目录

1 KRL 概述 4
1.1 程序结构 4
1.2 指令类型 6
1.3 常量 7
1.4 变量 8
1.4.1 项目变量 8
1.4.2 局部变量 9
1.4.3 系统变量 9
1.5 运算符 10
1.5.1 位操作 10
1.5.2 算术 10
1.5.3 逻辑 11
1.5.4 关系 11
1.5.5 赋值 12
1.5.6 类型转换 13
1.6 标签 14
1.7 注释 18
2 指令说明 19
2.1 IO 指令 21
2.1.1 DIN 21
2.1.2 DOUT 24
2.1.3 AIN 27
2.1.4 AOUT 29
2.1.5 PULSE 31
2.2 控制指令 34

2.2.1 JUMP 34
2.2.2 CALL 37
2.2.3 RET 40
2.2.4 PAUSE 43
2.2.5 ABORT 45
2.2.6 WAITIO 47
2.2.7 WAITMTN 50
2.2.8 TIMER 52
2.2.9 GETA 54
2.2.10 GETR 57
2.2.11 LABEL 61
2.3 运动指令 63
2.3.1 NOP 63
2.3.2 END 65
2.3.3 MOVJ 66
2.3.4 MOVL 68
2.3.5 MOVC 70
2.3.6 运动指令模块 73
2.4 运算指令 82
2.4.1 GETE 82
2.4.2 SETE 84
2.4.3 内置标准函数 86
2.5 设置指令 93
2.5.1 SPEED 93
2.5.2 RAMP 96
2.5.3 REFSYS 99
2.5.4 TOOL 101
2.5.5 AREA 103
2.6 系统指令 105

2.6.1 PRINT 105
2.6.2 CLS 107
2.6.3 GETPRM 108
2.6.4 SETPRM 111
2.6.5 ALM 113
2.7 模块 115
2.7.1 d const 115
2.7.2 s const 117
2.7.3 p const 118
2.7.4 d variable 119
2.7.5 s variable 120
2.7.6 p variable 121
2.7.7 system d variable 122
2.7.8 system s variable 123
2.7.9 system p variable 124
2.7.10 express 125
2.7.11 assignment 131
2.7.12 input/output 133
2.7.13 if condition 135
3 一览表 137
3.1 指令一览表 137
3.2 模块一览表 139

1 KRL 概述

KRL（KND Robot Language）是 KND 机器人编程语言，用于编程 job 文件。

- 关节 / 直线 / 圆弧插补
- 非运动指令可与运动指令并行执行，时间 / 距离等多种同步方式
- 子程序调用 / 返回 / 跳转
- 项目变量 / 局部变量 / 系统变量
- 数字变量自动类型（32 位整数 / 64 位整数 / 双精度浮点）
- 支持字串变量 / 位置变量
- 支持复杂表达式 / 数学函数
- 支持项目变量命名 / 自定义局部变量

1.1 程序结构

KRL 程序由步（step）构成，如下图所示：

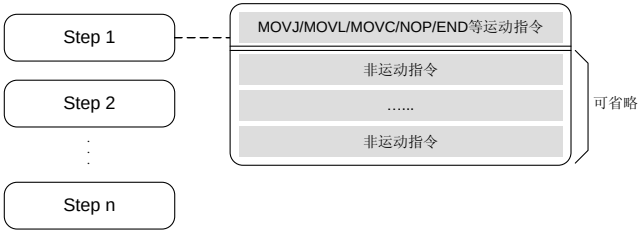


图1-1. KRL 程序结构图

- 步（step）以 1 条运动指令开头，之后可有 0 条或多条非运动指令
- 在运动指令执行的同时，可按时间 / 距离触发，启动并顺序执行步内的非运动指令
- 步（step）之间顺序执行

1.2 指令类型

表1-1. 指令类型表

指令类型	说明	例子
I/O 指令	读取或设置 I / O 端口。	DIN，DOUT，AIN，AOUT，PULSE
控制指令	控制程序执行流程。	JUMP，CALL，RET，PAUSE
运动指令	使机器人按关节 / 直线等方式移动。	MOVJ，MOVL，MOVC，NOP
运算指令	变量或常量的运算。	GETE，SETE，＋，－，×，/
设置指令	设置运行速度、工具、参考坐标系等。	SPEED，RAMP，REFSYS，TOOL
系统指令	获取/设置系统参数，产生报警等。	GETPRM，SETPRM，ALM

注：详请参考 3.1 指令一览表

1.3 常量

- 数字常量

表1-2. 数字常量表

类型	符号	例子	说明
32 位整数	S32	99	范围 -2^{31} ～ $2^{31}-1$
32 位正整数	U32	99U	范围 0 ～ $2^{32}-1$
64 位整数	S64	99L	范围 -2^{63} ～ $2^{63}-1$
64 位正整数	U64	99LU 99UL	范围 0 ～ $2^{64}-1$
双精度浮点数	DBL	99.9 9.99e1 .99e-2	支持科学计数法输入格式 除直接输入外，还可输入符号： M_E：常量 e M_PI：常量 π M_LN2：常量 ln2 值 M_LN10：常量 ln10 值 M_LOG2E：常量 $\log_2 e$ 值 M_LOG10E：常量 $\log_{10} e$ 值 M_SQRT2：常量 $\sqrt{2}$ M_SQRT3：常量 $\sqrt{3}$ M_INF：浮点 inf M_NAN：浮点 nan

- 字串常量

表1-3. 字符常量表

类型	举例	说明
字串	“This is a demo job” “运行到位置 1”	字串长度应不超过 127 字符，超过部分将被忽略，并提示警告。

- 位置常量

表1-4. 位置常量表

类型	举例	说明
位置	C0001 C1099	范围 C0000 ～ C9999 不支持通过指令初始化位置常量，或通过指令修改位置常量。只能由示教得到。

1.4 变量

1.4.1 项目变量

项目变量在项目内的所有 job 文件之间共享。

可按以下名字引用，或对变量命名后，按定义的变量名引用。

- 数字变量 （D0000 ~ D9999）
  可为 S32 / U32 / S64 / U64 / DBL 其中任一类型

- 字串变量 （S0000 ~ S9999）
  最长 127 个字符，支持运算符 = += + == !=

- 位置变量 （P0000 ~ P9999）
  支持运算符 = += -= + - == !=

1.4.2 局部变量

局部变量仅在当前 job 文件内有效。

预定义局部变量：直接按以下名字引用。

- 数字变量 （LD0000 ~ LD9999）
- 字串变量 （LS0000 ~ LS9999）
- 位置变量 （LP0000 ~ LP9999）

自定义局部变量 ：按定义的变量名引用。自动分配空间，与预定义的局部变量不重叠。

1.4.3 系统变量

系统变量在所有 job 文件之间共享。

只能按以下名字引用。其内容为只读，用于控制器向外输出运行信息，不能修改。

- 数字变量 （$D0000 ~ $D9999）
- 字串变量 （$S0000 ~ $S9999）
- 位置变量 （$P0000 ~ $P9999）

1.5 运算符

1.5.1 位操作

表1-5. 位操作表

符号	说明
&	按位与
^	按位异或
|	按位或
<<	左移位
>>	右移位
~	按位取反

1.5.2 算术

表1-6. 算术运算表

符号	说明
+	加
-	减
*	乘
/	除
%	模

1.5.3 逻辑

表1-7. 逻辑运算表

符号	说明
&&	逻辑与
||	逻辑或
!	逻辑非

1.5.4 关系

表1-8. 关系运算表

符号	说明
>	大于
>=	大于等于
<	小于

<=	小于等于
==	等于
!=	不等于

1.5.5 赋值

表1-9. 赋值运算表

符号	说明
=	赋值
+=	a += b <=> a = a + b
-=	a -= b <=> a = a - b
*=	a *= b <=> a = a * b
/=	a /= b <=> a = a / b
%=	a %= b <=> a = a % b
<<=	a <<= b <=> a = a << b
>>=	a >>= b <=> a = a >> b
&=	a &= b <=> a = a & b
|=	a |= b <=> a = a | b
^=	a ^= b <=> a = a ^ b
++	a ++ <=> a = a + 1
--	a -- <=> a = a - 1

1.5.6 类型转换

表1-10. 类型转换运算表

符号	说明
(U32)	转换为 32 位正整数
(S32)	转换为 32 位整数
(U64)	转换为 64 位正整数
(S64)	转换为 64 位整数
(DBL)	转换为双精度浮点数

1.6 标签

表1-11. 语句标签表

<table>
<tr><th>标签</th><th>功能</th><th>格式及说明</th><th>作用域</th></tr>
<tr><td>IN#</td><td>1 位输入端口</td><td>IN#(p.b)。

p 取值范围为：
<table>
<tr><td>起始地址</td><td>10000</td><td>30000</td><td>5x000</td></tr>
<tr><td>结束地址</td><td>10255</td><td>34095</td><td>5x999</td></tr>
<tr><td>助记符号</td><td>X</td><td>G</td><td>RV</td></tr>
</table>
b 取值范围为 0-7。

IN#(10012.1)：地址 10012 的 bit1 输入端口信号。
IN#(X2.0)：地址 10002 的 bit0 输入端口信号。</td><td>DIN
WAITIO
if condition
until condition</td></tr>
</table>

<table>
<tr>
<td>OT#</td>
<td>1 位输出端口</td>
<td>OT#(p.b)

p 取值范围为：
<table>
<tr><td>起始地址</td><td>20000</td><td>40000</td><td>6x000</td></tr>
<tr><td>结束地址</td><td>20255</td><td>44095</td><td>6x999</td></tr>
<tr><td>助记符号</td><td>Y</td><td>F</td><td>TS</td></tr>
</table>
b 取值范围为 0-7。

OT#(20001.2)：地址 20001 的 bit2 输入端口信号。
OT#(Y3.4)：地址 20003 的 bit4 输入端口信号。</td>
<td>DIN
DOUT
PULSE
WAITIO</td>
</tr>
<tr>
<td>IG#</td>
<td>8 位输入端口</td>
<td>IG#(p)

p 取值范围为：
<table>
<tr><td>起始地址</td><td>10000</td><td>30000</td><td>5x000</td></tr>
<tr><td>结束地址</td><td>10255</td><td>34095</td><td>5x999</td></tr>
<tr><td>助记符号</td><td>X</td><td>G</td><td>RV</td></tr>
</table>
IG#(50012)：地址 50012 的输入端口信号（共 8 位）。
IG#(G2)：地址 30002 的输入端口信号（共 8 位）。</td>
<td>DIN
JUMP
CALL
WAITIO
if condition</td>
</tr>
</table>

<table>
<tr>
<td>OG#</td>
<td>8 位输出端口</td>
<td>OG#(p)

p 取值范围为：
<table>
<tr><td>起始地址</td><td>20000</td><td>40000</td><td>6x000</td></tr>
<tr><td>结束地址</td><td>20255</td><td>44095</td><td>6x999</td></tr>
<tr><td>助记符号</td><td>Y</td><td>F</td><td>TS</td></tr>
</table>
OG#(60001)：地址 60001 的输入端口信号（共 8 位）。
OG#(F3)：地址 40003 的输入端口信号（共 8 位）。</td>
<td>DIN
DOUT
PULSE
WAITIO</td>
</tr>
<tr>
<td>AI#</td>
<td>模拟量输入端口</td>
<td>AI#(n)

n 取值范围为 0-63。

AI#(2)表示模拟输入端口 2。</td>
<td>AIN</td>
</tr>
<tr>
<td>AO#</td>
<td>模拟量输出端口</td>
<td>AO#(n)

n 取值范围为 0-63。

AO#(5)表示模拟输出端口 5。</td>
<td>AIN
AOUT</td>
</tr>
<tr>
<td>ON</td>
<td>端口状态</td>
<td>端口为 1。</td>
<td rowspan="2">DOUT
WAITIO
if condition
until
condition</td>
</tr>
<tr>
<td>OFF</td>
<td>端口状态</td>
<td>端口为 0。</td>
</tr>
</table>

INVERT	反转标记	当前信号为 ON 时，反转为 OFF。 当前信号为 OFF 时，反转为 ON。	DOUT PULSE
T	时间	例如：T = 0.1 表示时间为 0.1 秒。	PULSE TIMER
LABEL	跳转标记	例如：LABEL = 31 表示 L31。	JUMP
PRMNO	参数号标记	例如：PRMNO = 20 表示参数 P20。	GETPRM SETPRM
MAXV	最大值标记	参数最大值。	
MINV	最小值标记	参数最小值。	
DFTV	默认值标记	参数默认值。	
VJ	关节速度	例如：VJ = 50 表示关节速度为 50%。	MOVJ
V	工具位移速度	例如：V = 200 表示工具位移速度为 200mm/s。	MOVL MOVC
VR	工具旋转速度	例如：VR = 30 表示工具旋转速度为 30deg/s。	
VE	外部轴速度	例如：VE = 60 表示外部轴速度为 60%。	
PL	定位级别	例如：PL= 3 表示定位级别为 3。 默认时，值越小，越接近两条运动轨迹的转接点。	MOVJ MOVL MOVC
ACC	加速度调整	例如：ACC = 50 表示加速度为最大加速度的 50%。	

DEC	减速度调整	例如：DEC = 50 表示减速度为最大加速度的 50%。	
CST	圆弧起点	当连续的 MOVC 指令时，用 CST 标识圆弧起点。	MOVC
IF	条件标记	例如：IF (D1 == 1) 表示 D1 为 1 时条件满足。	if condition
IMOVE	增量移动	有此标记时运动段按增量移动，对应的位置输入应是增量值。	MOVJ MOVL
POLY0	加减速曲线标记	直线加减速。	RAMP
POLY1		多项式加减速。	
SINE0		三角函数加减速 0。	
SINE1		三角函数加减速 1。	
ST0	非运动指令执行条件	距离起点时间 （秒或%）。	MOVJ MOVL MOVC
ST1		距离终点时间 （秒或%）。	
SD0		距离起点距离 （米或%）。	
SD1		距离终点距离 （米或%）。	

1.7 注释

单行注释，例如：

// We should arrive at this point

MOVL C0009 **// 直线回到上一点(C0009)**

2 指令说明

指令结构图示例

如下图示例，指令结构图由【名称】、【说明序号】、【标签】、【模块】、【连接符】和【连接线】组成。当指令结构分支较多时，采用分块图描述。

结 构

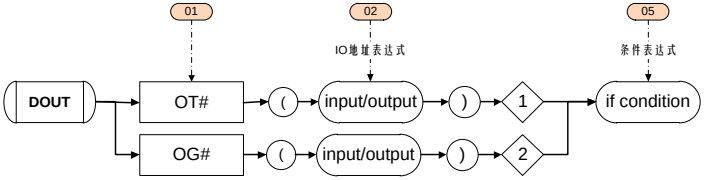


图2-1. 指令结构图

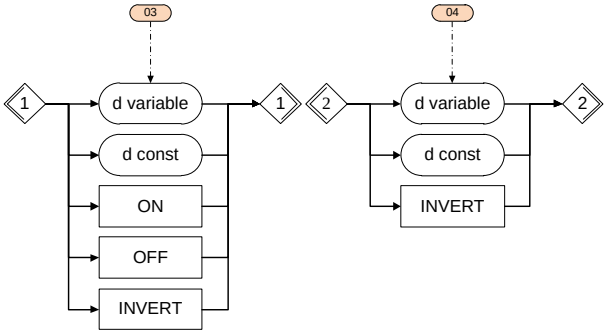
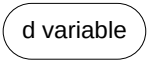


图2-2. 分块图

表2-1. 指令结构图元素定义表

图形	说明	备注
	指令名称	
	说明序号	按顺序编号。
	标签	字母大写， 请参考章节 1.6 标签。
	模块	字母小写， 请参考章节 2.7 模块。
	连接符(	包括()＝等。
	连接线	表明顺序和分支连接关系。
	分块	使用分块。
	分块开始	定义分块。
	分块结束	

2.1 IO 指令

2.1.1 DIN

功 能

获取 IO 端口状态。

结 构

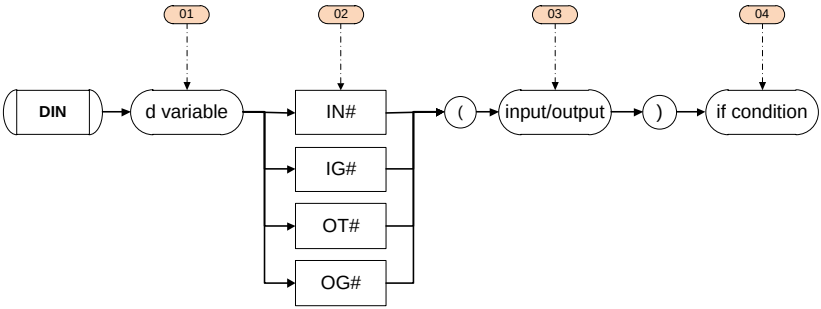


图2-3. DIN 指令结构图

说 明

获取指定 IO 地址端口的状态，存储到变量。

表2-2. DIN 指令说明表

序号	说明
01	d variable 用于存储 IO 端口状态值。
02	IN#： 1 位输入端子。 IG#： 8 位输入端子。 OT#：1 位输出端子。 OG#：8 位输出端子。
03	input/output 指定 IO 地址号， 地址应符合 IN# / IG# / OT# / OG#的设置。
04	if condition 设置执行条件，条件满足时执行。

例 子

（1） DIN D16 IN#(X1.4)

将输入信号端口号 X1.4 的状态值（开/关=1/0），存储在变量 D16 中。

（2） D2=10020

DIN LD1 IG(D2)

先对 D2 赋值 10020，之后将 D2 指定的输入信号端口组号 10020（端口 10020.0~10020.7）的状态值（开/关=1/0），存储在变量 LD1 中。

（3） DIN D10 OT#(Y1.2)

将输出信号端口号 Y1.2 的状态值（开/关=1/0），存储在变量 D10 中。

（4） DIN D12 OG#(Y2)

将输出信号端口组号 Y2（即端口 20002.0~20002.7）的状态值（开/关=1/0），存储在变量 D12 中。

2.1.2 **DOUT**

功 能

设置输出端口状态。

结 构

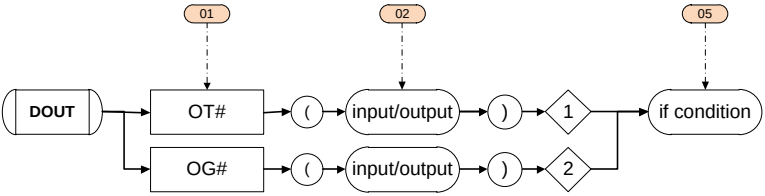


图2-4. DOUT 指令结构图

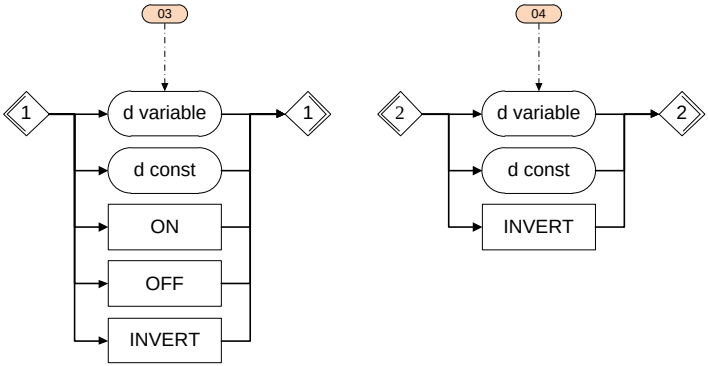


图2-5. DOUT 指令分块图

说 明

向指定地址的 1 位或 8 位端口输出指定的状态。

表2-3. DOUT 指令说明表

序号	说明
01	OT#：输出 1 位端子。 OG#：输出 8 位端子。
02	指定输出地址号， 地址应符合 OT# / OG#设置。
03	设置输出 1 位的状态值。 d variable / d const：非 0 时为状态 1，是 0 时为状态 0。 ON / OFF：1 / 0。 INVERT：取反先前状态。
04	设置输出 8 位的状态值。 d variable / d const：低 8 位有效。 INVERT：按位取反再输出。
05	if condition 设置执行条件，条件满足时执行。

例 子

（1） D1=1

DOUT OT#(Y2.1) D1

设置 Y2.1 端口为开启。

（2） DOUT OT#(Y2.1) ON

设置 Y2.1 端口为开启。

（3） DOUT OT#(Y2.1) 0

设置 Y2.1 端口为关闭。

（4） DOUT OT#(20002.1) OFF

设置输出端口 20002.1 为关闭。

（5） DOUT OT#(20002.1) INVERT

反转输出端口 20002.1。若当前为 ON 时，则反转为 OFF；当前为 OFF 时，则反转为 ON。

（6） D0=24

DOUT OG#(TS3) D0

设置输出端口 TS3 的状态为变量 D0 值。

2.1.3 **AIN**

功 能

获取模拟量端口电压值，并保存到指定的变量中。

结 构

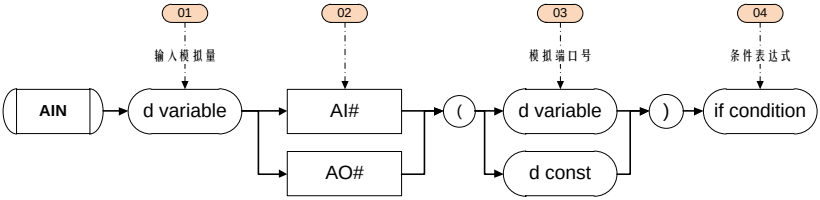


图2-6. AIN 指令结构图

说 明

获取指定的模拟端口的电压数值，存储到指定的 D 变量。

表2-4. AIN 指令说明表

序号	说明
01	用于存储模拟端口值的 D 变量地址。
02	AI#：模拟量输入端口。

	AO#：模拟量输出端口。
03	指定模拟地址号， 地址应符合 AI#、AO#的设置。
04	if condition 设置执行条件，条件满足时执行。

例 子

（1） AIN D5 AI#(1)

将模拟输入端口 1 中模拟电压值赋值到变量 D5 中。

（2） D6=3

AIN D5 AI#(D6)

将模拟输入端口 3 中模拟电压值赋值到变量 D5 中。

（3） AIN D5 AO#(1)

将模拟输出端口 1 中模拟电压值赋值到变量 D5 中。

（4） D6=3

AIN D5 AO#(D6)

将模拟输出端口 3 中模拟电压值赋值到变量 D5 中。

2.1.4 **AOUT**

功 能

将电压值输出至模拟量输出端口。

结 构

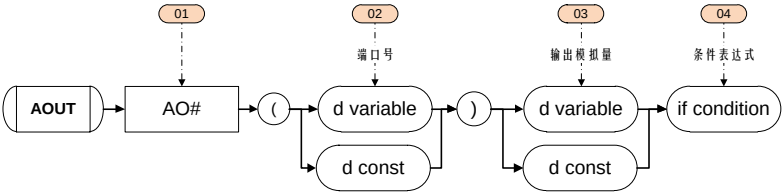


图2-7. 指令结构图

说 明

表2-5. AOUT 指令说明表

序号	说明
01	AO#：模拟量输出端口。
02	d variable / d const：指定模拟输出地址号， 地址应符合 AO#设置。
03	d variable / d const：设置输出模拟量值。模拟电压单位为伏特（V）。

04	if condition 设置执行条件，条件满足时执行。

例 子

（1） D3=2.7

AOUT AO#(1) D3

由模拟输出端口 1 输出电压值 2.7V。

（2） D3=3.3

LD2=3

AOUT AO#(LD2) D3

由模拟输出端口 3 输出电压值 3.3V。

2.1.5 **PULSE**

功 能

将脉冲信号输出到输出端口中。

结 构

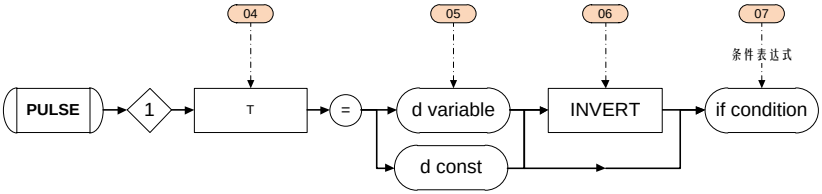


图2-8. PULSE 指令结构图

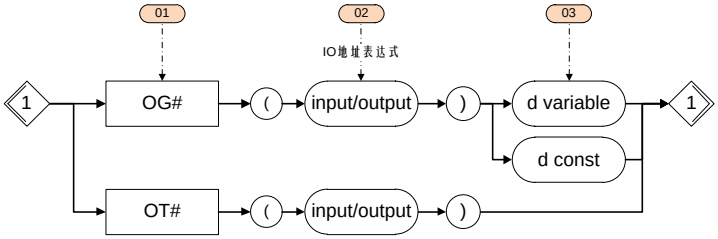


图2-9. PULSE 分块图

说 明

将脉冲信号输出到指定的输出信号端口（组）中。

表2-6. PULSE 指令说明表

序号	说明
01	OT#：输出 1 位端子。 OG#：输出 8 位端子。
02	input/output 指定输出地址号， 地址应符合 OT# / OG#设置。
03	按组输出时的屏蔽位。 d variable / d const：非 0 时为对应位端口输出，否则不输出。
04	T 时间标签。
05	d variable / d const：指定输出的脉冲宽度，单位为秒（s）。
06	INVERT：负脉冲（若无 INVERT 则为正脉冲）。
07	if condition 设置执行条件，条件满足时执行。

例 子

（1） PULSE OT#(20000.0) T=1

脉冲信号输出至 20000.0 端口，脉冲宽度为 1s。

（2） D3=20004.0

PULSE OT#(D3) T=1.5

脉冲信号输出至 20004.0 端口，脉冲宽度为 1.5s。

（3） D001=48

PULSE OG#(20005) D001 T=5.0

D001=48（十进制）=00110000（二进制），故在端口 20005.4 和 20005.5 输出脉冲，脉冲宽度为 5s。

表2-1. PULSE 示例表

Bit7	Bit6	**Bit5**	**Bit4**	**Bit3**	Bit2	Bit1	Bit0
OG#(20005)							

2.2 控制指令

2.2.1 JUMP

功 能

跳转至指定的标号（LABEL 或 L0~L9999）。

结 构

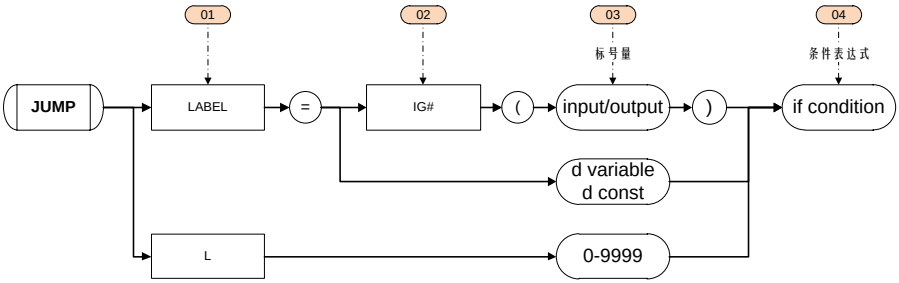


图2-10. JUMP 指令结构图

说 明

JUMP 指令跳转程序运行到指定的标号处运行。

表2-7. JUMP 指令说明表

序号	说明
01	L：通过常量指定标号。 LABEL：通过端口或 D 变量/常量指定标号。
02	IG#：8 位输入端口（组）。
03	【01】= L 时：指定 0-9999 标号。 【01】= LABEL 时：通过端口 input/output 或 d variable / d const 指定标号。
04	if condition 设置执行条件，条件满足时执行。

例 子

（1） JUMP L0003

跳转至 L0003 标号处

（2） D0000 =1

JUMP LABEL = D0000 IF IN#(10020.1) == ON

如果输入信号 10020.1 位为 ON 时，程序跳转至 L01 标号处。

（3）
```
L0：
JUMP L1 IF IN#(D0) == D2
MOVJ C0003 VJ=10
L1：
MOVL C0029 V=200
JUMP L0
END
```

当程序运行到 JUMP L1 处时，进行条件判断，如果 IN#(D0) == D2，则

跳转到 L1 处，否则继续往下执行 MOVJ C0003 VJ=10，执行到 JUMP L0 时，跳转到标号 L0 处。

2.2.2 **CALL**

功 能

调用指定的任务。

结 构

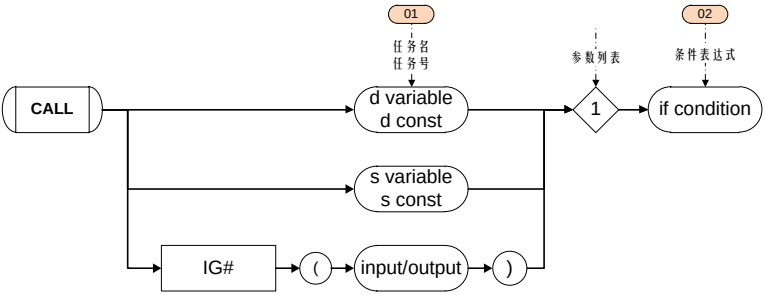


图2-11. 指令结构图

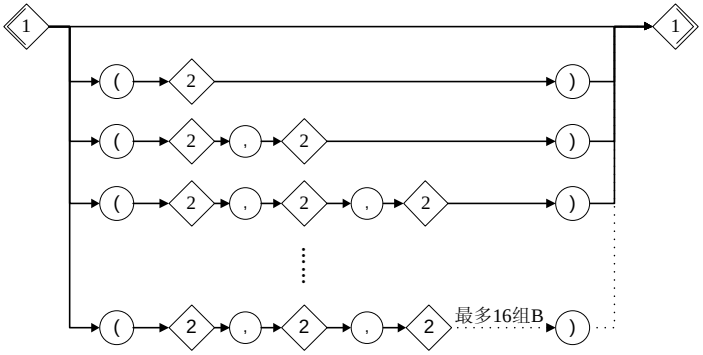


图2-12. CALL 指令分块图

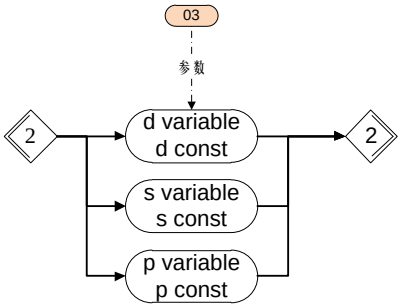


图2-13. CALL 指令参数分块图

说 明

CALL 指令通过指定的任务号或任务名调用子任务（JOB）。可以通过 1 指定参数列表传递到子任务。最多可以传递 16 个 2 限定的参数。

表2-8. CALL 指令说明表

序号	说明
01	任务号可以通过 d variable / d const 指定。 任务号也可以通过 input/output 8 位端口号指定，指定 0~255 号子任务。 任务名通过 s variable / s const 指定。
02	if condition 设置执行条件，条件满足时执行。
03	参数可设置为 d，s，p 变量或常量。

例 子

（1） CALL “TEST1”

调用程序名为 TEST1 的任务。

（2） D0000=1

CALL D0000 IF IN#(10000.0) == ON

如果输入信号端口 10000.0 为 ON，则调用程序号为 1 的任务。

（3） 对比有无参数调用

无参数调用

```
<Calling job>
NOP
D0=1
D1=5
D2=7
CALL “WAITIO INPUT”
```

```
<Called job:WAITIO INPUT>
WAITIO IN#(D0) ON
WAITIO IN#(D1) ON
WAITIO IN#(D2) ON
RET
```

有参数调用

```
<Calling job>
NOP
CALL “WAITIO INPUT”(1,5,7)
```

```
<Calledjob:WAITIO INPUT>
GETA LD0 (1)
GETA LD1(5)
GETA LD2(7)
WAITIO IN#(LD0) ON
WAITIO IN#(LD1) ON
WAITIO IN#(LD2) ON
RET
```

图2-14. CALL 参数例子图

无参数调用时用到了项目变量，而项目变量在项目内各任务间共享，容易造成误使用；有参数调用时只使用局部变量。

2.2.3 RET

功 能

返回调用任务。

任务结束前，将值保存在系统返回值存储区，以便调用任务获取。若该任务是被调用的任务，则返回调用任务的 CALL 指令下一条指令继续运行；否则返回本任务第一行继续运行。

结 构

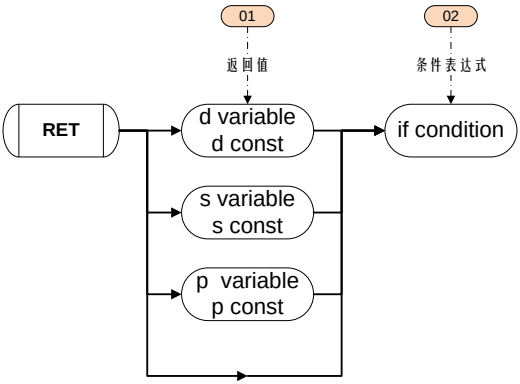


图2-15. RET 指令结构图

说 明

RET 指令执行后，将本程序中的 D、S、P 变量和常量保存在系统返回值存储区，可通过 GETR 指令来读取。

表2-9. RET 指令说明表

序号	说明
01	返回值。 可设置为 d，s，p 变量或常量。 或无返回值。
02	if condition 设置执行条件，条件满足时执行。

例 子

（1） RET “Calling job” IF IN#(10002) == ON

如果输入信号 10002 最低位为 ON，则返回调用的任务，返回值为字符串。

（2） 返回数值到主程序对应变量中。

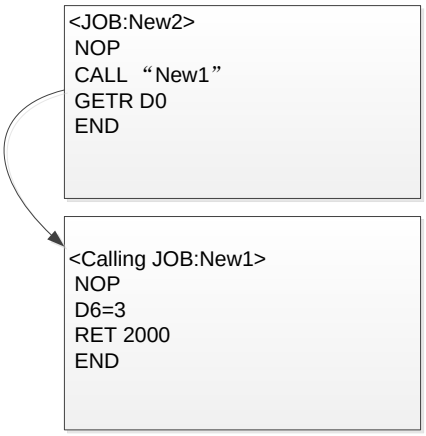


图2-16. RET 数值到变量

在程序“New2”中调用程序“New1”，当程序“New1”运行结束后，返回程序“New2”继续运行，并将数值 2000 赋值到变量 D0 中。

2.2.4 **PAUSE**

功 能

暂停任务，等待任务再次启动。

结 构

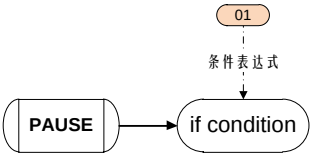


图2-17. PAUSE 指令结构图

说 明

表2-10. PAUSE 指令说明表

序号	说明
01	if condition 设置执行条件，条件满足时执行。

例 子

（1） MOVJ C0000 VJ=20
PAUSE
MOVL C0001 V=200
执行到 PAUSE 时，暂停任务，当再次启动执行时，程序继续。

（2） PAUSE IF IN#(10002.0) == ON
若输入端口 10002.0 为 ON 状态，则暂停任务，当再次启动执行时，程序继续。

（3） D0016=12
MOVJ C0000 VJ=20
PAUSE IF (D0016==12)
MOVL C0001 V=200
RET
END
执行到 PAUSE 行时若 D0016 为 12，则暂停任务，当再次启动执行时，程序继续。

2.2.5 **ABORT**

功 能

停止执行任务，并返回任务首行。

结 构

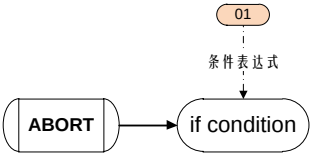


图2-18. 指令结构图

说 明

表2-11. ABORT 指令说明表

序号	说明
01	if condition 设置执行条件，条件满足时执行。

例 子

（1） ABORT

机器人停止运行，并返回程序首行。

（2） ABORT IF (D0000==1)

当 D0000 等于 1 时，机器人停止运行，并返回程序首行。

（3） D0001=10003.6

ABORT IF IN#(D0001)==ON

当 10003.6 位为 ON 状态时，机器人停止运行，并返回程序首行。

（4） ABORT 添加 IF 语句 IG#格式

a) D0001=10003

D0002=1

ABORT IF IG#(D0001) == D0002

当端口 10003 的值等于 1 时，机器人停止运行，并返回程序首行。

b) D0001=10003

D0002=1

ABORT IF IG#(D0001) != D0002

当端口 10003 的值不等于 1 时，机器人停止运行，并返回程序首行。

2.2.6 **WAITIO**

功 能

等待 I/O 信号为指定的状态。

结 构

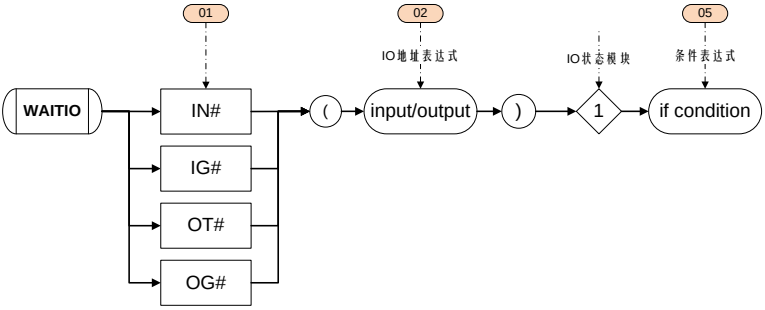


图2-19. WAITIO 指令结构图

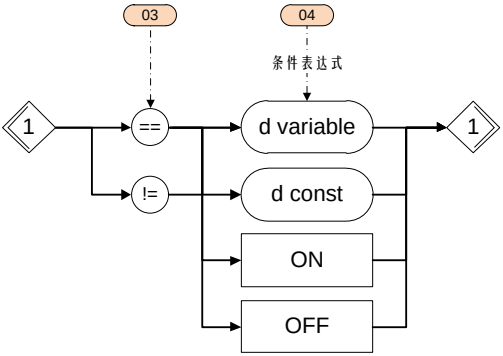


图2-20. WAITIO 分块图

说 明

等待直至指定的 IO 端口为指定的状态。

表2-1. WAITIO 指令说明表

序号	说明
01	IN#：1 位输入端子。 IG#：8 位输入端子。 OT#：1 位输出端子。 OG#：8 位输出端子。
02	input/output 指定 IO 地址， 地址应符合 IN# / IG# / OT# / OG#的设置。
03	【==】 ：等待的状态等于 04 指定的值。 【!=】 ：等待的状态不等于 04 指定的值。
04	IO 的目标状态。

	d variable / d const：按位非 0 时为状态 1，是 0 时为状态 0。 ON / OFF：1 / 0。
05	if condition 设置执行条件，条件满足时执行。

例 子

（1） WAITIO IN#(10000.0) == ON

程序等待，直到端口 10000.0 变为 ON 状态后，程序继续执行。

（2） WAITIO IN#(10000.0) != ON

程序等待，直到端口 10000.0 不是 ON 状态后，程序继续执行。

（3） D0000 = 10002

D0001 = 16

WAITIO IG#(D0000) == D0001

程序等待，直到端口 10002.0~10002.7 状态排列顺序为 00010000。

（4） WAITIO OT#(20000.0) == ON

程序等待，直到端口 20000.0 变为 ON 状态后，程序继续执行。

（5） WAITIO OT#(20000.0) == OFF

程序等待，直到端口 20000.0 变为 OFF 状态后，程序继续执行。

（6） WAITIO OT#(20000.0) != OFF

程序等待，直到端口 20000.0 不是 OFF 状态后，程序继续执行。

2.2.7 WAITMTN

功 能

等待当前步（step）的运动指令运行结束后，继续执行。

结 构

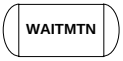


图2-21. WAITMTN 指令结构图

说 明

由于运动指令和非运动指令可以并行执行，当非运动指令需要等待运动指令完成时，使用 WAITMTN 指令。

例 子

（1） 在 step n 运行至 step n+1 的过程中，使输出信号 20000.0 设为 ON 状，等待 step n 运动指令运行结束，之后使输出信号 20000.0 设为 OFF 状态。

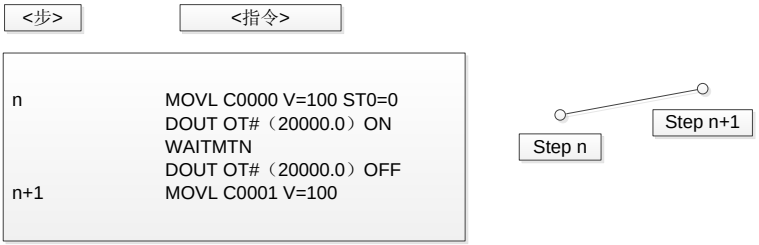


图2-22. WAITMTN 示例图

2.2.8 **TIMER**

功 能

等待指定的时间。

结 构

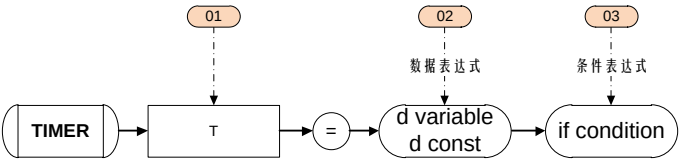


图2-23. TIMER 指令结构图

说 明

等待指定的时间后再继续执行下一指令。

表2-12. TIMER 指令说明表

序号	说明
01	T：时间标签。
02	等待时间，单位为秒（s）。

03	if condition 设置执行条件，条件满足时执行。

例 子

（1） TIMER T = 12.5

等待 12.5 秒。

（2） D0000 = 5

TIMER T = D0000

等待 5 秒。

2.2.9 GETA

功 能

在被调用任务中，GETA 用于获取主任务 CALL 指令中传递的参数。当 GETA 指令执行时，CALL 指令中的参数将被检索，并保存在 GETA 指定的变量中。

结 构

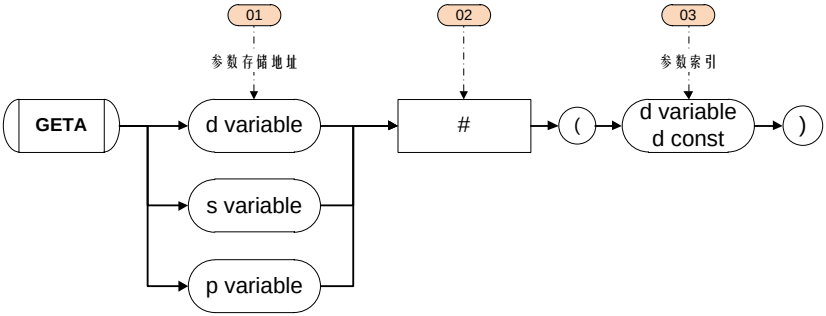


图2-24. GETA 指令结构图

说 明

应在子任务开始时就获取并保存传递的参数，以避免复杂的多级子任务调用时，传递的参数被意外改写。

表2-13. GETA 指令说明表

序号	说明
01	指定的变量。 d variable：指定 D 变量。 s variable：指定 S 变量。 p variable：指定 P 变量。 变量类型（D，S，P）必须与 CALL 指令提供的参数类型一致。
02	【#】标签。
03	指定的参数索引。 必须在 CALL 指定的参数数量范围内，否则报警。

例 子

（1） 使用 D_VAR 保存位置

Calling Job

```
NOP
MOVJ C0001 VJ=100
MOVJ C0002 VJ=50
CALL “test1”(8)
MOVL C0003 V=125
END
```

Called Job：test1

```
NOP
GETA LD0000 #(1)
LD0000 *= 10
AOUT AO#(1) LD0000
END
```

（2） 使用 S_VAR 保存位置

Calling Job

```
NOP
MOVJ C0001 VJ=100
MOVJ C0002 VJ=50
CALL “test1”(“beijing”)
```

Called Job：test1

```
NOP
GETA LS0000 #(1)
ALM 3001 LS0000
END
```

```
MOVL C0003 V=125
END
```

（3） 使用 P_VAR 保存位置

Calling Job

```
NOP
MOVJ C0001 VJ=10
MOVJ C0002 VJ=50
CALL “test1”(P0001)
MOVL C0003 V=125
END
```

Called Job：test1

```
NOP
GETA LP0000 #(1)
MOVJ LP0000 VJ=100
END
```

（4） 同时使用多种数据格式保存位置

Calling Job

```
NOP
MOVJ C0001 VJ=100
MOVJ C0002 VJ=50
CALL “test1”(3，“beijing”，C0003，P0003，S0200，D0030)
MOVL C0003 V=125
END
```

Called Job：test1

```
NOP
GETA LD0007 #(1)   //将数值 3 保存在局部变量 LD0007 中
GETA LP0000 #(4)   //将 P0003 的变量保存在局部变量 LP0000 中
GETA LS0000 #(2)   //将字符串“beijing”保存在局部变量 LS0000 中
GETA LP0006 #(3)   //将 C0003 的变量保存在局部变量 LP0006 中
GETA LS0040 #(5)   //将 S0200 的字符串保存在局部变量 LS0040 中
GETA LD0008 #(6)   //将 D0030 的数值保存在局部变量 LD0008 中
```

2.2.10 GETR

功 能

获取 RET 指令返回值。

结 构

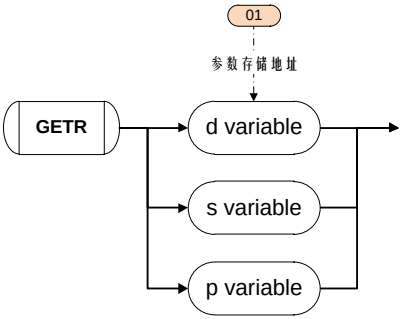


图2-25. GETA 指令结构图

说 明

当 GETR 指令执行时，被调用任务中 RET 指令中返回值将保存在指定的变量中。

若 RET 指令没有指定返回值，请勿使用 GETR 指令，否则返回值不可预期。

应在 CALL 指令后立即获取 RET 的返回值，以免多 CALL 指令调用时，返回

值被改写。

表2-14. GETR 指令说明表

序号	说明
01	指定的变量。 d variable：指定 D 变量 s variable：指定 S 变量 p variable：指定 P 变量 变量类型（D，S，P）必须与 RET 指令的返回类型一致，否则报警。

例 子

（1） 获取 D_VAR 返回值

a) **Calling Job**

```
NOP
MOVJ C0001 VJ=100
CALL “test1”
GETR LD0004
MOVL C0003 V=125
END
```

Called Job：test1

```
NOP
MOVL C0002 V=100
MOVL C0003 V=200
MOVL C0004 V=200
RET D0007
END
```

b) **Calling Job**

```
NOP
MOVJ C0001 VJ=100
CALL “test1”
GETR LD0004
```

Called Job：test1

```
NOP
MOVL C0002 V=100
MOVL C0003 V=200
MOVL C0004 V=200
```

MOVL C0003 V=125	RET 200
END	END

（2） 获取 S_VAR 返回值

a) **Calling Job**

```
NOP
MOVJ C0001 VJ=100
CALL “test1”
GETR LS0004
MOVL C0003 V=125
END
```

Called Job：test1

```
NOP
MOVL C0002 V=100
MOVL C0003 V=200
MOVL C0004 V=200
RET S0007
END
```

b) **Calling Job**

```
NOP
MOVJ C0001 VJ=100
CALL “test1”
GETR LS0004
MOVL C0003 V=125
END
```

Called Job：test1

```
NOP
MOVL C0002 V=100
MOVL C0003 V=200
MOVL C0004 V=200
RET “test string”
END
```

（3） 获取 P_VAR 返回值

a) **Calling Job**

```
NOP
MOVJ C0001 VJ=100
CALL “test1”
GETR LP0004
MOVL C0003 V=125
END
```

Called Job：test1

```
NOP
MOVL C0002 V=100
MOVL C0003 V=200
MOVL C0004 V=200
RET P0007
END
```

b) **Calling Job**

```
NOP
MOVJ C0001 VJ=100
CALL “test1”
GETR LP0004
MOVL C0003 V=125
END
```

Called Job：test1

```
NOP
MOVL C0002 V=100
MOVL C0003 V=200
MOVL C0004 V=200
RET C0002
END
```

2.2.11 **LABEL**

功 能

跳转位置标号定义。

结 构

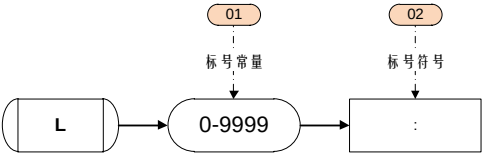


图2-26. LABEL 指令结构图

说 明

跳转位置标号定义，必须指定 0~9999 常量标号。

提示：L1，L01，L001，L0001 等价。L00001 超出四位标号限制，错误语法。

表2-15. LABEL 指令说明表

序号	说明
01	指定的标号常量 0~9999。与 L 之间没有空白字符。
02	【:】标签。

例 子

（1） L1：

MOVJ C0002 VJ=20

MOVL C0099 V=200

JUMP L1

当程序运行到 JUMP L1 时，程序跳回 L1 处，继续执行。

（2） L9999：

JUMP L9998 IF IN#(D0) == D2

MOVJ C0003 VJ=10

L9998：

MOVL C0029 V=200

JUMP L9999

END

当程序运行到 JUMP L9998 时，进行条件判断，如果 IN#(D0) == D2，则跳转到 L9998 处，如果条件不成立则继续往下执行。执行到 JUMP L9999 时，跳转到 L9999 处，继续执行程序。

2.3 运动指令

2.3.1 NOP

功 能

空操作。

结 构

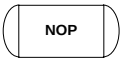


图2-27. NOP 指令结构图

说 明

空操作指令通常用在程序开始。

编程设计时，某些未实现详细设计的模块也可以先用 NOP 占用一个语句行。

例 子

（1） NOP
D2=122

```
1+2
SIN (90)
MOVL C0001 V=100
END
```

NOP 指令可以单独作为一步（step），也可以与非运动指令一起作为一步（step）。

2.3.2 END

功 能

程序结束。

结 构

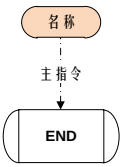


图2-28. END 指令结构图

说 明

END 指令结束本任务，END 之后的指令将无效。

例 子

（1） NOP

…

END

程序运行至 END 处时，结束运行。

2.3.3 MOVJ

功 能

以关节插补方式向目标位置移动。

结 构

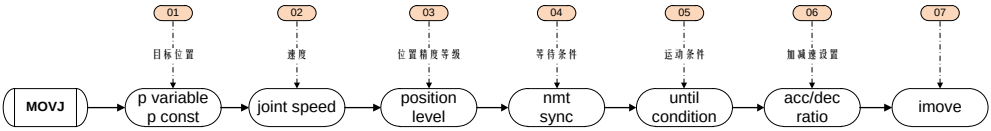


图2-29. MOVJ 指令结构图

说 明

参考 章节 2.3.6 运动指令模块。

表2-16. MOVJ 指令说明表

序号	说明
01	目标位置。
02	关节速度。

03	位置精度等级。
04	非运动指令执行条件.
05	运动结束条件。
06	加减速百分比设置。
07	增量运动。

例 子

（1） NOP

MOVJ C0002 VJ=50

从起始位置到 Step 1 之间的曲线为 MOVJ 指令的运动轨迹，执行过程中 VJ 速度为 50%，如下图所示：

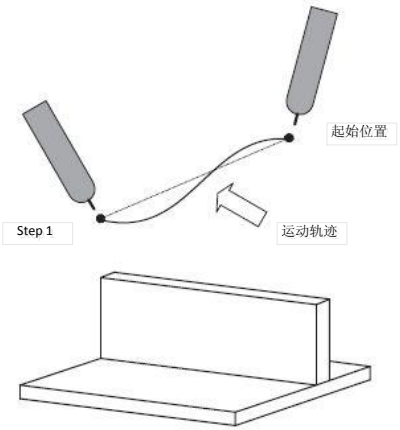


图2-30. MOVJ 运动轨迹例子

2.3.4 MOVL

功 能

以直线插补方式向目标位置移动。

结 构

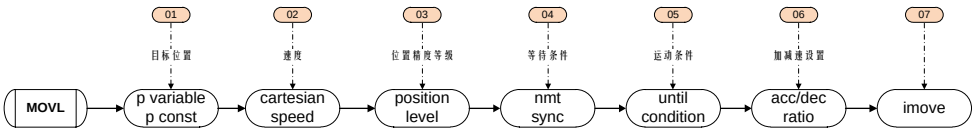


图2-31. MOVL 指令结构图

说 明

参考章节 2.3.6 运动指令模块。

表2-17. MOVL 指令说明表

序号	说明
01	目标位置。
02	速度。
03	位置精度等级。

04	非运动指令执行条件。
05	运动结束条件。
06	加减速百分比设置。
07	增量运动。

例 子

（1） NOP

```
MOVJ C0000 VJ=50          // Step 1
MOVJ C0001 VJ=25          // Step 2
MOVJ C0002 VJ=12.5        // Step 3
MOVL C0003 V=50           // Step 4
```

C0002 到 C0003 之间的运动方式为直线运动，运行速度为：50mm/s，如图所示：

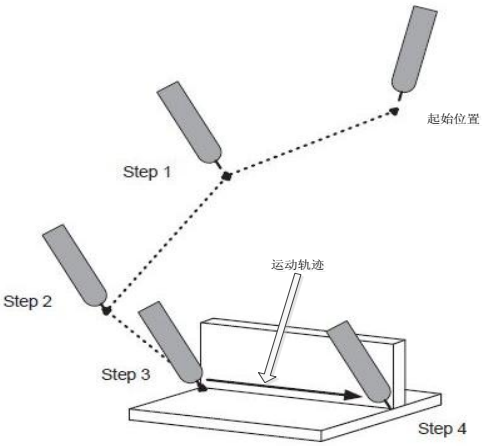


图2-32. MOVL 运行轨迹例子图

2.3.5 MOVC

功 能

以圆弧插补方式向目标位置移动。

结 构

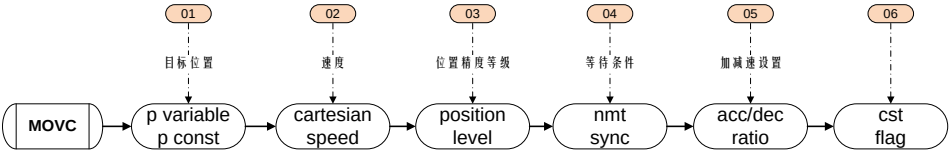


图2-33. MOVC 指令结构图

说 明

指令说明表如下，另请参考 章节 2.3.6 运动指令模块。

表2-18. MOVC 指令说明表

序号	说明
01	目标位置。
02	速度。
03	位置精度等级。

04	非运动指令执行条件。
05	加减速百分比设置。
06	下一个圆弧起点标记。

例 子

（1） 单一圆弧运动

```
0001  NOP
0002  MOVL C0000 V=100
0003  MOVL C0001 V=100
0004  MOVC C0002 V=100
0005  MOVC C0003 V=100
0006  MOVC C0004 V=100
0007  MOVC C0005 V=100
0008  END
```

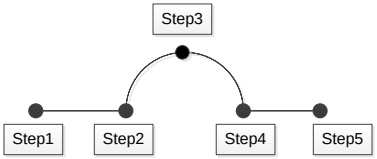


图2-34. MOVC 单一圆弧实例

如图所示，当单一圆弧时，示教 Step2 至 Step4 的 3 个点。Step1 至 Step2 的轨迹自动为直线。

（2） 连续圆弧运动

```
NOP
MOVL C0000 V=120                //Step1
MOVC C0001 V=120                //Step2
MOVC C0002 V=120                //Step3
MOVC C0003 V=120 CST            //Step4
MOVC C0004 V=120                //Step5
MOVC C0005 V=120                //Step6
END
```

连续圆弧时，在前一圆弧的第 3 个 MOVC 指令增加 CST 标签，使得这一 MOVC 同时作为下一圆弧的第 1 个 MOVC。

2.3.6 运动指令模块

2.3.6.1. cartesian speed

结 构

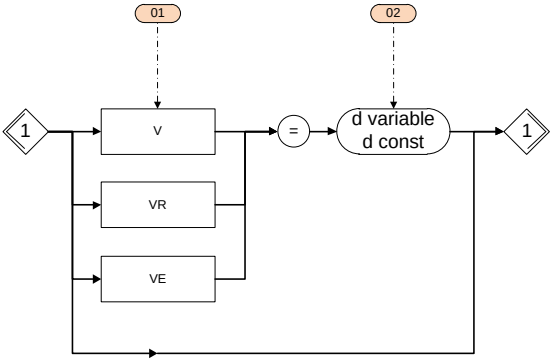


图2-35. cartesian speed 模块结构图

说 明

表2-19. cartesian speed 模块说明表

序号	说明
01	V： 工具移动速度 （单位 mm/s）。 VR：工具旋转速度 （单位 deg/s）。 VE：外部轴速度 （单位 %）。
02	d variable / d const 目标速度值。

2.3.6.2. joint speed

结 构

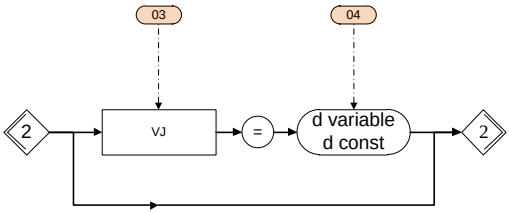


图2-36. Joint speed 模块结构图

说 明

表2-20. joint speed 模块说明表

序号	说明
03	VJ：关节速度值（单位 %）。
04	d variable / d const 目标速度值。

2.3.6.3. position level

结 构

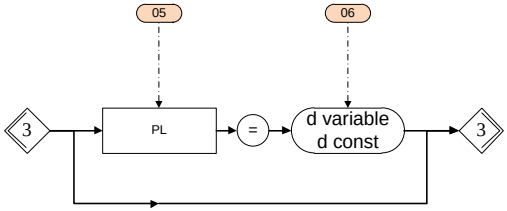


图2-37. position level 模块结构图

说 明

表2-21. position level 模块说明表

序号	说明
05	PL：位置精度等级。
06	d variable / d const 目标位置精度等级值。

2.3.6.4. nmt sync

结 构

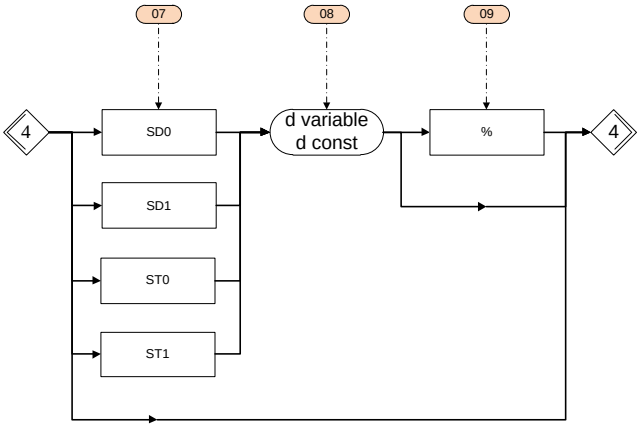


图2-38. nmt sync 模块结构图

说 明

表2-22. nmt sync 模块说明表

序号	说明
07	ST0：距离起点时间 （秒或%）。 ST1：距离终点时间 （秒或%）。

	SD0：距离起点距离 （米或%）。 SD1：距离终点距离 （米或%）。 设置当前步（step）的非运动指令启动执行的条件。 当不设置时，当前步的运动指令执行完成后，才启动执行非运动指令。
08	d variable / d const 目标值。
09	【%】标签。

2.3.6.5. until condition

结 构

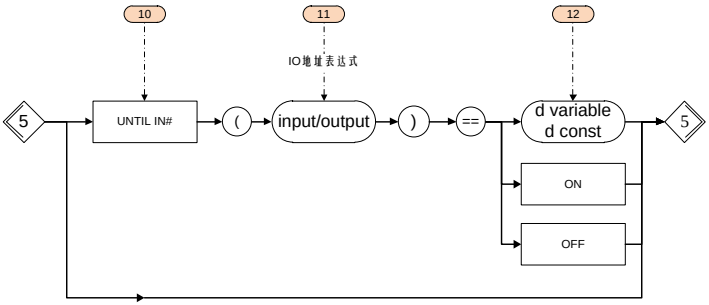


图2-39. until condition 模块结构图

说 明

当指定的端口满足指定的状态时，结束当前运动段。

表2-23. until condition 模块说明表

序号	说明
10	【UNTIL IN#】标签。
11	Input / output 目标端口值。
12	d variable / d const ：目标值。 ON：1。 OFF：0。

2.3.6.6. acc/dec ratio

结 构

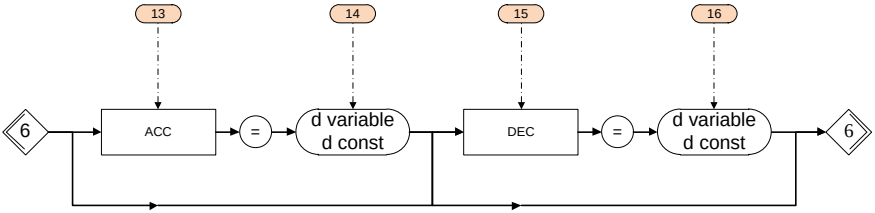


图2-40. acc/dec ratio 模块结构图

说 明

表2-24. acc/dec 模块说明表

序号	说明
13	【ACC】标签，加速度。
14	d variable / d const ：加速度百分比。
15	【DEC】标签，减速度。
16	d variable / d const ：减速度百分比。

2.3.6.7. imove flag

结 构

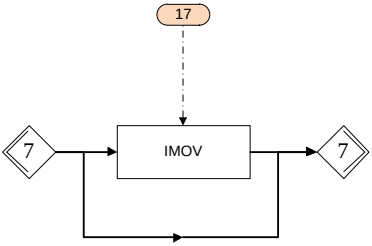


图2-41. imove flag 模块结构图

说 明

表2-25. imove flag 模块说明表

序号	说明
17	IMOV：位置增量运动。

2.3.6.8. cst flag

结 构

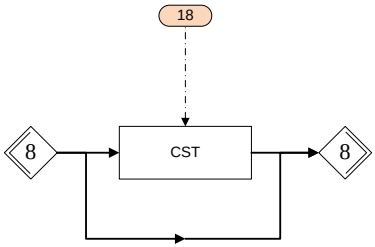


图2-42. cst flag 模块结构图

说 明

表2-26. cst flag 模块说明表

序号	说明
18	CST：本段 MOVC 作为下一段圆弧第 1 个 MOVC。

2.4 运算指令

2.4.1 GETE

功 能

提取位置变量的分量数据，并存储在变量中。

结 构

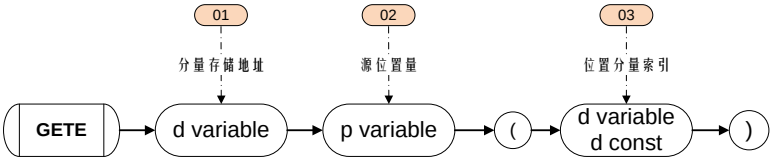


图2-43. GETE 指令结构图

说 明

对指定的位置变量，根据指定的分量索引号，获取分量并存储于 d 变量中。

表2-27. GETE 指令说明表

序号	说明
01	d variable 用于存储值的 D 变量。

02	p variable　位置变量。
03	d variable / d const 位置变量的分量索引。

例 子

（1）　位置变量：

P0006：P0(70)，P1(2)，P2(7)，P3(700)，P4(-30)，P5(90)，P6(50)，P7(2)

P0007：P0(70)，P1(2)，P2(7500)，P3(18)，P4(-12)，P5(30)，P6(540)，P7(10)

```
“PalletCalling”
NOP
D0010=2
LD0011=2
GETE D0001 P0006(2)          //将位置变量 P0006 的位置 2 数据存储在 D0001
GETE LD0002 P0007(2)         //将位置变量 P0007 的位置 2 数据存储在 LD0002
GETE LD0004 P0006(D0010)     //将位置变量 P0006 的位置 2 数据存储在 LD0004
GETE LD0005 P0007(LD0011)    //将位置变量 P0007 的位置 2 数据存储在 LD0005
GETE LD[2+4] P0006(D0010)    //将位置变量 P0006 的位置 2 数据存储在 LD0006
GETE LD[1*7] P[12-5] (D[200/20])
                             //将位置变量 P0007 的位置 2 数据存储在 LD0007
```

2.4.2 SETE

功 能

设置位置变量的分量数据。

结 构

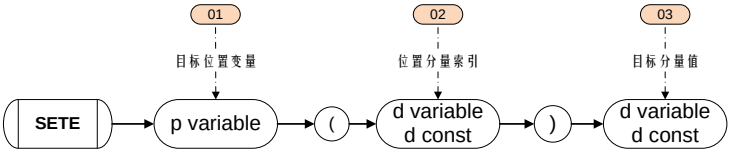


图2-44. SETE 指令结构图

说 明

对指定的位置变量，设置指定的分量索引号对应的分量数据。

表2-28. SETE 指令说明表

序号	说明
01	p variable 目标设置位置量。
02	d variable / d const 位置分量索引值。
03	d variable / d const 位置分量索引。

例 子

（1） “PalletCalled”

```
NOP
D0004=20
D0005=200
SETE P0000(2) D0004          //修改位置变量 P0000 的 2 位置为 D0004
SETE P[0+0](2) D[20-16]      //修改位置变量 P0000 的 2 位置为 D0004
SETE LP0001(2) D0005         //修改位置变量 LP0001 的 2 位置为 D0005
SETE P0000(2) D0004          //修改位置变量 P0000 的 2 位置为 D0004
MOVL P0001
END
```

2.4.3 内置标准函数

内置标准函数包括三角函数，反三角函数，常用数学函数三大类。

2.4.3.1. 三角函数

功 能

计算三角函数值。

表2-29. 正、余弦函数表

函数名称	函数参数	功能	返回值
SIN	U32/S32/U64/S64/DBL	正弦	[-1.0 ~ 1.0] (DBL)
COS	U32/S32/U64/S64/DBL	余弦	[-1.0 ~ 1.0] (DBL)
TAN	U32/S32/U64/S64/DBL	正切	-INF ~ INF(DBL)

结 构

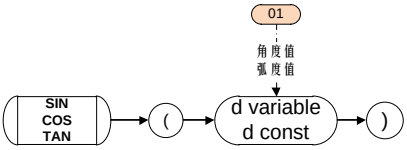


图2-45. SIN/COS/TAN 指令结构图

说 明

当函数参数为 DBL（浮点）时，表示单位为弧度值（radians）。否则单位为度数值（degrees）。可使用符号 M_PI 表示 π 弧度值。

表2-30. SIN/COS/TAN 指令说明表

序号	说明
01	d variable / d const 角度值，弧度或度数。

例 子

```
D0001 = SIN(+0*M_PI+M_PI*0)-0.0                    //0 degrees
D0002 = SIN(+0*M_PI+M_PI/6)-0.5                    //30 degree
D0003 = COS(+0*M_PI+M_PI/4)-M_SQRT2/2.0            //45 degree
D0004 = COS(+0*M_PI+M_PI/3)-0.5                    //60 degree
D0005 = TAN(+0*M_PI+M_PI/2)                        //90 degree
D0006 = TAN(+0*M_PI+M_PI/2+M_PI/6)+M_SQRT3         //120 degree
```

运算结果：D0001~D0006 均为 0.0（DBL）

2.4.3.2. 反三角函数

功 能

计算反三角函数值。

表2-31. 正、余弦函数表

函数名称	功能	参数范围	函数参数数据类型	返回值	备注
ASIN	反正弦	[-1.0 ~ 1.0]	DBL	(-PI/2, PI/2] radians(DBL)	
ACOS	反余弦			(0.0, PI] radians(DBL)	
ATAN	反正切	-INF ~ INF	U32/S32/U64/ S64/DBL	(-PI/2, PI/2] radians(DBL)	单参数
ATAN				(-PI, PI] radians(DBL)	双参数

结 构

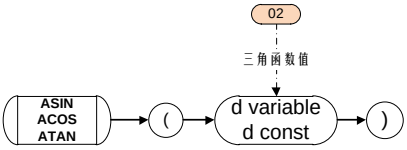


图2-46. ASIN/ACOS/ATAN 指令结构图

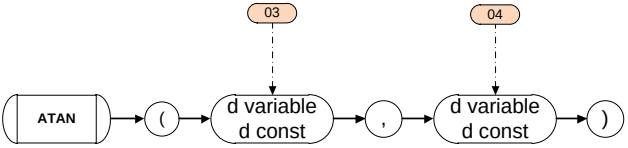


图2-47. ATAN 双参数指令结构图

说 明

ATAN 函数具有两个功能，分别为单参数和双参数，返回值参见表 2-31。

表2-32. ASIN/ACOS/ATAN 指令说明表

序号	说明
02	d variable / d const 三角函数值。
03	Y 分量。
04	X 分量。

例 子

```
D0001 = ASIN(-1.0) * 180/M_PI                    //D0001 = -90.0
D0002 = ASIN(-M_SQRT3/2.0) *180/M_PI             //D0002 = -60.0
```

2.4.3.3. 常用数学函数

功 能

函数名称	功能	参数个数	返回值	备注
RAND	产生[0, 1]之间的随机数	0	DBL	
CEIL	向上取整	1	DBL	
FLOOR	向下取整			
EXP	e 幂指数	1	DBL	
EXP2	2 幂指数			
EXP10	10 幂指数			
LOG	e 对数			
LOG2	2 对数			
LOG10	10 对数			
SQRT	平方根	1	DBL	
CBRT	立方根			
POW	x 的 y 次方	2		第一参数需>=0。
ABS	绝对值	1	U32/S32/U64/S64/DBL	返回值数据类型与参数一致。
MAX	两个数中的最大值	2		2 个参数必须是同一数据类型，返回值数据类型与参数一致。
MIN	两个数中的最小值	2		

表2-1. 数学函数表

结 构

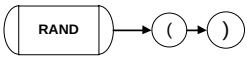


图2-48. RAND 数学函数指令结构图

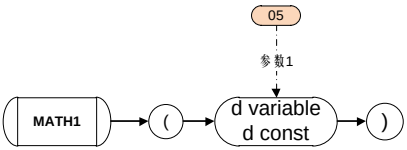


图2-49. 1 个参数数学函数指令结构图

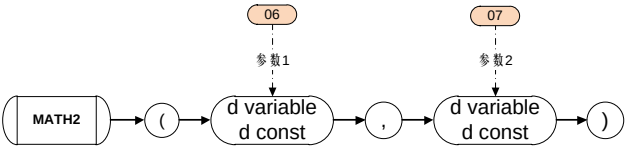


图2-50. 2 个参数数学函数指令结构图

说 明

MAX/MIN 数学函数的两参数数据类型必须一致，返回值数据类型为参数的数据类型。否则提示报警。

表2-33. 常用数学函数说明表

序号	说明
05	参数 1，数据类型 U32/S32/U64/S64/DBL。

06	参数 1，数据类型 U32/S32/U64/S64/DBL。
07	参数 2，数据类型 U32/S32/U64/S64/DBL。

例 子

```
D0001=MAX(-1,2)                 //D0001 = 2 (S32)
D0002=MAX(-1.0,2.0)             //D0002 = 2.0 (DBL)
D0003=CEIL(1.9999)              //D0003 = 2.0 (DBL)
D0004=FLOOR(2.9999)             //D0004 = 2.0 (DBL)
D0005=SQRT(10)*SQRT(10)-10      //D0005 = 0.0 (DBL)
D0006=SQRT(100)-10              //D0006 = 0.0 (DBL)
D0007=EXP(2)-M_E*M_E            //D0007 = 0.0 (DBL)
D0008=LOG(POW(M_E,1))-1.0       //D0008 = 0.0 (DBL)
D0009=LOG2(SQRT(2.0))-0.5       //D0009 = 0.0 (DBL)
```

2.5 设置指令

2.5.1 **SPEED**

功 能

设置运动指令速度。

结 构

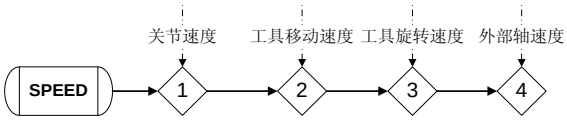


图2-51. SPEED 指令结构图

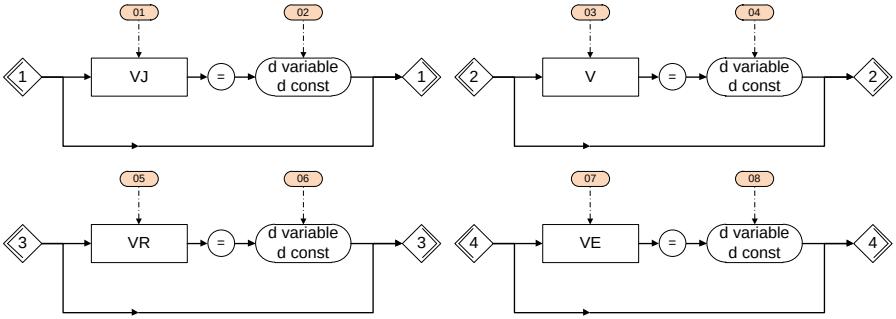


图2-52. SPEED 分块图

说 明

SPEED 指令为后续的所有运动指令设定默认的速度参数。该设置一直有效，直到下一个 SPEED 指令，或控制器复位。

表2-34. SPEED 指令说明表

序号	说明
01	VJ：关节速度（单位 %）。
02	d variable / d const 目标 VJ 值。
03	V：工具移动速度（单位 mm/s）。
04	d variable / d const 目标 V 值。
05	VR：工具旋转速度（单位 deg/s）。
06	d variable / d const 目标 VR 值。
07	VE：外部轴速度（单位 %）。
08	d variable / d const 目标 VE 值。

例 子

1. 设置再现速度

```
Job:test1

NOP
MOVJ C0001 VJ=100.00
MOVL C0002 V=120
SPEED VJ=50 V=200 VR=30
MOVJ C0004
MOVL C0005
MOVL C0006 VR=60
END
```

```
NOP
按关节速度100％运行到C0001
按工具移动速度120mm/s运行到C0002
设置速度：  关节速度为50%
            工具移动速度为200mm/s
            工具旋转速度为30deg/s
按设置的关节速度运行到C0004
按设置的工具移动速度运行到C0005
按工具旋转速度60deg/s运行到C0006
END
```

图2-53. SPEED 指令例子图

2.5.2 RAMP

功 能

设置运动指令的加速度曲线类型。

结 构

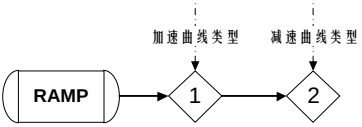


图2-54. RAMP 指令结构图

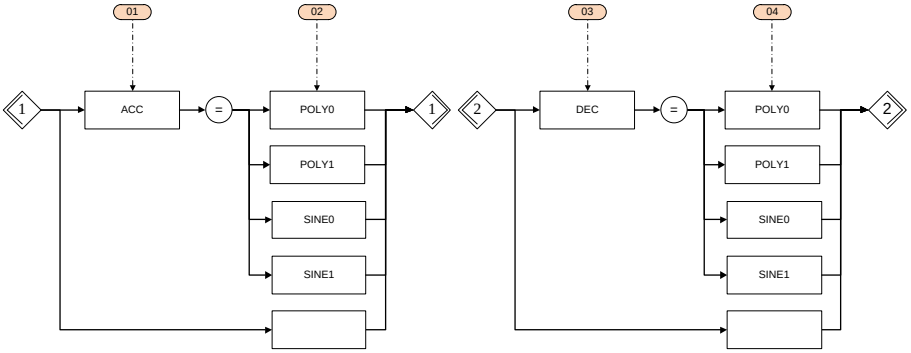


图2-55. RAMP 指令分块图

说 明

RAMP 指令将影响其后所有执行的运动指令的加减速曲线类型，直至下一 RAMP 指令，或控制器复位。

表2-35. RAMP 指令说明表

序号	说明
01	ACC：加速度。
02	POLY0：直线。 POLY1：多项式。 SIN0：三角函数 0。 SIN1：三角函数 1。
03	DEC：减速度。
04	POLY0：直线。 POLY1：多项式。 SIN0：三角函数 0。 SIN1：三角函数 1。

例 子

（1）
```
MOVL C0002 V=200
RAMP ACC=SINE1
MOVJ C0003 VJ=20
MOVC C0005 V=200
END
```

（2）
```
MOVL C0002 V=200
```

```
RAMP ACC=SINE1
MOVJ C0003 VJ=20
MOVC C0005 V=200
MOVJ C0006 VJ=20
MOVC C0007 V=200
RAMP ACC=SINE1 DEC=POLY0
MOVC C0005 V=200
END
```

2.5.3 **REFSYS**

功 能

设置参考坐标系。

结 构

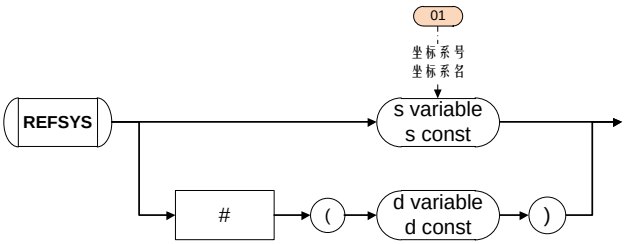


图2-56. REFSYS 指令结构图

说 明

REFSYS 指令为后续的所有运动指令设定一个新的参考坐标。该设置一直有效，直至下一 REFSYS 指令，或关闭项目。

若程序中没有设定参考坐标系，系统默认参考坐标系为世界坐标系。

表2-36. REFSYS 指令说明表

序号	说明
01	目标参考坐标系。 s variable / s const：通过坐标系名称设置。 d variable / d const：通过坐标系号设置。 预定义的坐标系： 坐标系名称 | 坐标系号 ROBOT | -1 WORLD | 0

例 子

```
NOP
REFSYS  “ROBOT”          //参考坐标系设置为机器人坐标系
MOVL C0001 V=1000
REFSYS  “WORLD”          //参考坐标系设置为世界坐标系
MOVL C0001 V=1000
REFSYS  #(2)             //参考坐标系设置为2号用户坐标系
MOVL C0001 V=1000
END
```

2.5.4 **TOOL**

功 能

设置使用的工具。

结 构

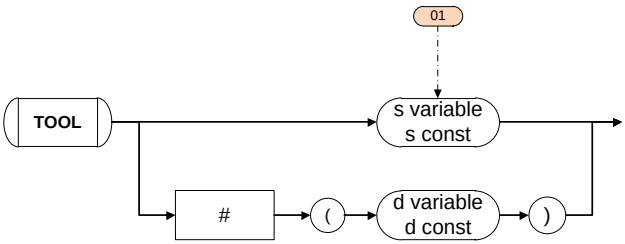


图2-57. TOOL 指令结构图

说 明

TOOL 指令设置使用的工具。该设置一直有效，直到下一个 TOOL 指令，或关闭项目。

表2-37. TOOL 指令说明表

<table>
<tr><th>序号</th><th>说明</th></tr>
<tr><td>01</td><td>目标工具。
s variable / s const：通过工具名称设置。
d variable / d const：通过工具号设置。

预定义的工具：
<table>
<tr><td>工具名称</td><td>工具号</td></tr>
<tr><td>FLANGE</td><td>0</td></tr>
</table></td></tr>
</table>

例 子

（1） D0003=2
TOOL #(D0003)
使用 2 号工具。

（2） TOOL “FLANGE” //或 TOOL #(0)
未安装工具。

2.5.5 AREA

功 能

选择当前要激活的区域。

结 构

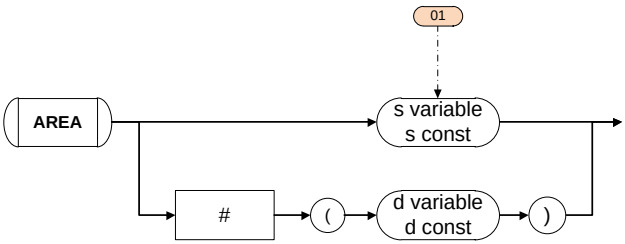


图2-58. AREA 指令结构图

说 明

表2-38. AREA 指令说明表

序号	说明
01	s variable / s const：通过区域名称设置。 d variable / d const：通过区域号设置。

例 子

```
NOP
AREA   #(1)
```

使用 1 号区域。

```
NOP
AREA   #(0)                    //或 AREA “NONE”
```

取消区域。

2.6 系统指令

2.6.1 **PRINT**

功 能

在控制台终端显示字符信息。

结 构

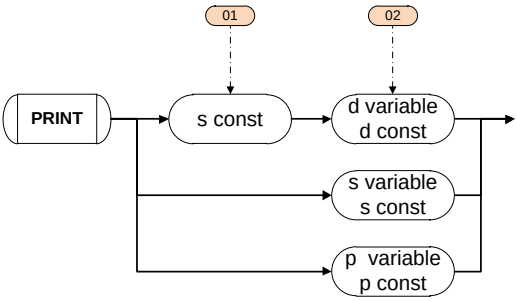


图2-59. PRINT 指令结构图

说 明

PRINT 指令在控制台显示终端输出指定的字符串或位置量对应的字符串。

表2-39. PRINT 指令说明表

序号	说明
01	s const 目标格式化字符串。
02	d variable / d const：要显示的 D 变量。 s variable / s const：要显示的 S 变量。 p variable / p const：要显示的 P 变量。

例 子

（1） PRINT “Test”

在终端显示 Test

（2） PRINT “Test1\nTest2”

在终端显示

Test1

Test2

（3） SET D0000 10

PRINT “Test %d” D0000

在终端显示 Test 10 （其中%d 代表变量 D0000 并作为字符串形式输出）

2.6.2 **CLS**

功 能

删除在控制台终端显示的所有字符。

结 构

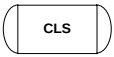


图2-60. CLS 指令结构图

说 明

删除在控制台终端显示的所有字符。

例 子

（1） PRINT “TEST”
CLS
执行 CLS 指令时，清除显示终端内容。

2.6.3 **GETPRM**

功 能

获取系统参数值到变量。

结 构

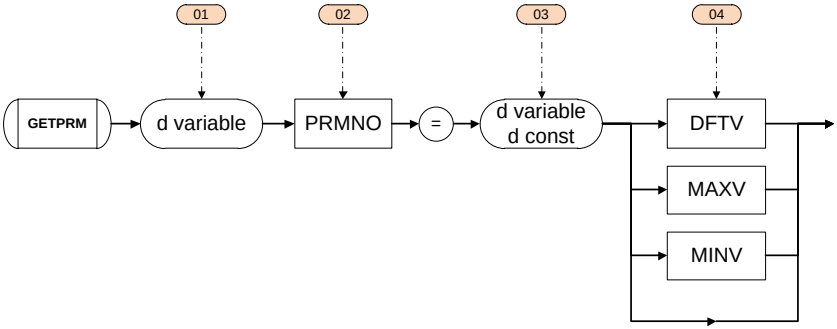


图2-61. GETPRM 指令结构图

说 明

GETPRM 指令获取指定的参数号的参数值到 D 变量。

表2-40. GETPRM 指令说明表

序号	说明
01	d variable：存放参数值的目标 D 变量。
02	PRMNO：参数号标签。
03	d variable / d const：参数号。
04	MAXV：最大值，获取该参数的最大值到【01】。 MINV：最小值，获取该参数的最小值到【01】。 DFTV：默认值，获取该参数的默认值到【01】。 空：当前值，获取该参数的当前值到【01】。

例 子

（1） D200=2
GETPRM D3 PRMNO = D200
将 2 号参数当前值赋值到变量 D3 中

（2） D200=2
GETPRM D3 PRMNO = D200 MAXV
将 2 号参数设置范围的最大值赋值到变量 D3 中

（3） D200=2
GETPRM D3 PRMNO = D200 MINV
将 2 号参数设置范围的最小值赋值到变量 D3 中

（4） D200=2
GETPRM D3 PRMNO = D200 DFTV
将 2 号参数出厂设置值赋值到变量 D3 中

2.6.4 **SETPRM**

功 能

设置系统参数值。

结 构

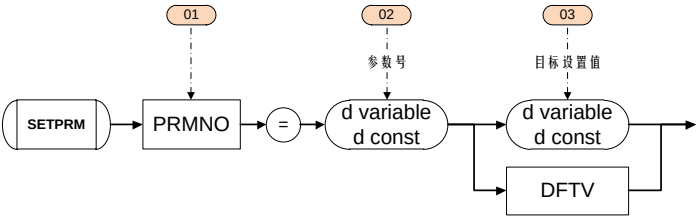


图2-62. SETPRM 指令结构图

说 明

SETPRM 指令设置【模块 03】指定的数据值到【模块 02】参数号指定的系统参数中。

应根据系统参数含义相关说明设置参数值，否则可能使系统运行于异常状态。

表2-41. SETPRM 指令说明表

序号	说明
01	PRMNO：参数号标签。
02	d variable / d const：参数号。
03	DFTV：恢复该参数的出厂默认值。 d variable / d const：参数目标设置值。

例 子

（1） SETPRM PRMNO = 6 D3

将变量 D3 中的数值设置到 6 号参数

（2） SETPRM PRMNO = 6 DFTV

将 6 号参数恢复为出厂设置的默认值

2.6.5 **ALM**

功 能

产生一个报警，包括报警号和报警信息字符串。

结 构

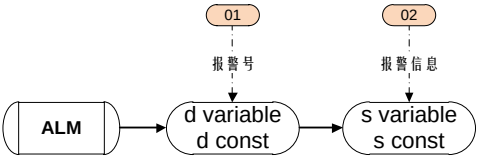


图2-63. ALM 指令结构图

说 明

产生一个报警，包括报警号和报警信息字符串。

表2-42. ALM 指令说明表

<table>
<tr><th>序号</th><th>说明</th></tr>
<tr><td>01</td><td>d variable / d const：报警号。
<table>
<tr><td colspan="2">报警号范围</td></tr>
<tr><td>警告</td><td>2000～2063</td></tr>
<tr><td>故障</td><td>5000～5063</td></tr>
</table></td></tr>
<tr><td>02</td><td>s variable / s const：报警信息。</td></tr>
</table>

例 子

（1） D0002=2000
S0002= “There is a warning”
ALM D0002 S0002
触发 2000 号报警（警告），报警内容：“There is a warning”

（2） ALM 5000 “There is an error”
触发 5000 号报警（故障），报警内容：“There is an error”

2.7 模块

本小节定义前述章节指令定义中使用到的【模块】。

2.7.1 d const

结 构

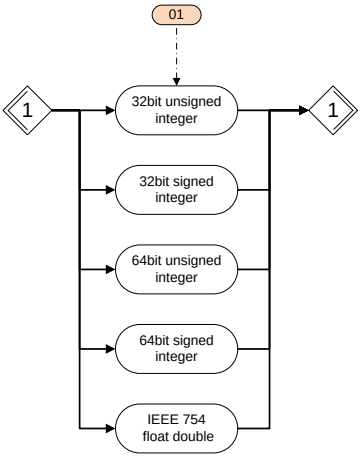


图2-64. d const 模块定义图

说 明

模块【d const】 为 digit const 的缩写，即数字常量定义。

表2-43. d const 模块说明表

序号	说明
01	32bit signed integer：32 位整数。 32bit unsigned integer：32 位正整数。 64bit signed integer：64 位整数。 64bit unsigned integer：64 位正整数。 IEEE 754 float double：64 位双精度浮点数（IEEE754 标准）。

2.7.2 s const

结 构

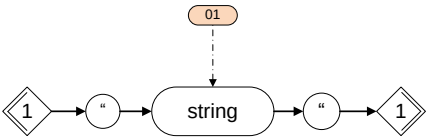


图2-65. s const 模块定义图

说 明

模块【s const】 为 string const 的缩写，即字符常量定义。

表2-44. s const 模块说明表

序号	说明
01	形如 ”hello world”，”ABC” 等。

2.7.3 **p const**

结 构

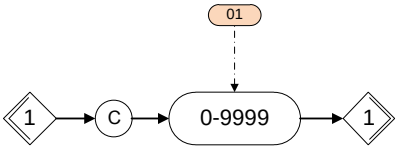


图2-66. p const 模块定义图

说 明

模块【p const】 为 position const 的缩写，即位置常量定义。

每一个任务，可以单独定义 C0000~C9999 位置常量。在任务内有效，任务之间位置常量互不影响。

2.7.4 d variable

结 构

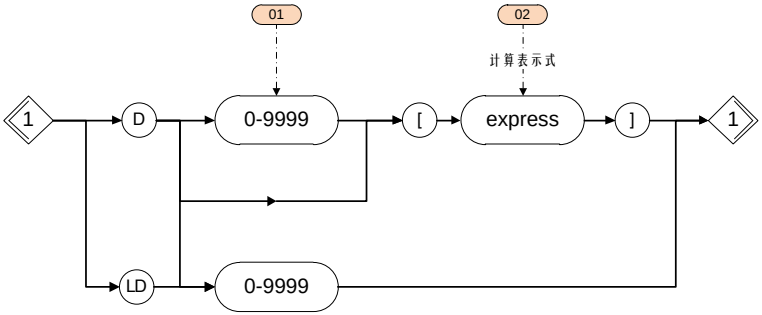


图2-67. d variable 模块定义图

说 明

模块【d variable】 为 digit variable 的缩写，即数字变量定义。

表2-45. d variable 模块说明表

序号	说明
01	0-9999：D 变量号。
02	express：相对偏移量，请参考 express 定义。

以 D 开始的为项目数字变量，以 LD 为开始的为局部数字变量。例如 D2，LD03。可以通过计算表达式相对其他数字变量的偏移量指定新的变量，如 D2[3]实际表示的是 D5，D[1+4]表示的也是 D5。

2.7.5 s variable

结 构

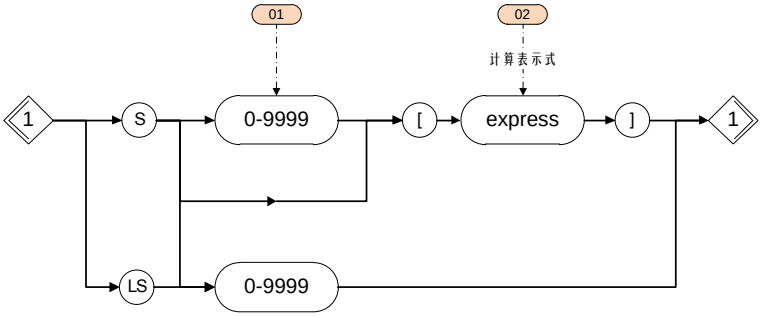


图2-68. s variable 模块定义图

说 明

模块【s variable】 为 string variable 的缩写，即字符串变量定义。

表2-46. s variable 模块说明表

序号	说明
01	0-9999： S 变量号。
02	express：相对偏移量，请参考 express 定义。

以 S 开始的为项目字符串变量，以 LS 为开始的为局部字符变量。例如 S2，LS03。 可以通过计算表达式相对其他字符变量的偏移量指定新的变量，如 S2[3]实际表示的是 S5，LS[1+4]表示的 LS5。

2.7.6 **p variable**

结 构

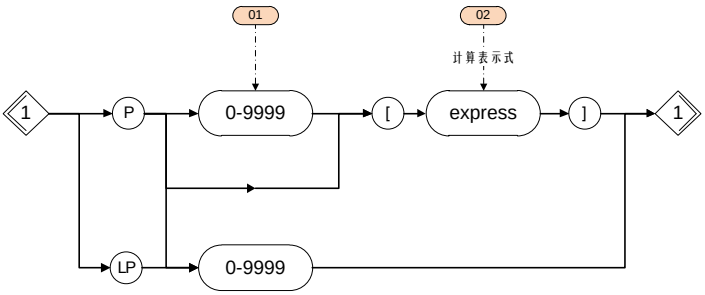


图2-69. p variable 模块定义图

说 明

模块【p variable】 为 position variable 的缩写，即位置变量定义。

表2-47. p variable 模块说明表

序号	说明
01	0-9999： P 变量号。
02	express：相对偏移量，请参考 express 定义。

以 P 开始的为项目位置变量，以 LP 为开始的为局部位置变量。例如 P2，LP03。

可以通过【模块 01】的计算表达式相对其他位置变量的偏移量指定新的变量，如 P2[3]实际表示的是 P5，LP[1+4]表示的 LP5。

2.7.7 system d variable

结 构

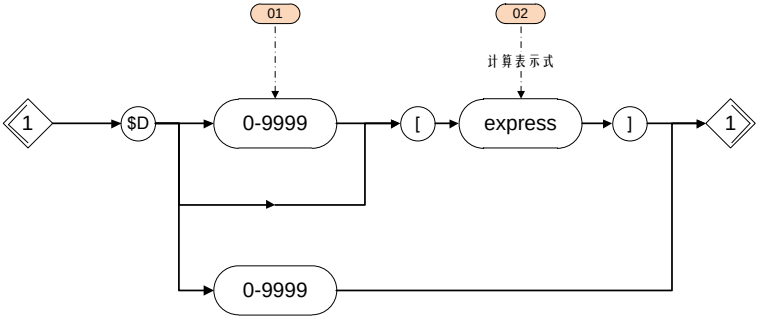


图2-70. system d variable 模块定义图

说 明

模块【system d variable】 为 system digit variable 的缩写，即系统数值变量。以$D 开始，例如$D2。

表2-48. system d variable 模块说明表

序号	说明
01	0-9999： $D 变量号。
02	express：相对偏移量，请参考 express 定义。

可以通过【模块 01】的计算表达式相对系统数字变量的偏移量指定新的变量，如$D2[3]实际表示的是$D5。系统变量对于用户程序只可读取，不可写入。

2.7.8 system s variable

结 构

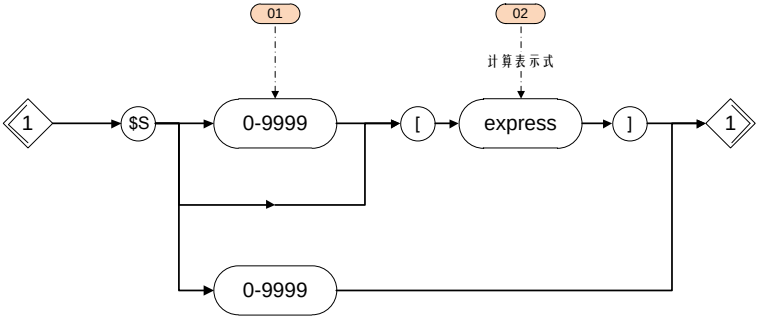


图2-71. system s variable 模块定义图

说 明

模块【system s variable】 为 system string variable 的缩写，即系统字符串变量。以$S 开始，例如$S2。

表2-49. system s variable 模块说明表

序号	说明
01	0-9999：$ S 变量号。
02	express：相对偏移量，请参考 express 定义。

可以通过【模块 01】的计算表达式相对系统字符串变量的偏移量指定新变量，如$S2[3]实际表示的是$S5。系统字符串变量对于用户程序只可读取，不可写入。

2.7.9 system p variable

结 构

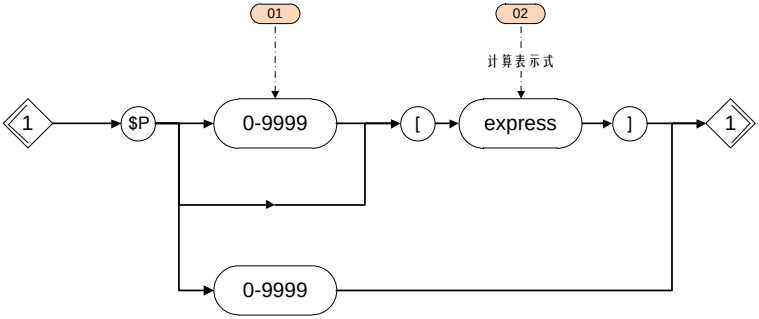


图2-72. system p variable 模块定义图

说 明

模块【system p variable】 为 system position variable 的缩写，即系统位置变量。以$P 开始，例如$P2。

表2-50. system p variable 模块说明表

序号	说明
01	0-9999：$ P 变量号。
02	express：相对偏移量，请参考 express 定义。

可以通过【模块 01】的计算表达式相对系统位置变量的偏移量指定新的变量，如$P2[3]实际表示的是$P5。系统位置变量对于用户程序只可读取，不可写入。

2.7.10 **express**

结 构

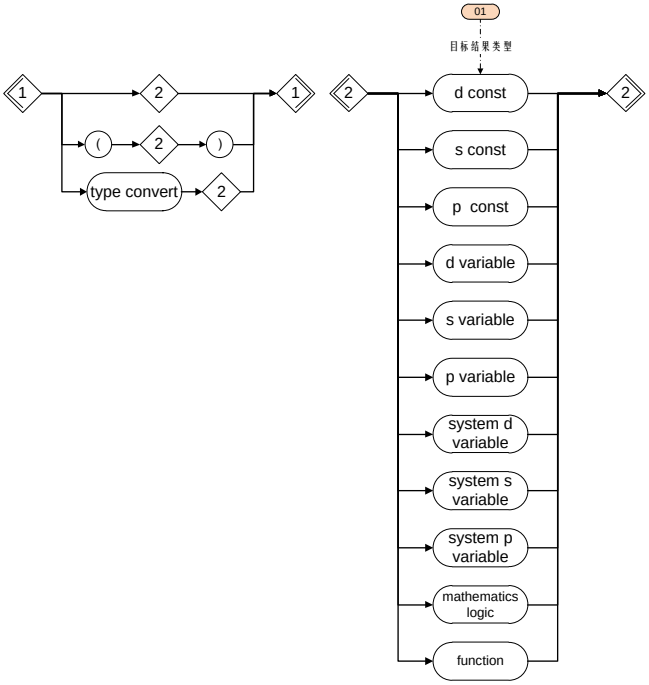


图2-73. express 模块定义图

说 明

模块【express】为计算表达式。由 D、S、P 类型的变量、常量、通过数学逻辑运算、函数、强制类型装换等运算表示。最终得到 D、S、P 的变量或常量。并可以作为计算表达式本身、指令、赋值指令等的输入。

请参考：2.7.10.1 function、2.7.10.2 mathematics logic、2.7.10.3 type convert。

2.7.10.1. function

结 构

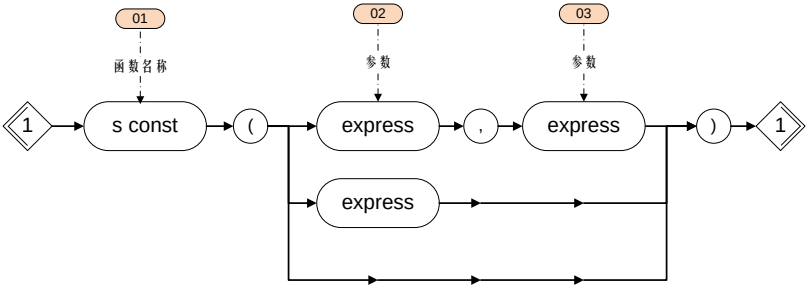


图2-74. function 模块定义图

说 明

模块【function】为函数模块，由函数名称和参数组成。

表2-51. function 模块说明表

序号	说明
01	s const 函数名称。请参考 2.4.3 内置标准函数。
02	express：参数 1。
03	express：参数 2。

2.7.10.2. mathematics logic

结 构

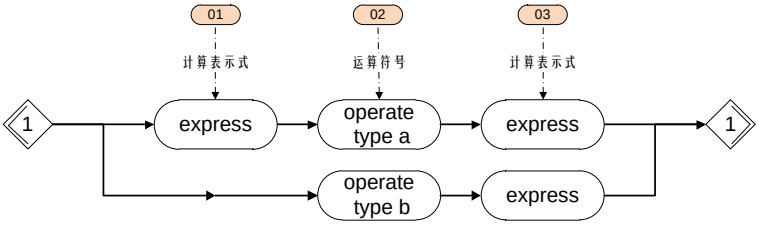


图2-75. mathematics logic 模块定义图

说 明

模块【mathematics logic】为算术逻辑运算模块。

表2-52. mathematics logic 模块说明表

序号	说明
01	express：操作数 1。
02	运算符。 operate type a：二元运算符号，操作数为【01】和【03】。可用的运算符号为 + - * / % << >> > < >= <= == != & ^ | && || operate type b：一元运算符号，操作数为【03】。可用的运算符号为 - + ~ ！
03	express：操作数 2。

2.7.10.3. type convert

结 构

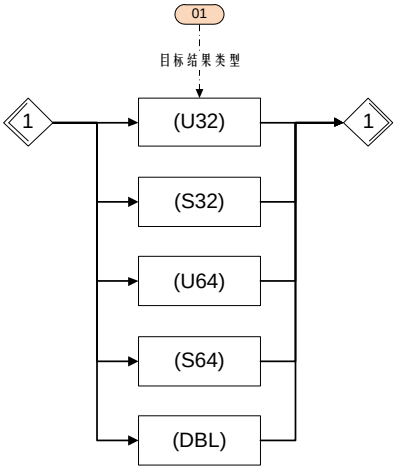


图2-76. type convert 模块定义图

说 明

模块【type convert】为强制类型转换模块。其后跟随数值计算表达式，将把表达式计算结果按该模块指定的数据类型进行强制转换。

表2-53. type convert 模块说明表

序号	说明
01	U32：把目标数据类型转为 32 位正整数。 S32：把目标数据类型转为 32 位整数。 U64：把目标数据类型转为 64 位正整数。 S64：把目标数据类型转为 64 位整数。 DBL：把目标数据类型转为双精度浮点数。

2.7.11 **assignment**

结 构

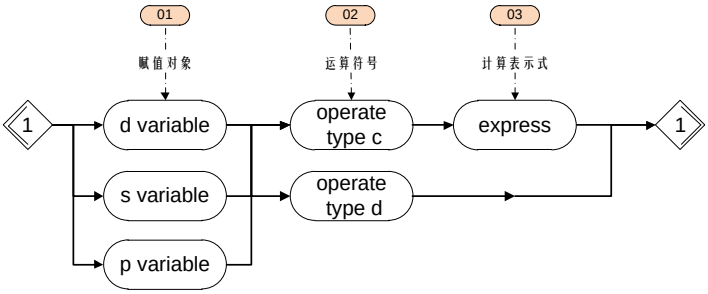


图2-77. assignment 模块定义图

说 明

模块【assignment】为赋值模块。

表2-54. assignment 模块说明表

序号	说明
01	d variable：目标赋值 D 变量地址。 s variable：目标赋值 S 变量地址。 p variable：目标赋值 P 变量地址。

02	运算符。 operate type c：可用的赋值运算符号为 = /= *= %= += -= >>= <<= &= |= ^= operate type d：可用的赋值运算符号为 ++ - -
03	express：赋值源数据计算表达式。

2.7.12 **input/output**

结 构

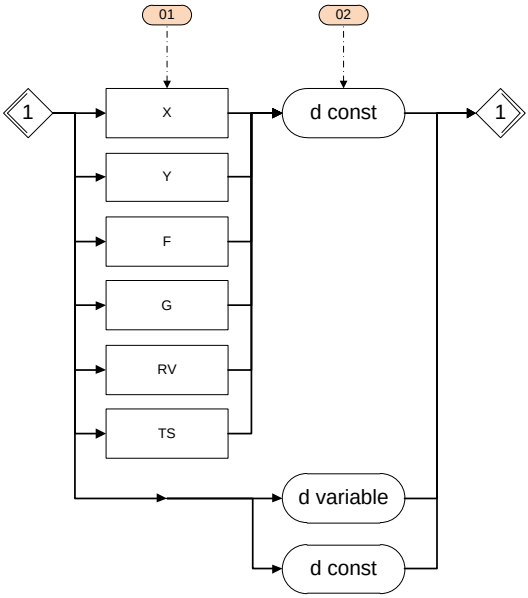


图2-78. Intput/output 模块定义图

说 明

模块【input/output】为 IO 端口。

表2-55. Input/output 模块说明表

序号	说明
01	当【01】为缺省空时，通过【02】指定的统一的 IO 号访问端口。 当【01】不为空时，通过【02】常量访问对应的端口。
02	d const：地址值。 d variable / d const：统一的地址编号值。

由于通用 IO 端口为 8 位，如果需要 d const 访问位请使用.0 ~ .7，形如 X110.7。F0.8 超出位范围，运行时提示报警。

2.7.13 **if condition**

结 构

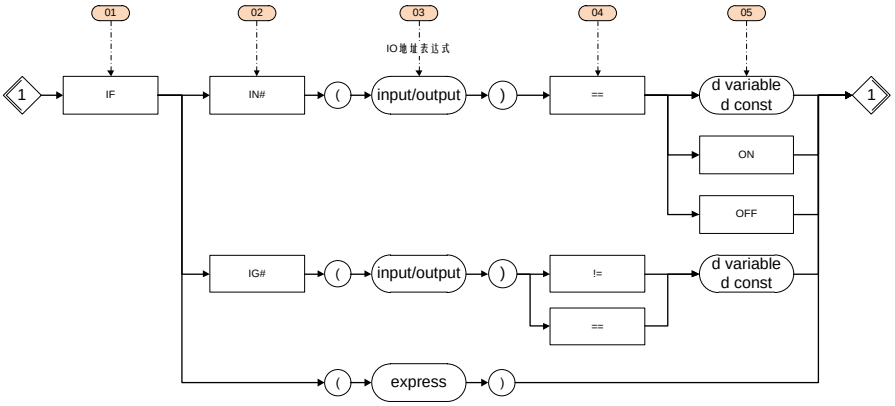


图2-79. if condition 模块定义图

说 明

模块【if condition】为条件表达式。当条件表达式为真时，指令行执行，否则不执行。

表2-56. if condition 逻辑判断方法表

序号	说明
A	【02】为输入端口【IN#】时，判断【03】指定的端口是否等于【05】,

	从而得到逻辑运算结果为真或假。
B	【02】为输入端口【IG#】时，判断【03】指定的端口是否等于或等于【05】，从而得到逻辑运算结果为真或假。
C	【02】为空时，判断【03】指定的计算表达式运算结果为非零或零，从而得到运算结果为真或假。

表2-57. if condition 模块说明表

序号	说明
01	IF 标签。
02	IN#：1 位输入端口。 IG#：8 位输入端口。
03	Input/output：端口地址。 express：逻辑表达式。
04	== 端口是否等于目标匹配值。 != 端口是否不等于目标匹配值。
05	目标匹配值。 d variable / d const: D 变量常量值。 ON: 1。 OFF: 0。

3 一览表

3.1 指令一览表

表3-1. 指令一览表 (待续-下页)

指令名称	功能概述	分组	备注
2.1.1 DIN	获取 IO 端口状态	IO 指令	
2.1.2 DOUT	设置输出端口状态		
2.1.3 AIN	获取模拟量端口电压值		
2.1.4 AOUT	将电压值输出至模拟量输出端口		
2.1.5 PULSE	将脉冲信号输出到输出端口中		
2.2.1 JUMP	跳转至指定的标号	控制指令	
2.2.2 CALL	调用指定的任务		
2.2.3 RET	返回调用任务		
2.2.4 PAUSE	暂停任务，等待任务再次启动		
2.2.5 ABORT	停止执行任务，并返回任务首行		
2.2.6 WAITIO	等待 I/O 信号为指定的状态		
2.2.7 WAITMTN	等待当前步的运动指令运行结束后继续执行		
2.2.8 TIMER	等待指定的时间		
2.2.9 GETA	获取 CALL 指令中传递的参数		
2.2.10 GETR	获取 RET 指令返回值		
2.2.11 LABEL	跳转位置标号定义		

表3-1. 指令一览表 (续-上页)

2.3.1 NOP	空操作	运动指令	
2.3.2 END	程序结束		
2.3.3 MOVJ	以关节插补方式向目标位置移动		
2.3.4 MOVL	以直线插补方式向目标位置移动		
2.3.5 MOVC	以圆弧插补方式向目标位置移动		
2.4.1 GETE	提取位置变量的分量数据，并存储在变量中	运算指令	
2.4.2 SETE	设置位置变量的分量数据		
2.5.1 SPEED	设置运动指令速度	设置指令	
2.5.2 RAMP	设置运动指令的加减速曲线类型		
2.5.3 REFSYS	设置参考坐标系		
2.5.4 TOOL	设置使用工具		
2.5.5 AREA	设置使用区域		
2.6.1 PRINT	在控制台终端显示字符信息	系统指令	
2.6.2 CLS	删除在控制台终端显示的所有字符		
2.6.3 GETPRM	获取系统参数		
2.6.4 SETPRM	设置系统参数		
2.6.5 ALM	产生报警		

3.2 模块一览表

表3-2. 模块一览表 (待续-下页)

模块章节	模块名称	模块结构图	备注
2.3.6.1	cartesian speed	图 2-35	运动指令使用
2.3.6.2	joint speed	图 2-36	
2.3.6.3	position level	图 2-37	
2.3.6.4	nmt sync	图 2-38	
2.3.6.5	until condition	图 2-39	
2.3.6.6	acc/dec ratio	图 2-40	
2.3.6.7	imove flag	图 2-41	
2.3.6.8	cst flag	图 2-42	
2.7.1	d const	图 2-64	模块表
2.7.2	s const	图 2-65	
2.7.3	p const	图 2-66	
2.7.4	d variable	图 2-67	
2.7.5	s variable	图 2-68	
2.7.6	p variable	图 2-69	
2.7.7	system d variable	图 2-70	
2.7.8	system s variable	图 2-71	
2.7.9	system p variable	图 2-72	
2.7.10	express	图 2-73	
2.7.10.1	function	图 2-74	
2.7.10.2	mathematics logic	图 2-75	
2.7.10.3	type convert	图 2-76	

表3-2. 模块一览表 (续-上页)

2.7.11	assignment	图 2-77	
2.7.12	input/output	图 2-78	
2.7.13	if condition	图 2-79	