驱控一体机

AR 语言篇

日期: 2017 年 06 月

版本: V2.0 (中文版)

目录

1. AR 语言总览

指令说明	指令类别	指令符号
算术运算符	+	加法运算
	-	减法运算
	*	乘法运算
	/	除法运算
	//	取整除法，即得到不大于结果的最大整数值
	%	取余除法
	^	指数运算
	-	取负运算
	~	异或运算
	&	与运算
	|	或运算
	~	取反运算
	<<	左移运算
	>>	右移运算
关系运算符	==	等于
	~=	不等于
	<=	小于等于
	>=	大于等于
	<	小于
	>	大于
逻辑运算符	or	逻辑或
	not	逻辑非
	and	逻辑与
	false	假 （注： nil 也为 false）
	true	真
一般符号	#	求 table 数组长度
	=	赋值操作
	--	单行注释
	--[[	多行注释开始行
	--]]	多行注释结束行
	( )	用于函数定义、调用，表达式运算
	{ }	用于定义 table 数组
	[ ]	table 数组元素操作符
	::	定义 goto 指令的跳转位置标签
	;	语句结束符，可以省略不写
	,	用于函数参数定义、调用，多变量赋值，table 数组定义等
	.	用于 table 数组元素访问
	..	字符串连接符
	…	定义函数可变参数项

过程控制指令	if then else elseif end	if 条件分支指令
	while do end	while 循环控制指令
	for do end	for 循环控制指令
	repeat until	repeat 循环控制指令
	goto	goto 无条件跳转指令
	function end	用户函数定义指令
一般关键字	break	跳出 for、 while、 repeat 循环
	local	定义局部变量
	nil	变量为空值
	return	函数调用返回
机器人轴定义	AX	笛卡尔坐标系 X 轴号
	AY	笛卡尔坐标系 Y 轴号
	AZ	笛卡尔坐标系 Z 轴号
	AC	笛卡尔坐标系 C 轴号
	J1	J1 关节
	J2	J2 关节
	J3	J3 关节
	J4	J4 关节
机器人全局变量	ON	打开状态
	OFF	关闭状态
	p0~p999	机器人默认点名称
过程控制指令	if then else elseif end	if 条件分支指令
	while do end	while 循环控制指令
	for do end	for 循环控制指令
	repeat until	repeat 循环控制指令
	goto	goto 无条件跳转指令
	function end	用户函数定义指令
运动指令	MovL	直线方式运行到笛卡尔坐标系绝对位置的指令
	MovLR	直线方式运行到笛卡尔坐标系相对位置的指令
	MovP	点到点方式运行到笛卡尔坐标系绝对位置的指令
	MovPR	点到点方式运行到笛卡尔坐标系相对位置的指令
	MovJ	控制各个关节移动到目标绝对角度的指令
	MArchP	点到点方式控制机器人进行拱形移动的指令
	MArc	从当前位置圆弧插补到笛卡尔坐标系绝对位置的指令
	MCircle	笛卡尔坐标系下的整圆插补指令
	linerun	自动运行状态下实现手动笛卡尔的运动
	stoprun	自动运行状态下实现停止手动直线运动
运动指令参数	AccJ	设置加速度比例指令，影响 MovJ、 MovJR、 MovP、 MovPR、 MArchP 指令的加速时间
	DecJ	设置减速度比例指令，影响 MovJ、 MovJR、 MovP、 MovPR、 MArchP 指令的减速时间
	SpdJ	设置速度比例指令，影响 MovJ、 MovJR、 MovP、 MovPR、 MArchP 指令的运行速度
	AccL	设置直线运动指令加速度，影响 MovL、 MovLR、 MArchL、 Marc、 MCircle 指令的加速时间。单位 mm/s^2

	DecL	设置直线运动指令减速度，影响 MovL、MovLR、MArchL、Marc、MCircle 指令的减速时间。单位 mm/s^2
	SpdL	设置直线运动指令速度，影响 MovL、 MovLR、MArchL、Marc、 MCircle 指令的运行速度。单位 mm/s
程序管理指令	Delay	延时指令。单位：毫秒
	Exit	程序运行终止
	Pause	暂停程序运行
一般指令	X	创建指定 X 轴笛卡尔坐标绝对位置点的指令
	Y	创建指定 Y 轴笛卡尔坐标绝对位置点的指令
	Z	创建指定 Z 轴笛卡尔坐标绝对位置点的指令
	C	创建指定 C 轴笛卡尔坐标绝对位置点的指令
	XYZC	创建指定 XYZC 轴笛卡尔坐标绝对位置点的指令
输入/输出指令	DI	读取输入端口状态
	DO	读写输出端口状态
	WDI	读取输入端口状态,直到等待某一信号有效，则继续执行后续的程序
	WDO	读取输出端口状态,直到等待某一信号有效，则继续执行后续的程序
坐标系指令	SetU	设置机器人当前用户坐标系
	SetT	设置机器人当前工具坐标系
	WrU	修改用户坐标系的数据
	WrT	修改工具坐标系的数据
	CacU	新建用户坐标系
	V2Tool	根据视觉坐标计算工具坐标
	getcart	获取当前笛卡尔坐标以及关节坐标
码垛指令	SetPlt	设置码垛数指令
	GetPlt	取码垛数据点指令
	ToPutPLT	放码垛
	FromPutPLT	放码垛后，离码垛
	ResetPLT	重置码垛个数
伺服管理指令	MotOn	伺服所有轴上电使能
	MotOff	关闭伺服所有轴使能
通信指令	RecCom	从 RS232 串口接收数据
	ClrCom	清除 RS232 串口接收缓冲区
	sysnetclr	清除网络数据
	sysnetget	查询方式读取网络数据
	sysnetsend	发送网络数据
	sysnetcatch	阻塞式读取网络数据
	CloseNet	关闭 TCP 网络连接
	OpenNet	创建 TCP 网络
	ConnectNet	连接 TCP 网络
	RecvNet	网络接收数据功能函数
	WriteNet	网络发送数据功能函数
	publicread	读取全局表数据值

	publicwrite	写入数据值到全局表
视觉指令	VisToRob	将像素坐标转化成指定的用户坐标系下的坐标
	CCDrecv	接收视觉返回的数据
	CCDtrigger	触发相机拍照
	CCDsent	发送字符串到视觉
	CCDclr	清除网络托管的 IP
	CCDoffset	视觉偏差补偿
	GetDynCCDPos	动态视觉位置计算
动态跟随指令	FollowInit	动态跟随初始化
	SetDynCatch	开启或关闭跟随抓取任务
	GetCatchSpace	获取物件是否进入抓取区域
	SetCatch	执行跟随
	GetCatchState	获取抓取状态
	SynOver	结束跟随
	GetTrigger	获取触发状态（同一物体拍两次使用）
	SetViewData	发送数据给控制器，存入缓存待跟随
调试指令	print	打印输出用户调试数据
	Error	终止程序运行，并输出错误信息
点位指令	Point	调用点位表中的点位
系统指令	syswork	设置系统工作状态
	sysstate	读取系统状态
	sysrate	设置全局速度倍率
	systime	获取系统时钟
Modbus 主站读写指令	ReadRegW	读指定 PLC 寄存器的 16 位字地址
	ReadRegDW	读指定 PLC 寄存器的 32 位双字地址
	WriteRegW	写入 16 位字地址到指定的 PLC 寄存器
	WriteRegDW	写入 32 位字地址到指定的 PLC 寄存器

◆ 注意： AR 语言是大小写敏感的语言，用户一定按照规定的操作符使用，表中的所有指令符号用户只能根据说明使用，不能重新定义。 例如:

```
local X --定义变量 X
X=10 --赋值给变量 X
```

X 已经定义为系统指令，用户重新定义可能会运行出错。

1.1 算术运算符

指令符号	指令说明
+	加法运算
-	减法运算
*	乘法运算
/	除法运算
//	取整除法， 即得到不大于结果的最大整数值
%	取余除法
^	指数运算
-	取负运算
~	异或运算
&	与运算
|	或运算
~	取反运算
<<	左移运算
>>	右移运算

算术运算符提供了各种实数的运算功能，整形数据的位运算功能等。“+”加法运算还可以实现两个点位数据的加法运算。举例说明:

```
local point1={X=10,Y=20,Z=30,C=10,H=0 }
local point2={X=10,Y=20,Z=30,C=10,H=1 }
local point3 =point1+point2
```

运行该程序后， point3 点的数据中 x 为 20， y 为 40， z 为 60， c 为 20， h 为点位的手系，相加后以 point1 点手系为准。

1.2 关系运算符

指令符号	指令说明
==	等于
~=	不等于
<=	小于等于
>=	大于等于
<	小于
>	大于

关系运算符用在流程控制语句的条件判断中。举例说明:

```
local a =10
local b=20
if a == b then
end
if a < b then
end
if a < 30 then
end
```

== 等号可以比较字符串类型变量。

```
local a = "ROBOT"
if a == "ROBOT" then
...
end
```

1.3 逻辑运算符

指令符号	指令说明
or	逻辑或
not	逻辑非
and	逻辑与
false	假 （注： nil 也为假）
true	真

逻辑运算符用在流程控制语句的条件判断中。举例说明:

```
if 5 or 6 then
...
end
local a =30
local b=true
if a and b then
...
end
local a=0
local b=1
if a and b then   ----第 11 行
...
end
local a=0
local b=nil
if a and b then
...
end
```

注意: AR 语言中,只有 nil 和 false 值为假,其它值都为真,包括 0 也为真。上面示例第 11 行的条件是成立的。

1.4 一般符号

指令符号	指令说明
#	求 table 数组长度
=	赋值操作
--	单行注释
--[[	多行注释开始行
--]]	多行注释结束行

( )	用于函数定义、调用，表达式运算
{ }	用于定义 table 数组
[ ]	table 数组元素操作符
::	定义 goto 指令的跳转位置标签
;	语句结束符，可以省略不写
,	用于函数参数定义、调用，多变量赋值， table 数组定义等
.	用于 table 数组元素访问
..	字符串连接符
…	定义函数可变参数项

举例说明:

```
local a,b,c = 1,2,3      --用,符号写多变量赋值语句
p.x                      --用.符号访问数组元素
local a={10,20,30}       --用{}符号定义数组
a[1]                     --用[]符号访问数组元素
::lab::                  --用::符号定义 goto 指令跳转位置标签
```

注意： table 类型数组变量下标是从 1 开始，如 a[1]的值为 10。

1.5 一般关键字

指令符号	指令说明
break	跳出 for、 while、 repeat 循环
local	定义局部变量
nil	变量为空值
return	函数调用返回

local 符号用来声明用户的变量举例说明:

```
local a                      --声明局部变量 a ，值为 nil
local b={10,20}              --声明 table 数组
local b={{10,20},{30,40}}    --声明 table 二维数组
local a="ROBOT"              --声明字符串变量 a
```

1.6 机器人轴定义

指令符号	指令说明
AX	笛卡尔坐标系 X 轴号
AY	笛卡尔坐标系 Y 轴号
AZ	笛卡尔坐标系 Z 轴号
AC	笛卡尔坐标系 C 轴号
J1	J1 关节
J2	J2 关节
J3	J3 关节
J4	J4 关节

轴号是全局变量，作为运动指令的一些参数使用，用户也不能重新定义。

1.7 机器人全局变量

指令符号	指令说明
ON	打开状态
OFF	关闭状态
p0~p999	机器人默认点名称

全局变量为系统定义好的变量，有具体的意义，用户不能在重新定义或赋值，否则可能运行出错。点位变量 p0~p999 为 table 类型的点变量，机器人启动时定义如下:

```
local p={x=VALUE1,y=VALUE2,z=VALUE3,c=VALUE4,h=VALUE5}
```

程序中， 用户可以以 p.x p.y 的方式访问点的数据值

```
local a = p1          -- a 为 p1 的引用
a.x = 10              --那么 p1.x 的值也变为 10
local a = #p1         --拷贝 p1 的值到 a
a.x= 10               --p1.x 保持原来的值
```

1.8 过程控制指令

指令符号	指令说明
if then else elseif end	if 条件分支指令
while do end	while 循环控制指令
for do end	for 循环控制指令
repeat until	repeat 循环控制指令
goto	goto 无条件跳转指令
function end	用户函数定义指令

1.8.1 if 条件分支指令

使用说明 if then else elseif end

语法说明:

Case1	Case2	Case3

if conditions then
then-part
end
	if conditions then
	then-part
	else
	else-part
	end
		if conditions then
		then-part
		elseif conditions then
		elseif-part
		else
		else-part
		end

Conditions 表示控制语句的条件， 如果为 true 则条件成立， part 为执行的代码部分。Conditions 条件可以是常量，变量，表达式，函数调用等，程序只判断最终结果为 false 或 true 去选择执行程序。

1.8.2 while 循环指令

使用说明 while do end

语法说明:

```
while condition do
statements
end
```

conditions 表示控制条件，如果为 true 则条件成立， statements 为执行的代码部分，如果为 false 则条件不成立，不执行 statements 代码部分。

```
a = 0
while a<10 do          --条件判断，如果为 true，则继续执行 a = a-1
a = a-1
end
```

1.8.3 for 循环指令

使用说明 for do end

语法说明:

```
for var=exp1,exp2,exp3 do
loop-part
end
```

for 将用 exp3 作为 step（步进值）从 exp1（初始值）到 exp2（终止值），执行 loop-part（循环体）。其中 exp3 可以省略，默认 step=1 exp 是一个表达式，可以是数值常量，变量，或函数调用的返回值，表达式运算。

有几点需要注意:

1. 三个表达式只会被计算一次，并且是在循环开始前。

```
for i=1,f(x) do                    --调用 f(x)函数一次，函数返回值作为循环的终止值
end of the cycle
print(i)
end
for i=10,1,-1 do
print(i)
end
```

2.控制变量 var 是局部变量自动被声明,并且只在循环内有效.

```
for i=1,10 do
print(i)
end
max = i                            --错误引用 i， i 是局部变量
```

如果需要保留控制变量的值，需要在循环中将其保存

```
local found = nil
```

```
for i=1,a.n do
    if a[i] == value then
        found = i
        break
    end
end
print(found)
```

3.循环过程中不要改变控制变量的值，那样做的结果是不可预知的。如果要退出循环，使用 break 语句。

1.8.4 repeat until 循环指令

使用说明 repeat-until
语法说明:

```
repeat
    statements
until conditions
```

while 语句大致相同，只是循环部分 statements 的执行先后不一样，repear-until 的循环体先执行在判断循环条件，而 while 语句是先判断循环条件。

1.8.5 goto 无条件跳转指令

使用说明 goto
语法说明 goto lab_name
lab_name 用户定义的文件行跳转位置名称， 字符串类型， 如果没有定义会出错。

```
local a=0
::lab::                         --定义跳转位置名称 lab
MovL(p1)
a = a + 1
print("跳转次数:",a)
goto lab                        --跳转到示例 2 循环执行
```

注意： goto 指令不能从一个函数跳转到另外一个函数,lab_name 跳转名称不能重复定义。

1.8.6 函数指令

使用说明 function end
语法说明 function func_name (arguments-list)

```
Function func_name(arguments-list)
statements-list
end
```

name 是函数名字，根据 AR 语言命名规则进行定义， 且函数名不能重复，否则会出错。
arguments-list 为函数的参数列表，列表中的参数可以是任何数据类型的变量，当有多个参数时，通过“,”分隔。函数还可以没有参数，在定义的时候参数列表为空即可，但是括号不能省略。

statements-list 为函数体部分，函数体就是实现该函数功能的具体程序细节，函数体结束部分

可以有返回值，也可以没有返回值，如果有返回值则通过关键字 return 说明。定义一个加法功能函数

```
function add (a,b)
return a+b
end
```

函数的调用 add(1,2) --函数相加并返回结果 3
返回多个值，求相加与相乘函数

```
1.function addmul(a,b)
2.return a+b，  a*b
3.end
4.addmul(2,5)                                --函数相加相乘，返回 7 10
```

1.9 运动指令

指令符号	指令说明
MovL	直线方式运行到笛卡尔坐标系绝对位置的指令
MovLR	直线方式运行到笛卡尔坐标系相对位置的指令
MovP	点到点方式运行到笛卡尔坐标系绝对位置的指令
MovPR	点到点方式运行到笛卡尔坐标系相对位置的指令
MovJ	控制各个关节移动到目标角度的指令
MArchP	点到点方式控制机器人进行拱形移动的指令
MArc	从当前位置圆弧插补到笛卡尔坐标系绝对位置的指令
MCircle	笛卡尔坐标系下的整圆插补指令
linerun	自动运行状态下实现手动笛卡尔的运动
stoprun	自动运行状态下实现停止手动直线运动

1.9.1 MovL

使用说明	以直线的方式运动到笛卡尔坐标系下的目标绝对位置
语法说明	MovL(A, “CP=20 Acc=20 Dec=20 Spd=100 AccC=20 SpdC=20 In=10 PULSE+/PULSE-“)
参数说明	该指令一共两个参数，第一个参数 A 为目标点，第二个参数为可选参数、省略时系统默认全局状态值 **下表各指令参数说明**

A	笛卡尔坐标目标位置。可以是点的名称 p1~p999，也可以是点的索引号 1~999。可选参数，可以指定运动到目标位置的各项参数
CP	说明运动到目标点时是否平滑过渡，范围为 0~100
Acc	指定运动到目标位置的加速度，单位为 mm/ s^2
Dec	指定运动到目标位置的减速度，单位为 mm/ s^2
Spd	指定运动到目标位置的速度，单位为 mm/s
AccC	指定运动到目标位置的姿态加速度(可选)，单位为 mm/ s^2
SpdC	指定运动到目标位置的姿态速度（可选），单位为 mm/ s
In	指定一个输入触发信号用来终止运动（可选）
PULSE+ /PULSE-	PULSE+: 上升沿有效； PULSE-: 下降沿有效

举例说明	1. MovL(p1) --机器人以直线的方式从当前位置运动到 p1 目标点 2. MovL(10) --机器人以直线的方式从当前位置运动到 p10 目标点 3. MovL(10, "Acc=100 Spd=1000") --机器人以直线的方式从当前位置运动到目标点 p10，其中加速度为 100 mm/s^2，速度为 1000 mm/s 4. MovL(p20, "CP=20") --机器人以直线的方式运动到 p20 位置，其中目标位置 p20 以平滑度 20 过渡 5. MovL(p20, "In=10 PULSE+") --机器人以直线的方式运动到 p20 目标位置，在运动过程中如果检测输入信号 In10 上升沿有效则当前指令运行结束。

注意： 可选参数编程时可以根据需要是否设置，其中 Acc、 Spd 省略时，系统默认全局值，当设置 In 选项触发停止时， 参数 PULSE+/ PULSE-才会有意义，且只能二选一设置信号有效停止或无效停止。

1.9.2 MovLR

使用说明	以直线的方式运动到笛卡尔坐标系下的目标相对位置
语法说明	MovLR(A,B,"CP=20 Acc=20 Dec=20 Spd=100 AccC=20 SpdC=20 In=10 PULSE+/ PULSE-")
参数说明	该指令一共三个参数，第一个为笛卡尔坐标轴号， 第二个参数为移动的相对距离， 第三个参数为可选参数、 省略时系统默认全局状态值 **下表各指令参数说明**

A	AX,AY,AZ,AC 各笛卡尔坐标轴号
B	各轴移动的相对距离
CP	说明运动到目标点时是否平滑过渡(可选)，范围为 0~100
Acc	指定运动到目标位置的直线加速度(可选)，单位为 mm/s^2

	Dec	指定运动到目标位置的直线减速度(可选)，单位为 mm/s^2
	Spd	指定运动到目标位置的直线速度(可选)，单位为 mm/s
	AccC	指定运动到目标位置的姿态加速度(可选)，单位为 mm/s^2
	SpdC	指定运动到目标位置的姿态速度(可选)，单位为 mm/s
	In	指定一个输入触发信号用来终止运动(可选)
	PULSE+ /PULSE-	PULSE+: 上升沿有效; PULSE-: 下降沿有效(可选)
举例说明:	1. MovLR(AX,10) --X 轴从当前位置往正方向移动 10mm 的距离 2. MovLR(AC,10) --C 轴从当前位置往正方向移动 10 度的角度 3. MovLR(AY,-10) --Y 轴从当前位置往负方向移动 10mm 的距离 4. MovLR(AZ,10) --Z 轴从当前位置往正方向移动 10mm 的距离	

1.9.3 MovP

使用说明	点到点方式移动到笛卡尔坐标系下的目标绝对位置	
语法说明	MovP(A,"CP=20 Acc=20 Dec=20 Spd=100")	
参数说明	该指令一共两个参数，第一个参数 A 为目标点，第二个参数为可选参数，省略时系统默认全局状态值 **下表各指令参数说明**	
	A	笛卡尔坐标目标位置。可以是点的名称 p1~p999，也可以是点的索引号 1~999。可选参数 " CP Acc=20 Dec=20 Spd=100"
	CP	可选参数，说明运动到目标点时是否平滑过渡，范围 0~100
	Acc	可选参数，指定运动到目标位置的加速度比例，范围 1~100
	Dec	可选参数，指定运动到目标位置的减速度比例，范围 1~100
	Spd	可选参数，指定运动到目标位置的速度比例，范围 1~100
举例说明:	1. MovP(p1) --机器人以点到点的方式从当前位置运动到 p1 目标点 MovP(10) --机器人以点到点方式从当前位置运动到 p10 目标点 2. MovP(10, "Acc=50 Spd=50") --机器人以点到点方式从当前位置运动到 p10 目标点，其中加速度为 50%的倍率，速度为 50%的倍率 3. MovP(p20, "CP=20") --机器人以点到点方式运动到 p20 位置，其中目标位置 p20 以平滑度 20 过渡 4. local p={X=200,Y=10} --用户定义 p 点变量 5. MovP(p) --运动到 p 点目标位置	

1.9.4 MovPR

使用说明	点到点方式移动到笛卡尔坐标系下的目标相对位置	
语法说明	MovPR(A,B," CP=20 Acc=20 Dec=20 Spd=100")	
参数说明	A	AX,AY,AZ,AC 各笛卡尔坐标轴号
	B	各轴移动的相对距离
	CP	可选参数，说明运动到目标点时是否平滑过渡，范围 0~100
	Acc	可选参数，指定运动到目标位置的加速度比例，范围 1~100
	Dec	可选参数，指定运动到目标位置的减速度比例，范围 1~100

	Spd	可选参数， 指定运动到目标位置的速度比例，范围 1~100
举例说明	1. MovPR(AX,10) --X 轴从当前位置以点到点的方式往正方向移动 10mm 的距离 2. MovPR(AY,10) --Y 轴从当前位置以点到点的方式往正方向移动 10mm 的距离 3. MovPR(AZ,-10) --Z 轴从当前位置以点到点的方式往负方向移动 10mm 的距离 4. MovPR(AC,10) --C 轴从当前位置以点到点的方式往正方向移动 10 度的角度	

1.9.5 MovJ

使用说明	点到点的方式移动机器人各个关节到指定的角度绝对位置	
语法说明	1. MovJ (A,B," Acc=20 Dec=20 Spd=20") 2. MovJ(A,"Acc=20 Dec=20 Spd=20")	
参数说明	用法 1: **下表各指令参数说明**	
	A	J1,J2,J3,J4 对应机器人各个关节号
	B	各关节移动的目标角度
	Acc	可选参数， 指定运动到目标位置的加速度比例，范围 1~100
	Dec	可选参数，指定运动到目标位置的减速度比例，范围 1~100
	Spd	可选参数，指定运动到目标位置的速度比例，范围 1~100
	用法 2: **下表各指令参数说明**	
	A	关节方式移动到笛卡尔坐标系下的目标绝对位置，范围 p1~p999
	Acc	可选参数， 指定运动到目标位置的加速度比例，范围 1~100
	Dec	可选参数，指定运动到目标位置的减速度比例，范围 1~100
	Spd	可选参数，指定运动到目标位置的速度比例，范围 1~100
举例说明	1. MovJ (J1,10) --机器人第一关节移动到 10 度的位置 2. MovJ (J3,-10) --机器人第三关节移动到-10 毫米的位置 3. MovJ(p1, "Acc=20 Dec=20 Spd=20") －关节运动到目标位值 p1 点	

注意： 关节运动时 J3 为毫米单位，其它关节为角度单位。

1.9.6 MArchP

使用说明	机器人以点到点的方式做拱形运动	
语法说明	MArchP(A,B,C,D," Acc=20 Dec=20 Spd=100")	
参数说明	**下表各指令参数说明**	
	A	为 p1~p999 点名称或 1~999 点索引号
	B	为 Z 轴最高限位绝对位置，单位为毫米
	C	为 Z 轴上升的高度，单位为毫米
	D	为 Z 轴下降的高度，单位为毫米
	Acc	可选参数， 指定运动到目标位置的加速度比例，范围 1~100
	Dec	可选参数，指定运动到目标位置的减速度比例，范围 1~100
	Spd	可选参数，指定运动到目标位置的速度比例，范围 1~100
举例说明	1. MArchP(p1,0,10,5) --机器人从当前位置 Z 轴上升 10mm 距离，然后以点到点的运动方式移动到距离目标位置 5mm 处， Z 轴下降 5mm 到达	

目标位置（拱形的高度不能超过 Z 轴的最高限 0mm （绝对位置））

2. MArchP(p1,0,10,5,” Acc=50 Spd=100”) --机器人以 50%的加速度，100%s 的速度做点到点方式拱形运动

1.9.7 MArc

使用说明	笛卡尔坐标系下的圆弧运动
语法说明	MArc(A,B," CP=20 Acc=20 Dec=20 Spd=100 Angle = 360")
参数说明	**下表各指令参数说明**

A	笛卡尔坐标圆弧经过点。可以是点的名称 p1~p999，也可以是点的索引 1~999
B	笛卡尔坐标圆弧终点。可以是点的名称 p1~p999，也可以是点的索引 1~999
CP	可选参数，说明运动到目标点时是否平滑过渡，范围 0~100
Acc	可选参数，指定运动到目标位置的加速度，单位为 mm/s^2
Dec	可选参数，指定运动到目标位置的减速度，单位为 mm/s^2
Spd	可选参数，指定运动到目标位置的速度，单位为 mm/s
Angle	可选参数，指定圆弧的角度，范围 1~360

举例说明

1. MArc(p1,p2) --机器人从当前位置以圆弧的方式经过 p1 点运动到 p2 ' 目标位置
2. MArc(1,2) --机器人从当前位置以圆弧的方式经过 p1 点运动到 p2 目标位置
3. MArc(1,2,"Acc=100 Spd=1000") --机器人从当前位置以圆弧的方式经过 p1 点运动到 p2 目标点，其中加速度为 100mm/s^2，速度为 1000mm/s
4. MArc(1,2,"CP=20") --机器人圆弧运动到 p2 点时平滑过渡

1.9.8 MCircle

使用说明	机器人在笛卡尔坐标系下的圆周运动
语法说明	MCircle(A,B," CP=20 Acc=20 Acc=20 Spd=100 Angle=360 ")
参数说明	**下表各指令参数说明**

A	笛卡尔坐标圆弧经过点。可以是点的名称 p1~p999，也可以是点的索引 1~999
B	笛卡尔坐标圆弧终点。可以是点的名称 p1~p999，也可以是点的索引 1~999
CP	可选参数，说明运动到目标点时是否平滑过渡，范围 0~100
Acc	可选参数，指定运动到目标位置的加速度，单位为 mm/s^2
Dec	可选参数，指定运动到目标位置的减速度，单位为 mm/s^2
Spd	可选参数，指定运动到目标位置的速度，单位为 mm/s
Angle	可选参数，指定圆弧的角度，范围 1~360

举例说明

1. MCircle(p1,p2) --机器人从当前位置经过 p1 点运动到 p2 目标位置的整圆运动
2. MCircle(1,2) --机器人从当前位置经过 p1 点运动到 p2 目标位置的整圆运动。
3. MCircle(1,2,"Acc=100 Spd=1000") --机器人从当前位置经过 p1 点运动到 p2 目标点的圆周运动，其中加速度为 100mm/s^2，速度为 1000 mm/s^2。

4. MCircle(1,2,"CP=20") --机器人圆周运动到 p2 点时平滑过渡 p2 点。
5. MCircle (1,2,"Angle=180") --机器人从当前位置往 p1 到 p2 的方向运动 180 度的半圆

1.9.9 linerun

使用说明	自动运行状态下实现手动笛卡尔的运动
语法说明	line(A,B,C)
参数说明	**下表各指令参数说明**

A	AX,AY,AZ,AC 各笛卡尔坐标轴号
B	运动的方向； 1 代表正方向， -1 代表负方向
C	移动的距离

举例说明

1. linerun(AX,1,10) --X+轴方向手动每次移动 10mm
2. linerun(AX,-1,10) --X-轴负方向手动每次移动 10mm
3. linerun(AY,1,10) --Y+轴方向手动每次移动 10mm
4. linerun(AY,-1,10) --Y-轴负方向手动每次移动 10mm
5. linerun(AZ,1,10) --Z+轴方向手动每次移动 10mm
6. linerun(AZ,-1,10) --Z-轴负方向手动每次移动 10mm
7. linerun(AC,1,10) --C+轴方向手动每次移动 10 度
8. linerun(AC,-1,10) --C-轴负方向手动每次移动 10 度

注：linerun 指令实现的是连续手动笛卡尔运动。

1.9.10 stoprun

使用说明	自动运行状态下实现停止手动直线运动
语法说明	stoprun()
参数说明	**无参数**

举例说明

```
function Bit(value,bit)
    if (value & (0x0001<<bit)) == 0 then
        return nil
    else
        return 1
    end
end
function ExtManu()
    local value
    value = publicread(0x100)
    if DI(0)==ON then --X+
        print("X+\n")
        linerun(AX,1,50)
    elseif DI(1)==ON then --X-
        print("X-\n")
        linerun(AX,-1,50)
    elseif DI(2)==ON then --Y+
        print("Y+\n")
        linerun(AY,1,50)
```

```
        elseif DI(3)==ON then --Y-
            print("Y-\n")
            linerun(AY,-1,50)
        elseif DI(4)==ON then --Z+
            print("Z+\n")
            linerun(AZ,1,50)
        elseif DI(5)==ON then --Z-
            print("Z-\n")
            linerun(AZ,-1,50)
        elseif DI(6)==ON then --C+
            print("C+\n")
            linerun(AC,1,50)
        elseif DI(7)==ON then --C
            print("C-\n")
            linerun(AC,-1,0)
        else
            stoprun()
        end
    end
    function main()
        while 1 do
            ExtManu()
            Delay(10)
        end
    end
```

注： stoprun 指令实现的是停止手动笛卡尔运动，故 linerun 指令和 stoprun 指令是配套使用的。若没有调用 stoprun 指令， linerun 指令会一直执行直到运动到限位位置。

1.10 运动参数设置指令

指令符号	指令说明
AccJ	设置加速度比例指令，影响 MovJ、 MovJR、 MovP、 MovPR、MArchP 指令的加速时间
DecJ	设置减速度比例指令，影响 MovJ、 MovJR、 MovP、 MovPR、MArchP 指令的减速时间
SpdJ	设置速度比例指令，影响 MovJ、 MovJR、 MovP、 MovPR、 MArchP 指令的运行速度
AccL	设置直线运动指令加速度,影响 MovL、MovLR、MArchL、MArc、MCircle 指令的加速时间。单位 mm/s^2
DecL	设置直线运动指令减速度,影响 MovL、MovLR、MArchL、MArc、MCircle 指令的减速时间。单位 mm/s^2
SpdL	设置直线运动指令速度，影响 MovL、 MovLR、 MArchL、 MArc、MCircle 指令的运行速度。单位 mm/s^2

1.10.1 AccJ

使用说明	设置点到点运动方式的加速度比例	
语法说明	AccJ(A)	
参数说明	A	百分比，范围 1~100
举例说明	1. AccJ(50) --设置点到点 50%的加速度比例 2. MovP(p2) --机器人以 50%的加速度比例运动到 p2 目标位置	

注意：设置后机器人一直保持该全局加速度比例为点到点运动的默认加速度比例值，直到下一次更新。

1.10.2 DecJ

使用说明	设置点到点运动方式的减速度比例	
语法说明	DecJ(A)	
参数说明	A	百分比， 范围 1~100
举例说明	3. DecJ(50) --设置点到点 50%的减速度比例 4. MovP(p2) --机器人以 50%的减速度比例运动到 p2 目标位置	

1.10.3 SpdJ

使用说明	设置点到点运动方式的速度比例	
语法说明	SpdJ(A)	
参数说明	A	百分比，范围 1~100
举例说明	1. SpdJ(50) --设置点到点 50%的速度比例 2. MovP(p2) --机器人以 50%的速度比例运动到 p2 目标位置	

1.10.4 AccL

使用说明	设置直线运动方式的加速度	
语法说明	AccL(A)	
参数说明	A	实际加速度，单位 mm/s^2，范围 1~10000
举例说明	1. AccL(500) --设置直线运动的加速度为 500 mm/s^2 2. MovL(p2) --机器人以 500 mm/s^2 的加速度运动到 p2 目标位置	

1.10.5 DecL

使用说明	设置直线运动方式的减速度	
语法说明	DecL(A)	
参数说明	A	实际减速度，单位 mm/s^2，范围 1~10000
举例说明	3. DecL(500) --设置直线运动的减速度为 500 mm/s^2 4. MovL(p2) --机器人以 500 mm/s^2 的减速度运动到 p2 目标位置	

1.10.6 SpdL

使用说明	设置点到点运动方式的速度比例	
语法说明	SpdL(A)	
参数说明	A	百分比，范围 1~100

举例说明	1. SpdL(500) --设置直线运动的速度为 500mm/s 2. MovL(p2) --机器人以 500mm/s 的速度运动到 p2 目标位置

1.11 程序管理指令

指令符号	指令说明
Delay	延时指令。单位：毫秒
Exit	程序运行终止
Pause	暂停程序运行

1.11.1 Delay

使用说明	延时指令
语法说明	Delay(A)
参数说明	A 延时时间，单位为毫秒,范围 1~100000
举例说明	1. Delay(1000) --程序延时 1000 毫秒 2. Delay(1) --机器人延时 1 毫秒 3. local time = 1000 4. Delay(time) --参数为变量

1.11.2 Exit

使用说明	退出程序执行指令
语法说明	Exit()
参数说明	该指令无参数
举例说明	1. Exit() --执行该指令后程序程序停止执行 2. MovL(1) --该指令不会再执行

1.11.3 Pause

使用说明	暂停程序执行指令
语法说明	Pause()
参数说明	该指令无参数
举例说明	1. Pause() --执行该指令后程序暂停执行 2. MovL(1) --按启动键后程序继续运行该指令

注意： 暂停后， 按启动键可以继续执行程序。

1.12 一般指令

指令符号	指令说明
X	创建指定 X 轴笛卡尔坐标绝对位置点的指令
Y	创建指定 Y 轴笛卡尔坐标绝对位置点的指令
Z	创建指定 Z 轴笛卡尔坐标绝对位置点的指令
C	创建指定 C 轴笛卡尔坐标绝对位置点的指令
XYZC	创建指定 XYZC 轴笛卡尔坐标绝对位置点的指令

1.12.1 X/ Y/ Z/ C

使用说明	创建指定某个轴笛卡尔坐标位置的点
语法说明	X(A) Y(A) Z(A) C(A)
参数说明	A 各个轴笛卡尔坐标位置， C 轴为角度，其它轴为 mm
举例说明	1. MovL(p10 + X(20)) --机器人运动到相对于 p10 点往 X 轴正向偏移 20 毫米的位置处 2. MovLR(X(20)) --机器人相对当前位置往 X 轴正向移动 20 毫米 3. MovLR(Z(-20)) --机器人相对当前位置往 Z 轴负向移动 20 毫米 4. MovLR(C(20)) --机器人相对当前位置往 C 轴正向移动 20 度

注意： 该指令是用来生成一个点数据，不会产生运动，一般与点数据运算后作为参数传递给运动指令。

1.12.2 XYZC

使用说明	创建指定各个轴笛卡尔坐标位置的点
语法说明	XYZC(A,B,C,D)
参数说明	**下表各指令参数说明**

A	指定 X 轴笛卡尔坐标位置，单位 mm
B	指定 Y 轴笛卡尔坐标位置，单位 mm
C	指定 Z 轴笛卡尔坐标位置，单位 mm
D	指定 C 轴笛卡尔坐标位置，单位 度

举例说明	MovL(p10 + XYZC(10,20,-5,30)) --机器人运动到相对于 p10 点往 X 轴正向偏移 10 毫米,Y 轴正向偏移 20 毫米、 Z 轴负向偏移 5 毫米、 C 轴正向偏移 30 度的位置移动

1.13 输入/输出指令

指令符号	指令说明
DI	读取输入端口状态
DO	写输出端口状态
WDI	读取输入端口状态,直到等待某一信号有效，则继续执行后续的程序
WDO	读取输出端口状态,直到等待某一信号有效，则继续执行后续的程序

1.13.1 DI

使用说明	读取输入端口状态
语法说明	**形式 1**： DI() **形式 2**： DI(A)
参数说明	**形式 1**： 返回值 一个 32 位的二进制数，从低位到高位分别代表输入端口的状态值， 0 代表关闭， 1 代表打开。 **形式 2**： 返回值 ON 或 OFF **下表各指令参参数说明** A 读取的输入端口号，范围 0~33
举例说明	1. local input = DI() --获取所有输入端口的状态 if ((input>>0)&0x0001)==1 then --判断输入端口 0 是否打开，若为 --1，则输入端口 0 打开

```
        MovP(p1)
    elseif ((input>>9)&0x0001)==1 then --判断输入端口 9 是否打开，若
                                        --为 1， 则输入端口 1 打开
        MovP(p2)
    end
2. if DI(10) == ON then --输入端口 10 有效时运动到 p1 点
        MovP(p1)
    end
```

注意： DI 指令只是读取输入端口的状态，状态有效或无效都不会死等待。

1.13.2 DO

使用说明	读取或写输出端口状态
语法说明	DO(A,B) DO(A,B,"Time = 1000")
参数说明	返回值 ON 或 OFF **下表各指令参数说明**

A	读写的输出端口号，范围 0~23
B	写输出端口的状态值， ON 或 OFF
Time	可选参数， 状态保持时间，单位为毫秒

举例说明

```
1. if DO(10) == ON then --读输出端口是否为 ON
2.       MovP(p1)
3. end
4. DO(10,ON) --打开 10 号输出端口为 ON
5. DO(10,ON, “Time=1000”) --打开 10 号输出端口为 ON,并保持 1 秒钟， 1 秒之后输出 10 切换为 OFF。
```

注意：可选参数 Time 为输出端口保持时间， 时间到位后会将该状态切换为相反状态

1.13.3 WDI

使用说明	读取输入端口状态,直到等待某一信号有效，则继续执行后续的程序
语法说明	WDI(A,B) WDI(A,B, “Time=1000”)
参数说明	返回值 ON 或 OFF **下表各指令参数说明**

A	读取的输入端口号
B	输入端口的状态值，ON 或 OFF
Time	可选参数，等待时间，单位为毫秒

举例说明

```
1. WDI(10,ON) --直到等到输入端口 10 状态为 ON 再执行 MovP 指令 MovP(p1)
2. WDI(10,ON, “Time=2000”) --等待输入端口 10 状态为 ON， 等待时间为 2 -------秒，若 2 秒之后输入端口 10 状态仍为 OFF，则继续执行 MovP 指令 MovP(p1)
```

1.13.4 WDO

使用说明	读取输出端口状态,等待某一信号有效，则继续执行后续的程序
语法说明	WDO(A,B) WDO(A,B, “Time=1000”)
参数说明	返回值 ON 或 OFF **下表各指令参数说明**

A	读取的输出端口号
B	输出端口的状态值，ON 或 OFF
Time	可选参数，等待时间，单位为毫秒

举例说明

1. WDO(10,ON) --直到等到输出端口 10 状态为 ON 再执行 MovP 指令
   MovP(p2)
2. WDO(10,ON, “Time=2000”) --等待输出端口 10 状态为 ON，等待时间为 2 秒，若 2 秒之后输出端口 10 状态仍为 OFF，则继续执行 MovP 指令
   MovP(p2)

1.14 坐标系指令

指令符号	指令说明
SetU	设置机器人当前用户坐标系
SetT	设置机器人当前工具坐标系
WrU	修改用户坐标系的数据
WrT	修改工具坐标系的数据
CacU	新建用户坐标系
V2Tool	根据视觉坐标计算工具坐标
getcart	获取当前笛卡尔坐标

1.14.1 SetU

使用说明	设置当前用户坐标系
语法说明	形式 1: SetU(A) 形式 2: SetU(A,B)

参数说明	A	设置的用户坐标系号，范围 0~9
	B	可选参数， 用户坐标偏移量{x=10,y=20,z=0,c=30}

举例说明

1. SetU(1) --选择用户坐标系 1
   MovL(p1) --运动到用户坐标系 1 中的 p1 位置
2. SetU(1,{ x=10,y=20,z=0,c=30}) --在用户坐标系 1的基础上增加偏移量，然后设定为新的用户坐标系
   MovL(p1) --在新的用户坐标系中运动到 p1 位置

注意：坐标系设置后立即有效并保持到下一次更新。

注意：在原有的用户坐标系的基础上设置偏移量生成新的用户坐标，原有的用户坐标没有被修改。

1.14.2 SetT

使用说明	设置当前工具坐标系	
语法说明	SetT(A)	
参数说明	A	设置的工具坐标系号，范围 1~9
举例说明	1. SetT (1) --选择工具坐标系 1 2. MovL(p1) --运动到工具坐标系 1 中的 p1 位置	

1.14.3 WrU

使用说明	修改用户坐标系的数据	
语法说明	**形式 1**: WrU(A,B,C) **形式 2**: WrU(A,B)	
参数说明	**下表各指令参数说明(形式 1)**	
	A	要修改的用户坐标系号，范围为 1~9
	B	修改的轴号，分别为 AX、AY、AZ、AC
	C	修改的值，X/Y/Z 轴单位为毫米,C 轴单位为角度
	下表各指令参数说明(形式 2)	
	A	要修改的第几组用户坐标系号，范围为 1~9
	B	Table 类型，包含用户坐标系的原点坐标，以及该用户相对于用户 0 旋转的角度
举例说明	1. WrU(1,AX,200) --修改第一组用户坐标系的 X 轴为 200 毫米 2. WrU(1,AC,100) --修改第一组用户坐标系的 C 轴为 100 度 3. WrU(2,{x=200,y=100,z=0,c=90}) --修改用户 2 的原点为(200,100,0), 用户 2 相对于用户 0 的旋转角度为 90 度 4. Pos = { x=300,y=100,z=0,c=-90} WrU(3,Pos)	

1.14.4 WrT

使用说明	修改工具坐标系的数据	
语法说明	**形式 1**: WrT (A,B,C) **形式 2**: WrT(A,B)	
参数说明	**下表各指令参数说明(形式 1)**	
	A	要修改的工具坐标系号，范围为 1~9
	B	修改的轴号，分别为 AX、 AY、 AZ、 AC
	C	修改的值，单位为毫米,C 轴为角度
	下表各指令参数说明(形式 2)	
	A	要修改的工具坐标系号，范围为 1~9
	B	Table 类型，包含要修改的工具末端相对于机器人末端在 X,Y,Z,C 轴方向的偏移量
举例说明	1. WrT(1,AX,20) --修改工具坐标系 1 的 X 轴偏移量为 20 毫米 2. WrT(1,AY,10) --修改工具坐标系 1 的 Y 轴偏移量为 10 毫米 3. WrT(1,{x= 10,y=20,z=0,c=30}) – 修改工具 1 相对于机器人末端在 X、Y、Z、 C 方向上的偏移量分别为 10mm,20mm,0mm,30 度。 4. Offset = { x=10,y=10,z=0,c=-30}	

WrT(2,Offset) --参数为变量，修改工具 2 相对于机器人末端在 X、Y、Z、 C 方向上的偏移量分别为 10mm,10mm,0mm,-30 度。

1.14.5 CacU

使用说明	新建用户坐标系	
语法说明	CacU (pos1,pos2)	
参数说明	**下表各指令参数说明**	
	pos1	用户坐标系的原点
	Pos2	用户坐标系 X 方向上的一点
举例说明	1. local pos1 = {300,100,0,0} --用户坐标系的原点 2. local pos2 = {300,120,0,0} --用户坐标系 X 轴方向上的一点 3. WrU(1,CacU(pos1,pos2)) --将新建的用户坐标指定为用户坐标系 1	

1.14.6 V2Tool

使用说明	根据视觉坐标计算工具坐标	
语法说明	V2Tool(A,B)	
参数说明	A	视觉输出的笛卡尔坐标值
	B	工具坐标系号，范围 1~9
举例说明	local vis = {x= 300, y=10,z=0,c=0} --视觉坐标 V2Tool(vis,3) --将计算出来的工具写入到工具坐标系 3 SetT(3) --设置当前工具坐标系为 3 号坐标系	

注意：视觉坐标和当前的机器人坐标必须在同一坐标系下。

1.14.7 getcart

使用说明	获取当前笛卡尔坐标以及关节坐标	
语法说明	pos1,pos2= getcart ()	
参数说明	无参数	
返回值	pos1	当前机器人的笛卡尔坐标
	pos2	当前机器人的关节坐标
举例说明	1. local pos1,pos2= getcart () --获取当前笛卡尔坐标以及关节坐标 --pos1.x,pos1.y,pos1.z,pos1.c,pos1.h 分别为笛卡尔 x/y/z/c/h 的值 -- pos2.x,pos2.y,pos2.z,pos2.c,pos2.h 分别为关节 J1/J2/J3/J4 以及手系的值 2. local pos = getcart () --获取当前笛卡尔坐标 -- pos.x,pos.y,pos.z,pos.c,pos.h 分别为笛卡尔 x/y/z/c/h 的值	

注意：若只返回一个值，则为当前机器人的笛卡尔坐标。

1.15 码垛指令

指令符号	指令说明
SetPlt	设置码垛数指令
GetPlt	取码垛数据点指令
ToPutPLT	放码垛
FromPutPLT	放码垛后，离码垛
ResetPLT	重置码垛个数

1.15.1 SetPlt

使用说明	设置码垛参数
语法说明	XY 轴码垛 SetPlt(A,B,C,D,E,F) XYZ 轴码垛 SetPlt(A,B,C,D,E,F,G,H)
参数说明	**下表各指令参数说明** XY 轴码垛 SetPlt(A,B,C,D,E,F)

A	码垛号，范围为 1~6
B	码垛原点位置
C	相对于码垛原点的 X 轴方向最后一个码垛点
D	相对于码垛原点的 Y 轴方向最后一个码垛点
E	码垛 X 轴方向等分数
F	码垛 Y 轴方向等分数

XYZ 轴码垛 SetPlt(A,B,C,D,E,F,G,H)

A	码垛号，范围为 1~6
B	码垛原点位置
C	相对于码垛原点的 X 轴方向最后一个码垛点
D	相对于码垛原点的 Y 轴方向最后一个码垛点
E	相对于码垛原点的 Z 轴方向最后一个码垛点
F	码垛 X 轴方向等分数
G	码垛 Y 轴方向等分数
H	码垛 Z 轴方向等分数

举例说明	1. SetPlt(1,p1,p2,p3,5,11) --设置 XY 轴码垛 2. SetPlt(1,p1,p2,p3,p4,11,5,3) --设置 XYZ 轴码垛

注意: 码垛设置指令只要在程序开始执行一次，指令自动根据参数的个数判断是 XY 轴码垛或 XYZ 轴码垛。

1.15.2 GetPlt

使用说明	获得码垛数据点
语法说明	XY 轴码垛 GetPlt(A,B,C) XYZ 轴码垛 GetPlt(A,B,C,D)
参数说明	返回值 返回码垛中的各个点位数据 **下表各指令参数说明** XY 轴码垛 GetPlt(A,B,C)

A	码垛号，范围为 1~6
B	码垛 X 方向的位置，从 1 开始
C	码垛 Y 方向的位置，从 1 开始

XYZ 轴码垛 GetPlt(A,B,C,D)

A	码垛号，范围为 1~6
B	码垛 X 方向的位置，从 1 开始
C	码垛 Y 方向的位置，从 1 开始
D	码垛 Z 方向的位置，从 1 开始

举例说明	function main() SetPlt(1,p1,p2,p3,5,11) ---设置 XY 轴码垛

```
i=1
j=1
while i <= 5 do
        j=1
    while j <= 11 do
        pos = GetPlt(1,i,j) --循环指令中读取码垛点位数据
        print(pos.X,pos.Y,pos.Z,pos.C,pos.H) --打印输出坐标
        j = j + 1
        MovL(pos) --直线运动到码垛位置
    end
     i = i + 1
end
end             --主程序结束
```

1.15.3 ToPutPLT

使用说明	放码垛
语法说明	ToPutPLT(PltName)
参数说明	无返回值 **输入参数说明** PltName　码垛名（ PLT0~PLT4 ）

注： 使用该语句，须在码垛配置中设置关于码垛的一些参数，包括码垛名、码垛顺序、参考坐标系等

1.15.4 FromPutPLT

使用说明	放码垛后，离码垛
语法说明	FromPutPLT(PltName)
参数说明	**返回值** O　盘码垛未满，可继续码垛 1　盘码垛已满 **输入参数说明** PltName　码垛名（ PLT0~PLT4 ）

注： FromPutPLT 指令和 ToPutPLT 指令是配套使用的。

1.15.5 ResetPLT

使用说明	重置码垛个数
语法说明	ResetPLT(PltName,count)
参数说明	**返回值** 无 **输入参数说明** PltName　码垛名（ PLT0~PLT4 ） count　码垛个数，可任意设置个数，码垛个数从 1 开始
举例说明	local posReady = p1 --待机点&安全点 local quliao = p2 --取料点

```
MArchP(posReady,5,1,1) --待机点&安全点
while true do
    MArchP(quliao,5,1,1)
    ToPutPLT("PLT1") --放码垛 PLT1
    full=FromPutPLT("PLT1") --离码垛 PLT1
    --每次码完一个物体， full 会用来判断码盘是否码满，码满则为 1
    if full==1 then
        ResetPLT("PLT1",1)
      end
end
```

1.16 伺服管理指令

指令符号	指令说明
MotOn	伺服所有轴打开使能
MotOff	伺服所有轴关闭使能

1.16.1 MotOn

使用说明　伺服所有轴使能打开
语法说明　MotOn()
参数说明　无参数
举例说明　MotOn() --所有轴使能打开

1.16.2 MotOff

使用说明　伺服所有轴使能关闭
语法说明　MotOff()
参数说明　无参数
举例说明　MotOff() --所有轴使能关闭

1.17 通信指令

指令符号	指令说明
RecCom	从 RS232 串口接收数据
SendCom	往 RS232 串口发送数据
ClrCom	清除 RS232 串口接收缓冲区
sysnetclr	清除网络数据
sysnetget	查询方式读取网络数据
sysnetsend	发送网络数据
sysnetcatch	阻塞式读取网络数据
CloseNet	关闭 TCP 网络连接
OpenNet	创建 TCP 网络
ConnectNet	连接 TCP 网络
RecvNet	网络接收数据功能函数
WriteNet	网络发送数据功能函数
publicread	读取全局表数据值

Publicwrite	写入数据值到全局表

1.17.1 RecCom

使用说明	RS232 串口接收数据
语法说明	Err,RecBuf=RecCom(A, "Time=5000")
参数说明	**指令参数说明**

A	通信端口号 1
Time	设置接收超时时间，单位为毫秒

返回值

RecBuf	通信端口号 1
Err	0 表示接收成功
	1 表示接收超时

举例说明

```
local RecBuf --定义接收缓冲区
local Err --定义接收错误号
Err,RecBuf = RecCom(1," Time=5000") --串口 1 接收数据，接收超时时间为 5 秒。
if Err == 0 then --接收成功
    print(RecBuf.buff) --RecBuf 缓冲区中接收到上位机发送的数据
end
```

1.17.2 SendCom

使用说明	RS232 串口发送数据
语法说明	SendCom(A, B)
参数说明	**指令参数说明**

A	通信端口号 1
B	要发送的数据，可以使 ASCII 码类型，也可以使 TABLE 十六进制类型数据，指令自动判断参数类型进行发送

举例说明

```
local SendBuf={0x01,0x05,0x00,0x1A,0xff,0x00}
SendCom(1,SendBuf) --串口 1 以十六进制方式发送数据
local SendBuf = "ROBOT"
SendCom(1,SendBuf)
SendCom(1,"ROBOT")
```

1.17.3 ClrCom

使用说明	RS232 串口接收缓冲区清除
语法说明	ClrCom(A)
参数说明	通信端口号 1
举例说明	ClrCom(1) --清除 1 号端口串口数据 Err,RecBuf = RecCom(1," Time=5000") --清除串口 1 接收缓冲区后再接收

1.17.4 sysnetclr

使用说明	清除网络数据
语法说明	Sysnetclr(ipaton{"A"},B)
参数说明	**返回值** 无返回值 **指令参数说明** A 网络通讯 IP 地址 B 网络通讯端口号
举例说明	local CameraNet={ipaton("192.168.0.100"),8080} -- IP: 192.168.0.100; 端口号 8080 sysnetclr(CameraNet) --清除网络数据

1.17.5 sysnetget

使用说明	查询方式读取网路数据
语法说明	Err,Data = sysnetget({ipaton("A"),B})
参数说明	**指令参数说明** A 网络通讯 IP 地址 B 网络通讯端口号 **返回值** Err 接收错误号 0 表示接收成功 1 表示接收失败 Data 接收数据缓冲区
举例说明	local CameraNet={ipaton("192.168.0.100"),8080} -- IP: 192.168.0.100; 端口号 8080 local Err local Data Err,Data = sysnetget(CameraNet) --扫描式接收数据

1.17.6 sysnetsend

使用说明	发送网络数据
语法说明	sysnetsend({ipaton("A"),B},C)
参数说明	返回值 无返回值 **指令参数说明** A 网络通讯 IP 地址 B 网络通讯端`口号 C 网络发送的数据
举例说明	local CameraNet={ipaton("192.168.0.100"),8080} --IP: 192.168.0.100; 端口号 8080 local data={0x23,0x11,0x33} sysnetsend(CameraNet,data) --发送数组数组 local data = "trigger" sysnetsend(CameraNet,data) --发送字符串

1.17.7 sysnetcatch

使用说明	阻塞式读取网路数据
语法说明	Err,Data = sysnetcatch({ipaton("A"),B})
参数说明	**指令参数说明**

A	网络通讯 IP 地址
B	网络通讯端口号

返回值

Err	接收错误号
	0 表示接收成功
	1 表示接收失败
Data	接收数据缓冲区

举例说明

```
local CameraNet={ipaton("192.168.0.100"),8080} --IP:  192.168.0.100; 端口号 8080
local Err
local Data
Err,Data = sysnetcatch(CameraNet) --阻塞式接收数据
```

1.17.8 CloseNet

使用说明	关闭 TCP 网络连接
语法说明	CloseNet ({ipaton(A),B})
参数说明	**指令参数说明**

A	TCP 通讯， RC400 控制器作为客户端时视觉设备的 IP 地址
B	TCP 通讯， RC400 控制器作为客户端时视觉设备的端口号

1.17.9 OpenNet

使用说明	创建 TCP 网络
语法说明	OpenNet({ipaton(A),B})
参数说明	**指令参数说明**

A	TCP 通讯， RC400 控制器作为客户端时视觉设备的 IP 地址
B	TCP 通讯， RC400 控制器作为客户端时视觉设备的端口号

返回值

Err	接收错误号
	0 表示 TCP 网络打开成功
	1 表示 TCP 网络打开失败

1.17.10 ConnectNet

使用说明	连接 TCP 网络
语法说明	ConnectNet({ipaton(A),B})
参数说明	**指令参数说明**

A	TCP 通讯， RC400 控制器作为客户端时视觉设备的 IP 地址
B	TCP 通讯， RC400 控制器作为客户端时视觉设备的端口号

返回值

Err	接收错误号
	0 表示 TCP 网络连接成功
	1 表示 TCP 网络连接失败

1.17.11 RecvNet

使用说明	网络接收数据功能函数
语法说明	Err_net,RecBuf = RecvNet ({ipaton(A),B},C)
参数说明	**指令参数说明**

A	TCP 通讯， RC400 控制器作为客户端时视觉设备的 IP 地址
B	TCP 通讯， RC400 控制器作为客户端时视觉设备的端口号
C	时时间，单位毫秒（ms）

返回值

Err_net	0 表示接收数据成功
	1 表示接收数据失败
RecBuf	接收数据缓冲区

1.17.12 WriteNet

使用说明	网络发送数据功能函数
语法说明	WriteNet ({ipaton(A),B},C)
参数说明	**指令参数说明**

A	TCP 通讯， RC400 控制器作为客户端时视觉设备的 IP 地址
B	TCP 通讯， RC400 控制器作为客户端时视觉设备的端口号
C	发送的网络数据

举例说明

```
local ipPort={ipaton(192.168.0.100),2000} --ip， port
CloseNet(ipPort) --关闭 TCP 网络
Delay(100)
print("关闭 TCP 网络！ ")
if OpenNet(ipPort) == 0 then --打开 TCP 网络
        print("打开 TCP 网络！正在连接...")
        repeat
            Delay(2)
        until ConnectNet(ipPort) == 0 --连接 TCP 网络
        print("已连接 TCP 网络！ ")
end
while 1 do
        local x,y,z,c
        local err_net
        err_net,RecBuf = RecvNet(ipPort,5000) --等待接收网络数据,5 秒
钟超时
        if err_net == 0 then --接收成功
            if RecBuf.buff == "?" then --判断接收字符串是否为"？ "
                    WriteNet(ipPort,"OK") --发送字符串 “OK”
            end
        else
```

```
            print("接收失败:",err_net)
        end
        Delay(10)
    end
```

1.17.13 publicread

使用说明	读取全局表数据值	
语法说明	C = publicread(A, “B”)	
参数说明	**指令参数说明**	
	A	全局表数据值的地址，地址长度为 2
	B	读取数据的方式，整型（可省略） /浮点型/十六进制
	返回值	
	C	读取全局表中地址对应的数据值
举例说明	local a = publicread(0x100,"float") --AR 以浮点型方式读地址 0x100 的值 local b = publicread(0x102) -- AR 以整型方式读地址 0x102 的值	

1.17.14 publicwrite

使用说明	写入数据值到全局表	
语法说明	publicwrite(A,B,"C")	
参数说明	返回值 无返回值 **指令参数说明**	
	A	全局表数据值的地址，地址长度为 2
	B	写入到对应地址的数据
	C	写入数据的方式，整型（可省略） /浮点型/十六进制
举例说明	publicwrite(0x102,10.5,"float") --AR 以浮点型方式写入到地址 0x100 的值 publicwrite(0x102,5) -- AR 以整型方式写入到地址 0x102 的值	

1.18 视觉指令

指令符号	指令说明
VisToRob	将像素坐标转化成指定的用户坐标系下的坐标
CCDrecv	接收视觉返回的数据
CCDtrigger	触发相机拍照
CCDsent	发送字符串到视觉
CCDclr	清除网络托管的 IP
CCDoffset	视觉偏差补偿
GetDynCCDPos	动态视觉位置

1.18.1 VisToRob

使用说明	将像素坐标转化成指定的用户坐标系下的坐标	
语法说明	D= VisToRob (A,B,C)	
参数说明	A	像素坐标
	B	9 点或 16 点标定的一组系数
	C	相机的像素标志位，范围 0~9 0: 30 万像素（ 640*480） 1: 50 万像素（ 800*600） 2: 80 万像素（ 1024*768） 3: 100 万像素（ 1140*900） 4: 130 万像素（ 1280*960） 5: 200 万像素（ 1600*1200） 6: 300 万像素（ 2048*1536） 7: 500 万像素（ 2576*1932） 8: 500 万像素（ 2592*1944） 9: 500 万像素（ 2560*1920）
返回值	D 用户坐标系下的坐标	

1.18.2 CCDrecv

使用说明	接收视觉返回的数据	
语法说明	n,data=CCDrecv(CamName)	
参数说明	CamName	视觉名称，在 UI 配置时生成
返回值	n	接收视觉数据的组数
	data	接收视觉返回的绝对坐标，可直接用于点位运动

举例说明

```
n,data=CCDrecv( “CAM1” ) --接收视觉 CAM1 的数据
for i=1,n do --n 决定循环的次数
    print(i,data[i][1],data[i][2],data[i][3]) --打印每组数据
    if data[i][1]~=0 or data[i][2]~=0 then --判断数据不同时为零
        pos.x=data[i][1] --数据 data[i][1]赋值给 pos.x
        pos.y=data[i][2] --数据 data[i][2]赋值给 pos.y
        pos.z=0;
        pos.c=data[i][3] --数据 data[i][3]赋值给 pos.c
    end
end
MovP(pos) --点到点方式运动到点 pos
```

1.18.3 CCDtrigger

使用说明	触发相机拍照	
语法说明	CCDtrigger(CamName)	
参数说明	CamName	视觉名称，在 UI 配置时生成，可根据设置自动识别 网络触发 or IO 触发
返回值	无返回值	

1.18.4 CCDsent

使用说明	发送字符串到视觉	
语法说明	CCDsent(CamName,Buff)	
参数说明	CamName	视觉名称，在 UI 配置时生成
	Buff	发送的字符串（若不填，则在视觉 UI 界面填写值）
返回值	无返回值	

1.18.5 CCDclr

使用说明	清除网络托管的 IP	
语法说明	CCDclr(CamName)	
参数说明	CamName	视觉名称，在 UI 配置时生成
返回值	无返回值	

1.18.6 CCDoffset

使用说明	视觉偏差补偿	
语法说明	pos = CCDoffset(CamName,view_pos)	
参数说明	CamName	视觉名称，在 UI 配置时生成
	view_pos	接收的视觉坐标
返回值	pos 输出补偿后的绝对坐标	

1.18.7 GetDynCCDPos

使用说明	动态视觉位置计算	
语法说明	robot_pos= GetDynCCDPos(CamName,view_pos)	
参数说明	CamName	视觉名称，在 UI 配置时生成
	view_pos	动态视觉坐标
返回值	robot_pos 输出计算后的绝对坐标（基坐标下或当前用户坐标系下）	

1.19 动态跟随指令

指令符号	指令说明
FollowInit	动态跟随初始化
SetDynCatch	开启或关闭跟随抓取任务
GetCatchSpace	获取物件是否进入抓取区域
SetCatch	执行跟随
GetCatchState	获取抓取状态
SynOver	结束跟随
GetTrigger	获取触发状态（同一物体拍两次使用）
SetViewData	发送数据给控制器，存入缓存待跟随

1.19.1 FollowInit

使用说明	动态跟随初始化	
语法说明	FollowInit(CamName)	
参数说明	CamName	视觉名称，在 UI 配置时生成
返回值	无返回值	
举例说明	1. FollowInit(“CAM1”) --初始化动态跟随视觉 CAM1	

1.19.2 SetDynCatch

使用说明	开启或关闭动态抓取任务	
语法说明	SetDynCatch(n)	
参数说明	n	开启或关闭标志(可选参数)，范围 0~1
		0 关闭跟随抓取任务
		1 开始跟随抓取任务
举例说明	1. SetDynCatch(0) --关闭跟随任务 2. SetDynCatch(1) --开始跟随任务	

1.19.3 GetCatchSpace

使用说明	获取物件是否进入抓取区域	
语法说明	state=GetCatchSpace()	
参数说明	无参数	
参数说明	0	未进入抓取区
	1	进入抓取区

1.19.4 SetCatch

使用说明	执行跟随 （待物件进入抓取区后可执行此命令进行跟随）
语法说明	SetCatch ()
参数说明	无参数
返回值	无返回值

1.19.5 GetCatchState

使用说明	获取抓取状态	
语法说明	state=GetCatchState ()	
参数说明	无参数	
返回值	0	抓取结束，即正常抓取结束
	1	开始运动，即从当前点运行到目标点，并同速同位
	2	进入同步，已经同步成功，已开始同速同位
	3	错误退出，超出抓取区，无法完成同步抓取

1.19.6 SynOver

使用说明	结束跟随（待完成同步中工艺后，可执行此命令进行退出同步）
语法说明	SynOver()
参数说明	无参数
返回值	无返回值

1.19.7 GetTrigger

使用说明	获取触发状态，用来分辨同一物体拍两次时的先后顺序	
语法说明	num=GetTrigger()	
参数说明	无参数	
参数说明	0	为第一次拍照后到第二次拍照前标志
	1	为第二次拍照后再到第一次拍照前标志

1.19.8 SetViewData

使用说明	发送数据给控制器， 存入缓存待跟随	
语法说明	SetViewData (pos)	
参数说明	pos	视觉接收到的点位数据

举例说明

```
local pos={x=0,y=0,z=0,c=0,h= 0}
local n,data=CCDrecv("CAM1") --动态抓取接收相机 CAM1 发送的网络数据
if data then                          --如果接收数据不为空，则进行 for 循环
    for i=1,n do
        if data[i][1]~=0 and data[i][2]~=0 then
            pos.x=data[i][1] --data[i][1]赋值给 pos.x
            pos.y=data[i][2] --data[i][2]赋值给 pos.y
            pos.c=data[i][3] --data[i][3]赋值给 pos.c
            SetViewData(pos) --发送数据给控制器
        end
    end
end
```

1.20 调式指令

指令符号	指令说明
print	打印输出用户调试数据
Error	终止程序运行，并输出错误信息

1.20.1 print

使用说明	从 RS232 串口输出调试数据
语法说明	print(…)
参数说明	可变参数，参数个数和类型可以任意

举例说明

```
1. print(12) --从串口输出数据 12
2. print ("Robot") --从串口输出字符串数据 Robot
3. local a=10
4. print ("Robot",a) --从串口输出字符串数据 Robot 和数字 10
```

1.20.2 Error

使用说明	终止程序运行，并输出错误信息
语法说明	Error(A)
参数说明	用于输出用户定义的错误信息，字符串类型
举例说明	Error(“程序运行错误”) --执行后在界面输出报警信息并终止程序运行

1.21 点位指令

指令符号	指令说明
Point	调用点位表中的点位

1.21.1 Point

使用说明	调用点位表中的点位	
语法说明	B= Point (A)	
参数说明	A	点位表 DATA 中的点位数据，范围 1~999
返回值	B	将点位表中点位对出赋值给 B
举例说明	local pos={} pos=Point(10) --调用点位数据表中的 P10 点赋值给 pos print(pos.x,pos.y,pos.z,pos.c)	

注： A 的值只能为 1~999 中的数字，不能写成 p1~p999。

1.22 系统指令

指令符号	指令说明
syswork	设置系统工作状态
sysstate	读取各个轴的电流值或获取系统状态
sysrate	设置全局速度速率
systime	获取系统时钟

1.22.1 syswork

使用说明	设置系统工作状态	
语法说明	syswork(A)	
参数说明	无返回值 输入参数说明	
	A	1 启动程序运行
		2 暂停程序运行
		3 停止程序运行
		4 程序复位
举例说明	MovP(p1) --点到点运动到 p1 点 syswork(2) --程序暂停 Delay(100) --延时 100ms syswork(1) --启动程序	

1.22.2 sysstate

使用说明	读取各轴的电流值或获取系统状态
语法说明	state = sysstate(n)
参数说明	无输入 **返回值 state** 0 运行状态 非 0 异常状态 有输入 n,范围 1~4 分别代表 SCARA 机器人的四个轴 **返回值 state** 返回指定轴的电流值，单位为毫安
举例说明	1. local state state = sysstate() --获取系统状态 if state ~=0 then --若非 0 syswork(4) --复位 end 2. local state1 = sysstate(1) --获取第一轴的电流值 local state2 = sysstate(2) --获取第二轴的电流值 local state3 = sysstate(3) --获取第三轴的电流值 local state4 = sysatate(4) --获取第四轴的电流值

1.22.3 sysrate

使用说明	设置全局速度倍率
语法说明	sysrate(A)
参数说明	返回值 无返回值 **输入参数说明** A 速度倍率，范围 1~100
举例说明	sysrate(50) --设置全局速度倍率为 50%

1.22.4 systime

使用说明	获取系统时钟
语法说明	time = systime()
参数说明	无输入 **返回值 time** time 系统时钟值
举例说明	local time1=systime() --获取当前的系统时钟 MovP(p1) MovP(p2) local time2 = systime()-time1 --获取程序运行的时间 print(time2)

1.23 主站读写指令

指令符号	指令说明
ReadRegW	读指定 PLC 寄存器的 16 位字地址
ReadRegDW	读指定 PLC 寄存器的 32 位双字地址
WriteRegW	写入 16 位字地址到指定的 PLC 寄存器
WriteRegDW	写入 32 位字地址到指定的 PLC 寄存器

1.23.1 ReadRegW

使用说明	读指定 PLC 寄存器的 16 位字地址	
语法说明	Value = ReadRegW ({A,B},C,D)	
参数说明	A	数据通讯协议， A=1 代表异步收发器（ UART）
	B	站号
	C	寄存器地址
	D	读取变量的个数（可选），默认为 1
返回值	Value	返回读取寄存器对应地址中变量的值
举例说明	1. local plc={1,1} --PLC 通信参数使用 1 为 UART,站号为 1 2. ReadRegW(plc,250) --读 PLC 寄存器 250 16 位字地址 3. local a=ReadRegW(plc,250,20) --连续读 PLC 寄存器 250 开始的 20 个 16 位变量 4. for i=1,20 do print(a[i]) end	

1.23.2 ReadRegDW

使用说明	读指定 PLC 寄存器的 32 位双字地址	
语法说明	Value=ReadRegDW ({A,B},C,D,E)	
参数说明	A	数据通讯协议， A=1 代表异步收发器（ UART）
	B	站号
	C	寄存器地址
	D	读取变量的个数（可选），默认为 1
	E	读取变量的类型（可选），整型和浮点型，默认为整型
返回值	Value	返回读取寄存器对应地址中变量的值
举例说明	1. local plc={1,1} --PLC 通信参数使用 1 为 UART,站号为 1 2. print(ReadRegDW(plc,250)) --读 PLC 寄存器 250 32 位双字地址, 整型变量 3. print(ReadRegDW(plc,250,"float")) --读 PLC 寄存器 250 32 位双字地址,浮点型变量 4. local a=ReadRegDW(plc,250,20) --连续读 PLC 寄存器 250 开始的 20 个 32 位整型变量 5. for i=1,20 do print(a[i]) end 6. a = nil 7. local a=ReadRegDW(plc,250,20,"float") --连续读 PLC 寄存器 250	

开始的 20 个 32 位浮点变量

8. for i=1,20 do
   print(a[i])
   end
9. ReadRegDW(plc,250,0x100,10) --读 PLC 250 开始的 32 位变量到本地 0x100 变量中，共 10 个变量且为整型值
10. ReadRegDW(plc,250,0x100,10,"float") --读 PLC 250 开始的 32 位变量到本地 0x100 变量中，共 10 个变量且为浮点数
11. ReadRegDW(plc,250,{x=true,y=true,z=true,c=true,n=1},10) --读 PLC 250 开始的 32 位变量到 点位表中，共 10 点个变量且为整型
12. ReadRegDW(plc,250,{x=true,y=true,z=true,c=true,n=1},10,"float") --读 PLC 250 开始的 32 位变量到 点位表中，共 10 点个变量且为浮点

1.23.3 WriteRegW

使用说明	写入 16 位字地址到指定的 PLC 寄存器	
语法说明	Value =WriteRegW ({A,B},C,D)	
参数说明	A	数据通讯协议， A=1 代表异步收发器（ UART）
	B	站号
	C	寄存器地址
	D	写入变量的个数
返回值	Value	返回写入到寄存器对应地址中变量的值
举例说明	1. local plc={1,1} --PLC 通信参数使用 1 为 UART,站号为 1 2. WriteRegW(plc,250,1) --写 PLC 寄存器 250 16 位字地址 3. WriteRegW(plc,250,{10,20,30})--连续写 PLC 寄存器 250 16 位字地址	

1.23.4 WriteRegDW

使用说明	写入 32 位字地址到指定的 PLC 寄存器	
语法说明	WriteRegDW ({A,B},C,D,E,F)	
参数说明	A	数据通讯协议， A=1 代表异步收发器（ UART）
	B	站号
	C	寄存器地址
	D	本地地址（可选）
	E	读取变量的个数
	F	读取变量的类型（可选），整型和浮点型，默认为整型
返回值	无	
举例说明	1. local plc={1,1} --PLC 通信参数使用 1 为 UART,站号为 1 2. WriteRegDW(plc,250,1) --写 PLC 寄存器 250 32 位字地址 3. WriteRegDW(plc,250,0x100,10) --写本地 0x100 变量到 PLC 250 开始的 32 位变量到,共 10 个变量且为整型值 4. WriteRegDW(plc,250,{10,20,30})--连续写 PLC 寄存器 250 32 位字地址 5. WriteRegDW(plc,250,1,"float")--写 PLC 寄存器 250 32 位字地址浮点数 6. WriteRegDW(plc,250,0x100,10,"float") --写本地 0x100 变量到 PLC 250 开始的 32 位变量到,共 10 个变量且为浮点数 7. WriteRegDW(plc,250,{10,20,30},"float") --连续写 PLC 寄存器 250 32 位字地址浮点数	