fend
```

BClr

[功能]

将数值指定的 Bit 置 0 并返回该数值。

[语法]

`BClr` (数值, 位数)

[参数说明]

数值 可以十进制整数或浮点数指定。

位数 以整数指定第几位。

[使用案例]

```
Function main
```

```
    BClr(123 , 2)
```

```
Fend
```

BSet

[功能]

将数值指定的 Bit 置 1 并返回该数值。

[语法]

`BSet(数值, 位数)`

[参数说明]

数值 可以十进制整数或浮点数指定。

位数 以整数指定第几位。

[使用案例]

```
Function main
    BSet(123 , 2)
Fend
```

BTst

[功能]

返回数值指定 Bit 的值。

[语法]

`BTst(数值, 位数)`

[参数说明]

数值 可以十进制整数或浮点数指定。

位数 以整数指定第几位。

[使用案例]

```
Function main
    BTst(123 , 2)
Fend
```

Ubound

[功能]

返回数组长度。

[语法]

`Ubound` (数组名 [, 维度])

[参数说明]

数组名 以十进制整数或浮点数指定。

维度 1 为一维。2 为二维。3 为三维。可省略，默认一维。

[使用案例]

```
Function main
```

```
    Integer i , a(10)
```

```
    For i = 0 To UBound(a)
```

```
        a(i) = i
```

```
    Next
```

```
End
```

Dist

[功能]

返回两点之间的直线距离。

[语法]

`Dist`(点位 1 , 点位 2)

[参数说明]

点位 1, 点位 2 ... 以点位表编号指定需要比较的点数据。

[使用案例]

`Function` main

```
    Print Dist(P1 , P2)        '检查 P1 点至 P2 的直线距离
```

`Fend`

字符串运算指令详细说明

ParseStr

[功能]

字符串分割。

[语法]

`ParseStr` 字符串, 字符串数组\$(), 分割符\$

字符串数组长度 = `ParseStr`(字符串, 字符串数组\$(), 分割符\$)

[参数说明]

| | |
|-----------|-----------------|
| 字符串 | 需要用于分割的字符串。 |
| 字符串数组\$() | 用于接收被分隔符分割的字符串。 |
| 分割符\$ | 字符串根据该分隔符进行分割。 |
| 字符串数组长度 | 字符串分割后得到的数组的长度。 |

[使用案例]

```
Function main
    String myStr$ , strArr$(0)
    Integer i
    myStr$ = "1$2$3$4"
    ParseStr myStr$ , strArr$() , "$"
    For i = 0 To UBound(strArr$())
        Print "数组内容 = " , strArr$(i)
    Next
End
```

+

[功能]

字符串拼接。

[语法]

字符串 + 字符串

[参数说明]

字符串 需要用于拼接的字符串。

[使用案例]

```
Function main
    String myStr$
    myStr$ = "Robot" + "OS"
End
```

<>, =

[功能]

不等于、等于、赋值。

[语法]

字符串 <> 字符串

字符串 = 字符串

字符串变量 = 字符串

[参数说明]

字符串 需要用于运算的字符串。

[使用案例]

```
Function main
    String myStr$
    myStr$ = "stop"
    IF myStr$ = "Run" Then
        Print "Run"
    ElseIf myStr$ <> "Run" Then
        Print "stop"
    EndIF
End
```

Val

[功能]

字符串转数值。

[语法]

Val (字符串)

[参数说明]

字符串 需要用于转换的字符串。

[使用案例]

```
Function main
    String myStr$
    myStr$ = "3.1415926"

    Double num
    num = Val(myStr$)
Fend
```

Str\$

[功能]

数值转字符串。

[语法]

Str\$(数值)

[参数说明]

数值 需要用于转换的数值。

[使用案例]

```
Function main
    String myStr$
    Double num
    num = Val(myStr$)
    myStr$ = Str$(num)
Fend
```

Len

[功能]

获取字符串长度。

[语法]

Len(字符串)

[参数说明]

字符串 需要用于转换的字符串。

[使用案例]

```
Function main
    String myStr$
    Print Len(myStr$)
End
```

Asc

[功能]

字符串转 ASCII 码。

[语法]

Asc(字符串)

[参数说明]

字符串 需要用于转换的字符串。

[使用案例]

```
Function Main
    Integer AA1 , BB1 , CC1
    AA1 = Asc("a")
    BB1 = Asc("b")
    CC1 = Asc("c")
    Print "The ASCII Value of AA1 is", AA1
    Print "The ASCII Value of BB1 is", BB1
    Print "The ASCII Value of CC1 is", CC1
End
```

Chr\$

[功能]

ASCII 码转字符串。

[语法]

Chr\$(ASCII 码)

[参数说明]

ASCII 码 1~255 指定 ASCII 码。

[使用案例]

```
Function Main
    String Test$
    Test$ = Chr$(&H41) + Chr$(&H42) + Chr$(&H43)
    Print "The value of Test= " , Test$
Fend
```

Left\$

[功能]

从字符串左侧开始提取指定的字符串。

[语法]

Left\$(字符串, 数量)

[参数说明]

字符串 需要被提取的字符串。

数量 提取数量。

[使用案例]

```
Function Main
    Print Left$("ABCDEFG", 2)
    '>>> AB
    Print Left$("ABCDEFG", 3)
    '>>> ABC
Fend
```

Mid\$

[功能]

从字符串指定范围开始提取指定的字符串。

[语法]

Mid\$(字符串, 起始位置[, 数量])

[参数说明]

| | |
|------|-----------------------|
| 字符串 | 需要被提取的字符串。 |
| 起始位置 | 提取字符串的起始位置。 |
| 数量 | 提取数量。省略则从起始位置开始提取到结束。 |

[使用案例]

```
Function main
    Print Mid$("123456" , 3 , 3)
Fend
```

Right\$

[功能]

从字符串右侧开始提取指定的字符串。

[语法]

Right\$(字符串, 数量)

[参数说明]

| | |
|-----|------------|
| 字符串 | 需要被提取的字符串。 |
| 数量 | 提取数量。 |

[使用案例]

```
Function Main
    Print Right$("ABCDEFG" , 2)
'>>> FG
    Print Right$("ABC" , 3)
'>>> ABC
Fend
```

LSet\$

[功能]

从字符串左侧开始提取指定长度的字符串，字符串长度不足时补充空格。

[语法]

`LSet$(字符串, 数量)`

[参数说明]

字符串 需要被提取的字符串。

数量 提取数量。

[使用案例]

```
Function Main
    String temp$
    temp$ = "123"
    temp$ = LSet$(temp$, 10)      'temp$ = "123"
End
```

RSet\$

[功能]

从字符串右侧开始提取指定长度的字符串，字符串长度不足时补充空格。

[语法]

`RSet$(字符串, 数量)`

[参数说明]

字符串 需要被提取的字符串。

数量 提取数量。

[使用案例]

```
Function Main
    String temp$
    temp$ = "123"
    temp$ = RSet$(temp$, 10)     'temp$ = "123"
End
```

Space\$

[功能]

返回指定数量的空格字符串。

[语法]

Space\$(数量)

[参数说明]

数量 空格数量。

[使用案例]

```
Function Main
    Print "XYZ" + Space$(1) + "ABC"
    ' >>> XYZ ABC
    Print Space$(3) + "ABC"
    ' >>> ABC
Fend
```

LCase\$

[功能]

返回小写字母的字符串。

[语法]

LCase\$(字符串)

[参数说明]

字符串 需要用于转换的字符串。

[使用案例]

```
Function Main
    String Test$
    Test$ = "Data"
    Test$ = LCase$(Test$)           ' Test$ = "data"
Fend
```

UCase\$

[功能]

返回大写字母的字符串。

[语法]

UCase\$(字符串)

[参数说明]

字符串 需要用于转换的字符串。

[使用案例]

```
Function Main
    String Test$
    Test$ = "Data"
    Test$ = UCase$(Test$)           'Test$ = "Data"
End
```

LTrim\$

[功能]

用于删除字符串左侧空格并返回。

[语法]

LTrim\$(字符串)

[参数说明]

字符串 需要用于转换的字符串。

[使用案例]

```
Function Main
    String Test$
    Test$ = " Data "
    Test$ = LTrim$(Test$)           'Test$ = "Data  "'
End
```

RTrim\$

[功能]

用于删除字符串右侧空格并返回。

[语法]

`RTrim$(字符串)`

[参数说明]

字符串 需要用于转换的字符串。

[使用案例]

```
Function Main
    String Test$
    Test$ = "  Data  "
    Test$ = RTrim$(Test$)           'Test$ = "  Data"
End
```

Trim\$

[功能]

用于删除字符串两侧空格并返回。

[语法]

`Trim$(字符串)`

[参数说明]

字符串 需要用于转换的字符串。

[使用案例]

```
Function Main
    String Test$
    Test$ = "  Data  "
    Test$ = Trim$(Test$)           'Test$ = "Data"
End
```

InStr

[功能]

从字符串中检索指定字符的位置并返回。

[语法]

`InStr`(字符串, 检索字符串)

[参数说明]

字符串 需要用于检索的字符串。

检索字符串 需要检索的字符串。

[使用案例]

```
Function main
    Print InStr("ABCDEF", "C")
Fend
```

Tab\$

[功能]

用于返回指定数量制表符的字符串。

[语法]

`Tab$(数量)`

[参数说明]

数量 指定制表符数量。

[使用案例]

```
Function Main
    Integer i
    Print "X", Tab$(1), "Y"
    For i = 1 To 10
        Print x(i), Tab$(1), y(i)
    Next i
Fend
```

Hex\$

[功能]

用于将 16 进制数转成字符串。

[语法]

Hex\$(数值)

[参数说明]

数值 需要转成字符串的数值。

[使用案例]

```
Function Main
    Integer stat
    stat = 10
    Print Hex$(stat)
'>>>A
    Print Hex$(255)
'>>>FF
Fend
```

点位运算指令详细说明

+、-

[功能]

相对坐标增量运动。

[语法]

运动指令 点位 + 分量

运动指令 点位 - 分量

[参数说明]

分量 方向分量。

[使用案例]

Function main

 Go P1 + X(10)

 Move P2 - U(5)

Fend

[功能]

修改手系、修改用户坐标。

[语法]

运动指令 /用户坐标系编号

运动指令 /[R/L]

[参数说明]

用户坐标系编号 指定切换的用户坐标系编号，范围 1~64。

R/L 指定切换的手系，R 右手，L 左手。

[使用案例]

Function main

```
Go P1 + X(10) /R      '以右手系移动到 P1 点 X+5mm 处
Go P1 + X(10) /3      '以用户坐标系 3 移动到 P1 点 X+5mm 处
```

Fend

[功能]

绝对坐标运动。

[语法]

运动指令 点位 : 分量

[参数说明]

分量 方向分量。

[使用案例]

Function main

```
Go P1 :X(10)      '移动到 P1 点且 X=10mm 处
```

Fend

=

[功能]

赋值。

[语法]

点位 = 指令

[参数说明]

[使用案例]

```
Function main
    P1 = XY(10 , 20 , 30 ,40)
Fend
```

@

[功能]

转换到指定用户坐标系。

[语法]

@用户坐标系编号

[参数说明]

用户坐标系编号 指定切换的用户坐标系编号，范围 1~64。

[使用案例]

```
Function main
    GO XY(10 , 20 , 30 ,40) @2
Fend
```

X、Y、Z、U、V、W

[功能]

方向分量。

[语法]

+/-/:方向分量

[参数说明]

[使用案例]

Function main

Go P1 + X(10)

Move P2 - U(5)

Go P1 +X(5) /R

Go P1 +X(5) /3

Go P1 :X(10)

'以右手系移动到 P1 点 X+5mm 处

'以用户坐标系 3 移动到 P1 点 X+5mm 处

'移动到 P1 点且 X=10mm 处

Fend

RX、RY、RZ

[功能]

指定用户坐标系 X、Y、Z 轴的旋转分量。

[语法]

+/-[RX/RZ]

[参数说明]

[使用案例]

Function main

Go P1 + X(10) + RX(10)

'P1 点的用户坐标绕 X 轴旋转 10°，并移动到 P1 点 X+5mm 处

Fend

TLX、TLY、TLZ、TLU、TLV、TLW

[功能]

指定工具坐标系 X、Y、Z、U、V、W 轴的旋转分量。

[语法]

`+/-[TLX/TLY/TLZ/TLU/TLV/TLW]`

[参数说明]

[使用案例]

Function main

```
Go P1 + X(10) + TLX(10)
```

'P1 点的用户坐标绕 X 轴旋转 10°，并移动到 P1 点 X+5mm 处

Fend

SavePoints

[功能]

将当前程序内存中的点位数据保存至点位文件中。文件不存在则会新建文件。

[语法]

`SavePoints` 文件名

[参数说明]

文件名 以字符串命名点位文件名。

[使用案例]

Function main

```
P1 = XY(10 , 20 , 30 , 40)
```

```
P2 = XY(40 , 30 , 20 , 10)
```

```
SavePoints "ABCD" '将 P1、P2 保存至 ABCD 点位文件中
```

Fend

LoadPoints

[功能]

加载点位文件。

[语法]

`LoadPoints` 文件名 {, Merge}

[参数说明]

文件名 以字符串命名点位文件名。

Merge 用于读入新的点之前，不想清除当前的点时进行设置。如果进行设置，则将新的点添加到设置的点中。文件中已存在要添加的点时，执行覆写，可省略。

[使用案例]

`Function` main

 ' 将通用的点读入到当前的机器人中

`LoadPoints` "R1Common.pts"

 ' 合并部件模型 1 的点

`LoadPoints` "R1Model.pts" , Merge

 ' 读入机器人 2 的点文件

`LoadPoints` "R2Model.pts"

`End`

ClearPoints

[功能]

用于清除程序内存储存的点位数据。

[语法]

`ClearPoints`

[参数说明]

[使用案例]

`Function main`

```
P1 = XY(10 , 20 , 30 , 40)
```

```
P2 = XY(40 , 30 , 20 , 10)
```

```
ClearPoints      '前面定义的 P1、P2 点将不复存在
```

`Fend`

修订记录

| 修订版本 | 日期 | 记录员 | 内容 |
|--------|------------|-----------|--|
| V0.0.1 | 2026.01.22 | 练思海 - LSH | --初定版本 1. OS 指令一览 2. 通用语句详细说明 3. 数据类型详细说明 4. 运动指令详细说明 5. 独立轴指令详细说明 6. 点位、IO、编码器备注指令说明 7. IO 指令详细说明 8. 寄存器读写指令详细说明 9. 通讯指令详细说明 10. 系统指令详细说明 11. 码垛语句详细说明 12. 传送带跟踪语句详细说明 13. 数学运算指令详细说明 14. 字符串运算指令详细说明 15. 点位运算详细说明 |
| V0.0.2 | 待修订 | - | |