DENSO机械手

编程手册Ⅰ
基础知识与指令

Copyright © 2008-2011 DENSO WAVE INCORPORATED
All rights reserved.

本使用说明书的著作权属于 DENSO WAVE INCORPORATED。

本说明书所登载的公司名称和产品，均属各公司的商标或注册商标。

规格如有变更，恕不另行通知。

用于本说明书中的图片与实际操作时显示的画面会有所不同。

前言

承蒙您对DENSO机械手的厚爱，在此表示由衷的感谢。

本产品汇聚了本公司的先进技术，是高速度、高精度而且具备高度功能的 "装配用机械手"。

请在充分阅读理解本说明书的基础上，安全高效地使用机械手。

本书说明的对象产品

■配置 RC7 型控制器

- 垂直多关节型机械手　　V* 系列
- 水平多关节型机械手　　H* -G 系列
- 直角坐标型机械手　　XYC - 4G 系列
- 内置型机械手　　XR 系列

■选件品

- 视觉设备　　μVision 系列

注：机械手控制器的版本记载于粘贴在控制器表面的 "控制器设定表" 的软件Ver.栏中。
此外，还可通过多功能教导器的［主画面］-［F6 设定］-［F6 维护］-［F2 版本］的ROM版本栏进行确认。

要求

在使用之前，请务必阅读 "安全注意事项"，以便能够正确安全使用DENSO机械手。

本书的构成

本书的构成如下所示。

第 1 部 程序设计

第 1 章 程序示例

记述了机械手编程语言PAC的程序示例。请作为用于了解指令使用方法的资料使用。

第 2 章 程序的动作原理

对与程序执行时规定的相关内容进行说明。该程序是通过编写程序所必需的程序语言PAC编制而成的。

第 3 章 机械手动作的种类

机械手的动作以定位其基准位置的方法或判断是否到达目标位置的方法，分为若干类型。就机械手的动作种类进行说明。

第 4 章 速度、加速度、减速度的指定

就速度、加速度、减速度的含义和设定进行说明。

第 5 章 视觉控制

对编写程序时所必需的与视觉相关的用语进行说明。

第 2 部 指令参考

第 6 章 指令的表示方法与分类

记述了机械手编程语言PAC的指令的表示方法与指令一览。

通过本章的指令一览表能够迅速地查找到个别指令的具体说明。

第 7 章 PAC 语言的构成要素

在本章中，就构成PAC语言的要素（标识符、变量、常量、运算符、表达式、指令等）的规定进行说明。

第 8 章 PAC 语言的语法

对基于PAC语言进行编程时的语法上的规定进行说明。

第 9 章～第 21 章

对机械手编程语言PAC的各个指令进行说明。按照不同功能进行总结。

若要迅速找到特定的指令说明，可从第8章的指令一览表中找到指令的说明页。

第 22 章 附录

总结了编码及参数等与程序编写、运用密切相关的必要信息。

目录

按照字母顺序排列的指令一览表

以功能为单位的指令一览表

第 1 部 程序的基础知识

第 1 章 程序示例

1.1 典型事例应用 1-1
1.2 程序流程 1-2
1.3 程序列表 1-3

第 2 章 程序的动作原理

2.1 程序的调出与子程序 2-1
2.1.1 子程序的调出 2-2
2.1.2 程序的调出 2-3
2.1.3 程序的递归调用 2-4
2.1.4 文件夹结构的程序调出方法［RC7_Ver. 2.2 以后版本］ 2-5
2.2 程序的启动 2-7
2.2.1 从多功能教导器、小型教导器进行的启动 2-7
2.2.2 从外部机器进行启动 2-7
2.3 多项任务 2-8
2.3.1 优先顺序 2-8
2.3.2 任务之间通信 2-8
2.4 串行通信 2-10
2.4.1 线路编号 2-10
2.4.2 通信指令 2-11
2.4.3 通信缓冲区的清空 2-11
2.4.4 通信端口指令 2-11
2.4.5 示例、应用 2-12
2.5 共用程序库 (Library) 2-16
2.5.1 程序库 (Program Bank) 2-16
2.5.2 码垛堆积共用程序库 (Library) 2-17
2.6 TP 简易操作盘［Ver.1.5 以上版本］ 2-28
2.6.1 操作盘画面创建的程序示例 2-30

第 3 章 机械手动作的种类

3.1 绝对动作与相对动作 3-1
3.1.1 绝对动作 3-1
3.1.2 相对动作 3-1
3.1.3 绝对动作与相对动作的动作示例 3-1

3.2 到达位置的确认方法 3-3
3.2.1 通过 (Pass) 动作 3-3
3.2.2 结束动作 3-3
3.2.3 编码器确认动作 3-3
3.2.4 通过 (Pass) 动作、结束动作、编码器数值确认动作的动作示例 3-4
3.2.5 通过 (Pass) 动作、结束动作、编码器数值确认动作执行时间的差异 3-5
3.2.6 不进行通过 (Pass) 动作的情况 3-6
3.2.7 通过 (Pass) 动作效果减弱的情况 3-7
3.2.8 加速度影响通过 (Pass) 动作的路径的情况 3-8
3.2.9 通过 (Pass) 开始位移 3-9
3.2.10 弧形运转功能 "仅适用 Ver.1.9 以上版本、4 轴机械手" 3-11
3.3 插补控制 3-12
3.3.1 PTP 控制 3-12
3.3.2 CP 控制 3-13
3.3.3 圆弧插补控制 3-13
3.3.4 自由曲线插补［Ver2.3 以上版本］ 3-14
3.3.5 高轨迹控制功能［Ver.2.61 以后］ 3-19
3.3.6 特异点回避功能［Ver.2.61 以后，6 轴专用］ 3-20
3.4 在动作指令之后有输出指令时 3-24
3.5 力限制功能 3-25
3.5.1 概要 3-25
3.5.2 各个轴的电流限制功能［V1.2 以上版本］ 3-25
3.5.3 前端力限制功能［V1.4 以上版本］ 3-27
3.5.4 碰撞检测功能 3-33

第 4 章 速度、加速度、减速度的指定

4.1 外部速度、内部速度 4-1
4.2 速度指定 4-2
4.3 外部加速度、外部减速度、内部加速度、内部减速度 4-3
4.4 加速度、减速度的设定 4-4
4.5 速度、加速度设定示例 4-5
4.6 最佳可搬运质量设定功能 4-8
4.6.1 模式 0 4-8
4.6.2 模式 1 4-9
4.6.3 模式 2 4-10
4.6.4 模式 3 4-11
4.6.5 设定所需要的使用条件 4-11
4.6.6 在设定时 4-12
4.7 "有关使用条件" 的最佳可搬质量设定功能 4-13
4.7.1 外部负荷条件值（前端负荷质量、负荷重心位置）与外部模式的设定方法 4-13
4.7.2 内部负荷条件值（前端负荷质量、负荷重心位置）与内部模式的设定方法 4-17
4.7.3 机械手的安装条件设定方法 4-18
4.7.4 最佳可搬运质量初始化设定的设定方法［V1.4 以后版本］ 4-21

第 5 章 视觉控制

5.1 视觉控制 5-1
5.1.1 视觉控制的用语 5-1
5.1.2 在使用视觉命令时的注意事项（视觉卡） 5-10
5.1.3 与原有的μVision-15 间的互换性 5-11

第 2 部 指令参考

第 6 章 指令的表示方法与分类

6.1 指令解说的表示方法 6-1
6.2 指令一览表 6-2
6.2.1 按字母顺序排列的指令一览 6-2
6.2.2 按功能单位排列的指令一览 6-2

第 7 章 PAC 语言的构成要素

7.1 新机械手语言 PAC 7-1
7.2 机械手语言 PAC 与原来的语言的关系 7-2
7.3 语言要素 7-3
7.4 名称 7-3
7.5 标识符 7-4
7.5.1 变量 7-4
7.5.2 函数 7-8
7.5.3 标签 7-8
7.5.4 程序 7-9
7.6 数据型 7-10
7.6.1 字符串 7-10
7.6.2 数值 7-10
7.6.3 矢量 7-10
7.6.4 姿势 (Pose) 7-10
7.6.5 I / O (ON / OFF) 7-10
7.7 数据型的转换 7-11
7.7.1 数值 7-11
7.7.2 字符和数值 7-11
7.7.3 姿势 (Pose) 型数据 7-11
7.8 常量 7-12
7.8.1 数值常量 7-12
7.8.2 字符串常量 7-14
7.8.3 矢量型常量 7-14
7.8.4 姿势 (Pose) 常量 7-14

7.9 式和运算符 7-16
7.9.1 赋值运算符 7-16
7.9.2 算术运算符 7-16
7.9.3 关系运算符 7-17
7.9.4 逻辑运算符 7-18
7.9.5 字符串运算符 7-18
7.9.6 矢量运算 7-19
7.9.7 位置运算 7-19
7.9.8 关节运算 7-20
7.9.9 齐次变换矩阵运算 7-20
7.9.10 算术运算符、逻辑运算符、关系运算符的优先顺序［Ver.1.5 版本以上］ 7-20
7.10 PAC 语言中单位的处理 7-22

第 8 章 PAC 语言的语法

8.1 语句与行 8-1
8.2 程序名称与定义 8-2
8.3 标签 8-3
8.4 字符组 8-4
8.5 保留字 8-5
8.6 定义语句 8-6
8.6.1 类型定义 8-6
8.6.2 函数 / 程序定义 8-9
8.6.3 用户坐标系的定义 8-9
8.6.4 文件夹变量定义 / EXTERN 变量定义［RC7_Ver. 2.2 以上版本］ 8-9
8.7 赋值语句 8-10
8.7.1 数值赋值语句 8-10
8.7.2 字符串赋值语句 8-10
8.7.3 矢量赋值语句 8-10
8.7.4 姿势 (Pose) 赋值语句 8-11
8.8 流程控制语句 8-13
8.8.1 无条件分支 8-13
8.8.2 条件分支 8-13
8.8.3 选择 8-13
8.8.4 重复 8-14
8.8.5 定义完成处理的调出 8-15
8.9 机械手控制语句 8-17
8.9.1 动作控制语句 8-17
8.9.2 停止控制语句 8-17
8.9.3 速度控制语句 8-18
8.9.4 时间控制语句 8-18
8.9.5 坐标转换语句 8-18

8.10 输入输出控制语句 8-19
8.10.1 DI / DO 控制语句 8-19
8.10.2 RS232C 控制语句 8-19
8.10.3 多功能教导器控制语句 8-19
8.11 多项任务控制语句 8-20
8.11.1 任务控制语句 8-20
8.11.2 信号 (Semaphore) 控制语句 8-20
8.11.3 专用信号 (Semaphore) 控制语句 8-20
8.12 时刻、日期控制 8-21
8.13 错误控制 8-22
8.13.1 错误控制指令 8-22
8.13.2 错误存放功能［Ver. 1.98 以上版本］ 8-22
8.14 系统信息 8-25
8.15 预处理程序 8-26
8.16 根据值进行调出与根据参照进行调出 8-27
8.16.1 根据值进行调出 8-27
8.16.2 根据参照进行调出 8-28
8.17 视觉控制 8-29
8.17.1 图像输入输出 8-29
8.17.2 视窗设定 8-29
8.17.3 描绘 8-30
8.17.4 图像处理 8-30
8.17.5 编码识别 8-31
8.17.6 标注 8-31
8.17.7 检索功能 8-31
8.17.8 结果获取 8-32
8.17.9 视觉 CAL 8-32

第 9 章 定义语句

9.1 程序名 9-1
9.2 干扰区域坐标 9-2
9.3 用户函数 9-6
9.4 初始坐标 9-7
9.5 工具坐标 9-8
9.6 工件坐标 9-9
9.7 局部 (Local) 变量 9-11
9.8 排列 9-19
9.9 文件夹功能 9-21

第 10 章 赋值语句

10.1 变量 10-1
10.2 矢量 10-2
10.3 形态 10-5
10.4 连接角 10-6
10.5 姿势 10-7
10.6 旋转分量 10-8
10.7 轴分量 10-12

第 11 章 流程控制语句

11.1 程序的停止 11-1
11.2 调出 11-4
11.3 重复 11-9
11.4 条件分支 11-17
11.5 无条件分支 11-21
11.6 备注 11-23

第 12 章 机械手控制语句

12.1 动作控制 12-1
12.2 形态控制 12-35
12.3 停止控制 12-41
12.4 速度控制 12-45
12.5 时间控制 12-60
12.6 坐标转换 12-63
12.7 干扰检查 12-67
12.8 伺服内部数据 12-71
12.9 电机电源 12-73
12.10 CAL 12-74
12.11 特殊控制 12-75

第 13 章 输入输出控制语句

13.1 I / O 端口 13-1
13.2 RS232C 及 Ethernet 端口 13-10
13.3 串行二进制通信（RS-232C 以及 Ethernet 端口） 13-16
13.4 多功能教导器 13-23

第 14 章 多项任务控制语句

14.1 任务控制 14-1
14.2 信号 (semaphore) 14-15
14.3 臂信号 (semaphore) 14-22
14.4 特权任务 14-30

第 15 章 函数

15.1 算术函数 15-1
15.2 三角函数 15-12
15.3 角度转换 15-19
15.4 速度转换 15-22
15.5 时间函数 15-24
15.6 矢量 15-25
15.7 姿势数据型转换 15-29
15.8 距离抽出 15-37
15.9 形态分量 15-38
15.10 角度分量 15-39
15.11 轴分量 15-40
15.12 旋转分量 15-43
15.13 姿势分量 15-47
15.14 位置函数 15-48
15.15 字符串函数 15-52

第 16 章 常量

16.1 内置常量 16-1

第 17 章 时间 / 日期控制

17.1 时间 / 日期 17-1

第 18 章 错误控制

18.1 错误信息 18-1

第 19 章 系统信息

19.1 系统 19-1
19.2 Log 19-4
19.3 动作模式 19-13

第 20 章 预处理器

20.1 符号常量、宏定义 20-1
20.2 文件读取 20-4
20.3 最佳化 20-6

第 21 章 视觉控制（机械手控制器：选择）

21.1 图像输出入 21-1
21.2 视窗设定 21-11
21.3 描绘 21-21
21.4 图像处理 21-41
21.5 编码识别 21-65
21.6 标注 21-68
21.7 检索功能 21-77
21.8 获取结果 21-94

第 22 章 附录

22.1 字符编码表 22-1
22.2 机械手形态 22-2
22.3 环境设定值 22-13
22.4 使用条件参数 22-14
22.5 保留字一览 22-24
22.6 旧语言指令对照表 (VS) 22-26
22.7 版本对应表 22-31
22.8 设定参数表 22-32

按照字母顺序排列的指令一览表

4 軸	6 軸	視覚装置	
◎	◎	◎	在所有机械手及视觉设备中均可使用
○	○	○	在所有机械手中均可使用，但是在 4 轴机械手、6 轴机械手或视觉设备中规格不同。
◎	V1.2		在 4 轴机械手和版本 1.2 之后的 6 轴机械手中均可使用

指令	功能	4 轴	6 轴	视觉装置	说明页
#					
#define	使用指定字符串替换程序中所指定的常量或宏名。	◎	◎	◎	20-1
#error	如果执行 #error 指令，则强制性置为编译错误。	◎	◎	◎	20-3
#include	读入预处理程序。	◎	◎	◎	20-4
#pragma optimize	指定按程序单位执行的最优化处理。	◎	◎	◎	20-6
#undef	将通过 #define 定义的符号常量或宏的定义置为无效。	◎	◎	◎	20-2
A					
ABS	得出算式的值的绝对值。	◎	◎	◎	15-1
ACCEL	指定当前所获取的臂组轴的内部加速度、内部减速度。	◎	◎		12-48
ACOS	得出反余弦 (arc cosine)。	◎	◎	◎	15-12
APPROACH	执行工具坐标系指定的绝对动作。	○	○		12-1
AREA	定义进行干扰检查的区域。	○	○		9-2
AREAPOS	将进行干扰检查的区域的中心位置和长方体的方向以位置型进行返回。	◎	◎		15-48
AREASIZE	将定义进行干扰检查的区域的长方体的大小 （各个边的长度）以矢量型进行返回。	◎	◎		15-49
ARRIVE	通过设定相对于动作命令的全移动距离的动作比例，使程序待机直到到达机械手所设定的动作比例。	◎	V1.2		12-29
ASC	变换为字符编码。	◎	◎	◎	15-52
ASIN	得出反正弦 (arc sine)。	◎	◎	◎	15-13
ATN	得出反正切 (arc tangent)。	◎	◎	◎	15-14
ATN2	得出以算式 1 除以算式 2 之后的反正切 (arc tangent)。	◎	◎	◎	15-15
AVEC	抽出近似矢量。	◎	◎		15-25
B					
BIN$	将算式的值转换为 2 进制数的字符串。	◎	◎	◎	15-53
BLOB	执行标注。	◎	◎	◎	21-68
BLOBCOPY	复制目标标签编号。	◎	◎	◎	21-75
BLOBLABEL	获取指定坐标的标签编号。	◎	◎	◎	21-73
BLOBMEASURE	进行目标标签编号的特征测量。	◎	◎	◎	21-71
BUZZER	鸣响蜂鸣器。	◎	◎		13-24
C					
CALL	调出程序并执行。	◎	◎	◎	11-4
CAMIN	将来自照相机的影像存放在图像储存器 （处理画面）。	◎	◎	◎	21-1
CAMLEVEL	设定照相机影像的输入等级。	◎	◎	◎	21-4
CAMMODE	设定存放照相机影像时的功能。	◎	◎	◎	21-2
change_bCap	进行按钮的字幕设定。	V1.5	V1.5		13-32
change_pCap	进行页面的字幕设定。	V1.5	V1.5		13-33
CHANGETOOL	变更工具坐标系。	◎	◎		12-63
CHANGEWORK	变更用户坐标系。	◎	◎		12-64
CHGEXTMODE	进行外部自动模式切换。	V1.98	V1.98		19-13
CHGINTMODE	进行内部自动模式切换。	V1.98	V1.98		19-14

4 轴	6 轴	视觉装置	
◎	◎	◎	在所有机械手及视觉设备中均可使用
○	○	○	在所有机械手中均可使用，但是在 4 轴机械手、6 轴机械手或视觉设备中规格不同。
◎	V1.2		在 4 轴机械手和版本 1.2 之后的 6 轴机械手中均可使用

指令	功能	4 轴	6 轴	视觉装置	说明页
CHR$	将 ASCII 编码转换为字符。	◎	◎	◎	15-54
CLEARLOG	对伺服控制 LOG 的记录进行初始化。	◎	◎		19-5
CLEARTRACELOG	清空跟踪 LOG 数据。	V2.7	V2.7		19-9
CLEARVARLOG	清空变量 LOG 数据。	V2.7	V2.7		19-12
CLRERR	清空错误。	V1.98	V1.98		18-3
CLRSPLINEPOINT	将自由曲线动作的通过点全部删除。	V2.3	V2.3		12-33
com_discom	为了结束二进制通信释放 RS-232C 端口。（释放 COM 端口）使用 Ethernet 外端口时，关闭外端口，切断连接。	V1.5	V1.5	V1.9	13-21
com_encom	为使用 RS-232C 端口或 Ethernet 用户端口进行二进制通信做准备。RS-232C 端口的情况下，为了进行二进制通信而占有端口。使用 Ethernet 用户端口时，打开用户端口，在端口与服务器之间建立连接。	V1.5	V1.5	V1.9	13-20
com_state	RS-232C 及获取 Ethernet 端口的状态。	V1.5	V1.5	V1.9	13-22
CONTINUERUN	进行连续启动。	V1.98	V1.98		14-10
COS	得出余弦 (cosine)。	◎	◎	◎	15-16
CREATESEM	生成信号 (semaphore)。	◎	◎		14-15
CURACC	获取当前所获取的臂组轴的内部加速度。	◎	◎		12-52
CURDEC	获取当前所获取的臂组轴的内部减速度。	◎	◎		12-54
CURERRSTATUS	返回错误发生状态。（特权任务专用命令）	V2.2	V2.2		14-34
CUREXJ	以 F 型获取附加轴的当前角度。	V1.5	V1.5		12-24
CUREXTACC	得出外部加速度的当前值。	V1.4	V1.4		12-58
CUREXTDEC	得出外部减速度的当前值。	V1.4	V1.4		12-58
CUREXTSPD	得出外部速度的当前值。	V1.4	V1.4		12-59
CURFIG	得出机械手形态的当前值。	◎	◎		12-35
CURJACC	功能获取当前所获取的臂组轴的内部轴加速度。	◎	◎		12-53
CURJDEC	获取当前所获取的臂组轴的内部轴减速度。	◎	◎		12-55
CURJNT	以 J 型取得机械手的当前角度。	○	○		12-21
CURJSPD	获取当前所获取的臂组轴的内部移动轴速度。	◎	◎		12-56
CUROPTMODE	获取动作模式。	V1.98	V1.98		19-15
CURPOS	以 P 型数据得出工具坐标系中的当前位置。	○	○		12-22
CURSPD	获取当前所获取的臂组轴的内部移动速度。	◎	◎		12-57
CURTOOL	得出当前所设定的 TOOL 编号。	V1.4	V1.4		12-65
CURTRN	以 T 型数据得出工具坐标系中的当前位置。	◎	◎		12-23
CURWORK	得出当前所设定的 WORK 编号。	V1.4	V1.4		12-66
D					
DATE$	得出当前的日期。	◎	◎		17-1
DEADMANSTATE	双重安全开关的状态。 0: OFF、 1: ON	V2.2	V2.2		14-13
DECEL	指定当前所获取的臂组轴的内部减速度。	◎	◎		12-50
DEF FN	定义用户定义函数。	◎	◎	◎	9-6
DEFDBL	定义 D 型变量 （双精度实数型变量）。双精度实数的范围是 -1.7D + 308 ～ 1.7D + 308。	◎	◎	◎	9-12
DEFEND	保护任务。	◎	◎		14-5
DEFINT	定义 I 型变量 （整数型变量）。整数的范围是 –2147483648 ～ 2147483647。	◎	◎	◎	9-11
DEFIO	定义对应输出入端口的 IO 变量 （I / O 变量）。	◎	◎	◎	9-17

4 轴	6 轴	视觉装置	
◎	◎	◎	在所有机械手及视觉设备中均可使用
○	○	○	在所有机械手中均可使用，但是在 4 轴机械手、6 轴机械手或视觉设备中规格不同。
◎	V1.2		在 4 轴机械手和版本 1.2 之后的 6 轴机械手中均可使用

指令	功能	4 轴	6 轴	视觉装置	说明页
DEFJNT	定义 J 型变量 （关节型变量）。	○	○		9-15
DEFPOS	定义 P 型变量 （位置型变量）。	○	○		9-14
DEFSNG	定义 F 型变量 （单精度实数型变量）。单精度实数的范围是 -3.4E + 38 ～ 3.4E + 38。	◎	◎	◎	9-11
DEFSTR	定义 S 型变量 （字符串型变量）。字符串的长度为最大 243 个字符。	◎	◎	◎	9-12
DEFTRN	定义 T 型变量 （齐次变换型变量）。	◎	◎		9-16
DEFVEC	定义 V 型变量 （矢量型变量）。	◎	◎		9-13
DEGRAD	将单位转换为弧度。	◎	◎	◎	15-19
DELAY	在指定的时间内，停止程序的执行。	◎	◎	◎	12-60
DELETESEM	删除信号 (semaphore)。	◎	◎		14-18
DEPART	通过工具坐标系的指定执行相对动作。	○	○		12-4
DESTEXJ	以 F 型获取附加轴的以当前的动作命令所指定的目标位置。停止时，获取当前的位置 （指令值）。	V1.5	V1.5		12-28
DESTJNT	以 J 型获取当前的动作命令目标位置。机械手停止时，获取当前的位置 （指令值）。	○	○		12-25
DESTPOS	以 P 型获取当前的动作命令目标位置。机械手停止时，获取当前的位置 （指令值）。	○	○		12-26
DESTTRN	以 T 型获取当前的动作命令目标位置。机械手停止时，获取当前的位置 （指令值）。	◎	◎		12-27
DETECT ON/OFF	检测夹治具输入的上升以及下降，将此时机械手的位置存放至变量。	V3.2	V3.2		13-8
DIM	定义排列。	◎	◎	◎	9-19
disp_page	进行页面的显示。	V1.5	V1.5		13-34
DIST	返回 2 点间的距离。	V1.8	V1.8		15-37
DO ～ LOOP	进行迭代 （循环）。	◎	◎	◎	11-9
DRAW	通过工件作业坐标系的指定执行相对动作。	◎	◎		12-6
DRIVE	进行各轴的相对动作。	◎	◎		12-8
DRIVEA	进行各轴的绝对动作。	◎	◎		12-10
E					
END	定义基于程序的动作结束。	◎	◎	◎	11-1
ERRMSG$	赋予错误讯息。	◎	◎	◎	18-1
EXECAL	CAL 执行	V1.98	V1.98		12-74
EXIT DO	执行从 DO ～ LOOP 中的强制退出。	◎	◎	◎	11-11
EXIT FOR	执行从 FOR ～ NEXT 中的强制退出。	◎	◎	◎	11-14
EXP	得出将自然对数作为基数的指数函数。	◎	◎	◎	15-2
EXTERN	定义参照用其他的程序所定义的 FOLDER 变量的行为。	V2.2	V2.2		9-24
EXTSPEED	外部速度设定	V1.98	V1.98		12-59
F					
FALSE	赋予布尔值的假 (0) 的值。	◎	◎	◎	16-4
FIG	抽出形态。	○	○		15-38
FIGAPRL	计算能够执行 CP 动作的接近位置和基准位置的形态。	○	○		12-37
FIGAPRP	计算能够执行 PTP 动作的接近位置与基准位置的形态。	○	○		12-39
FLUSH	清空输入缓冲区。	◎	◎	◎	13-15
FLUSHSEM	释放信号 (semaphore) 等待任务。	◎	◎		14-19

4 轴	6 轴	视觉装置	
◎	◎	◎	在所有机械手及视觉设备中均可使用
○	○	○	在所有机械手中均可使用，但是在 4 轴机械手、6 轴机械手或视觉设备中规格不同。
◎	V1.2		在 4 轴机械手和版本 1.2 之后的 6 轴机械手中均可使用

指令	功能	4 轴	6 轴	视觉装置	说明页
FOLDER	定义从外部程序可访问的局部 (Local) 变量。	V2.2	V2.2		9-21
FOR ～ NEXT	循环执行位于 FOR ～ NEXT 之间的一系列的指令。	◎	◎	◎	11-12
G					
GETENV	获取系统的环境设定值。	◎	◎	◎	19-1
GETERR	从通过错误存放功能定义的缓冲中获取错误编码。	V1.98	V1.98		18-2
GETERRLVL	赋予错误编码的等级。	V1.98	V1.98		18-4
GetJntData	获取指定轴的伺服内部数据。（对应 Ver. 1.5 以上版本）				12-72
GETLANGUAGE	获取当前的语言设定。	V2.2	V2.2		19-3
GETSPLINEPOINT	取出已经登录的自由曲线动作的通过点。	V2.3	V2.3		12-34
GetSrvData	获取机械手轴的伺服内部数据。（对应 Ver. 1.5 以上版本）				12-71
GIVEARM	释放当前正在获取的臂组。臂组 0 （仅限机械手）时，开放机械手控制权。	◎	◎		14-27
GIVESEM	释放信号 (semaphore) 等待任务。	◎	◎		14-20
GIVEVIS	释放视觉处理权。	◎	◎		14-29
GOHOME	向用 HOME 语句定义的位置点 （原点）移动。	◎	◎		12-12
GOSUB	调用子程序。	◎	◎	◎	11-6
GOTO	无条件执行程序的分支。	◎	◎	◎	11-21
H					
HALT	停止程序的执行。	◎	◎		12-42
HEX$	以字符串形式得出从 10 进制数变换为 16 进制数后的值。	◎	◎	◎	15-57
HOLD	暂时停止程序的执行。	◎	◎		12-41
HOME	将任意的坐标作为初始位置定义。	○	○		9-7
I					
IF ～ END IF	执行 IF ～ END IF 之间的条件式中的条件判断。	◎	◎	◎	11-17
IF ～ THEN ～ ELSE	执行逻辑表达式的条件判断。	◎	◎	◎	11-18
IN	从通过 I / O 变量表示的 I / O 端口中读取数据。	◎	◎	◎	13-1
INIT	根据特权任务参数的状态，进行电机 ON、 CAL、速度的设定。	V1.7	V1.7		14-30
INITWAITERR	初始化错误检测。（特权任务专用命令）	V2.2	V2.2		14-32
INPUT	得出来自 RS232C 及 Ethernet 端口的数据。	◎	◎	◎	13-10
INPUTB	从 RS-232C 及 Ethernet 端口输入 1 字节 (Byte) 数据。	V1.5	V1.5	V1.9	13-17
INT	得出不超过被指定值的最大的整数值。	◎	◎	◎	15-3
INTERRUPT ON/OFF	中断机械手的动作。	◎	◎		12-43
IOBLOCK ON/OFF	在执行动作指令时，同步执行 I / O 指令、运算指令等非动作指令。	◎	◎		13-3
J					
J2P	从关节型转换为位置型。	○	○		15-29
J2T	从关节型转换为齐次变换型。	○	○		15-30
JACCEL	指定当前所获取的臂组轴的内部轴加速度、内部轴减速度。	◎	◎		12-49
JDECEL	指定当前所获取的臂组轴的内部轴减速度。	◎	◎		12-51
JOINT	从关节型坐标抽出角度。	○	○		15-39
JSPEED	指定当前所获取的臂组轴的内部轴速度。	◎	◎		12-47

4 轴	6 轴	视觉装置	
◎	◎	◎	在所有机械手及视觉设备中均可使用
○	○	○	在所有机械手中均可使用，但是在 4 轴机械手、6 轴机械手或视觉设备中规格不同。
◎	V1.2		在 4 轴机械手和版本 1.2 之后的 6 轴机械手中均可使用

指令	功能	4 轴	6 轴	视觉装置	说明页
K					
KILL	强制结束任务。	◎	◎		14-3
KILLALL	强制结束特权任务以外的所有的任务。（相当于程序复位）	V1.98	V1.98		14-9
L					
LEFT$	抽出左部分字符串。	◎	◎	◎	15-58
LEN	以字节 (Byte) 数得出字符串的长度。	◎	◎	◎	15-59
LET	对变量赋值。	○	○	○	10-1
LETA	赋值于齐次变换型的近似矢量中。		◎		10-2
LETENV	设定系统的环境设定值。	◎	◎	◎	19-2
LETF	赋值于位置型或齐次变换型的形态分量中。	◎	◎		10-5
LETJ	赋值于关节型的指定连接角中。	◎	◎		10-6
LETO	赋值于齐次变换型的定向矢量中。	◎	◎		10-3
LETP	赋值于位置型或齐次变换型的位置矢量中。	◎	◎		10-4
LETR	赋值于位置型的 3 个旋转分量中。		◎		10-7
LETRX	赋值于位置型的 X 轴旋转分量中。		◎		10-8
LETRY	赋值于位置型的 Y 轴旋转分量中。		◎		10-9
LETRZ	赋值于位置型的 Z 轴旋转分量中。		◎		10-10
LETT	赋值于位置型的 T 分量中。	◎			10-11
LETX	对矢量型 / 赋值于位置型 / 齐次变换型的 X 轴分量中。	◎	◎		10-12
LETY	赋值于矢量型 / 位置型 / 齐次变换型的 Y 轴分量中。	◎	◎		10-13
LETZ	赋值于矢量型 / 位置型 / 齐次变换型的 Z 轴分量中。	◎	◎		10-14
LINEINPUT	从 RS232C 及 Ethernet 端口中读取数据直到间隔符，并对字符串型变量赋值。	◎	◎	◎	13-12
LINPUTB	从 RS-232C 及 Ethernet 端口输入多字节 (Byte) 数据。	V1.5	V1.5	V1.9	13-19
LOCKSTATE	获取锁定的状态。 0: OFF、 1: ON	V2.2	V2.2		14-12
LOG	得出自然对数。	◎	◎	◎	15-4
LOG10	得出常用对数。	◎	◎	◎	15-5
LPRINTB	从 RS-232C 及 Ethernet 端口输出多字节 (Byte) 数据。	V1.5	V1.5	V1.9	13-18
M					
MAGNITUDE	得出矢量的大小。	◎	◎		15-28
MAX	抽出最大值。	◎	◎	◎	15-7
MID$	从字符串中取出指定的字符数部分的字符串。	◎	◎	◎	15-60
MIN	抽出最小值。	◎	◎	◎	15-8
MOTOR {ON | OFF}	执行电机电源的 ON / OFF。	V1.98	V1.98		12-73
MOVE	将机械手向指定坐标移动。通过使用 EX 选择 （附加轴相对动作）或 EXA 选择 （附加轴绝对动作），可以使机械手和附加轴同步 （同时出发和到达）动作。	○	○		12-13
MPS	将指定的速度值 （mm/sec）转换为对 CP 动作时的最大内部速度的比率 （%）。	◎	◎		15-22
N					
NORMTRN	进行齐次变换型的正规化计算。	V1.8	V1.8		15-36

4 轴	6 轴	視覚装置	
◎	◎	◎	在所有机械手及视觉设备中均可使用
○	○	○	在所有机械手中均可使用，但是在 4 轴机械手、6 轴机械手或视觉设备中规格不同。
◎	V1.2		在 4 轴机械手和版本 1.2 之后的 6 轴机械手中均可使用

指令	功能	4 轴	6 轴	视觉装置	说明页
O					
OFF	赋予 OFF (0) 值。	◎	◎	◎	16-1
ON	赋予 ON (1) 值。	◎	◎	◎	16-2
ON ～ GOSUB	根据式的值，调出相应的子程序的调出。	◎	◎	◎	11-7
ON ～ GOTO	执行根据表达式的值的无条件分支。	◎	◎	◎	11-22
ORD	变换为字符编码。	◎	◎	◎	15-61
OUT	向通过 I / O 变量表示的 I / O 端口输出数据。	◎	◎	◎	13-2
OVEC	抽出定向矢量。	◎	◎		15-26
P					
P2J	从位置型转换为关节型。	○	○		15-31
P2T	从位置型转换为齐次变换型。	○	○		15-32
PI	赋予 π 值。	◎	◎	◎	16-3
POSCLR	将轴的当前位置强制性置为 0mm 或 0 度。	V1.5	V1.5		12-31
POSRX	抽出 X 轴旋转分量。		◎		15-43
POSRY	抽出 Y 轴旋转分量。		◎		15-44
POSRZ	抽出 Z 轴旋转分量。		◎		15-45
POST	抽出 T 轴旋转分量。	◎			15-46
POSTUREAREA	定义高感度位置姿势检测功能的设定。	V3.0	V3.0		9-4
POSX	抽出 X 分量。	○	○		15-40
POSY	抽出 Y 分量。	○	○		15-41
POSZ	抽出 Z 分量。	○	○		15-42
POW	求取幂乘。	◎	◎	◎	15-6
PRINT	从 RS232C 及 Ethernet 端口输出数据。	◎	◎	◎	13-13
PRINTB	向 RS-232C 及 Ethernet 端口输出 1 字节 (Byte) 数据。	V1.5	V1.5	V1.9	13-16
PRINTDBG	向调试视窗输出数据。	◎	◎		13-24
PRINTLBL	向用户定义按钮设定标签 （字幕）。	◎	◎		13-25
PRINTMSG	将讯息附上字幕和图标，在多功能教导器的彩色液晶中显示。	◎	◎		13-23
PRINTWARNING	在教导器警告讯息显示部分显示讯息。	V2.2	V2.2		13-25
PROGRAM	定义程序名。	◎	◎	◎	9-1
PVEC	抽出位置矢量。	○	○		15-27
R					
RAD	将通过弧度赋予的数值转换为角度。	◎	◎	◎	15-20
RADDEG	将单位转换为度。	◎	◎	◎	15-21
REM	进行备注的记述。	◎	◎	◎	11-23
REPEAT ～ UNTIL	执行后迭代。	◎	◎	◎	11-15
RESET	将 I / O 端口置为 OFF。	◎	◎	◎	13-7
RESETAREA	初始化干扰检查。	◎	◎		12-68
RESETPOSTUREAREA	将高感度位置姿势检测功能设为无效。	V3.0	V3.0		12-70
RETURN	从子程序中返回。	◎	◎	◎	11-8
RIGHT$	抽出字符串的右部分。	◎	◎	◎	15-62
RND	发生大于 0 小于 1 的随机数。	◎	◎	◎	15-9
ROBOTSTOP	进行机械手停止。	V1.98	V1.98		14-11

4 轴	6 轴	视觉装置	
◎	◎	◎	在所有机械手及视觉设备中均可使用
○	○	○	在所有机械手中均可使用，但是在 4 轴机械手、6 轴机械手或视觉设备中规格不同。
◎	V1.2		在 4 轴机械手和版本 1.2 之后的 6 轴机械手中均可使用

指令	功能	4 轴	6 轴	视觉装置	说明页
ROTATE	执行指定的旋转轴的旋转动作。	○	○		12-17
ROTATEH	执行以近似矢量为轴的旋转动作。	◎	◎		12-19
RPM	将指定轴编号的旋转数 （rpm）转换为对 PTP 动作时的最大内部速度的比率 （%）。	V2.8	V2.8		15-23
RUN	并列启动其他程序。	◎	◎		14-1
RVEC	抽出姿势。		◎		15-47
S					
SEC	将以秒为单位赋予的数值变换为毫秒。	◎	◎	◎	15-24
SELECT CASE	执行多个条件判断。	◎	◎	◎	11-19
SEMIDSTATE	返回所指定的信号 (semaphore) ID 是有效还是无效。	V2.2	V2.2		14-14
SET	将 I / O 端口置为 ON。	◎	◎	◎	13-5
set_button	进行按钮参数的设定。	V1.5	V1.5		13-26
set_page	进行页面参数的设定。	V1.5	V1.5		13-30
SETAREA	选择进行干扰检查的区域。	◎	◎		12-67
SETERR	将用户定义错误设定到 I 型变量区域。	V1.98	V1.98		18-1
SETOCCUPATIONTIME	变更特权任务的占有时间。	V2	V2		14-31
SETPOSTUREAREA	将高感度位置姿势检测功能设为有效。	V3.0	V3.0		12-69
SETSPLINEPOINT	登录自由曲线动作的通过点。	V2.3	V2.3		12-32
SGN	调查符号。	◎	◎	◎	15-10
SHCIRCLE	检索圆。	◎	◎	◎	21-92
SHCLRMODEL	消除登录模型。	◎	◎	◎	21-81
SHCOPYMODEL	复制登录模型。	◎	◎	◎	21-80
SHCORNER	检索棱角。	◎	◎	◎	21-88
SHDEFCIRCLE	设定圆检索的条件。	◎	◎	◎	21-91
SHDEFCORNER	设定棱角检索的条件。	◎	◎	◎	21-87
SHDEFMODEL	进行检索模型的登录。	◎	◎	◎	21-77
SHDISPMODEL	将登录模型显示在画面上。	◎	◎	◎	21-82
SHMODEL	检索模型。	◎	◎	◎	21-83
SHREFMODEL	参照登录模型数据。	◎	◎	◎	21-79
SIN	得出正弦 (sine)。	◎	◎	◎	15-17
SPEED	指定当前所获取的臂组轴的内部移动速度。	◎	◎		12-45
SPRINTF$	将表达式变换为所指定的格式化，并作为字符串返回。	◎	◎	◎	15-55
SQR	得出平方根。	◎	◎	◎	15-11
ST_aspACLD	变更内部负荷条件值。以前端负荷质量 (g)、负荷重心位置 (mm) （注 1）指定所有负荷条件值。	V1.9	V1.9		12-75
ST_aspChange	选择最佳搬运质量设定模式的内部模式。	V1.9	V1.9		12-76
ST_OffSrvLock	解除指定轴的伺服锁定。（4 轴机械手专用）	V1.9			12-87
ST_OnSrvLock	将指定轴置为伺服锁定状态。（4 轴机械手专用）	V1.9			12-86
ST_ResetCompControl	将力限制功能设为无效。（6 轴专用命令）		V1.9		12-90
ST_ResetCompEralw	将力限制时的工具端的位置、姿势偏差允许值初始化。（6 轴专用命令）		V1.9		12-103
ST_ResetCompJLimit	将力限制时的电流限制值初始化。（力限制特殊功能） （6 轴专用命令）		V1.9		12-99
ST_ResetCompRate	将力限制时的柔度的比例初始化。（6 轴专用命令）		V1.9		12-95

4 轴	6 轴	視覚装置	
◎	◎	◎	在所有机械手及视觉设备中均可使用
○	○	○	在所有机械手中均可使用，但是在 4 轴机械手、6 轴机械手或视觉设备中规格不同。
◎	V1.2		在 4 轴机械手和版本 1.2 之后的 6 轴机械手中均可使用

指令	功能	4 轴	6 轴	视觉装置	说明页
ST_ResetCompVMode	将力限制时的速度控制模式设为无效。（力限制特殊功能）（6 轴专用命令）		V1.9		12-101
ST_ResetCurLmt	解除指定轴的电机电流限制。	V1.9	V1.9		12-83
ST_ResetDampRate	将力限制时的粘性比例初始化。（6 轴专用命令）		V1.9		12-105
ST_ResetEralw	将指定轴的偏差允许值返回到默认值。	V1.9	V1.9		12-85
ST_ResetFrcAssist	将力限制时的补偿力初始化。（力限制特殊功能）（6 轴专用命令）		V1.9		12-97
ST_ResetFrcLimit	将力限制比例初始化。（6 轴专用命令）		V1.9		12-93
ST_ResetGravity	将重力平衡设为无效。	V1.9	V1.9		12-78
ST_ResetGrvOffset	将重力补偿值的修正设为无效。	V1.9	V1.9		12-80
ST_ResetZBalance	将重力补偿值的修正设为无效。（4 轴专用）	V1.9			12-107
ST_SetCompControl	将力限制功能设为有效。（6 轴专用命令）		V1.9		12-88
ST_SetCompEralw	设定力限制时的工具端的位置、姿势偏差允许值。（6 轴专用命令）		V1.9		12-102
ST_SetCompFControl	将力限制功能设为有效。（6 轴专用命令）		V1.9		12-89
ST_SetCompJLimit	设定力限制时的电流限制值。（力限制特殊功能）（6 轴专用命令）		V1.9		12-98
ST_SetCompRate	设定力限制时的柔度的比例。（6 轴专用命令）		V1.9		12-94
ST_SetCompVMode	设定力限制时的速度控制模式。（力限制特殊功能） （6 轴专用命令）		V1.9		12-100
ST_SetCurLmt	限制指定轴的电机电流值。	V1.9	V1.9		12-81
ST_SetDampRate	设定力限制时的粘性比例。（6 轴专用命令）		V1.9		12-104
ST_SetEralw	变更指定轴的偏差允许值。	V1.9	V1.9		12-84
ST_SetFrcAssist	设定力限制时的补偿力。（力限制特殊功能语句）（6 轴专用命令）		V1.9		12-96
ST_SetFrcCoord	选择力限制设定坐标系。（6 轴专用命令）		V1.9		12-91
ST_SetFrcLimit	设定力限制比例。（6 轴专用命令）		V1.9		12-92
ST_SetGravity	修正各个关节的静负荷 （重力转矩），设定重力平衡。	V1.9	V1.9		12-77
ST_SetGrvOffset	通过各个关节的重力转矩修正重力补偿值。	V1.9	V1.9		12-79
ST_SetZBalance	设定 Z 轴、 T 轴的重力补偿值。（4 轴专用）	V1.9			12-106
STARTLOG	开始伺服控制 LOG 的记录。	◎	◎		19-4
STARTTRACELOG	开始记录跟踪 LOG。	V2.7	V2.7		19-7
STARTVARLOG	开始记录变量 LOG。	V2.7	V2.7		19-10
STATUS	得出程序的状态。	◎	◎		14-6
STOP	结束程序的执行。	◎	◎	◎	11-2
STOPEND	是对连续启动或随附 CYCLE 选项启动的程序进行循环停止的语句。循环启动包含该语句的程序时，即使执行该语句也不会影响动作。	◎	◎		11-3
STOPLOG	停止伺服控制 LOG 的记录。	◎	◎		19-6
STOPTRACELOG	结束对跟踪 LOG 的记录。	V2.7	V2.7		19-8
STOPVARLOG	结束对变量 LOG 的记录。	V2.7	V2.7		19-11
STR$	将数值变换为字符串。	◎	◎	◎	15-64
STRPOS	得出字符串的位置。	◎	◎	◎	15-63
SUSPEND	暂停任务。	◎	◎		14-4
SUSPENDALL	停止特权任务以外的所有的程序。	V1.98	V1.98		14-8
SYSSTATE	获取控制器的状态。	V1.98	V1.98		19-16

4 軸	6 軸	視觉装置	
◎	◎	◎	在所有机械手及视觉设备中均可使用
○	○	○	在所有机械手中均可使用，但是在 4 轴机械手、6 轴机械手或视觉设备中规格不同。
◎	V1.2		在 4 轴机械手和版本 1.2 之后的 6 轴机械手中均可使用

指令	功能	4 轴	6 轴	视觉装置	说明页
T					
T2J	从关节型转换为齐次变换型。	◎	◎		15-33
T2P	从齐次变换型转换为位置型。	◎	◎		15-34
TAKEARM	获取臂组。当省略了臂组 NO 时，获取臂组 0 （仅限机械手）的控制权。获取时，将内部速度、加速度、减速度设定为 100。当获取的臂组包含机械手轴时，将工具坐标、工件坐标返回到原点。	◎	◎		14-22
TAKEARMSTATE	返回臂组控制权的获取状态。 0: 尚未获取、1: 已经获取	V2.2	V2.2		14-12
TAKESEM	获取具有指定信号 (semaphore) ID 的信号 (semaphore)。	◎	◎		14-21
TAKEVIS	获取视觉处理权。	◎	◎		14-28
TAN	得出正切 (tangent)。	◎	◎	◎	15-18
TIME$	得出当前的时间。	◎	◎		17-2
TIMER	得出经过时间。	◎	◎		17-3
TINV	计算齐次变换型的逆矩阵。	◎	◎		15-35
TOOL	定义工具坐标系。	○	○		9-8
TOOLPOS	将工具坐标系以位置型进行返回。	◎	◎		15-50
TRUE	赋予布尔值的真 (1) 的值。	◎	◎	◎	16-5
V					
VAL	从字符串变换为数值。	◎	◎	◎	15-65
VER$	获取各个模块的版本。	◎	◎	◎	19-3
VISBINA	将画面进行 2 值化处理。	◎	◎	◎	21-47
VISBINAR	将画面进行 2 值化显示。	◎	◎	◎	21-49
VISBRIGHT	指定描绘时的辉度值。	◎	◎	◎	21-23
VISCAMOUT	将来自照相机的影像显示在监视器上。	◎	◎	◎	21-5
VISCIRCLE	在画面上描绘圆。	◎	◎	◎	21-31
VISCLS	用指定辉度全面涂抹 （清空）模式所指定的画面。	◎	◎	◎	21-24
VISCOPY	复制画面。	◎	◎	◎	21-54
VISCROSS	在画面上描绘十字标记。	◎	◎	◎	21-35
VISDEFCHAR	指定字符的大小和显示方法。	◎	◎	◎	21-38
VISDEFTABLE	设定照相机影像的输入、图像输出时的一览表数据。	◎	◎	◎	21-8
VISEDGE	测量视窗内的边缘。	◎	◎	◎	21-60
VISELLIPSE	在画面上描绘椭圆。	◎	◎	◎	21-32
VISFILTER	对图像进行过滤器处理。	◎	◎	◎	21-50
VISGETNUM	从存放存储器中获取图像处理结果。	◎	◎	◎	21-94
VISGETP	从存放存储器 （处理画面）中获取指定坐标的辉度。	◎	◎	◎	21-42
VISGETSTR	获取编码识别结果。	◎	◎	◎	21-95
VISHIST	得到画面的直方柱状体 （辉度分布）。	◎	◎	◎	21-43
VISLEVEL	根据直方柱状体结果求取 2 值化标准值。	◎	◎	◎	21-45
VISLINE	在画面上描绘直线。	◎	◎	◎	21-27
VISLOC	指定字符的显示位置。	◎	◎	◎	21-36
VISMASK	进行图像间运算。	◎	◎	◎	21-52
VISMEASURE	测量视窗内的特征 （面积、重心、主轴角）。	◎	◎	◎	21-55
VISOVERLAY	将描绘画面的信息显示在监视器上。	◎	◎	◎	21-7
VISPLNOUT	将存放存储器的图像显示在监视器上。	◎	◎	◎	21-6

4 轴	6 轴	視覚装置	
◎	◎	◎	在所有机械手及视觉设备中均可使用
○	○	○	在所有机械手中均可使用，但是在 4 轴机械手、6 轴机械手或视觉设备中规格不同。
◎	V1.2		在 4 轴机械手和版本 1.2 之后的 6 轴机械手中均可使用

指令	功能	4 轴	6 轴	视觉装置	说明页
VISPOSX	从存放存储器中取得图像处理结果 （ X 坐标）。	◎	◎	◎	21-96
VISPOSY	从存放存储器中取得图像处理结果 （ Y 坐标）。	◎	◎	◎	21-97
VISPRINT	在画面上显示字符、数字。	◎	◎	◎	21-40
VISPROJ	测量视窗内的投影数据。	◎	◎	◎	21-58
VISPTP	在画面上描绘连接 2 点间的直线。	◎	◎	◎	21-28
VISPUTP	在画面上描绘点。	◎	◎	◎	21-26
VISREADQR	读取 QR 编码。	◎	◎	◎	21-65
VISRECT	在画面上描绘矩形。	◎	◎	◎	21-29
VISREFCAL	获取 CAL （视觉－机械手坐标转换）数据。	◎	◎	◎	21-99
VISREFHIST	读出直方柱状体结果。	◎	◎	◎	21-44
VISREFTABLE	参照一览表的数据。	◎	◎	◎	21-10
VISSCREEN	指定描绘的画面。	◎	◎	◎	21-21
VISSECT	在画面上描绘扇形。	◎	◎	◎	21-34
VISSTATUS	监视各个命令的处理结果。	◎	◎	◎	21-98
VISWORKPLN	指定处理目标的存放存储器 （处理画面）。	◎	◎	◎	21-41
W					
WAIT	执行条件程序的停止。	◎	◎		12-62
WAITERROR	检测错误发生。（特权任务专用命令）	V2.2	V2.2		14-33
WHILE ～ WEND	执行前迭代。	◎	◎	◎	11-16
WINDCLR	消除设定的视窗信息。	◎	◎	◎	21-16
WINDCOPY	复制视窗的数据。	◎	◎	◎	21-17
WINDDISP	描绘所设定的视窗。	◎	◎	◎	21-20
WINDMAKE	指定图像处理的范围。	◎	◎	◎	21-11
WINDREF	取得视窗的信息。	◎	◎	◎	21-19
WORK	定义用户坐标系。	○	○		9-9
WORKATTRIBUTE	获取指定工件坐标系的属性值。	V3.2	V3.2		15-51
WORKPOS	将用户坐标系以位置型进行返回。	◎	◎		15-51
WRITE	从 RS232C 及 Ethernet 端口输出数据。	◎	◎	◎	13-14

以功能为单位的指令一览表

4 軸	6 軸	視覚装置	
◎	◎	◎	在所有机械手及视觉设备中均可使用
○	○	○	在所有机械手中均可使用，但是在 4 轴机械手、6 轴机械手或视觉设备中规格不同。
◎	V1.2		在 4 轴机械手和版本 1.2 之后的 6 轴机械手中均可使用

功能分类	指令	功能	4 轴	6 轴	视觉装置	说明页
定义语句						
程序名	PROGRAM	定义程序名。	◎	◎	◎	9-1
干扰区域坐标	AREA	定义进行干扰检查的区域。	○	○		9-2
	POSTUREAREA	定义高感度位置姿势检测功能的设定。	V3.0	V3.0		9-4
用户函数	DEF FN	定义用户定义函数。	◎	◎	◎	9-6
初始坐标	HOME	将任意的坐标作为初始位置定义。	○	○		9-7
工具坐标	TOOL	定义工具坐标系。	○	○		9-8
工件坐标	WORK	定义用户坐标系。	○	○		9-9
局部 (Local) 变量	DEFINT	定义 I 型变量 （整数型变量）。整数的范围是 –2147483648 ～ 2147483647。	◎	◎	◎	9-11
	DEFSNG	定义 F 型变量 （单精度实数型变量）。单精度实数的范围是 -3.4E + 38 ～ 3.4E + 38。	◎	◎	◎	9-11
	DEFDBL	定义 D 型变量 （双精度实数型变量）。双精度实数的范围是 -1.7D + 308 ～ 1.7D + 308。	◎	◎	◎	9-12
	DEFSTR	定义 S 型变量 （字符串型变量）。字符串的长度为最大 243 个字符。	◎	◎	◎	9-12
	DEFVEC	定义 V 型变量 （矢量型变量）。	◎	◎		9-13
	DEFPOS	定义 P 型变量 （位置型变量）。	○	○		9-14
	DEFJNT	定义 J 型变量 （关节型变量）。	○	○		9-15
	DEFTRN	定义 T 型变量 （齐次变换型变量）。	◎	◎		9-16
	DEFIO	定义对应输出入端口的 IO 变量 （I / O 变量）。	◎	◎	◎	9-17
排列	DIM	定义排列。	◎	◎	◎	9-19
文件夹功能	FOLDER	定义从外部程序可访问的局部 (Local) 变量。	V2.2	V2.2		9-21
	EXTERN	定义参照用其他的程序所定义的 FOLDER 变量的行为。	V2.2	V2.2		9-24
赋值语句						
变量	LET	对变量赋值。	○	○	○	10-1
矢量	LETA	赋值于齐次变换型的近似矢量中。		◎		10-2
	LETO	赋值于齐次变换型的定向矢量中。	◎	◎		10-3
	LETP	赋值于位置型或齐次变换型的位置矢量中。	◎	◎		10-4
形态	LETF	赋值于位置型或齐次变换型的形态分量中。	◎	◎		10-5
连接角	LETJ	赋值于关节型的指定连接角中。	◎	◎		10-6
姿势	LETR	赋值于位置型的 3 个旋转分量中。		◎		10-7
旋转分量	LETRX	赋值于位置型的 X 轴旋转分量中。		◎		10-8
	LETRY	赋值于位置型的 Y 轴旋转分量中。		◎		10-9
	LETRZ	赋值于位置型的 Z 轴旋转分量中。		◎		10-10
	LETT	赋值于位置型的 T 分量中。	◎			10-11

4 轴	6 轴	視覚装置	
◎	◎	◎	在所有机械手及视觉设备中均可使用
○	○	○	在所有机械手中均可使用，但是在 4 轴机械手、6 轴机械手或视觉设备中规格不同。
◎	V1.2		在 4 轴机械手和版本 1.2 之后的 6 轴机械手中均可使用

功能分类	指令	功能	4 轴	6 轴	视觉装置	说明页
轴分量	LETX	对矢量型 / 赋值于位置型 / 齐次变换型的 X 轴分量中。	◎	◎		10-12
	LETY	赋值于矢量型 / 位置型 / 齐次变换型的 Y 轴分量中。	◎	◎		10-13
	LETZ	赋值于矢量型 / 位置型 / 齐次变换型的 Z 轴分量中。	◎	◎		10-14
流程控制语句						
程序的停止	END	定义基于程序的动作结束。	◎	◎	◎	11-1
	STOP	结束程序的执行。	◎	◎	◎	11-2
	STOPEND	是对连续启动或随附 CYCLE 选项启动的程序进行循环停止的语句。循环启动包含该语句的程序时，即使执行该语句也不会影响动作。	◎	◎		11-3
调出	CALL	调出程序并执行。	◎	◎	◎	11-4
	GOSUB	调用子程序。	◎	◎	◎	11-6
	ON ～ GOSUB	根据式的值，调出相应的子程序的调出。	◎	◎	◎	11-7
	RETURN	从子程序中返回。	◎	◎	◎	11-8
重复	DO ～ LOOP	进行迭代 （循环）。	◎	◎	◎	11-9
	EXIT DO	执行从 DO ～ LOOP 中的强制退出。	◎	◎	◎	11-11
	FOR ～ NEXT	循环执行位于 FOR ～ NEXT 之间的一系列的指令。	◎	◎	◎	11-12
	EXIT FOR	执行从 FOR ～ NEXT 中的强制退出。	◎	◎	◎	11-14
	REPEAT ～ UNTIL	执行后迭代。	◎	◎	◎	11-15
	WHILE ～ WEND	执行前迭代。	◎	◎	◎	11-16
条件分支	IF ～ END IF	执行 IF ～ END IF 之间的条件式中的条件判断。	◎	◎	◎	11-17
	IF ～ THEN ～ ELSE	执行逻辑表达式的条件判断。	◎	◎	◎	11-18
	SELECT CASE	执行多个条件判断。	◎	◎	◎	11-19
无条件分支	GOTO	无条件执行程序的分支。	◎	◎	◎	11-21
	ON ～ GOTO	执行根据表达式的值的无条件分支。	◎	◎	◎	11-22
备注	REM	进行备注的记述。	◎	◎	◎	11-23
机械手控制语句						
动作控制	APPROACH	执行工具坐标系指定的绝对动作。	○	○		12-1
	DEPART	通过工具坐标系的指定执行相对动作。	○	○		12-4
	DRAW	通过工件作业坐标系的指定执行相对动作。	◎	◎		12-6
	DRIVE	进行各轴的相对动作。	◎	◎		12-8
	DRIVEA	进行各轴的绝对动作。	◎	◎		12-10
	GOHOME	向用 HOME 语句定义的位置点 （原点）移动。	◎	◎		12-12
	MOVE	将机械手向指定坐标移动。通过使用 EX 选择 （附加轴相对动作）或 EXA 选择 （附加轴绝对动作），可以使机械手和附加轴同步 （同时出发和到达）动作。	○	○		12-13
	ROTATE	执行指定的旋转轴的旋转动作。	○	○		12-17

4 軸	6 軸	視觉装置	
◎	◎	◎	在所有机械手及视觉设备中均可使用
○	○	○	在所有机械手中均可使用，但是在 4 轴机械手、6 轴机械手或视觉设备中规格不同。
◎	V1.2		在 4 轴机械手和版本 1.2 之后的 6 轴机械手中均可使用

功能分类	指令	功能	4 轴	6 轴	视觉装置	说明页
	ROTATEH	执行以近似矢量为轴的旋转动作。	◎	◎		12-19
	CURJNT	以 J 型取得机械手的当前角度。	○	○		12-21
	CURPOS	以 P 型数据得出工具坐标系中的当前位置。	○	○		12-22
	CURTRN	以 T 型数据得出工具坐标系中的当前位置。	◎	◎		12-23
	CUREXJ	以 F 型获取附加轴的当前角度。	V1.5	V1.5		12-24
	DESTJNT	以 J 型获取当前的动作命令目标位置。机械手停止时，获取当前的位置（指令值）。	○	○		12-25
	DESTPOS	以 P 型获取当前的动作命令目标位置。机械手停止时，获取当前的位置（指令值）。	○	○		12-26
	DESTTRN	以 T 型获取当前的动作命令目标位置。机械手停止时，获取当前的位置（指令值）。	◎	◎		12-27
	DESTEXJ	以 F 型获取附加轴的以当前的动作命令所指定的目标位置。停止时，获取当前的位置 （指令值）。	V1.5	V1.5		12-28
	ARRIVE	通过设定相对于动作命令的全移动距离的动作比例，使程序待机直到到达机械手所设定的动作比例。	◎	V1.2		12-29
	POSCLR	将轴的当前位置强制性置为 0mm 或 0 度。	V1.5	V1.5		12-31
	SETSPLINEPOINT	登录自由曲线动作的通过点。	V2.3	V2.3		12-32
	CLRSPLINEPOINT	将自由曲线动作的通过点全部删除。	V2.3	V2.3		12-33
	GETSPLINEPOINT	取出已经登录的自由曲线动作的通过点。	V2.3	V2.3		12-34
形态控制	CURFIG	得出机械手形态的当前值。	◎	◎		12-35
	FIGAPRL	计算能够执行 CP 动作的接近位置和基准位置的形态。	○	○		12-37
	FIGAPRP	计算能够执行 PTP 动作的接近位置与基准位置的形态。	○	○		12-39
停止控制	HOLD	暂时停止程序的执行。	◎	◎		12-41
	HALT	停止程序的执行。	◎	◎		12-42
	INTERRUPT ON/OFF	中断机械手的动作。	◎	◎		12-43
速度控制	SPEED	指定当前所获取的臂组轴的内部移动速度。	◎	◎		12-45
	JSPEED	指定当前所获取的臂组轴的内部轴速度。	◎	◎		12-47
	ACCEL	指定当前所获取的臂组轴的内部加速度、内部减速度。	◎	◎		12-48
	JACCEL	指定当前所获取的臂组轴的内部轴加速度、内部轴减速度。	◎	◎		12-49
	DECEL	指定当前所获取的臂组轴的内部减速度。	◎	◎		12-50
	JDECEL	指定当前所获取的臂组轴的内部轴减速度。	◎	◎		12-51
	CURACC	获取当前所获取的臂组轴的内部加速度。	◎	◎		12-52

4 轴	6 轴	視覚装置	
◎	◎	◎	在所有机械手及视觉设备中均可使用
○	○	○	在所有机械手中均可使用，但是在 4 轴机械手、6 轴机械手或视觉设备中规格不同。
◎	V1.2		在 4 轴机械手和版本 1.2 之后的 6 轴机械手中均可使用

功能分类	指令	功能	4 轴	6 轴	视觉装置	说明页
	CURJACC	功能获取当前所获取的臂组轴的内部轴加速度。	◎	◎		12-53
	CURDEC	获取当前所获取的臂组轴的内部减速度。	◎	◎		12-54
	CURJDEC	获取当前所获取的臂组轴的内部轴减速度。	◎	◎		12-55
	CURJSPD	获取当前所获取的臂组轴的内部移动轴速度。	◎	◎		12-56
	CURSPD	获取当前所获取的臂组轴的内部移动速度。	◎	◎		12-57
	CUREXTACC	得出外部加速度的当前值。	V1.4	V1.4		12-58
	CUREXTDEC	得出外部减速度的当前值。	V1.4	V1.4		12-58
	CUREXTSPD	得出外部速度的当前值。	V1.4	V1.4		12-59
	EXTSPEED	外部速度设定	V1.98	V1.98		12-59
时间控制	DELAY	在指定的时间内，停止程序的执行。	◎	◎	◎	12-60
	WAIT	执行条件程序的停止。	◎	◎		12-62
坐标转换	CHANGETOOL	变更工具坐标系。	◎	◎		12-63
	CHANGEWORK	变更用户坐标系。	◎	◎		12-64
	CURTOOL	得出当前所设定的 TOOL 编号。	V1.4	V1.4		12-65
	CURWORK	得出当前所设定的 WORK 编号。	V1.4	V1.4		12-66
干扰检查	SETAREA	选择进行干扰检查的区域。	◎	◎		12-67
	RESETAREA	初始化干扰检查。	◎	◎		12-68
	SETPOSTUREAREA	将高感度位置姿势检测功能设为有效。	V3.0	V3.0		12-69
	RESETPOSTUREAREA	将高感度位置姿势检测功能设为无效。	V3.0	V3.0		12-70
伺服内部数据	GetSrvData	获取机械手轴的伺服内部数据。（对应 Ver. 1.5 以上版本）				12-71
	GetJntData	获取指定轴的伺服内部数据。（对应 Ver. 1.5 以上版本）				12-72
电机电源	MOTOR {ON | OFF}	执行电机电源的 ON / OFF。	V1.98	V1.98		12-73
CAL	EXECAL	CAL 执行	V1.98	V1.98		12-74
特殊控制	ST_aspACLD	变更内部负荷条件值。以前端负荷质量 (g)、负荷重心位置 (mm) （注 1）指定所有负荷条件值。	V1.9	V1.9		12-75
	ST_aspChange	选择最佳搬运质量设定模式的内部模式。	V1.9	V1.9		12-76
	ST_SetGravity	修正各个关节的静负荷 （重力转矩），设定重力平衡。	V1.9	V1.9		12-77
	ST_ResetGravity	将重力平衡设为无效。	V1.9	V1.9		12-78
	ST_SetGrvOffset	通过各个关节的重力转矩修正重力补偿值。	V1.9	V1.9		12-79
	ST_ResetGrvOffset	将重力补偿值的修正设为无效。	V1.9	V1.9		12-80
	ST_SetCurLmt	限制指定轴的电机电流值。	V1.9	V1.9		12-81
	ST_ResetCurLmt	解除指定轴的电机电流限制。	V1.9	V1.9		12-83
	ST_SetEralw	变更指定轴的偏差允许值。	V1.9	V1.9		12-84
	ST_ResetEralw	将指定轴的偏差允许值返回到默认值。	V1.9	V1.9		12-85

4 轴	6 轴	視覚装置	
◎	◎	◎	在所有机械手及视觉设备中均可使用
○	○	○	在所有机械手中均可使用，但是在 4 轴机械手、6 轴机械手或视觉设备中规格不同。
◎	V1.2		在 4 轴机械手和版本 1.2 之后的 6 轴机械手中均可使用

功能分类	指令	功能	4 轴	6 轴	视觉装置	说明页
	ST_OnSrvLock	将指定轴置为伺服锁定状态。（4 轴机械手专用）	V1.9			12-86
	ST_OffSrvLock	解除指定轴的伺服锁定。（4 轴机械手专用）	V1.9			12-87
	ST_SetCompControl	将力限制功能设为有效。（6 轴专用命令）		V1.9		12-88
	ST_SetCompFControl	将力限制功能设为有效。（6 轴专用命令）		V1.9		12-89
	ST_ResetCompControl	将力限制功能设为无效。（6 轴专用命令）		V1.9		12-90
	ST_SetFrcCoord	选择力限制设定坐标系。（6 轴专用命令）		V1.9		12-91
	ST_SetFrcLimit	设定力限制比例。（6 轴专用命令）		V1.9		12-92
	ST_ResetFrcLimit	将力限制比例初始化。（6 轴专用命令）		V1.9		12-93
	ST_SetCompRate	设定力限制时的柔度的比例。（6 轴专用命令）		V1.9		12-94
	ST_ResetCompRate	将力限制时的柔度的比例初始化。（6 轴专用命令）		V1.9		12-95
	ST_SetFrcAssist	设定力限制时的补偿力。（力限制特殊功能语句）（6 轴专用命令）		V1.9		12-96
	ST_ResetFrcAssist	将力限制时的补偿力初始化。（力限制特殊功能）（6 轴专用命令）		V1.9		12-97
	ST_SetCompJLimit	设定力限制时的电流限制值。（力限制特殊功能）（6 轴专用命令）		V1.9		12-98
	ST_ResetCompJLimit	将力限制时的电流限制值初始化。（力限制特殊功能）（6 轴专用命令）		V1.9		12-99
	ST_SetCompVMode	设定力限制时的速度控制模式。（力限制特殊功能）（6 轴专用命令）		V1.9		12-100
	ST_ResetCompVMode	将力限制时的速度控制模式设为无效。（力限制特殊功能）（6 轴专用命令）		V1.9		12-101
	ST_SetCompEralw	设定力限制时的工具端的位置、姿势偏差允许值。（6 轴专用命令）		V1.9		12-102
	ST_ResetCompEralw	将力限制时的工具端的位置、姿势偏差允许值初始化。（6 轴专用命令）		V1.9		12-103
	ST_SetDampRate	设定力限制时的粘性比例。（6 轴专用命令）		V1.9		12-104
	ST_ResetDampRate	将力限制时的粘性比例初始化。（6 轴专用命令）		V1.9		12-105
	ST_SetZBalance	设定 Z 轴、T 轴的重力补偿值。（4 轴专用）	V1.9			12-106
	ST_ResetZBalance	将重力补偿值的修正设为无效。（4 轴专用）	V1.9			12-107
输入输出控制语句						
I / O 端口	IN	从通过 I / O 变量表示的 I / O 端口中读取数据。	◎	◎	◎	13-1
	OUT	向通过 I / O 变量表示的 I / O 端口输出数据。	◎	◎	◎	13-2

4 轴	6 轴	视觉装置	
◎	◎	◎	在所有机械手及视觉设备中均可使用
○	○	○	在所有机械手中均可使用，但是在 4 轴机械手、6 轴机械手或视觉设备中规格不同。
◎	V1.2		在 4 轴机械手和版本 1.2 之后的 6 轴机械手中均可使用

功能分类	指令	功能	4 轴	6 轴	视觉装置	说明页
	IOBLOCK ON/OFF	在执行动作指令时，同步执行 I / O 指令、运算指令等非动作指令。	◎	◎		13-3
	SET	将 I / O 端口置为 ON。	◎	◎	◎	13-5
	RESET	将 I / O 端口置为 OFF。	◎	◎	◎	13-7
	DETECT ON/OFF	检测夹治具输入的上升以及下降，将此时机械手的位置存放至变量。	V3.2	V3.2		13-8
RS232C 及 Ethernet 端口	INPUT	得出来自 RS232C 及 Ethernet 端口的数据。	◎	◎	◎	13-10
	LINEINPUT	从 RS232C 及 Ethernet 端口中读取数据直到间隔符，并对字符串型变量赋值。	◎	◎	◎	13-12
	PRINT	从 RS232C 及 Ethernet 端口输出数据。	◎	◎	◎	13-13
	WRITE	从 RS232C 及 Ethernet 端口输出数据。	◎	◎	◎	13-14
	FLUSH	清空输入缓冲区。	◎	◎	◎	13-15
串行二进制通信（RS-232C 以及 Ethernet 端口）	PRINTB	向 RS-232C 及 Ethernet 端口输出 1 字节 (Byte) 数据。	V1.5	V1.5	V1.9	13-16
	INPUTB	从 RS-232C 及 Ethernet 端口输入 1 字节 (Byte) 数据。	V1.5	V1.5	V1.9	13-17
	LPRINTB	从 RS-232C 及 Ethernet 端口输出多字节 (Byte) 数据。	V1.5	V1.5	V1.9	13-18
	LINPUTB	从 RS-232C 及 Ethernet 端口输入多字节 (Byte) 数据。	V1.5	V1.5	V1.9	13-19
	com_encom	为使用 RS-232C 端口或 Ethernet 用户端口进行二进制通信做准备。RS-232C 端口的情况下，为了进行二进制通信而占有端口。使用 Ethernet 用户端口时，打开用户端口，在端口与服务器之间建立连接。	V1.5	V1.5	V1.9	13-20
	com_discom	为了结束二进制通信释放 RS-232C 端口。（释放 COM 端口）使用 Ethernet 外端口时，关闭外端口，切断连接。	V1.5	V1.5	V1.9	13-21
	com_state	RS-232C 及获取 Ethernet 端口的状态。	V1.5	V1.5	V1.9	13-22
多功能教导器	PRINTMSG	将讯息附上字幕和图标，在多功能教导器的彩色液晶中显示。	◎	◎		13-23
	PRINTDBG	向调试视窗输出数据。	◎	◎		13-24
	BUZZER	鸣响蜂鸣器。	◎	◎		13-24
	PRINTWARNING	在教导器警告讯息显示部分显示讯息。	V2.2	V2.2		13-25
	PRINTLBL	向用户定义按钮设定标签 （字幕）。	◎	◎		13-25
TP 简易操作盘	set_button	进行按钮参数的设定。	V1.5	V1.5		13-26
	set_page	进行页面参数的设定。	V1.5	V1.5		13-30
	change_bCap	进行按钮的字幕设定。	V1.5	V1.5		13-32
	change_pCap	进行页面的字幕设定。	V1.5	V1.5		13-33
	disp_page	进行页面的显示。	V1.5	V1.5		13-34
多项任务控制语句						
任务控制	RUN	并列启动其他程序。	◎	◎		14-1

4 轴	6 轴	視覚装置	
◎	◎	◎	在所有机械手及视觉设备中均可使用
○	○	○	在所有机械手中均可使用，但是在 4 轴机械手、6 轴机械手或视觉设备中规格不同。
◎	V1.2		在 4 轴机械手和版本 1.2 之后的 6 轴机械手中均可使用

功能分类	指令	功能	4 轴	6 轴	视觉装置	说明页
	KILL	强制结束任务。	◎	◎		14-3
	SUSPEND	暂停任务。	◎	◎		14-4
	DEFEND	保护任务。	◎	◎		14-5
	STATUS	得出程序的状态。	◎	◎		14-6
	SUSPENDALL	停止特权任务以外的所有的程序。	V1.98	V1.98		14-8
	KILLALL	强制结束特权任务以外的所有的任务。（相当于程序复位）	V1.98	V1.98		14-9
	CONTINUERUN	进行连续启动。	V1.98	V1.98		14-10
	ROBOTSTOP	进行机械手停止。	V1.98	V1.98		14-11
	TAKEARMSTATE	返回臂组控制权的获取状态。 0: 尚未获取、1: 已经获取	V2.2	V2.2		14-12
	LOCKSTATE	获取锁定的状态。 0: OFF、 1: ON	V2.2	V2.2		14-12
	DEADMANSTATE	双重安全开关的状态。 0: OFF、 1: ON	V2.2	V2.2		14-13
	SEMIDSTATE	返回所指定的信号 (semaphore) ID 是有效还是无效。	V2.2	V2.2		14-14
信号 (semaphore)	CREATESEM	生成信号 (semaphore)。	◎	◎		14-15
	DELETESEM	删除信号 (semaphore)。	◎	◎		14-18
	FLUSHSEM	释放信号 (semaphore) 等待任务。	◎	◎		14-19
	GIVESEM	释放信号 (semaphore) 等待任务。	◎	◎		14-20
	TAKESEM	获取具有指定信号 (semaphore) ID 的信号 (semaphore)。	◎	◎		14-21
臂信号 (semaphore)	TAKEARM	获取臂组。当省略了臂组 NO 时，获取臂组 0 （仅限机械手）的控制权。获取时，将内部速度、加速度、减速度设定为 100。当获取的臂组包含机械手轴时，将工具坐标、工件坐标返回到原点。	◎	◎		14-22
	GIVEARM	释放当前正在获取的臂组。臂组 0 （仅限机械手）时，开放机械手控制权。	◎	◎		14-27
	TAKEVIS	获取视觉处理权。	◎	◎		14-28
	GIVEVIS	释放视觉处理权。	◎	◎		14-29
特权任务	INIT	根据特权任务参数的状态，进行电机 ON、 CAL、速度的设定。	V1.7	V1.7		14-30
	SETOCCUPATIONTIME	变更特权任务的占有时间。	V2	V2		14-31
	INITWAITERR	初始化错误检测。（特权任务专用命令）	V2.2	V2.2		14-32
	WAITERROR	检测错误发生。（特权任务专用命令）	V2.2	V2.2		14-33
	CURERRSTATUS	返回错误发生状态。（特权任务专用命令）	V2.2	V2.2		14-34
函数						
算术函数	ABS	得出算式的值的绝对值。	◎	◎	◎	15-1
	EXP	得出将自然对数作为基数的指数函数。	◎	◎	◎	15-2
	INT	得出不超过被指定值的最大的整数值。	◎	◎	◎	15-3
	LOG	得出自然对数。	◎	◎	◎	15-4

4 轴	6 轴	視覚装置	
◎	◎	◎	在所有机械手及视觉设备中均可使用
○	○	○	在所有机械手中均可使用，但是在 4 轴机械手、6 轴机械手或视觉设备中规格不同。
◎	V1.2		在 4 轴机械手和版本 1.2 之后的 6 轴机械手中均可使用

功能分类	指令	功能	4 轴	6 轴	视觉装置	说明页
	LOG10	得出常用对数。	◎	◎	◎	15-5
	POW	求取幂乘。	◎	◎	◎	15-6
	MAX	抽出最大值。	◎	◎	◎	15-7
	MIN	抽出最小值。	◎	◎	◎	15-8
	RND	发生大于 0 小于 1 的随机数。	◎	◎	◎	15-9
	SGN	调查符号。	◎	◎	◎	15-10
	SQR	得出平方根。	◎	◎	◎	15-11
三角函数	ACOS	得出反余弦 (arc cosine)。	◎	◎	◎	15-12
	ASIN	得出反正弦 (arc sine)。	◎	◎	◎	15-13
	ATN	得出反正切 (arc tangent)。	◎	◎	◎	15-14
	ATN2	得出以算式 1 除以算式 2 之后的反正切 (arc tangent)。	◎	◎	◎	15-15
	COS	得出余弦 (cosine)。	◎	◎	◎	15-16
	SIN	得出正弦 (sine)。	◎	◎	◎	15-17
	TAN	得出正切 (tangent)。	◎	◎	◎	15-18
角度转换	DEGRAD	将单位转换为弧度。	◎	◎	◎	15-19
	RAD	将通过弧度赋予的数值转换为角度。	◎	◎	◎	15-20
	RADDEG	将单位转换为度。	◎	◎	◎	15-21
速度转换	MPS	将指定的速度值 （mm/sec）转换为对 CP 动作时的最大内部速度的比率（%）。	◎	◎		15-22
	RPM	将指定轴编号的旋转数 （rpm）转换为对 PTP 动作时的最大内部速度的比率 （%）。	V2.8	V2.8		15-23
时间函数	SEC	将以秒为单位赋予的数值变换为毫秒。	◎	◎	◎	15-24
矢量	AVEC	抽出近似矢量。	◎	◎		15-25
	OVEC	抽出定向矢量。	◎	◎		15-26
	PVEC	抽出位置矢量。	○	○		15-27
	MAGNITUDE	得出矢量的大小。	◎	◎		15-28
姿势数据型转换	J2P	从关节型转换为位置型。	○	○		15-29
	J2T	从关节型转换为齐次变换型。	○	○		15-30
	P2J	从位置型转换为关节型。	○	○		15-31
	P2T	从位置型转换为齐次变换型。	○	○		15-32
	T2J	从关节型转换为齐次变换型。	◎	◎		15-33
	T2P	从齐次变换型转换为位置型。	◎	◎		15-34
	TINV	计算齐次变换型的逆矩阵。	◎	◎		15-35
	NORMTRN	进行齐次变换型的正规化计算。	V1.8	V1.8		15-36
距离抽出	DIST	返回 2 点间的距离。	V1.8	V1.8		15-37
形态分量	FIG	抽出形态。	○	○		15-38
角度分量	JOINT	从关节型坐标抽出角度。	○	○		15-39
轴分量	POSX	抽出 X 分量。	○	○		15-40
	POSY	抽出 Y 分量。	○	○		15-41
	POSZ	抽出 Z 分量。	○	○		15-42
旋转分量	POSRX	抽出 X 轴旋转分量。		◎		15-43
	POSRY	抽出 Y 轴旋转分量。		◎		15-44

4 轴	6 轴	視觉装置	
◎	◎	◎	在所有机械手及视觉设备中均可使用
○	○	○	在所有机械手中均可使用，但是在 4 轴机械手、6 轴机械手或视觉设备中规格不同。
◎	V1.2		在 4 轴机械手和版本 1.2 之后的 6 轴机械手中均可使用

功能分类	指令	功能	4 轴	6 轴	视觉装置	说明页
	POSRZ	抽出 Z 轴旋转分量。		◎		15-45
	POST	抽出 T 轴旋转分量。	◎			15-46
姿势分量	RVEC	抽出姿势。		◎		15-47
位置函数	AREAPOS	将进行干扰检查的区域的中心位置和长方体的方向以位置型进行返回。	◎	◎		15-48
	AREASIZE	将定义进行干扰检查的区域的长方体的大小（各个边的长度）以矢量型进行返回。	◎	◎		15-49
	TOOLPOS	将工具坐标系以位置型进行返回。	◎	◎		15-50
	WORKPOS	将用户坐标系以位置型进行返回。	◎	◎		15-51
	WORKATTRIBUTE	获取指定工件坐标系的属性值。	V3.2	V3.2		15-51
字符串函数	ASC	变换为字符编码。	◎	◎	◎	15-52
	BIN$	将算式的值转换为 2 进制数的字符串。	◎	◎	◎	15-53
	CHR$	将 ASCII 编码转换为字符。	◎	◎	◎	15-54
	SPRINTF$	将表达式变换为所指定的格式化，并作为字符串返回。	◎	◎	◎	15-55
	HEX$	以字符串形式得出从 10 进制数变换为 16 进制数后的值。	◎	◎	◎	15-57
	LEFT$	抽出左部分字符串。	◎	◎	◎	15-58
	LEN	以字节 (Byte) 数得出字符串的长度。	◎	◎	◎	15-59
	MID$	从字符串中取出指定的字符数部分的字符串。	◎	◎	◎	15-60
	ORD	变换为字符编码。	◎	◎	◎	15-61
	RIGHT$	抽出字符串的右部分。	◎	◎	◎	15-62
	STRPOS	得出字符串的位置。	◎	◎	◎	15-63
	STR$	将数值变换为字符串。	◎	◎	◎	15-64
	VAL	从字符串变换为数值。	◎	◎	◎	15-65
常量						
内置常量	OFF	赋予 OFF (0) 值。	◎	◎	◎	16-1
	ON	赋予 ON (1) 值。	◎	◎	◎	16-2
	PI	赋予 π 值。	◎	◎	◎	16-3
	FALSE	赋予布尔值的假 (0) 的值。	◎	◎	◎	16-4
	TRUE	赋予布尔值的真 (1) 的值。	◎	◎	◎	16-5
时间 / 日期控制						
时间 / 日期	DATE$	得出当前的日期。	◎	◎		17-1
	TIME$	得出当前的时间。	◎	◎		17-2
	TIMER	得出经过时间。	◎	◎		17-3
错误控制						
错误信息	ERRMSG$	赋予错误讯息。	◎	◎	◎	18-1
	SETERR	将用户定义错误设定到 I 型变量区域。	V1.98	V1.98		18-1
	GETERR	从通过错误存放功能定义的缓冲中获取错误编码。	V1.98	V1.98		18-2
	CLRERR	清空错误。	V1.98	V1.98		18-3
	GETERRLVL	赋予错误编码的等级。	V1.98	V1.98		18-4

4 轴	6 轴	视觉装置	
◎	◎	◎	在所有机械手及视觉设备中均可使用
○	○	○	在所有机械手中均可使用，但是在 4 轴机械手、6 轴机械手或视觉设备中规格不同。
◎	V1.2		在 4 轴机械手和版本 1.2 之后的 6 轴机械手中均可使用

功能分类	指令	功能	4 轴	6 轴	视觉装置	说明页
系统信息						
系统	GETENV	获取系统的环境设定值。	◎	◎	◎	19-1
	LETENV	设定系统的环境设定值。	◎	◎	◎	19-2
	VER$	获取各个模块的版本。	◎	◎	◎	19-3
	GETLANGUAGE	获取当前的语言设定。	V2.2	V2.2		19-3
Log	STARTLOG	开始伺服控制 LOG 的记录。	◎	◎		19-4
	CLEARLOG	对伺服控制 LOG 的记录进行初始化。	◎	◎		19-5
	STOPLOG	停止伺服控制 LOG 的记录。	◎	◎		19-6
	STARTTRACELOG	开始记录跟踪 LOG。	V2.7	V2.7		19-7
	STOPTRACELOG	结束对跟踪 LOG 的记录。	V2.7	V2.7		19-8
	CLEARTRACELOG	清空跟踪 LOG 数据。	V2.7	V2.7		19-9
	STARTVARLOG	开始记录变量 LOG。	V2.7	V2.7		19-10
	STOPVARLOG	结束对变量 LOG 的记录。	V2.7	V2.7		19-11
	CLEARVARLOG	清空变量 LOG 数据。	V2.7	V2.7		19-12
动作模式	CHGEXTMODE	进行外部自动模式切换。	V1.98	V1.98		19-13
	CHGINTMODE	进行内部自动模式切换。	V1.98	V1.98		19-14
	CUROPTMODE	获取动作模式。	V1.98	V1.98		19-15
	SYSSTATE	获取控制器的状态。	V1.98	V1.98		19-16
预处理器						
符号常量、宏定义	#define	使用指定字符串替换程序中所指定的常量或宏名。	◎	◎	◎	20-1
	#undef	将通过 #define 定义的符号常量或宏的定义置为无效。	◎	◎	◎	20-2
	#error	如果执行 #error 指令，则强制性置为编译错误。	◎	◎	◎	20-3
文件读取	#include	读入预处理程序。	◎	◎	◎	20-4
最佳化	#pragma optimize	指定按程序单位执行的最优化处理。	◎	◎	◎	20-6
视觉控制 （机械手控制器：选择）						
图像输出入	CAMIN	将来自照相机的影像存放在图像储存器 （处理画面）。	◎	◎	◎	21-1
	CAMMODE	设定存放照相机影像时的功能。	◎	◎	◎	21-2
	CAMLEVEL	设定照相机影像的输入等级。	◎	◎	◎	21-4
	VISCAMOUT	将来自照相机的影像显示在监视器上。	◎	◎	◎	21-5
	VISPLNOUT	将存放存储器的图像显示在监视器上。	◎	◎	◎	21-6
	VISOVERLAY	将描绘画面的信息显示在监视器上。	◎	◎	◎	21-7
	VISDEFTABLE	设定照相机影像的输入、图像输出时的一览表数据。	◎	◎	◎	21-8
	VISREFTABLE	参照一览表的数据。	◎	◎	◎	21-10
视窗设定	WINDMAKE	指定图像处理的范围。	◎	◎	◎	21-11
	WINDCLR	消除设定的视窗信息。	◎	◎	◎	21-16
	WINDCOPY	复制视窗的数据。	◎	◎	◎	21-17
	WINDREF	取得视窗的信息。	◎	◎	◎	21-19
	WINDDISP	描绘所设定的视窗。	◎	◎	◎	21-20

4 轴	6 轴	视觉装置	
◎	◎	◎	在所有机械手及视觉设备中均可使用
○	○	○	在所有机械手中均可使用，但是在 4 轴机械手、6 轴机械手或视觉设备中规格不同。
◎	V1.2		在 4 轴机械手和版本 1.2 之后的 6 轴机械手中均可使用

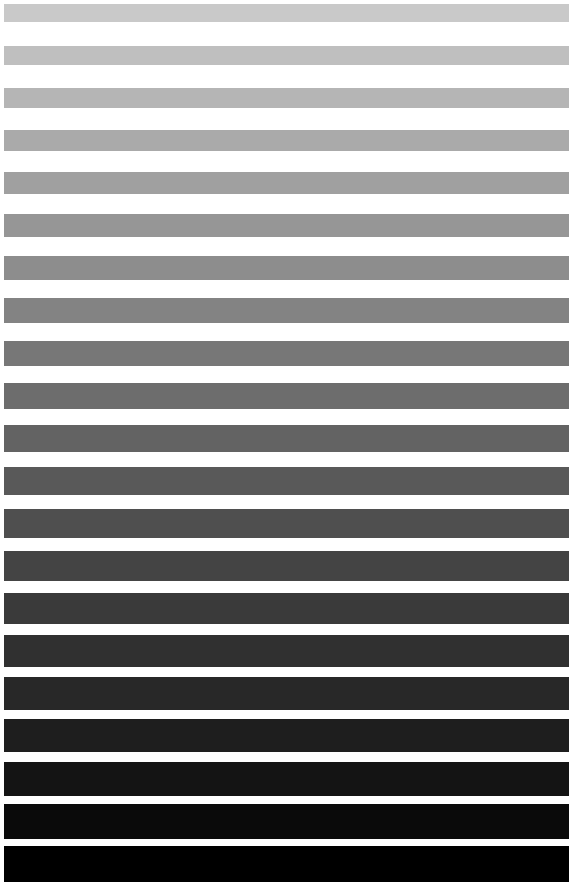
功能分类	指令	功能	4 轴	6 轴	视觉装置	说明页
描绘	VISSCREEN	指定描绘的画面。	◎	◎	◎	21-21
	VISBRIGHT	指定描绘时的辉度值。	◎	◎	◎	21-23
	VISCLS	用指定辉度全面涂抹（清空）模式所指定的画面。	◎	◎	◎	21-24
	VISPUTP	在画面上描绘点。	◎	◎	◎	21-26
	VISLINE	在画面上描绘直线。	◎	◎	◎	21-27
	VISPTP	在画面上描绘连接 2 点间的直线。	◎	◎	◎	21-28
	VISRECT	在画面上描绘矩形。	◎	◎	◎	21-29
	VISCIRCLE	在画面上描绘圆。	◎	◎	◎	21-31
	VISELLIPSE	在画面上描绘椭圆。	◎	◎	◎	21-32
	VISSECT	在画面上描绘扇形。	◎	◎	◎	21-34
	VISCROSS	在画面上描绘十字标记。	◎	◎	◎	21-35
	VISLOC	指定字符的显示位置。	◎	◎	◎	21-36
	VISDEFCHAR	指定字符的大小和显示方法。	◎	◎	◎	21-38
	VISPRINT	在画面上显示字符、数字。	◎	◎	◎	21-40
图像处理	VISWORKPLN	指定处理目标的存放存储器（处理画面）。	◎	◎	◎	21-41
	VISGETP	从存放存储器（处理画面）中获取指定坐标的辉度。	◎	◎	◎	21-42
	VISHIST	得到画面的直方柱状体（辉度分布）。	◎	◎	◎	21-43
	VISREFHIST	读出直方柱状体结果。	◎	◎	◎	21-44
	VISLEVEL	根据直方柱状体结果求取 2 值化标准值。	◎	◎	◎	21-45
	VISBINA	将画面进行 2 值化处理。	◎	◎	◎	21-47
	VISBINAR	将画面进行 2 值化显示。	◎	◎	◎	21-49
	VISFILTER	对图像进行过滤器处理。	◎	◎	◎	21-50
	VISMASK	进行图像间运算。	◎	◎	◎	21-52
	VISCOPY	复制画面。	◎	◎	◎	21-54
	VISMEASURE	测量视窗内的特征（面积、重心、主轴角）。	◎	◎	◎	21-55
	VISPROJ	测量视窗内的投影数据。	◎	◎	◎	21-58
	VISEDGE	测量视窗内的边缘。	◎	◎	◎	21-60
编码识别	VISREADQR	读取 QR 编码。	◎	◎	◎	21-65
标注	BLOB	执行标注。	◎	◎	◎	21-68
	BLOBMEASURE	进行目标标签编号的特征测量。	◎	◎	◎	21-71
	BLOBLABEL	获取指定坐标的标签编号。	◎	◎	◎	21-73
	BLOBCOPY	复制目标标签编号。	◎	◎	◎	21-75
检索功能	SHDEFMODEL	进行检索模型的登录。	◎	◎	◎	21-77
	SHREFMODEL	参照登录模型数据。	◎	◎	◎	21-79
	SHCOPYMODEL	复制登录模型。	◎	◎	◎	21-80
	SHCLRMODEL	消除登录模型。	◎	◎	◎	21-81
	SHDISPMODEL	将登录模型显示在画面上。	◎	◎	◎	21-82
	SHMODEL	检索模型。	◎	◎	◎	21-83
	SHDEFCORNER	设定棱角检索的条件。	◎	◎	◎	21-87
	SHCORNER	检索棱角。	◎	◎	◎	21-88

4 軸	6 軸	視覚装置	
◎	◎	◎	在所有机械手及视觉设备中均可使用
○	○	○	在所有机械手中均可使用，但是在 4 轴机械手、6 轴机械手或视觉设备中规格不同。
◎	V1.2		在 4 轴机械手和版本 1.2 之后的 6 轴机械手中均可使用

功能分类	指令	功能	4 轴	6 轴	视觉装置	说明页
获取结果	SHDEFCIRCLE	设定圆检索的条件。	◎	◎	◎	21-91
	SHCIRCLE	检索圆。	◎	◎	◎	21-92
	VISGETNUM	从存放存储器中获取图像处理结果。	◎	◎	◎	21-94
	VISGETSTR	获取编码识别结果。	◎	◎	◎	21-95
	VISPOSX	从存放存储器中取得图像处理结果（X 坐标）。	◎	◎	◎	21-96
	VISPOSY	从存放存储器中取得图像处理结果（Y 坐标）。	◎	◎	◎	21-97
	VISSTATUS	监视各个命令的处理结果。	◎	◎	◎	21-98
	VISREFCAL	获取 CAL（视觉－机械手坐标转换）数据。	◎	◎	◎	21-99

第1部
程序的基础知识

第 1 章
程序示例



在本章中将使用简单的应用事例演示各个指令。

第 1 章 程序示例

1.1 典型事例应用

作为最初的典型事例，举出如图1-1所示的应用实例，表示程序示例。请仔细阅读该程序，并掌握编写程序所需的知识。

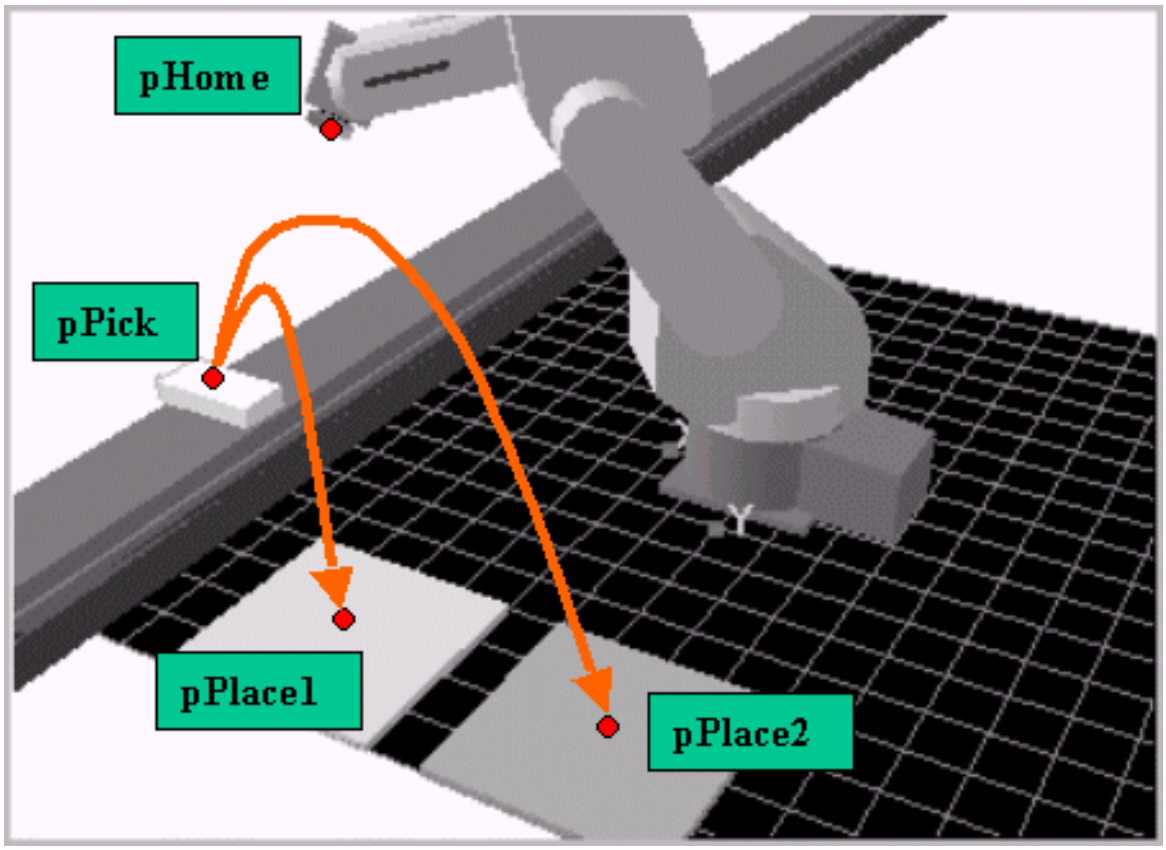


图1-1 典型事例应用

在该典型事例中，读取pPick中放置的目标物的QR编码，根据QR编码判断将其抓取到pPlace1还是pPlace2，将其搬运至各自的位置并放置。pHome为初始位置即成为基点的位置。

注意：关于编写程序的操作方法，请参照入门篇第11章程序的创建。此内容就采用相同的应用典型事例应用对编写程序的操作步骤进行了说明。

1.2 程序流程

在图1-2中表示了用于典型事例的程序流程。

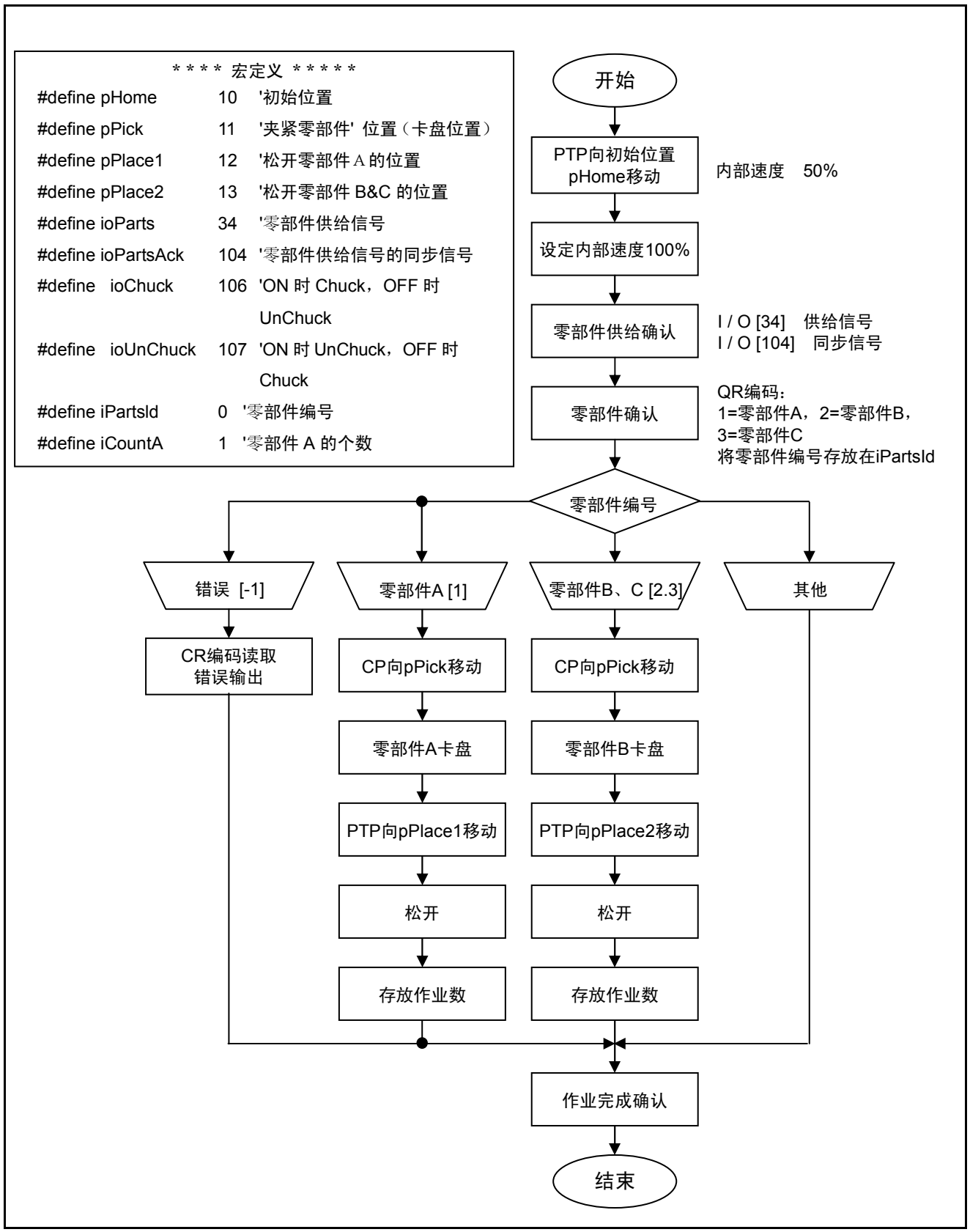


图1-2 典型事例的程序流程

1.3 程序列表

记述了典型事例的程序列表。
程序有 "PRO1"、"PRO2"、"dioSetAndWait"、"dioWaitAndSet" 等4种。
"PRO1"为主程序。
"PRO2"是有关读取QR编码的辅助程序。
"dioSetAndWait" 和 "dioWaitAndSet" 是操作确认零部件供给用的I / O的子程序。这2个程序被收录在WINCAPSⅢ的程序库 (Programme Bank) 中。

程序列表 "PRO1"

```
!TITLE   "选取＆装入 (pick-and-place)"
#INCLUDE   "dio_tab.h"                    'DIO宏定义文件的读入
#INCLUDE   "var_tab.h"                    '变量宏定义文件的读入
#DEFINE    appLen 100                     '接近长的定义
                宏定义appLen和100为同义。
PROGRAM pro1  程序定义（定义程序名称。详细情况参照p.9-1）
   TAKEARM  TAKEARM 文(是为了动作机械手所需要的文
            字。详细情况参照 p.14-19）
   MOVE P, P[pHome], S=50                 '将内部速度设为50%，向初始位置PTP移动
              与宏 P [10] 或 P10 有相同的意义。关于宏参照
              p.20-1
   SPEED 100 '将内部速度设为100%
   CALL dioWaitAndSet (ioParts, ioPartsAck)   '零部件供给确认
              程序调出（参照 p.2-1 和 p.2-3）
   CALL pro2  'QR编码的读取
   SELECT CASE I [iPartsId]  SELECT CASE 文(是以 I[OO]
                        宏   为条件的分支命令。
                             参照 p.11-19
     CASE –1I [OO] 为－1 时，按以下执行。理
       CALL dioSetAndWait (ioErrQR, ioErrQRAck)  '错误输出
     CASE 1
       GOSUB *PlacePartsA                '零部件A的处理
         调出子程序（参照p.2-2）
     CASE 2, 3
       GOSUB *PlacePartsBC               '零部件B、C的处理
   END SELECT
   CALL dioSetAndWait (ioComplete,ioCompleteAck) '动作结束信号输出
   GIVEARM 用 TAKEARM 定义，开放机械手手臂的占有使用
           权。
END
```

续下页

程序列表 "PRO1"（续）

```
' =====夹紧零部件（零部件卡盘）=====
*ChuckItem：用*OO：指定标签。然后用CALL *ChuckItem调出。
    SET IO [ioChuck] 同意为 IO [106]
                宏
    SET 命令（接通 IO。参照 p.13-5）
    RESET IO [ioUnChuck] 同意为 IO [106]
                宏
    RESET 命令（断开 IO。参照 p.13-7）
    RETURN RETURN语句（在和*ChuckItem之间构成子程序。）

' =====松开零部件（零部件松开卡盘）=====
*UnchuckItem：标签
    RESET IO [ioChuck]
    SET IO [ioUnChuck]
    RETURN 返回文

'=====零部件A的处理=====
*PlacePartsA：标签名称（子程序名称的定义）
    APPROACH P, P [pPick], appLen          '向离开卡盘点100mm的位置进行CP移动
  APPROACH 命令 同意为宏 在第 4 行定义的宏。定义为
  （将机械手向 P P [11] 100。通过这样的定义，可以
  [pPick1] 的正上方 在 1 处变更接近长度。
  移动。）
    MOVE L, P[pPick], S=80                 '将内部速度设为80%向卡盘点进行CP移动
      MOVE命令（p.12-14参照）
    GOSUB *ChuckItem                       '夹紧零部件
    DEPART L, appLen                       '向离开当前位置100mm的位置移动
      DEPART 命令 在第 4 行定义的宏。定义为 100。通过
      （p.12-4 参照） 这样的定义，可以在 1 处变更接近长度。

    APPROACH P, P[pPlace1], appLen         '向离开松开卡盘点100mm的位置进行CP移动
      APPROACH 命令 在第 4 行定义的宏。定义为
      （将机械手向 P 100。通过这样的定义，可以
      [pPick1] 的正上方 在 1 处变更接近长度。
      移动。）
    MOVE P, P [pPlace1], S=50              '将内部速度设为50%向松开卡盘点进行CP移动
      MOVE命令（p.12-14参照）
    GOSUB *UnchuckItem                     '松开零部件
    DEPART L, appLen                       '向离开当前位置100mm的位置进行CP移动
      DEPART 命令 在第 4 行定义的宏。定义为 100。通过这样
      （p.12-4 参照） 的定义，可以在 1 处变更接近长度。

    I [iCountA] = I [iCountA] + 1          '计数零部件A的个数
                  使I [iCountA] 增加1。
    RETURN
```

续下页

程序列表 "PRO1"（续）

```
'=====零部件BC的处理=====
*PlacePartsBC：标签名称（子程序名称的定义）
    APPROACH P, P [pPick], appLen          '向离开卡盘点100mm的位置进行CP移动
      APPROACH 命令（将机械手向 P [pPick1] 的正上方移动。）  在第 4 行定义的宏。定义为 100。通过这样的定义，可以在 1 处变更接近长度。
    MOVE L, P [pPick], S=90                '将内部速度设为90%向卡盘点进行CP移动
    GOSUB *ChuckItem                       '夹紧零部件
    DEPART L, appLen                       '向离开当前位置100mm的位置移动
      DEPART 命令（p.12-4 参照）  在第 4 行定义的宏。定义为 100。通过这样的定义，可以在 1 处变更接近长度。

    APPROACH P, P [pPlace2], appLen        '向离开松开卡盘点100mm的位置进行CP移动
      APPROACH 命令(将机械手向 P[pPick1]的正上方移动。（参照 p.12-1）  在第 4 行定义的宏。定义为 100。通过这样的定义，可以在 1 处变更接近长度。

    MOVE L, P[pPlace2], S=80               '将内部速度设为80%向松开卡盘点进行CP移动
      MOVE命令（p.12-14参照）
    GOSUB *UnchuckItem                     '松开零部件
    DEPART L, appLen                       '向离开当前位置100mm的位置进行CP移动
      DEPART 命令（p.12-4 参照）  在第 4 行定义的宏。定义为 100。通过这样的定义，可以在 1 处变更接近长度。

    I [iCountBC] = I [iCountBC] + 1        '计数零部件B、C的个数
RETURN
```

续下页

程序列表 "PRO2"

```
!TITLE "QR编码读取"
#INCLUDE   "var_tab.h"                              '变量宏定义文件的读入

'将编号存放在I [iPartsId] 中
PROGRAM pro2
  DIM len1 AS INTEGER 可以将局部 (Local) 变量 'lenl' 作为整数
                      型局部 (Local) 变量声明使用。
  TAKEVIS 定义占有视觉装置。在使用视觉装置时需要          '获取视觉信号 (semaphore)
  VISSCREEN 1,0,1                                   '将描绘画面0指定为描绘位置
    参照p.21-24）
  VISCLS 0                                          '将描绘画面0清空
    参照p.21-27）
  VISOVERLAY 1                                      '显示描绘画面0
    参照p.21-9）
  CAMIN   1,0,0                                     '将照相机图像输入处理画面0
    参照p.21-3）
  VISPLNOUT   0,1                                   '显示处理画面0
    参照p.21-8）
  VISREADQR   1,WINDREF (1, 2),WINDREF (1, 3),0,0,1     'QR编码的读取
    参照p.21-64）
  WINDDISP   1                                      '描绘视窗
    参照p.21-23）
  VISLOC   10   , 10                                '指定显示位置
    参照p.21-37）
  IF VISSTATUS (0) = 0 THEN                         '测量结果OK的情况下
    len1 = VISGETNUM (0, 0)                         '测量文字数的获取
    VISPRINT   VISGETSTR (1, len1)                  '测量结果的显示
    I [iPartsId] = VAL (VISGETSTR (1, len1))        '存放编号
  ELSE
    VISPRINT   "不能识别编码"                        '错误的显示
    I [iPartsId] = -1                               '错误时存放-1
  ENDIF
  IF…ELSE…进行条件分支。参照p.11-17
  GIVEVIS 解除占有视觉装置。                        '释放视觉信号 (semaphore)
END
```

注意：QR编码读取处理视窗1视已被定义过的视窗。

程系列表 "WAIT → SET"（共用程序库 (Library)）

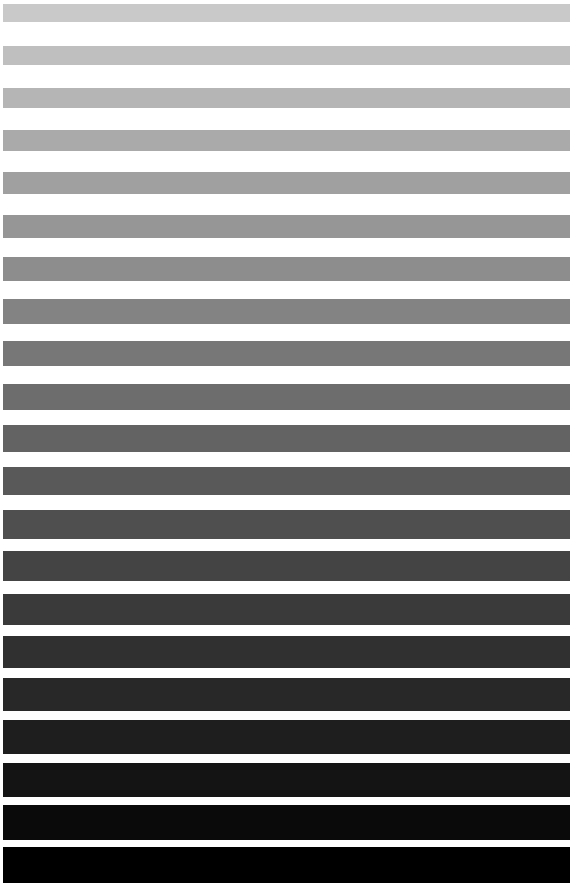
```
!TITLE "WAIT → SET"
PROGRAM dioWaitAndSet (waitIndex%, ackIndex%)
                                自变量 (argument) 定义    自变量 (argument) 定义参照p.9-1
  RESET IO [ackIndex] RESET文参照p.13-7          '同步信号OFF
  WAIT IO [waitIndex] = ON  WAIT文IO [] 置于ON之前    '等待同步信号ON
                等待。参照p.12-64
  SET IO [ackIndex] SET文参照p.13-5              '同步信号ON

END
```

程系列表 "SET → WAIT"（共用程序库 (Library)）

```
!TITLE "SET → WAIT"
PROGRAM dioSetAndWait (ackIndex%, waitIndex%)
                              自变量 (argument) 定义    自变量 (argument) 定义参照p.9-1
  SET IO [ackIndex] SET文参照p.13-5              '同步信号ON
  WAIT IO [waitIndex] = ON  WAIT文IO [] 置于ON之前      '等待同步信号ON
                等待。参照p.12-64
  RESET IO [ackIndex] RESET文参照p.13-7          '同步信号OFF
END
```

第 2 章
程序的动作原理



对编写时所需的程序动作原理进行说明。

第 2 章 程序的动作原理

2.1 程序的调出与子程序

可以在程序中事先抽出循环执行特定动作的部分，视必要性进行调出。
在同一个程序文件中放置该部分的方法称为子程序。独立于别的文件，作为另外的程序调出的行为称为程序调出。
子程序必须与调出侧的程序在相同的文件中。
调出已独立的另外一个文件时，从多个程序中调出其中一个程序，作为通用程序来使用。
将一系列的工作作为子程序或另外的程序总结到一处，由此可以无需多次记述相同内容的程序即可而且对于错误记述、削减程序编写时间、提高程序的可读性等都是有效的。

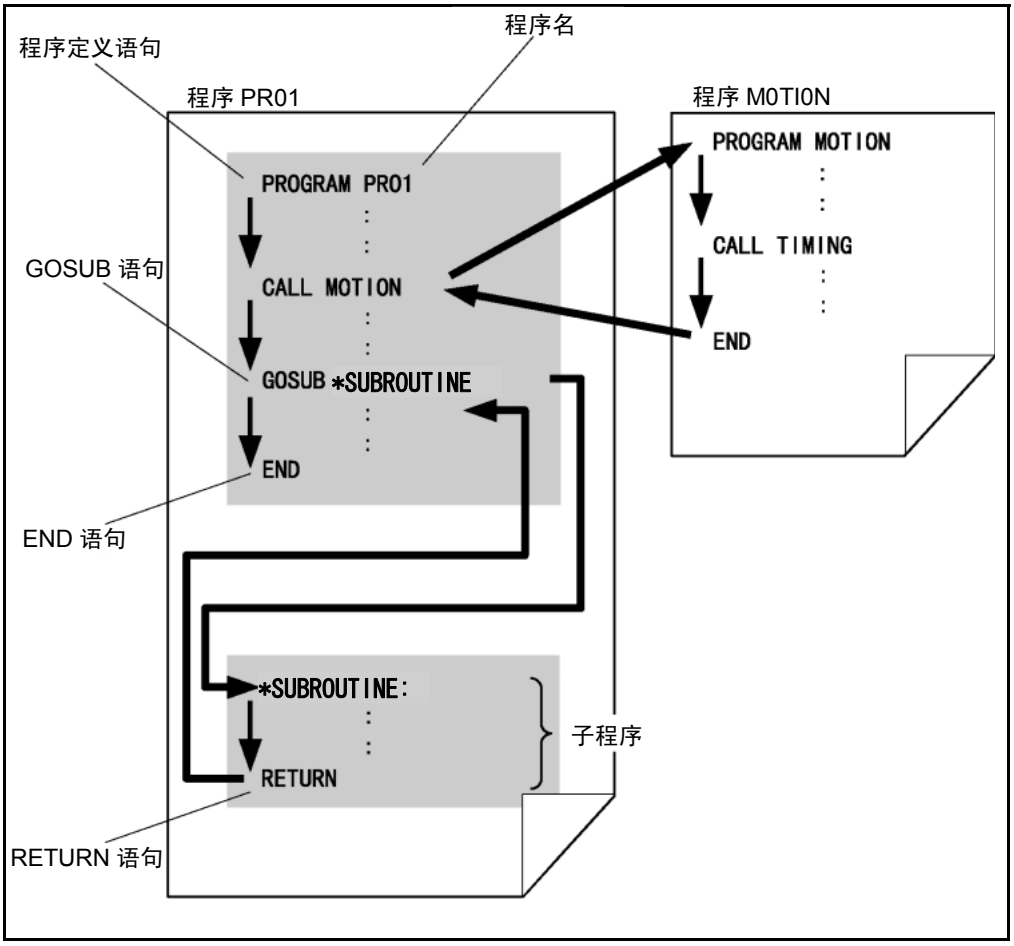


图2-1 程序与子程序的调出方法的不同

2.1.1 子程序的调出

在一个程序中，若想在不同的多个场所使用同样的处理时，将该处理作为子程序进行记述，在多个场所调出使用。

子程序必须与调出的程序在相同的文件中。

若用GOSUB语句调出子程序，则控制将转移到子程序。如果执行了位于所调出子程序的最末行的RETURN语句，则控制将返回到调出子程序的程序原来行的下一行。

从子程序能调出其他的子程序。

子程序与调出源处于相同的文件当中且共享局部变量，因此在调出时可以不考虑变量的交接。但是，不能从其他的文件中调出使用，也不能直接从多功能教导器和外部装置进行启动。

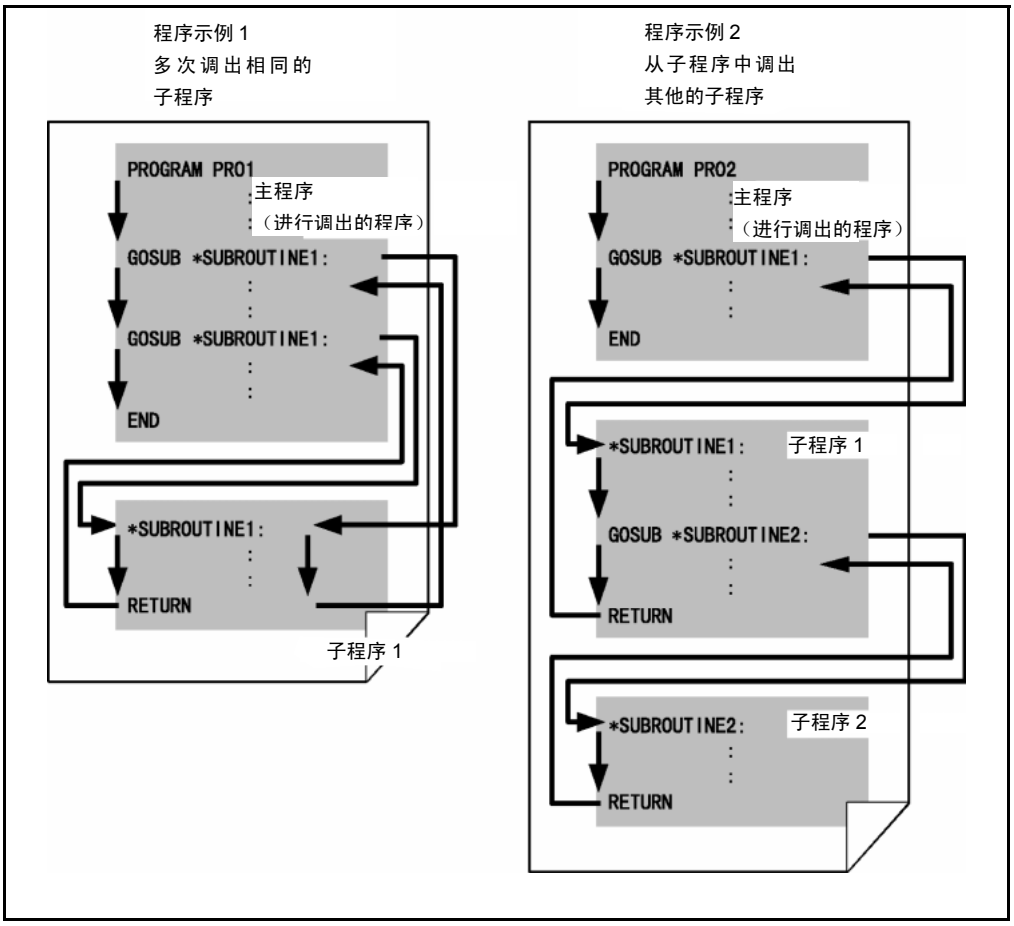


图2-2 子程序的调出

2.1.2 程序的调出

除主要执行的程序之外，另创建一个如子程序一样可以进行调出的程序。
在调出程序时，通过CALL语句指定程序名来执行。
如果使用CALL语句调出程序，则控制将转移到所调出的程序。如果执行了位于所调出程序的最末行的END语句，则控制将返回到调出源程序的下一行。
所调出的程序也可以再调出其他的程序。但是，被调出的程序不可以调出处于调出源的程序。
某一个程序可以被多个程序调出，所以具有通用性的程序可以提高开发效率。
在程序的调出中只有全局变量可以通用使用。因为局部 (Local) 变量没有共享，所以可视必要性作为自变量 (argument) 传递。请参照 "8.16 通过值进行调出和参照传址进行调出"。

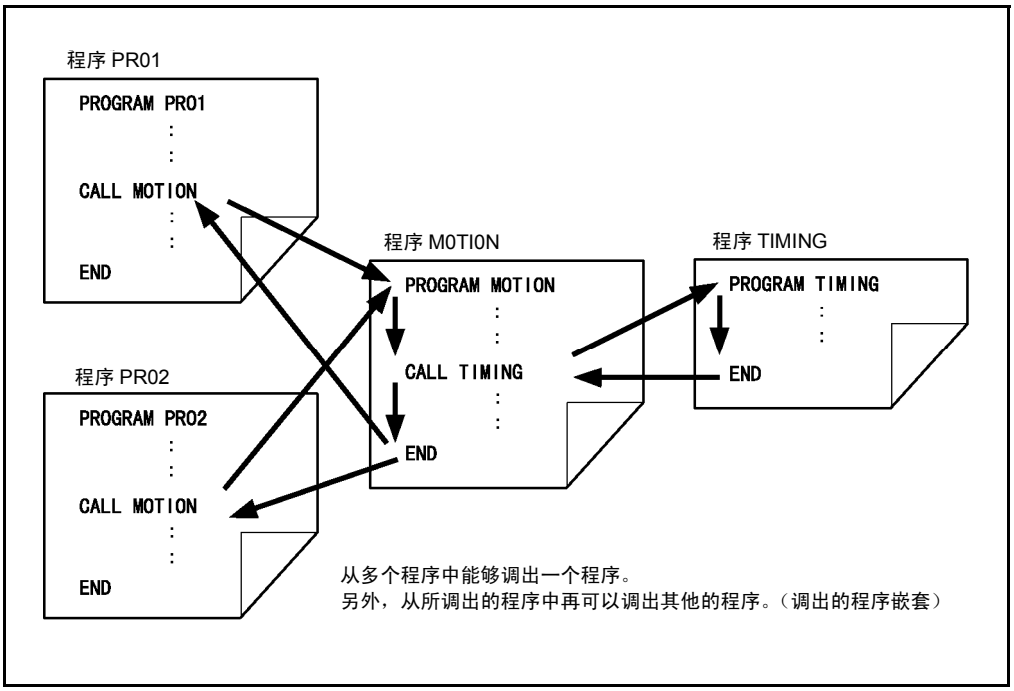


图2-3 程序的调出结构

2.1.3 程序的递归调用

在调出程序时，用CALL语句指定的程序名称，能够指定调出的程序本身。把这种操作称为递归调用。

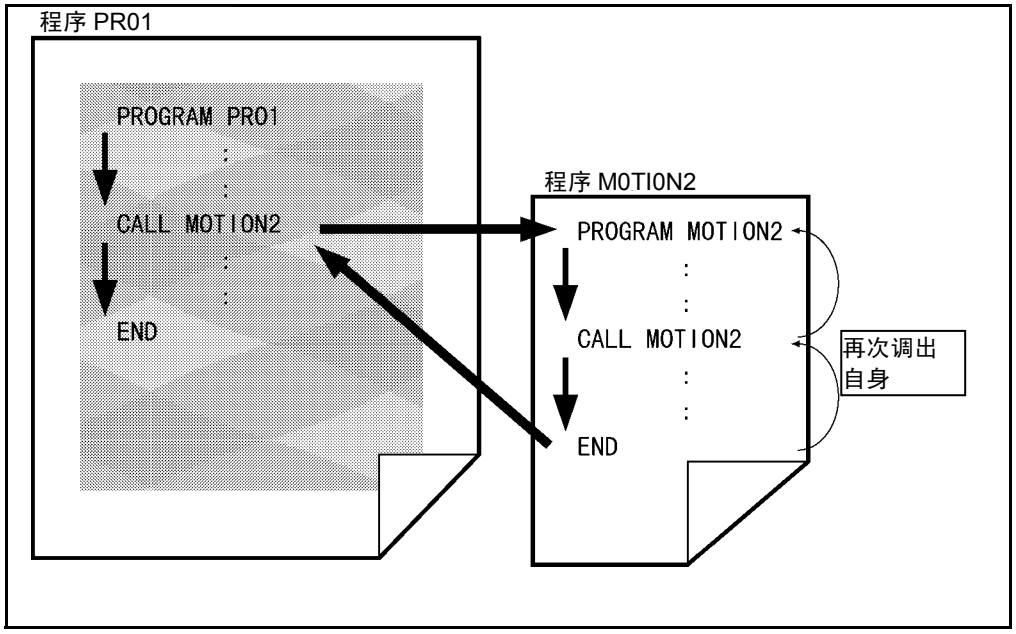


图2-4 程序的递归调用结构

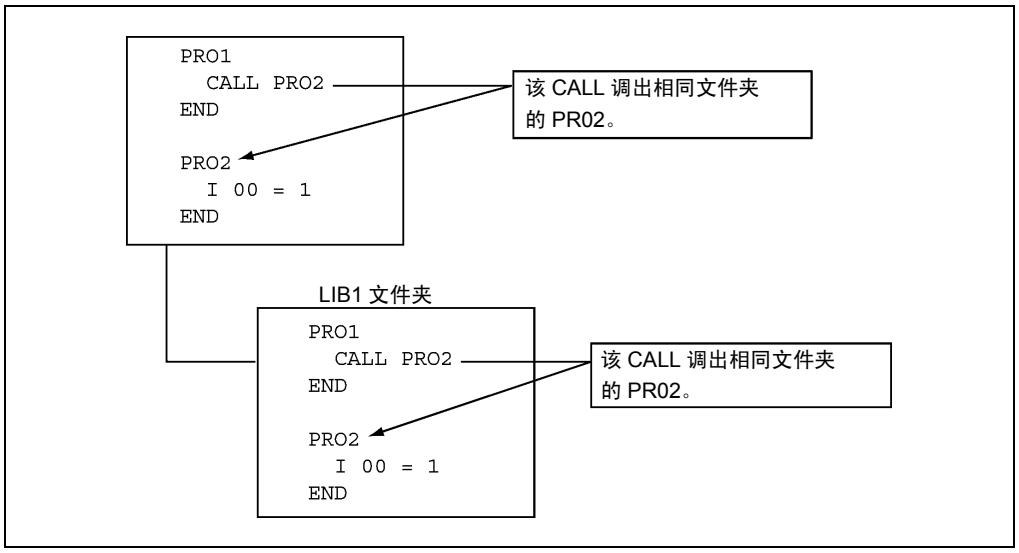
注意：因PAC程序变量被确保在每个程序的静态位置中，所以在进行递归调用时，要充分注意变量的使用方法。在通过局部 (Local) 变量设定初始值的情况下，在每次被调出时，其值都会返回到初始值。

2.1.4　文件夹结构的程序调出方法［RC7_Ver. 2.2 以后版本］

2.1.4.1　　CALL、RUN 语句

[1] 仅为程序名时

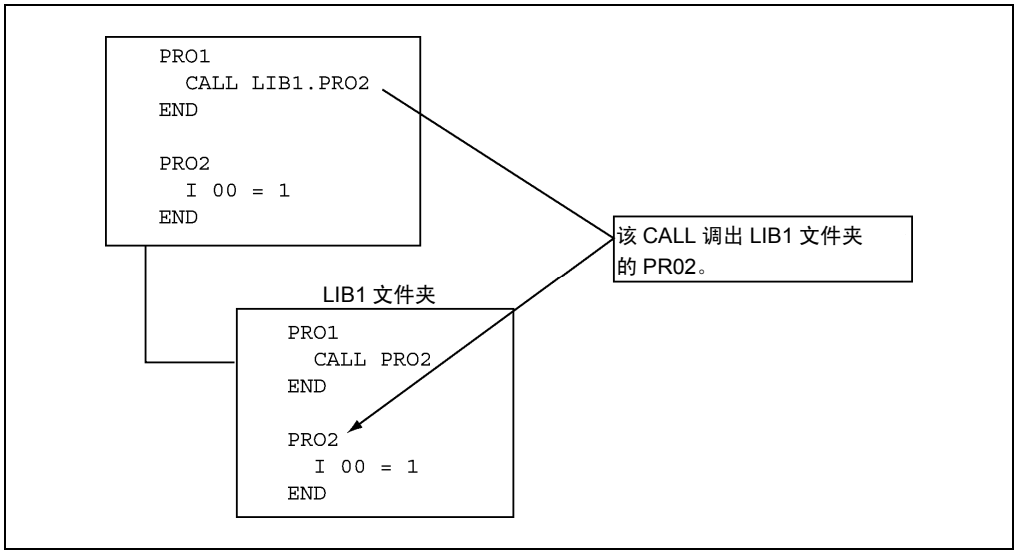
对存在调用PAC的文件夹内部的程序进行调出。



[2] 为包含文件夹名的程序名时

・ 如下所示指定程序。
　　CALL（文件夹名称）、（程序名称）

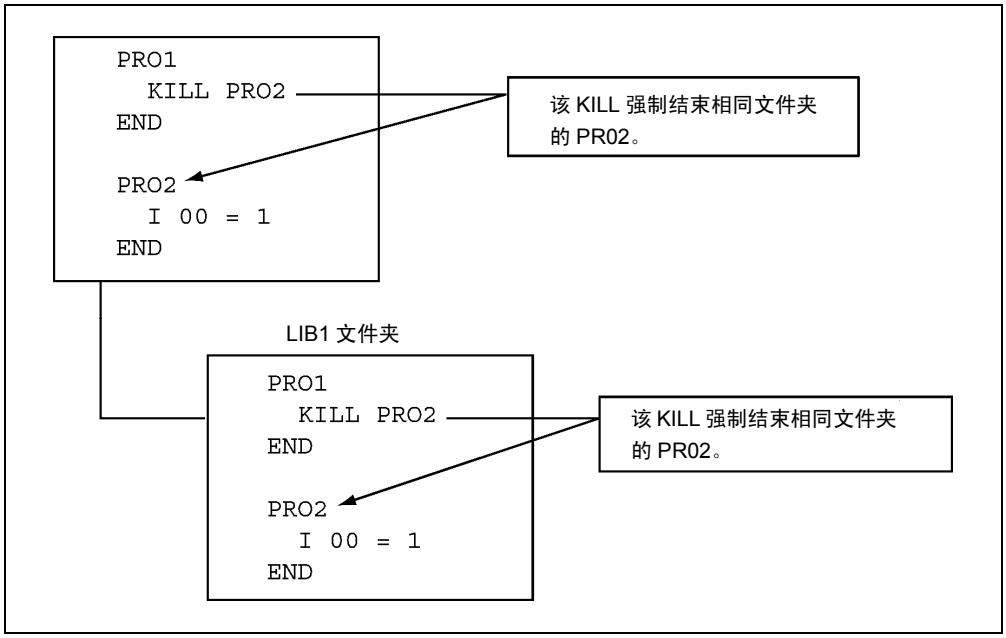
・ 可调出的文件夹只能为到下一层为止的文件夹。
・CALL（文件夹名称）、（文件夹名称）、（程序名称）为编译错误。



2.1.4.2 KILL、SUSPEND、STATUS 语句

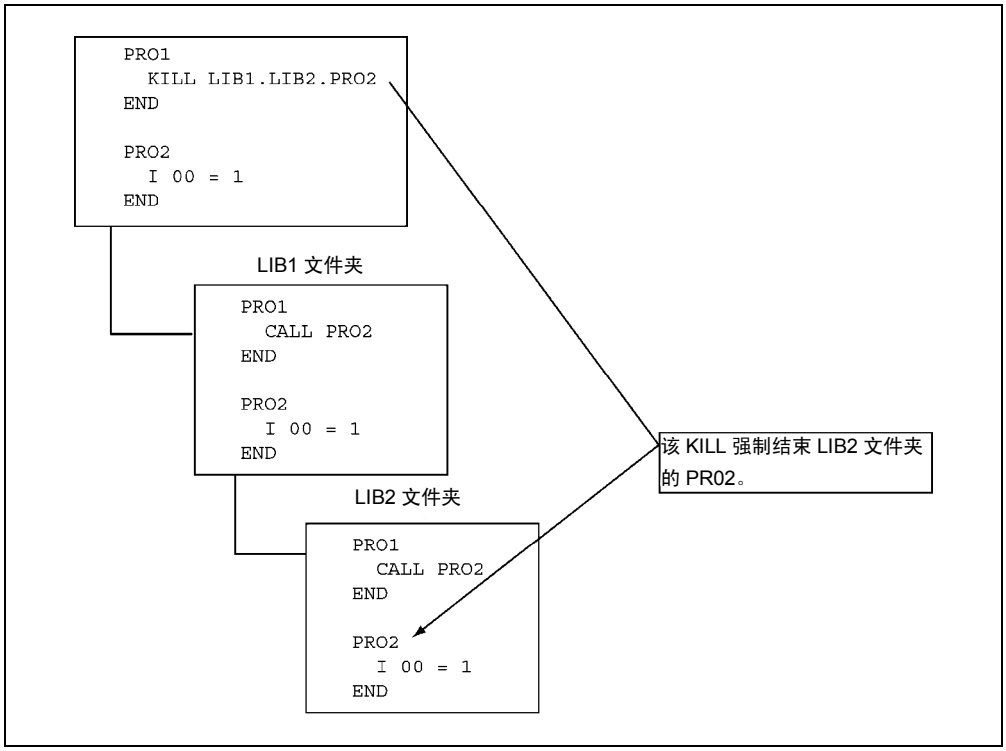
[1] 仅为程序名时

对存在调用PAC的文件夹内部的程序进行处理。



[2] 为包含文件夹名的程序名时

- 如下所示指定程序。
  KILL（文件夹名称）、（程序名称）
- 可以进行多个文件夹的指定。
  （例） KILL（文件夹名称）、（文件夹名称）、（程序名称）



2.2 程序的启动

启动程序时

●通过操作多功能教导器来启动
●通过操作小型教导器来启动
●从外部机器通过I / O来启动
共有以上3种方法。
根据启动方法的不同，能够启动的程序有以下的限制。
关于启动的程序调出其他程序的方法，在 "2.1 程序的调出和子程序" 中进行说明。

2.2.1 从多功能教导器、小型教导器进行的启动

从多功能教导器、小型教导器对程序进行启动时，要在教导检查模式或者内部自动模式下进行。
能够启动的程序仅限于没有自变量 (argument) 的程序。用任何程序名均能启动。

2.2.2 从外部机器进行启动

在外部自动模式的情况下，从外部I / O进行输入，能够启动程序。
能够启动的程序仅限于含有 "PRO <编号>" 形式程序名的程序。
能够从外部进行启动的程序，在标准模式的情况下仅限于PRO0～PRO32767；
在互换模式情况下仅限于PRO0～PRO127。

注意：含有 "PRO <编号>" 形式程序名的程序，不能具有自变量 (argument)。

2.3 多项任务

PAC语言的程序可以同时管理多个程序的执行。各个程序形成独自的运行进程，并将其称为任务。
因为同时管理多个程序的执行，所以有多个任务存在。称之为多项任务。

2.3.1 优先顺序

用RUN指令启动任务时，指定优先顺序。据此，所有的任务都分别具有各自的优先顺序。
DENSO机械手在执行程序期间，每隔一定的时间对任务的优先顺序进行监视，从等待状态的任务中切换任务，以便执行优先顺序高的任务。
由此，按照从高到低的优先顺序执行程序，如果优先顺序相同，则每隔一定时间对执行进行交替管理。用较短的时间间隔监视任务的替换，所以在表面上看是同时执行多个程序。

2.3.2 任务之间通信

通过多项任务控制机械手时，根据需要有进行同步任务控制或不允许同步任务而进行的排他控制的情况。不论什么情况，任务之间为了交换信号必须进行任务之间的通信。
在进行任务之间的通信时，通常使用信号 (semaphore)，而DENSO机械手也可以使用I / O的方法。

2.3.2.1 信号 (semaphore)

信号 (semaphore) 可以根据需要最多做到32个。在信号 (semaphore) 使用之前，如果首先用CREATESEM指令生成信号 (semaphore)，则可以获取信号 (semaphore) ID。用该信号 (semaphore) ID，就可以从多个的信号 (semaphore) 中指定特定的信号 (semaphore) 了。
在进行同步控制或者排他控制时，等待其他的任务指令一方的任务执行TAKESEM指令，发出指令一方的任务等待执行GIVESEM指令。发出指令一方的任务在准备完成之后，执行GIVESEM指令，允许等待信号 (semaphore) 的任务进行处理。
一个GIVESEM指令仅对于一个等待信号 (semaphore) 任务有效。
多个任务在等待相同信号 (semaphore) ID的信号 (semaphore) 时，对于从哪一个任务开始执行，可以从先后顺序或者优先顺序的2个等待执行方式中进行选择。用CREATESEM指令指定等待方式。

以下的2个程序，"MOTION1" 和 "TIMING1" 是使用信号 (semaphore) 进行同步控制的示例。
如果启动程序 "TIMING1"，则并行启动程序 "MOTION1"。"MOTION1" 通过接受发出的信号 (semaphore) "TIMING1"，执行MOVE指令。

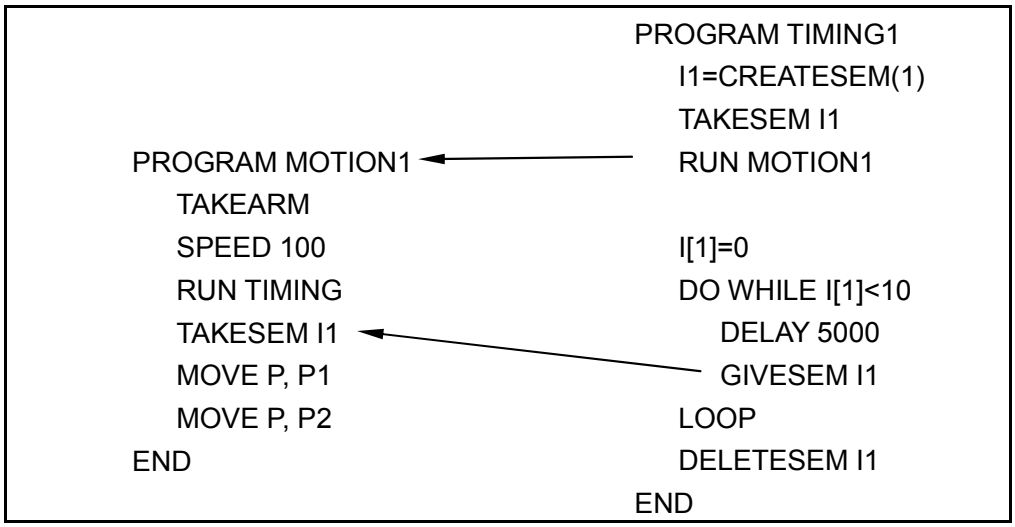


图2-5 由信号 (semaphore) 进行的同步控制的示例

2.3.2.2 I / O

使用I / O，可以进行任务之间的通信。
以下的2个程序，"MOTION2" 和 "TIMING2" 是使用I / O的进行同步控制的示例。与 "2.3.2.1 信号 (semaphore)" 中所说明的 "根据信号 (semaphore) 所进行的同步控制示例" 进行同样的动作。
用I / O进行任务之间通信的情况下，等待只按照先后顺序进行。
如果启动程序 "TIMING2"，"MOTION2" 则更新IO [118] ，如果O [118]被更新，"MOTION2" I则执行MOVE指令。然后，进行重复直至 "TIMING2" 的循环结束。

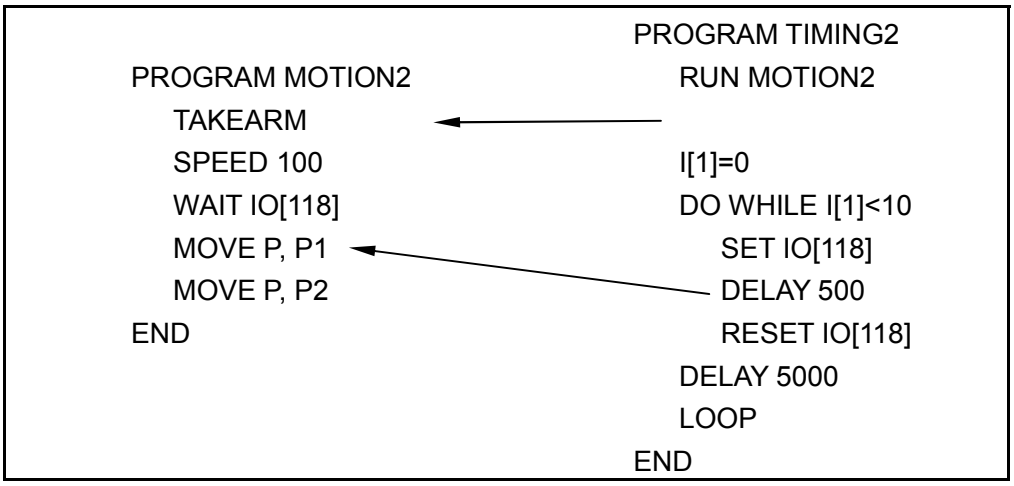


图2-6 由I / O进行的同步控制的示例

2.4 串行通信

2.4.1 线路编号

机械手控制器的线路编号和频道编号的对应情况如下所示。

机械手控制器的线路编号

电线编号	频道编号	用途
-1（注1）	-	μVision端口　RS-232C端口
0	ch1	多功能教导器连接端口
1	ch2	通用端口
2	ch3	（预约）
3	ch4	（预约）
4～7	ch5～ch8	Ethernet服务器端口
8～15	ch9～ch16	Ethernet顾客端口（注2）

注1： 仅以下指令可以使用。

- FLUSH
- printb
- inputb
- lprintb
- linputb
- com_state

此外，在使用 μVision端口 RS-232C端口时，请对RS-232C的通信进行如下设定。

传送速度：9600bps
比特长：8
奇偶校验比特：无
停止比特：1
流程控制：无

注2： 使用Ethernet顾客端口时，需要使用［7.2 串行二进制通信］的com_encom语句、com_discom语句，对端口进行开启和关闭。

注3： 机械手控制器的顾客 (Client)、服务器 (Server) 的设定，请参照操作指南 "5.7 项"。

2.4.2 通信指令

在利用RS232C以及Ethernet进行通信的PAC语言指令中，包含以下指令。发送接收的数据有2种，每种都有各自的指令。各指令的说明请参照 "13.2项"、"13.3项"。

发送接收数据	指令
文字编码数据（ASCII编码数据）	PRINT（输出）
	WRITE（输出）
	INPUT（输入）
	LINEINPUT（行输入）
二进制数据	printb（1字节 (Byte) 数据输出）
	inputb（1字节 (Byte) 数据输入）
	lprintb（多字节 (Byte) 数据输出）
	linputb（多字节 (Byte) 数据输入）

PRINT语句和WRITE语句的输出形式不同。用WRITE语句的形式输出的字符串，可以用INPUT语句正确输入。

2.4.3 通信缓冲区的清空

通过FLUSH语句，使RS232C及Ethernet的通信缓冲区清空。
请务必在开始通信之前实施。

- 参照FLUSH："13.2项"。

2.4.4 通信端口指令

RS232C以及Ethernet的各端口设置及状态获取的指令如下所示。各指令的功能因是RS232C或Ethernet端口而有所不同。
指令的说明请参照 "13.3项"。

指令	功能	
	RS232C	Ethernet
com_encom	发送接收二进制数据的设定	通过用户端口发送接收数据的设定
com_discom		
com_state	发送接收各数据时端口状态的获取	

2.4.5 示例、应用

在该项中，设计了简单的利用串行通信的应用程序，示例出多个程序。请在实际程序应用中作为参考使用。

在该例的应用程序中，机械手控制器计数动作次数，每隔一定时间发送一次动作次数的数据。在计算机侧，接受机械手控制器所发出的动作数据。机械手控制器一方的程序，用PAC语言编写。作为计算机侧的示例，使用Visual Basic。

在机械手控制器和计算机上分别安装了在 "2.4.4.1 机械手控制器的程序" 以及 "2.4.4.2 计算机的程序" 中所表示的程序，通过让其运行，可以通过RS232C线路发送机械手的动作次数，使计算机接收。

2.4.5.1 机械手控制器的程序

图2-7和图2-8是机械手控制器一方的程序示例。

这2个程序，可以以多项任务方式动作。

在对计数器和坐标进行初始化之后，程序 "PRO1" 并行启动 "PRO2"。同时，"PRO1" 使机械手在 "P2" "P1" 之间进行往复运动。在此期间内，每隔2秒（2000毫秒），"PRO2" 从RS232C端口 (ch2) 将存放在I [1] 中的机械手动作参数进行发送。

```
!TITLE "机械手动作"
PROGRAM PRO1
    I1=0                          '计数器清空
    P1=(370,400,750,-90,90,0,5)   '坐标设定
    P2=(510,400,750,-90,90,0,5)   '坐标设定
    RUN PRO2                      '任务并行启动
    TAKEARM                         '机器手控制权的获取
    DO
      MOVE P,P2                     '移向 P2 点。
      DELAY 1000                    '1 秒钟等待。
      MOVE P,P1                     '移向 P1 点
      I1=I1+1                       '计数移动次数
    LOOP                          '进行循环。
END
```

图 2-7 机械手控制器的程序 "PRO1" "机械手动作"

```
!TITLE "发送"
PROGRAM PRO2
        DO
      PRINT #2,I1
      DELAY 2000
        LOOP
END
```

图 2-8 机械手控制器的程序 "PRO2" "发送"

2.4.5.2 计算机侧的程序

计算机侧的程序如下所示。

程序清单的开始之处为定义用于进行通信的程序模块。在清单的最后，作为使用示例，有用于使用各个模块的数据接收程序。

注意：该程序使用文化Orient株式会社的通信控制装置 (PDQComm)，使用微软公司的Visual Basic编写。因此，请在安装有Visual Basic及PDQComm的计算机中使用。

```
'    模块定义语句
'
'    通信端口及通信速度的设定
'    PortNo%       ：通信端口的指定      1 or 2
'    ComInitString$：通信速度等设定      "9600,N,8,1" 等
'
Private Sub CommOpen (PortNo%,ComInitString$)
    '---------------------------------------
    ' 通信速度等设定
    '---------------------------------------
    PDQComm1.CommPort = PortNo%
    PDQComm1.Settings= ComInitString$        ' "9600,N,8,1"
    '---------------------------------------
    ' 无信号 (handshake) 交换
    ' 输入超时值设定（毫秒单位）
    ' 接收缓冲区大小的设定
    ' 发送缓冲区大小的设定
    '---------------------------------------
    PDQComm1.Handshaking = 0
    PDQComm1.InTimeout = 50
    PDQComm1.InBufferSize = 2048
    PDQComm1.OutBufferSize = 2048

    '---------------------------------------
    ' 打开端口
    '---------------------------------------
```

续下页

```
        PDQComm1.PortOpen = True
End Sub


'
'    关闭通信端口。
'
'
Private Sub CommClose ()
        '---------------------------------------
        ' 关闭端口。
        '---------------------------------------
        PDQComm1.PortOpen = False
End Sub

'
'    用双引号将赋予通信端口的字符串括起来，
'    在最后附加换行回车符后发送。
'    SendText$:    想要发送的字符串
'
Private Sub CommWrite (SendText$)
        '---------------------------------------
        ' 发送字符串和控制编码 ich (CR+LF) 时
        '---------------------------------------
        PDQComm1.Output = Chr (&H22) &SendText$ & Chr (&H22) & Chr
        (13) & Chr (10)

        '---------------------------------------
        ' 等待直到输出缓冲区为空
        '---------------------------------------
        Do
                DoEvents
        Loop Until PDQComm1.OutBufferCount = 0
        End Sub

'
'   在回车之前获取从通信端口接受的字符串，并将字符串返回。
'
Private Function CommRead ()

        '---------------------------------------
        ' 数据读取处理
        '---------------------------------------
        InString$ = ""
        Do While 1
                DoEvents
```

续下页

```
            '----------------------------------------
            '读取接收数据
            '----------------------------------------
            If PDQComm1.InBufferCount > 0 Then
               InString$ = InString$ +PDQComm1.Input
               If InStr (1, InString$,vbCr, vbBinaryCompare) <> 0 Then
Exit Do
            End If
      Loop
      CommRead = InString$

End Function

使用示例
Private Sub mnuCh1_Click ()
      ' 使用端口 1，将波特率置为 19200。
      CommOpen 1, "19200,N,8,1"
      ' 从控制器接收移动计数器。
      ReadBuf$ = CommRead
      ' 关闭所使用的端口。
      CommClose
End Sub
```

图 2-9 计算机侧的程序

2.5 共用程序库 (Library)

为了便于按照用途进行利用，将具有通用性的程序象零部件一样进行集中的程序库称为程序的共用程序库 (Library)。在PAC语言中，因为可以从程序中调出其他的程序，所以可以利用共用程序库 (Library) 中的程序，或者将编写的程序登录到共用程序库 (Library)，实现高效率开发程序。

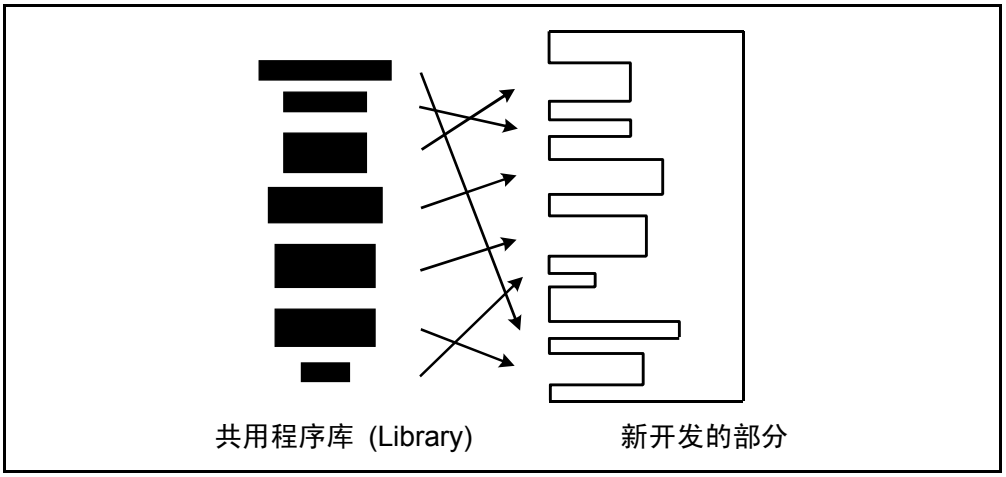


图2-10 利用共用程序库 (Library) 进行程序开发的图像

2.5.1 程序库 (Program Bank)

在WINCAPSⅢ中，备有用于共用程序库 (library) 的程序库 (Program Bank)。程序库 (Program Bank) 是将程序 (Program) 作为共用程序库 (Library) 进行登录或用于将所登录的程序加入到项目中的工具。

关于程序库 (Program Bank) 的操作方法，在WINCAPSⅢ指南 "程序库 (Program Bank)" 中进行了说明，请予参照。

2.5.1.1 共用程序库 (Library)的分类

已经事先收录在程序库 (Program Bank) 中的程序是按范畴分类的。

由用户新登录到共用程序库(library) 中的程序，请创建一个新的范畴进行登录。详情请参照编程手册Ⅱ。

2.5.2 码垛堆积共用程序库 (Library)

在PAC语言中，虽然没有用于码垛堆积的专用指令，但是通过使用作为共用程序库 (Library) 准备的程序可以实现码垛堆积动作。

用于码垛堆积的共用程序库 (Library)，可以使用程序库 (Program Bank) 追加到项目中。

2.5.2.1 码垛堆积

如图2-12所示的有间隔的托板上，顺次进行零部件的装入和取出的操作，称为码垛堆积。

码垛堆积的共用程序库 (Library) 程序为只要教导托板的间隔数和四个角的位置即可进行码垛堆积动作。

码垛堆积程序，在每次动作调出时，会按顺序改变托板的取出位置。

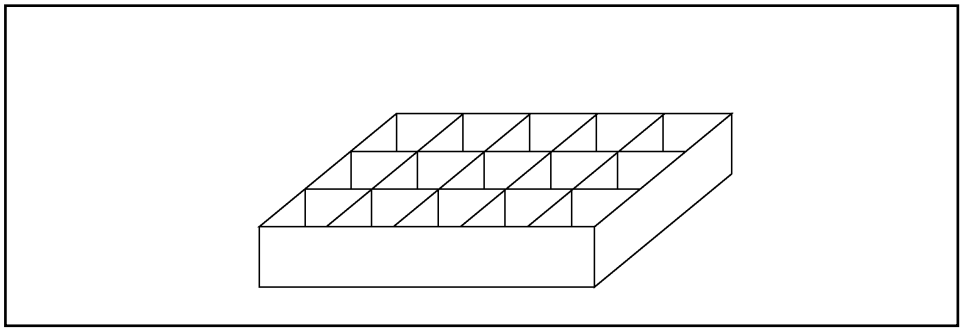


图 2-12 有间隔的托板

[1] 码垛堆积的参数

为了执行码垛堆积，所需的参数如图2-13、图2-14、图2-15、表2-1所示。
在PAC语言中，将这些参数作为变量的值保存起来。

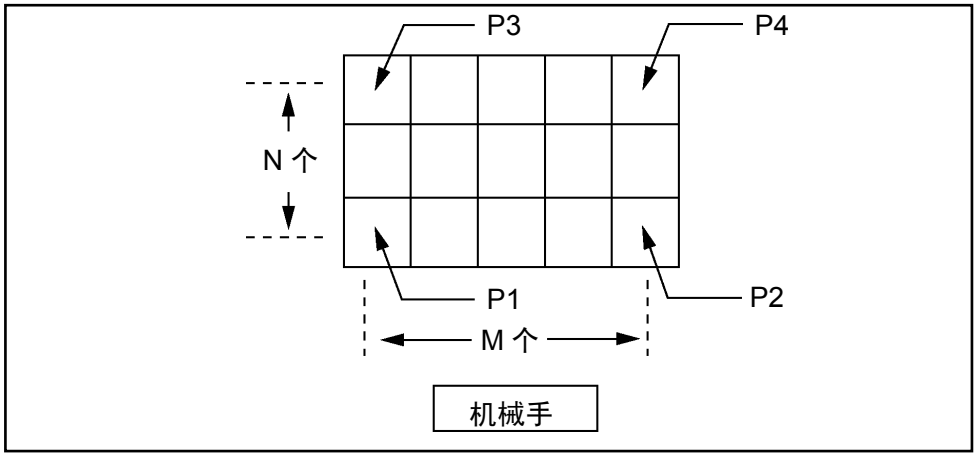


图 2-13　台架的俯视图

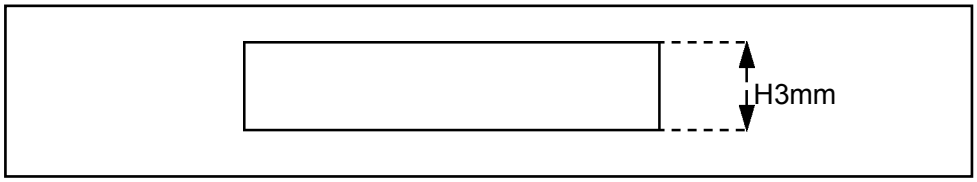


图 2-14　台架的侧视图

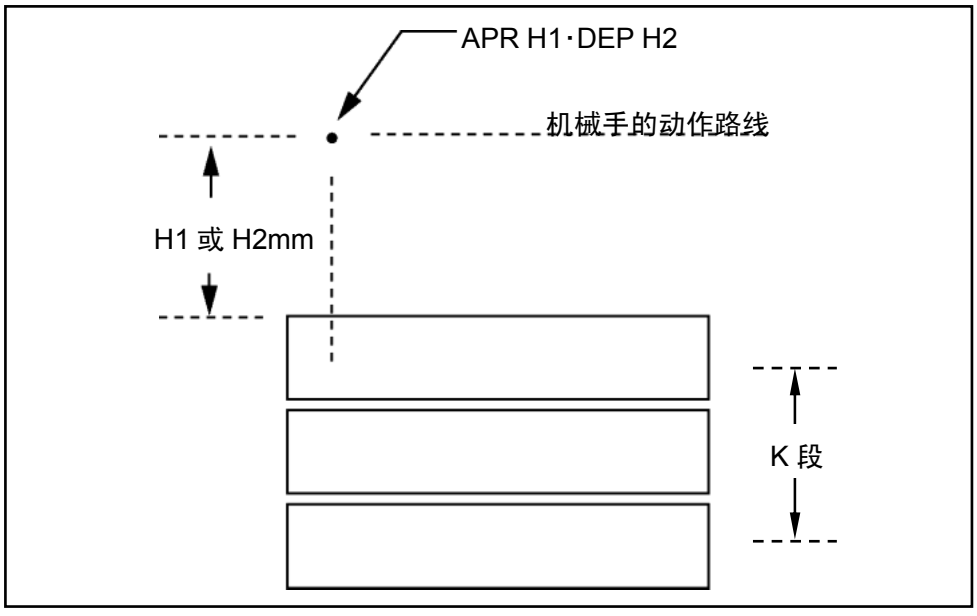


图 2-15　台架的段积图

表 2-3 码垛堆积所需的参数

记号	名称	意义	单位
	码垛堆积编号	码垛堆积的索引编号	无（整数）
N	横拆分数	从 P1 到 P3 方向的拆分数	个（整数）
M	纵拆分数	从 P1 到 P2 方向的拆分数	个（整数）
K	段积数	托板的段积数	个（整数）
H1	接近长度	机械手靠近托板时的接近长度	mm（单精度实数）
H2	起程长度	机械手离开托板时的起程长度	mm（单精度实数）
H3	托板高度	托板的 1 段的高度	mm（单精度实数）
	但是，H1 和 H2 必须满足以下的条件。 H1 > {H3 × (K-1)} + 5 H2 > {H3 × (K-1)} + 5		
P1 P2 P3 P4	图 2－13 所示的托板四角的点。各点的相对位置关系，不能进行互换。 另外，机械手的姿势为教导 P1 的位置时的姿势，在所有的点上均维持该姿势。		

N 横拆分数

表示托板的横向拆分数。

在图2-13中成为3列。

M 纵拆分数

表示托板的纵向拆分数。

在图2-13中成为5列。

K 段积数

表示托板的段积数。

在图2-15中成为3段。

H1 接近长度

表示机械手靠近托板时的接近长度。

每次调出码垛堆积程序时均使用相同的接近长度。

H2 起程长度

表示机械手离开托板时的起程长度

每次调出码垛堆积程序时均使用相同的起程长度。

H3 托板高度

表示托板的1段的高度。

托板逐渐向上累积时，输入正的数值。

托板逐渐减少时，输入负的数值。

托板的段数不变化时，输入0。

注意：H1和H2必须满足以下的条件。

$$H1 > \{H3 \times (K-1)\} + 5$$

$$H2 > \{H3 \times (K-1)\} + 5$$

如果不满足该条件，则在初始化时会显示错误。此条件的存在是为了不使机械手碰撞到托板，同时也为了将高出托板段数最多时的5mm点作为接近、起程的位置。不管段累积是否增减，均如图2-16所示，接近点、起程点都是相同的。

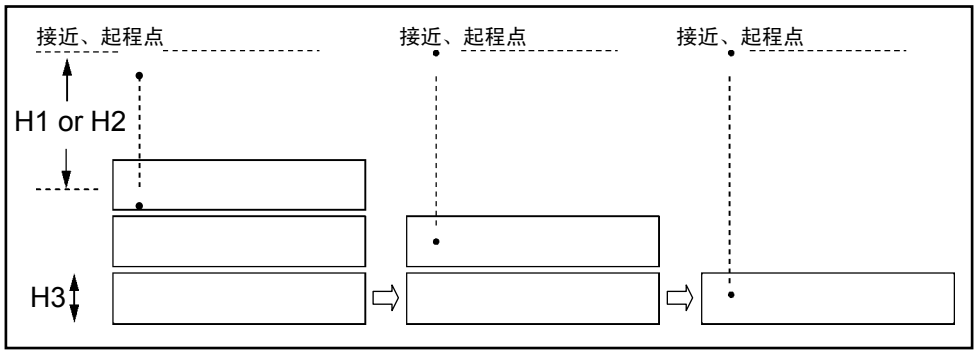


图 2-6 段累积的变化和接近点、起程点

P1、P2、P3、P4四个角的点

表示托板的四个角的零部件位置。各个点和执行顺序的关系如图2-17所示。

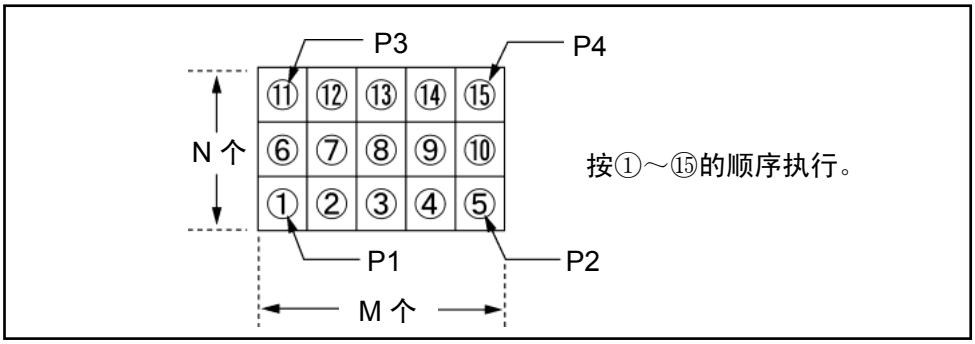


图 2-17 码垛堆积的顺序

[2] 参数值的设定

码垛堆积的纵横拆分数及段积数等参数值的设定通过调出共用程序库 (Library) 程序的 "pltInitialize"进行。

在进行 "系统项目的新建" 时，在［新建项目］对话框的［设备类型］栏中，如果选择了[1-码垛堆积],则自动将标题为[码垛堆积初始化模板1]的程序 "PRO2" 登录到共用程序库 (Library) 中。该程序因为调出 "pltInitialize"，所以将CALL语句的自变量 (argument) 的值改为适当的值来执行。

```
'!TITLE "码垛堆积 初始化模板 1"
'!AUTHOR "DENSO 机械手技术部"

PROGRAM PRO2
    CALL pltInitialize (0,4,3,1,50,50,50,52,53,54,55)    '请变更为适当的名称。
END
```

图 2-18 共用程序库 (Library) 程序 "码垛堆积初始化模板 1"

CALL pltInitialize (0, 3, 5, 3, 50, 50, 10, 52, 53, 54, 55) 的参数的说明

第 1 参数... 码垛堆积程序编号 (0)
第 2 参数... 横拆分数 (3)
第 3 参数... 纵拆分数 (5)
第 4 参数... 段积数 (3)
第 5 参数... 接近长度 (50mm)
第 6 参数... 起程长度 (50mm)
第 7 参数... 托板的高度 (10mm)
第 8 参数... P1 位置 (P52)
第 9 参数... P2 位置 (P53)
第 10 参数... P3 位置 (P54)
第 11 参数... P4 位置 (P55)

} 仅指定 P 型变量的编号。

图 2-19 "pltInitialize" 的参数的说明

该参数的含义也可以通过WINCAPSⅢ中的一个名为 "程序库" (Program Bank) 的工具查找到。

[3] 码垛堆积计数器

码垛堆积在动作过程中计数托板的间隔数，并将计数值保存在变量中。
计数器共有4个，分别是横向 (N)、纵向 (M)、高度方向 (K) 和总数 (cnt)。这些计数器在控制码垛堆积动作的核心程序、"pltKernl" 中被定义。
共用程序库 (Library) 程序的 "pltMove"，在每次对托板的位置完成一个操作，总数计数器的数值则加1，其他的计数器的数值也进行调整。
共用程序库 (Library) 程序的 "pltDecCnt"，每调出一次，则总数计数器的数值减1，各个计数器的数值也进行加减调整。
码垛堆积程序，在默认设定中可以创建到30个。因此，码垛堆积计数器也有31套。关于变更码垛堆积程序数的方法，请参照p.2-22 "[1] 码垛堆积程序的用户设计"。

计数规则

码垛堆积计数器，在每次执行 "pltMove" 时，总计数器的数值加1，并对其他计数器的数值进行调整。由此，下一个的托板位置得到保证。
总数计数器如果加1，则纵向计数器 (M) 的位置向后移动1。纵向计数器 (M) 的位置到达边缘、达到最大值时，则横向计数器 (N) 的位置向后移动1，纵向计数器 (M) 的位置成为设定的最小值。横向计数器 (N) 的位置到达边缘、达到最大值时，则高度方向的计数器 (K) 的位置向后移动1，纵向计数器 (M) 和横向计数器 (N)，成为设定的最小值。
如果中途停止码垛堆积程序之后，再重新开始，则计数器变量的数值被计算在内，向下一个位置移动。
即使切断电源，码垛堆积计数器的内容也会被保持。电源再次接通之后，如果不对码垛堆积计数器进行初始化，则接着上次结束时的状态进行码垛堆积动作。

> 注意：编译之后，加载执行程序可使变量的值初始化。

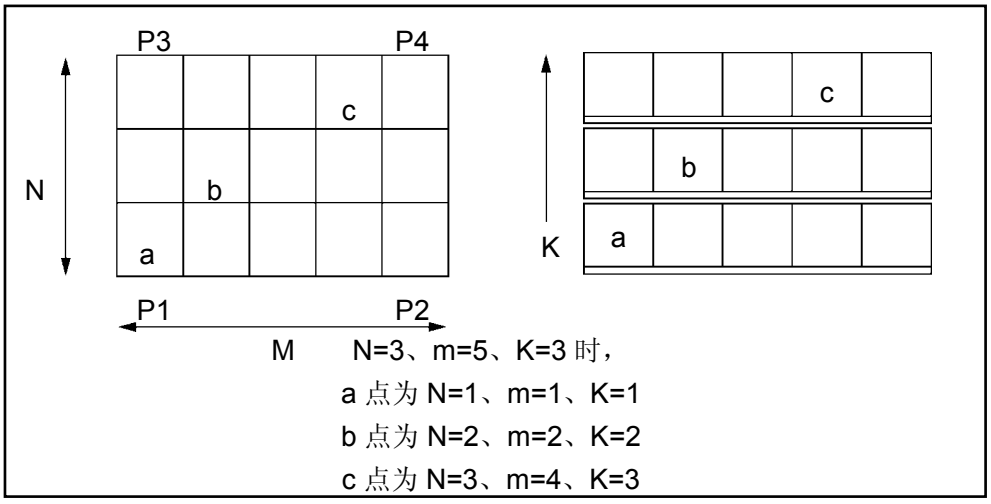


图 2-20 码垛堆积位置和计数器的关系

计数器的初始化

托板的更换和不使用托板的所有的框时，将计数器初始化。
在计数器初始化时，所有的码垛堆积计数器均赋值 "1"。
如果使用共用程序库 (Library) 程序的 "pltResetAll"，则可以将所有的码垛堆积计数器同时进行初始化。
例如：为了对托板编号为1的所有的计数器进行初始化，现说明如下。

```
CALL    pltResetAll (1)
```

为了对各个码垛堆积计数器分别进行初始化，使用 "pltLetN1"、"pltLetM1"、"pltLetK1"、"pltLetCnt"。
例如：为了对托板编号为1的N计数器进行初始化，现说明如下。

```
CALL    pltLetN1 (1, 1)
```

注意：第 2 自变量 (argument) 是在计数器上设定的值。也能设定为 1 以外的数值。

码垛堆积程序的结束信号

如果1段结束以及整个段结束，码垛堆积程序则安装1段结束标志 (Flag) 以及整个段结束标志 (Flag)。
为了获取1段结束标志 (Flag) 的状态，使用共用程序库 (Library) 程序 "pltGetPLT1END"。
为了获取整个段结束标志 (Flag) 的状态，使用共用程序库 (Library) 程序 "pltGetPLTEND"。
为了将1段结束标志 (Flag) 复位 (0)，使用共用程序库 (Library) 程序 "pltResetPLT1END"。
为了将整个段结束标志 (Flag) 复位 (0)，使用共用程序库 (Library) 程序 "pltResetPLTEND"。

2.5.2.2 码垛堆积程序

虽然实际的码垛堆积程序根据该应用程序所处的特殊环境呈现出多样性，但在标题为 "码垛堆积模板2" 的程序名 "PRO1" 中准备有使用共用程序库 (Library) 组建程序时的标准步骤。

请以该 "码垛堆积模板2" 为模版，并追加各应用的所需事项，以协助程序的开发。

图2-21中表示了程序名 "PRO1" 的 "码垛堆积模板2"。在该示例中，假定如下所示的设备。

- 码垛堆积点为P50～P55。
- 编写的程序向固定位置P50移动，执行码垛堆积程序编号0，向抓取位置P51移动，执行松开卡盘动作，确认段结束，如果结束信号被输出，则视必要性执行托板的替换动作，直至没有工件时结束。

```
!TITLE "码垛堆积模板 2"
!AUTHOR "DENSO 机械手技术部"
#DEFINE pltIndex      0                    '码垛堆积程序编号
                '可以选择'0～30 范围内的任意值。
#DEFINE ChuckNG       40                   '抽取检查 IO 的编号
                '可以选择任意值。

PROGRAM PRO1                               '请更改为适当的名称。
  TAKEARM   '臂信号 (semaphore) 的获取
  MOVE P, P50                              '向规定位置 P50 移动。
                '向码垛堆积的 P50 移动。
  IF IO[ChuckNG] = ON THEN                 '前次的抽取检查。
    CALL pltDecCnt (pltIndex)              '将总计数器设置于-1。
  END IF
  CALL pltMove (pltIndex)                  '执行码垛堆积编号 0。
  MOVE P, P51                              '向抓取位置 P51 移动。
                '向码垛堆积的 P51 移动。
  '<--- 插入松开卡盘的动作等 --->
  CALL pltGetPLT1END (pltIndex,0)          '在 I [0] 获取 1 段终了信号
                '在下一行判断第 2 个自变量 (argument) "0"
                ' "I"的矩阵编号
  IF I[0] THEN 'ON 时（即<>0 时）
    '<--- 插入托板更换动作等 --->
    CALL pltResetPLT1END (pltIndex)        '1 清空 1 段结束信号
  END IF
  GIVEARM   '臂信号 (semaphore) 的释放
END
```

图 2-21 程序名 "PRO1"、"码垛堆积模板 2"

[1] 码垛堆积程序的用户设计

Q. 将执行的码垛堆积程序变更为 1 时

A. 将 "PRO1"、"PRO2" 的 "pltIndex" 及 "0" 的值变更为 "1"。

Q. 将码垛堆积的拆分变更为横 5 阶、纵 3 阶时

A. 请将 "pltInitialize" 的第2号、第3号的参数 "3" 和 "5" 分别变更为 "5" 和 "3"。

☆ CALL pltInitialize (0, 5, 3, 3, 50, 50, 10, 52, 53, 54, 55)

Q. 将能够使用的码垛堆积程序数进行变更时

A. 通过将 "pltKernel" 程序的 "#DEFINE mcPaltMax 31" 的 "31" 变更为任意数值，能够变更码垛堆积程序数。

Q. 要设定码垛堆积的回避点时

A. 对于接近方向，追加 X=10、Y=10、Z=-10 的回避点时，在 "paltmove1.pac" 文件的 P [mcNextPos] 点上，要追加移动指令，该移动指令是向加上了平移偏差量 " (10, 10, -10, 0, 0, 0) H" 的点移动指令。

☆ 向 MOVE P, P [mcNextPos] + (10, 10, -10, 0, 0, 0) H
'X=10、Y=10、Z=-10 的回避点移动

Q. 要获取 N1、M1、K1 时

A. 请追加共用程序库 (Library) "pltGetK1"、"pltGetM1"、"pltGetN1"。

☆ call pltGetK1 (Index, iValue)

☆ call pltGetM1 (Index, iValue)

☆ call pltGetN1 (Index, iValue)

'Index ... 码垛堆积程序编号。

'iValue... K1, M1, N1 的值被返回的 I 型变量的编号。

例）#define pltIndex 0 '码垛堆积程序编号

```
Program Exmp
DefInt   Index, iValue
Index = pltIndex    '在 Index 寄存器上插入码垛堆积程序编号。
call pltGetK1 (Index, iValue)        'K1 值被返回到用 iValue
                                     '所指定的 I 型变量上。
call pltGetM1 (Index, iValue)        'M1 值被返回到用 iValue
                                     '所指定的 I 型变量上。
call pltGetN1 (Index, iValue)        'N1 值被返回到用 iValue
                                     '所指定的 I 型变量上。
End
```

Q. 要变更 N1、M1、K1 时

A. 请追加共用程序库 (Library) "pltLetK1"、"pltLetM1"、"pltLetN1"。

☆ call pltLetK1 (Index, iValue)

☆ call pltLetM1 (Index, iValue)

☆ call pltLetN1 (Index, iValue)

'Index ... 码垛堆积程序编号。

'iValue... 该值被输入在 K1、M1、N1 上。

例）#define pltIndex 0 '码垛堆积程序编号

```
Program Exmp
DefInt  Index, iValue
Index   = pltIndex          '在 Index 寄存器上，插入码垛堆
                            '积程序编号。
iValue = 10                 '输入要在 K1、M1、N1 上所要
                            '输入的值。
call pltLetK1 (Index, iValue)       '在 K1 上，iValue 的值
                                    '被输入。
call pltLetM1 (Index, iValue)       '在 M1 上，iValue 的值
                                    '被输入。
call pltLetN1 (Index, iValue)       '在 N1 上，iValue 的值
                                    '被输入。
End
```

Q. 要获取总计数器时

A. 请追加共用程序库 (Library) "pltGetCnt"。

☆ call pltGetCnt (Index, iValue)

'Index ... 码垛堆积程序编号。

'iValue... 总计数器的值返回的 I 型变量的编号。

例）#define pltIndex 0 '码垛堆积程序编号

```
Program Exmp
DefInt  Index, iValue
Index = pltIndex            '在 Index 寄存器上插入码垛堆
                            '积程序编号。
call pltGetCnt (Index, iValue) '总计数器的值返回到 iValue。
End
```

Q. 要变更总计数器时

A. 请追加共用程序库 (Library) "pltLetCnt"。

☆ call pltLetCnt (Index, iValue)

'Index ... 码垛堆积程序编号。

'iValue... 该值被输入在总计数器上。

例）

```
#define pltIndex      0                    '码垛堆积程序编号
Program Exmp
DefInt   Index, iValue
Index    = pltIndex                        '在Index寄存器上插入码垛堆积
                                           '程序编号。
iValue = 0                                 '输入要在总计数器上所要输入
                                           的值。
call pltLetCnt (Index, iValue)             'iValue 的值输入在
                                           '总计数器上。
End
```

Q. 总计数器是用什么样的时间选择进行加算?

A. 在执行 "pltKernel (Index, -15, mcNextPos, ndErr)" 时进行加算。因此，如果执行共用程序库 (Library) 的 "pltGetNextPos"、"pltMove"、"pltMove0"，则总计数器被加算。

☆ call pltMove (Index)

☆ call pltMove0

☆ call pltGetNextPos (Index, NextPos)

'Index ... 码垛堆积程序编号。

'NextPos .. 下一个位置被输入到用户指定位置的 P 型变量编号上。

2.6 TP简易操作盘［Ver.1.5以上版本］

任意的大小、位置、颜色的按钮的创建以及在画面上的粘贴以PAC语言实现。同时，通过动作一次按钮、页设定PAC程序，使操作盘有效。另外，一度设定后，如果不清空操作，则操作盘将持续动作。

可使用的按钮、画面

- 按钮…500个以内
- 画面….50画面以内

可创建操作盘画面的指令

set_button	按钮参数的设定（根据属性也兼表示ON / OFF）
set_page	页参数的设定（根据属性也兼表示ON / OFF）
change_bCap	按钮字幕变更
change_pCap	页字幕变更
disp_page	页显示

通过指令所设定的参数

按钮参数	• 索引 • 表示位置（左上 x 坐标、左上 y 坐标、右下 x 坐标、右下 y 坐标）， • 按钮类型（触摸开关［有形状的版本］，数值表示框，数值输入框，数字开关、指示灯［有形状的版本］、页切换按钮） • 状态（预约） • 背景颜色 • 文字颜色 • 有效标志 (Flag) • 表示标志 (Flag) • 字幕 • 变量类型 • 变量编号 • IO 编号 • 页编号 • 变更标志 (Flag)（预约） • 结果（预约）
页参数	索引、画面类型（固定）、状态（预约）、背景颜色、文字颜色、有效标志 (Flag)、表示标志 (Flag)、字幕、变更标志 (Flag)（预约）、结果（预约）

编制 TP 简易操作盘程序

为了表示操作盘画面，进行以下的操作。

1) 按钮参数的设定

使用set_button命令，决定按钮的参数。实行该全部参数。
（例）设定编号1的按钮的背景颜色 (7) 为黑 (0)。

```
set_button 1、7、0
```

2) 页参数的设定

使用set_page命令，决定页的参数。实行该全部参数。
（例）设定第2页的页背景颜色 (3) 为红 (4)。

```
set_page 2、3、4
```

3) 按钮字幕的设定

使用change_bCap命令，决定按钮的字幕。
（例）将编号2的按钮的字幕设定在 "准备"。

```
change_bCap 2，"准备"
```

4) 页字幕的设定

使用change_pCap命令，决定页的字幕。
（例）将第3页的字幕设定在 "画面3"。

```
change_pCap 3，"画面3"
```

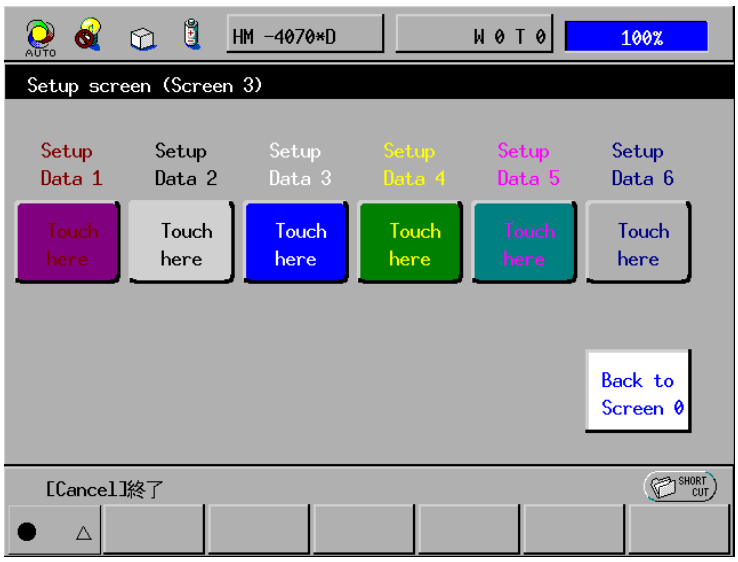
5) 页的表示

使用disp_page命令，表示页。

6) 操作盘画面的选择

从多功能教导器的基本画面选择［F9：操作盘］，表示操作盘。

（画面示例）



2.6.1 操作盘画面创建的程序示例

以下表现使用全体命令的操作盘画面创建的程序示例。

```
'!TITLE  "<标题>"
PROGRAM BUTTON3D_VAL_3
'颜色的定义
#define BLACK                0          '黑
#define BLUE                 1          '蓝
#define GREEN                2          '緑
#define CYAN                 3          '青绿色
#define RED                  4          '红
#define MAGENTA              5          '桃红色
#define BROWN                6          '棕
#define LIGHTGLAY            7          '浅灰色
#define GLAY                 8          '灰色
#define LIGHTBLUE            9          '浅蓝色
#define LIGHTGREEN           10         '浅绿色
#define LIGHTCYAN            11         '浅青绿色
#define LIGHTRED             12         '浅红色
#define LIGHTMAGENTA         13         '浅桃红色
#define YELLOW               14         '黄
#define WHITE                15         '白

'按钮的定义
#define LABEL                1          '标签
#define LINE                 2          '线
#define 2DBUTTON_V           17         '2D按钮（变更变量）
#define 3DBUTTON_V           18         '3D按钮（变更变量）
#define 3DBUTTON_V2          19         '3D按钮2（变更变量）
#define CIRCLE               20         '圆（变量变更）
#define 2DLED                33         '2DLED（指示灯）
#define CIRCLELED            34         '圆LED（指示灯）
#define 3DBUTTON_IO          35         ‘3D按钮（IO变更）

'页参数
#define P_BGCOLOR            3          '页背景颜色
#define P_CHARCOLOR          4          '页文字颜色
#define P_USESTATE           5          '可使用状态
#define P_VISSTATE           6          '可视状态
```

```
'按钮参数
#define X_UPPERLEFT_P      1          '左上X坐标
#define Y_UPPERLEFT_P      2          '左上Y坐标
#define X_LOWERRIGHT_P     3          '右下X坐标
#define Y_LOWERRIGHT_P     4          '右下Y坐标
#define B_KIND             5          ‘按钮种类
#define B_BGCOLOR          7          '按钮背景颜色
#define B_FGCOLOR          8          '按钮文字颜色
#define B_USESTATUS        9          '按钮使用可能状态
#define B_VISSTATUS        10         '按钮可视状态
#define B_VALUEKIND        11         '按钮变量种类（固定）
#define B_VALUE_NO         12         '按钮变量编号
#define B_IO_NO            13         '按钮I / O编号
#define B_DISP_PNO         14         '按钮的表示页编号

'状态定义
#define ON                 1          '使用
#define OFF                0          '不使用
#define I_VAL              1          '整数型

'按钮地址
#define IO_PB_ADRS         170        '操作盘按钮先头I / O编号

defint btn_no, minx, maxx, miny, maxy, loopcnt
defint enable, visible, var_type, var_index
defint panel_no, io_no, io_adrs, btn_adrs
defstr panel_cap
defstr bcap0, bcap1, bcap2, bcap3, bcap4, bcap5, bcap6, bcap7
    panel_no = 3

    panel_cap = "工序画面设定（画面3）"
    loopcnt = 0

    change_pCap panel_no, panel_cap          '页的标题设定
    set_page panel_no, P_BGCOLOR, GLAY       '页的背景颜色设定
    set_page panel_no, P_USESTATE, ON        '页使用的设定
    set_page panel_no, P_VISSTATE, ON        '页可视的设定
```

```
'各参数的初始化
    btn_adrs = 30
    io_no = IO_PB_ADRS
    reset io  [128 to 133]
    enable = ON
    visible = ON
    var_type = I_VAL
    var_index = 1

    bcap0= "工序" + chr$ (10) + "数据1"
    Bcap1= "工序" + chr$ (10) + "数据2"
    Bcap2= "工序" + chr$ (10) + "数据3"
    Bcap3= "工序" + chr$ (10) + "数据4"
    Bcap4= "工序" + chr$ (10) + "数据5"
    Bcap5= "工序" + chr$ (10) + "数据6"
    Bcap6= "工序0" + chr$ (10) + "返回"
    Bcap7= "在此" + chr$ (10) + "接触"

    while loopcnt < 6                                    '6回循环
        btn_no = btn_adrs + loopcnt
        minx = 10 + ((loopcnt mod 6)*100)
        miny = 50 + ((loopcnt / 6)*100)
        maxx = 100 + ((loopcnt mod 6)*100)
        maxy = 120 + ((loopcnt / 6)*100)
        io_adrs = IO_PB_ADRS + loopcnt

    '在此进行通用处理
        set_button (btn_no), (1), (minx)
        set_button (btn_no), (2), (miny)
        set_button (btn_no),  (3), (maxx)
        set_button (btn_no),  (4),  (maxy)
        set_button (btn_no),  (9),  (enable)
        set_button (btn_no),  (10),  (visible)
        set_button (btn_no),  (11),  (var_type)
        set_button (btn_no),  (12),  (var_index)
        set_button (btn_no),  (14),  (panel_no)
```

```
‘显示标签
    select case loopcnt
    case 0
        set_button btn_no, B_KIND, LABEL            '按钮类型设定
        set_button btn_no, B_IO_NO, io_adrs         'I / O地址设定
        set_button btn_no, B_BGCOLOR, GLAY          '背景颜色设定
        set_button btn_no, B_FGCOLOR, RED           '前面颜色设定
        change_bCap btn_no, bcap0                   '按钮名设定

    case 1
        set_button btn_no, B_KIND, LABEL            '按钮类型设定
        set_button btn_no, B_IO_NO, io_adrs         'I / O地址设定
        set_button btn_no, B_BGCOLOR, GLAY          '背景颜色设定
        set_button btn_no, B_FGCOLOR, BLACK         '前面颜色设定
        change_bCap btn_no, bcap1                   '按钮名设定

    case 2
        set_button btn_no, B_KIND, LABEL            '按钮类型设定
        set_button btn_no, B_IO_NO, io_adrs         'I / O地址设定
        set_button btn_no, B_BGCOLOR, GLAY          '背景颜色设定
        set_button btn_no, B_FGCOLOR, WHITE         '前面颜色设定
        change_bCap btn_no, bcap2                   '按钮名设定

    case 3
        set_button btn_no, B_KIND, LABEL            '按钮类型设定
        set_button btn_no, B_IO_NO, io_adrs         'I / O地址设定
        set_button btn_no, B_BGCOLOR, GLAY          '背景颜色设定
        set_button btn_no, B_FGCOLOR, YELLOW        '前面颜色设定
        change_bCap btn_no, bcap3                   '按钮名设定

    case 4
        set_button btn_no, B_KIND, LABEL            '按钮类型设定
        set_button btn_no, B_IO_NO, io_adrs         'I / O地址设定
        set_button btn_no, B_BGCOLOR,GLAY           '背景颜色设定
        set_button btn_no, B_FGCOLOR,               '前面颜色设定
        LIGHTMAGENTA
        change_bCap btn_no, bcap4                   '按钮名设定
```

```
            case 5
                set_button btn_no, B_KIND, LABEL              '按钮类型设定
                set_button btn_no, B_IO_NO, io_adrs           'I / O地址设定
                set_button btn_no, B_BGCOLOR, GLAY            '背景颜色设定
                set_button btn_no, B_FGCOLOR, LIGHTBLUE       '前面颜色设定
                change_bCap btn_no, bcap5                     '按钮名设定
            end select
            loopcnt = loopcnt + 1
        wend

    '下段表示
        loopcnt = 0
        while loopcnt < 6                                     '6回循环
            btn_no = btn_adrs + 10 + loopcnt
            minx = 10 + ((loopcnt mod 6)*100)
            miny = 120 + ((loopcnt / 6)*100)
            maxx = 100 + ((loopcnt mod 6)*100)
            maxy = 190 + ((loopcnt / 6)*100)
            io_adrs = IO_PB_ADRS + 6 + loopcnt
            var_index = var_index + 1

        '在此进行通用处理
                set_button (btn_no), (1), (minx)
                set_button (btn_no),  (2), (miny)
                set_button (btn_no), (3), (maxx)
                set_button (btn_no), (4), (maxy)
                set_button (btn_no), (9), (enable)
                set_button (btn_no), (10), (visible)
                set_button (btn_no), (11), (var_type)
                set_button (btn_no), (12), (var_index)
                set_button (btn_no), (14), (panel_no)
```

```
        '显示标签
            select case loopcnt
            case 0
'               set_button btn_no, B_KIND, 3DBUTTON_V      '按钮类型设定
                set_button btn_no, B_KIND, 3DBUTTON_V2     '按钮类型设定
                set_button btn_no, B_IO_NO, io_adrs        'I / O地址设定
                set_button btn_no, B_BGCOLOR, MAGENTA      '背景颜色设定
                set_button btn_no, B_FGCOLOR, RED          '前面颜色设定
                var_index = loopcnt
                set_button btn_no, B_VALUE_NO, var_index
                change_bCap btn_no, bcap7

            case 1
'               set_button btn_no, B_KIND, 3DBUTTON_V      '按钮类型设定
                set_button btn_no, B_KIND, 3DBUTTON_V2     '按钮类型设定
                set_button btn_no, B_IO_NO, io_adrs        'I / O地址设定
                set_button btn_no, B_BGCOLOR, LIGHTGLAY    '背景颜色设定
                set_button btn_no, B_FGCOLOR, BLACK        '前面颜色设定
                var_index = loopcnt
                set_button btn_no, B_VALUE_NO, var_index
                change_bCap btn_no, bcap7

            case 2
'               set_button btn_no, B_KIND, 3DBUTTON_V      '按钮类型设定
                set_button btn_no, B_KIND, 3DBUTTON_V2     '按钮类型设定
                set_button btn_no, B_IO_NO, io_adrs        'I / O地址设定
                set_button btn_no, B_BGCOLOR, BLUE         '背景颜色设定
                set_button btn_no, B_FGCOLOR, WHITE        '前面颜色设定
                var_index = loopcnt
                set_button btn_no, B_VALUE_NO, var_index
                change_bCap btn_no, bcap7

            case 3
'               set_button btn_no, B_KIND, 3DBUTTON_V      '按钮类型设定
                set_button btn_no, B_KIND, 3DBUTTON_V2     '按钮类型设定
                set_button btn_no, B_IO_NO, io_adrs        'I / O地址设定
                set_button btn_no, B_BGCOLOR, GREEN        '背景颜色设定
                set_button btn_no, B_FGCOLOR, YELLOW       '前面颜色设定
                var_index = loopcnt
                set_button btn_no, B_VALUE_NO, var_index
                change_bCap btn_no, bcap7

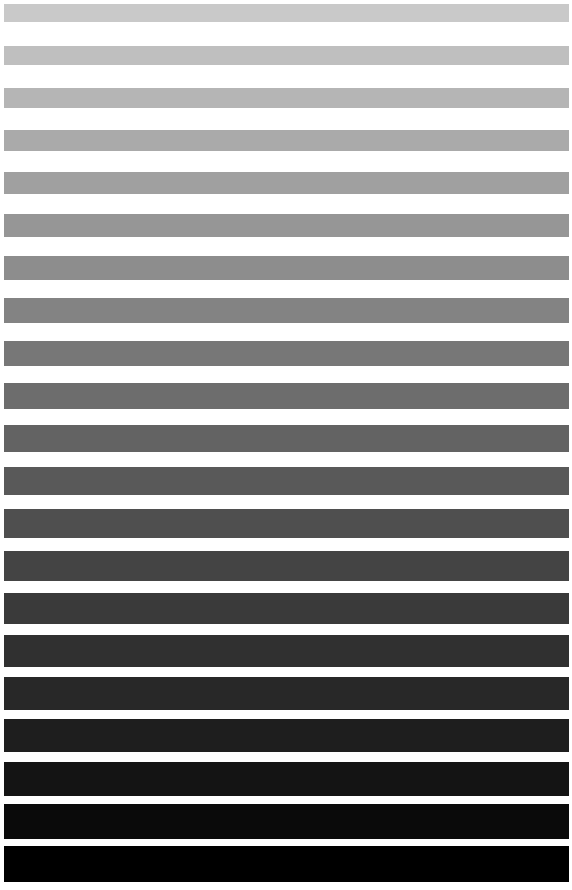
            case 4
'               set_button btn_no, B_KIND, 3DBUTTON_V      '按钮类型设定
                set_button btn_no, B_KIND, 3DBUTTON_V2     '按钮类型设定
                set_button btn_no, B_IO_NO, io_adrs        'I / O地址设定
                set_button btn_no, B_BGCOLOR, CYAN         '背景颜色设定
                set_button btn_no, B_FGCOLOR,              '前面颜色设定
                LIGHTMAGENTA
                var_index = loopcnt
                set_button btn_no, B_VALUE_NO, var_index
                change_bCap btn_no,bcap7
```

```
            case 5
'               set_button btn_no, B_KIND, 3DBUTTON_V   '按钮类型设定
                set_button btn_no, B_KIND, 3DBUTTON_V2  '按钮类型设定
                set_button btn_no, B_IO_NO, io_adrs     'I / O地址设定
                set_button btn_no, B_BGCOLOR, GLAY      '背景颜色设定
                set_button btn_no, B_FGCOLOR,           '前面颜色设定
                LIGHTBLUE
                var_index = loopcnt
                set_button btn_no, B_VALUE_NO, var_index
                change_bCap btn_no, bcap7

            end select
            loopcnt = loopcnt + 1
        wend
'返回到画面0创建3D按钮
            io_adrs = 128
            minx = 510
            maxx = 600
            miny = 250
            maxy = 320
            btn_no = 92
            set_button (btn_no), (1), (minx)
            set_button (btn_no), (2), (miny)
            set_button (btn_no), (3), (maxx)
            set_button (btn_no), (4), (maxy)
            set_button (btn_no), (9), (enable)
            set_button (btn_no), (10), (visible)
            set_button (btn_no), (11), (var_type)
            set_button (btn_no), (12), (var_index)
            set_button (btn_no), (14), (panel_no)
            set_button btn_no, B_KIND, 3DBUTTON_IO      '按钮类型设定
            set_button btn_no, B_IO_NO, io_adrs         'I / O地址设定
            set_button btn_no, B_BGCOLOR, BLUE          '背景颜色设定
            set_button btn_no, B_FGCOLOR, WHITE         '前面颜色设定
            change_bCap btn_no, bcap6                   '按钮名设定

        disp_page panel_no                              '指定画面的表示
END
```

第 3 章
机械手动作的种类



机械手的动作根据基准位置的定位方法或到达目标位置的方法分为若干类型。该章就机械手的动作种类进行说明。

第 3 章 机械手动作的种类

3.1 绝对动作与相对动作

3.1.1 绝对动作

向被教导的动作位置移动的动作称为绝对动作。
绝对动作不会受其前一动作影响，一定会向被教导的位置移动。
执行绝对动作的指令如下所示。

APPROACH、MOVE、GOHOME、DRIVEA

3.1.2 相对动作

从当前位置开始，仅移动被教导的移动量的动作称为相对动作。
相对动作是以执行前一个动作指令后的结果的当前位置为基准的，为此相对动作受前一动作指令的影响。
执行相对动作的指令如下所示。

DEPART、DRAW、DRIVE、ROTATE、ROTATEH

3.1.3 绝对动作与相对动作的动作示例

例举了从当前位置P1开始经由P2点向P3点移动的2种程序。
"MOVEMENT1" 绝对动作记述。
"MOVEMENT2" 依据绝对动作和相对动作记述。
无论执行哪一个程序均如图3-3所示进行相同动作。

```
PROGRAM MOVEMENT1
    TAKEARM
    MOVE L, P2
    MOVE L, P3
END
```

图3-1 绝对动作的程序示例

```
PROGRAM MOVEMENT2
    TAKEARM
    MOVE L, P2
    DRAW L, V3      'V3是P2与P3的相对距离
END
```

图3-2 相对动作的程序示例

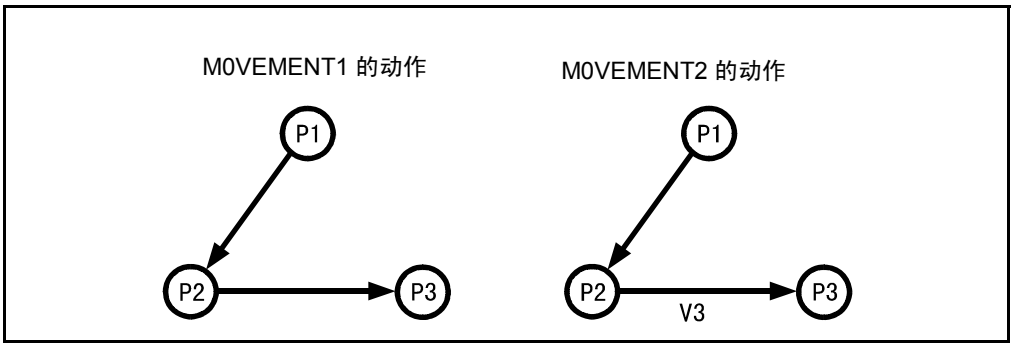


图3-3 2个程序的动作示例

在此，如果消除 "MOVEMENT1" 与 "MOVEMENT2" 最初的动作命令 "MOVE P, P2"，则动作如图3-4所示，变为不同的动作。
在 "MOVEMENT1" 中以绝对动作向点P3移动，但在 "MOVEMENT2" 中则只有V3进行相对动作。

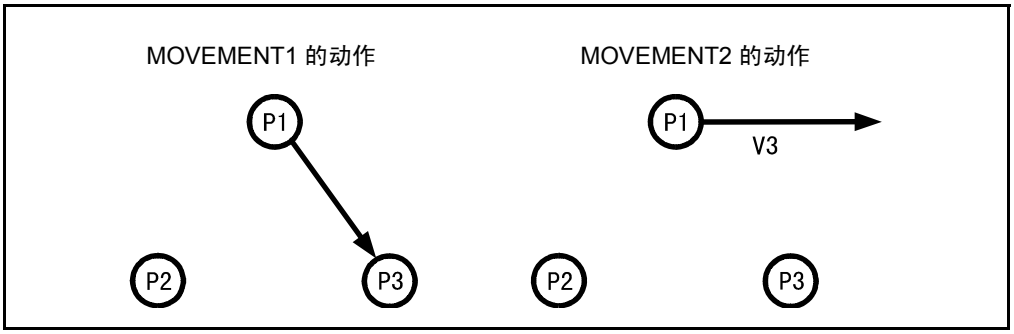


图3-4 2个程序的动作示例（消除后）

注意：相对动作执行从当前的位置移动所指定的相对距离的动作。
因此，在 INTERRUPT ON / OFF 指令（参照 12.3 项），使跳跃动作指令之后立即执行相对动作时，根据中断信号变为 ON 的时间，动作结束位置会有变化。当固定动作结束位置时，请使用绝对动作。

3.2 到达位置的确认方法

机械手手臂从一个动作移向下一个动作时，判断最初的动作是否完成的方法有3种。分别叫做通过 (Pass) 动作、结束动作、编码器数值确认动作。关于各个动作进行以下说明。

3.2.1 通过 (Pass) 动作

将通过被教导的动作位置或相对位置附近的动作称为通过 (Pass) 动作。
在所有的动作指令中，将通过 (Pass) 开始位移量指定为 "@P" 或 "@数值（数值>0）" 来执行通过 (Pass) 动作。

3.2.2 结束动作

到达被教导的动作位置或相对位置的动作中，伺服系的指令值与目标位置相一致时，判断到达目标位置的运行被称为结束动作。
在所有的动作指令中，将通过 (Pass) 开始位移量指定为 "@0" 来执行结束动作。

3.2.3 编码器确认动作

在到达被教导的动作位置或相对位置的动作中，编码器针对目标位置，进入指定脉冲（初始值20）以内时，判断已到达目标位置的运行称为编码器值确认动作。指定脉冲数可以以各轴为单位在程序中加以改变。
在所有的动作指令中，将通过 (Pass) 开始位移量指定为 "@E" 来执行编码器确认动作。

注意①：如果编码器针对目标位置，偏移指定脉冲以上时，则为没有到达，不能进入下一步骤。使指定脉冲变小或负荷变大时等，请考虑伺服系的偏差量。
②：在机器锁定运行中，"@E" 指定的动作全部以 "@0" 动作。因此，所显示的程序执行时间与实际动作时相比更短。
③：在没有记述到达位置的确认方法时，结束动作 "@0" 被自动设定到通过 (Pass) 开始位移量。

3.2.4 通过 (Pass) 动作、结束动作、编码器数值确认动作的动作示例

以3种类型示出了从当前位置P1经由P2点向P3点移动的程序的示例。分别是通过 (Pass) 动作、结束动作、编码器数值确认动作的程序示例。

```
PROGRAM PASS_MOVE
    TAKEARM
    MOVE P, @P P2
    MOVE P, @0 P3
END
```

图3-5 通过 (Pass) 动作的程序示例

```
PROGRAM END_MOVE
    TAKEARM
    MOVE L, @0 P2
    MOVE L, @0 P3
END
```

图3-6 结束动作的程序示例

```
PROGRAM ENCODER_MOVE
    TAKEARM
    MOVE L, @E P2
    MOVE L, @0 P3
END
```

图3-7 编码器数值确认动作的程序示例

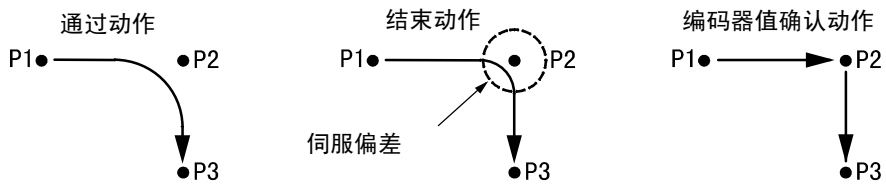


图3-8 通过 (Pass) 动作、结束动作、编码器数值确认动作的动作示例

3.2.5 通过 (Pass) 动作、结束动作、编码器数值确认动作执行时间的差异

3种类型的动作执行时间顺序为，最早的是通过 (Pass) 动作，其次是结束动作，最费时间的是编码器数值的确认动作。

通过 (Pass) 动作如图所示，在到达P2前的减速时间中，开始下一步的加速动作。因此，与各自执行减速与加速的结束动作相比，动作时间更短。

在编码器数值确认动作中，因为要严密地通过 (Pass) 编码器数值进行确认目标位置到达，因此与结束动作相比，消除伺服偏差的部分更为费时。

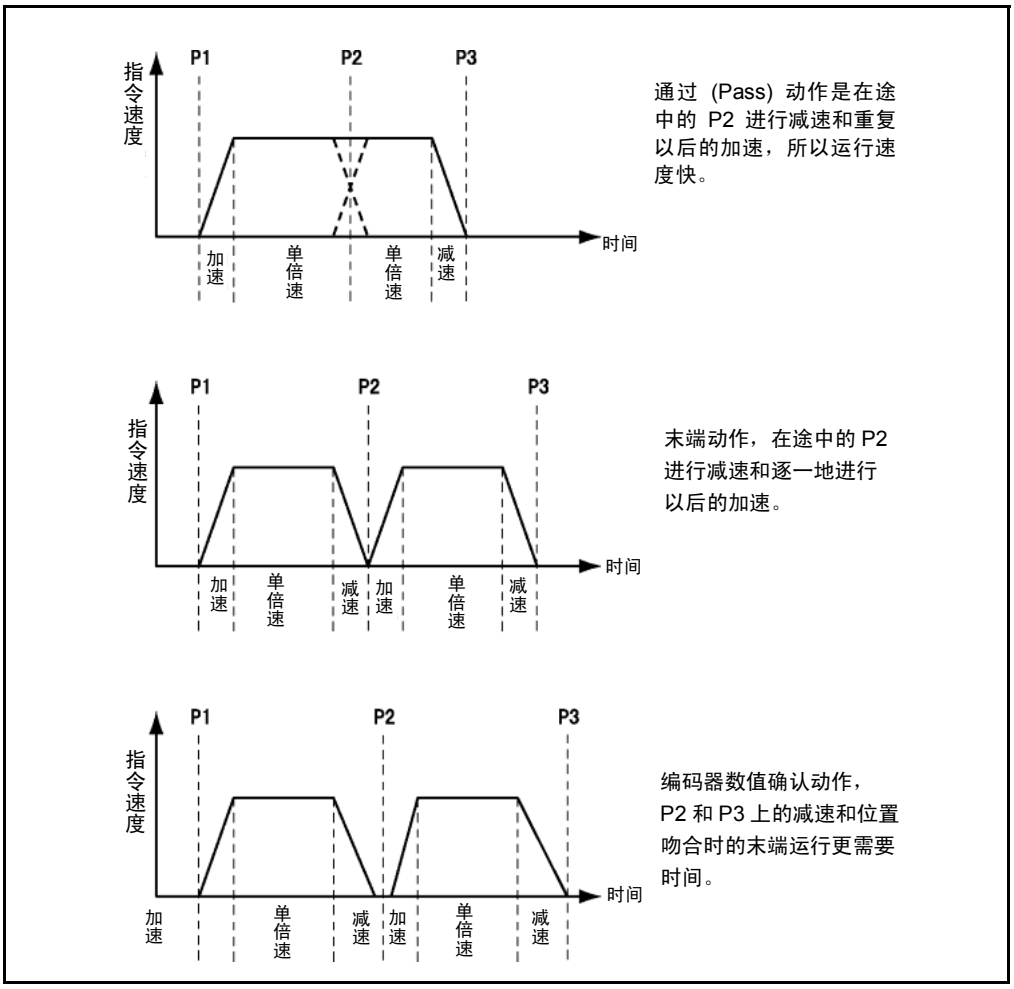


图3-9 通过 (Pass) 动作、结束动作、编码器数值确认动作的执行时间

3.2.6 不进行通过 (Pass) 动作的情况

在以下情况下，即使指定通过 (Pass) 动作，也会以结束动作方式进行运行。

3.2.6.1 在主程序的最后有通过 (Pass) 动作指令的情况

在主程序的最后，被执行的通过 (Pass) 动作指令作为结束动作指令被执行。
图3-10所示的情况下，以结束动作到达最后的P3。

```
PROGRAM PRO10
    TAKEARM
    MOVE L, @P P1
    MOVE L, @P P2
    MOVE L, @P P3
END
```

图3-10 在主程序的最后有通过 (Pass) 动作指令的程序示例

3.2.6.2 通过 (Pass) 动作指令之后有 GIVEARM 的情况

一旦执行GIVEARM（参照p.14-27），则要等待机械手的动作完全停止。因此，
一旦在通过 (Pass) 动作指令之后执行GIVEARM，则不进行通过 (Pass) 动作。

3.2.7 通过 (Pass) 动作效果减弱的情况

3.2.7.1 在通过 (Pass) 动作后有非动作指令时

如果在通过 (Pass) 动作指令与下一个动作指令之间有非动作指令，则通过 (Pass) 动作的执行时间缩短效果变弱。所谓非动作指令是指不伴随机械手本体的动作的指令。

在图3-12中，示出在通过 (Pass) 动作指令与下一个动作指令之间有非动作指令的示例。在该示例所示的情况下，如图3-13所示，因为在通过 (Pass) 动作指令的减速时间中执行非动作指令，因此通过 (Pass) 动作的执行时间缩短效果变弱。

```
PROGRAM PRO13
    MOVE L, @P P2      '通过 (Pass) 动作指令
    SET IO1            '非动作指令
    RESET I02          '非动作指令
      :
    MOVE L, @0 P3      '结束动作指令
END
```

图3-12 在通过 (Pass) 动作指令与下一个动作指令之间，有非动作指令的示例

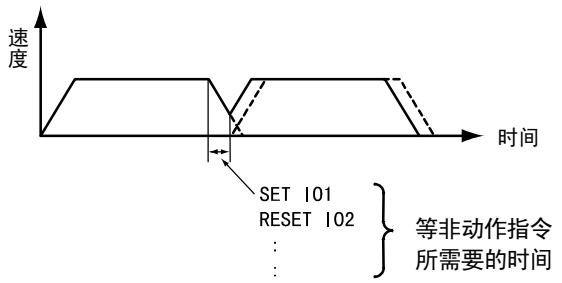


图3-13 通过 (Pass) 动作的执行时间缩短效果变弱的示例

3.2.7.2 通过 (Pass) 后的轨道短的情况

当通过 (Pass) 后的轨道短且速度模式为三角模式时，通过 (Pass) 开始位置在通过 (Pass) 后的轨道的加速结束时延迟以使通过 (Pass) 动作结束。因此，如图3-14所示当通过 (Pass) 前轨道的减速度小的时候，通过 (Pass) 效果显著变弱。

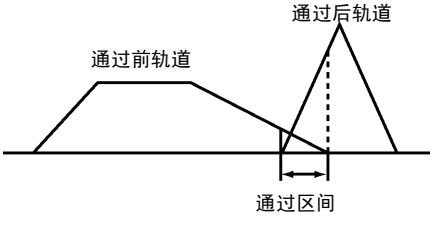


图 3-14 通过 (Pass) 后轨道短且为三角模式的情况

3.2.8 加速度影响通过 (Pass) 动作的路径的情况

DENSO系列机械手设定速度时，自动将速度平方以100分之一的加速度和减速度设定。
如果使用SPEED指令，加速度和减速度也同样被设定。
使用机械手自动设定的加速度进行通过 (Pass) 动作时，动作路径经常是一定的。
图3-15，表示将速度设定为60%和80%的示例。

⚠注意：在设定加速度时，请确认随通过 (Pass) 动作的路径变化不会产生碰撞等危险。

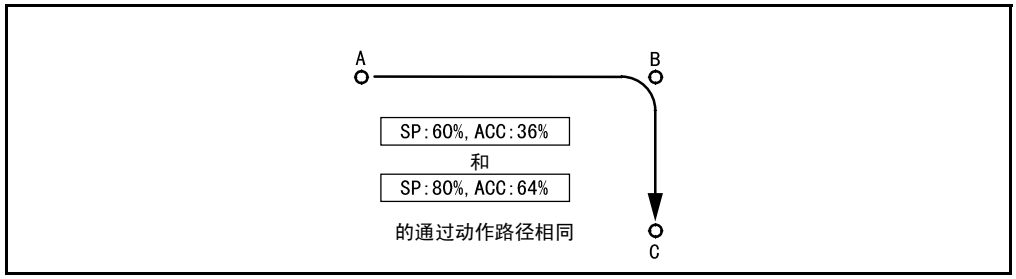


图3-15 自动设定加速度的示例

一旦任意设定加速度，则动作路径发生变化。在图3-16，对于速度60%的设定，表示加速度和减速度双方分别为100%和10%的情况。

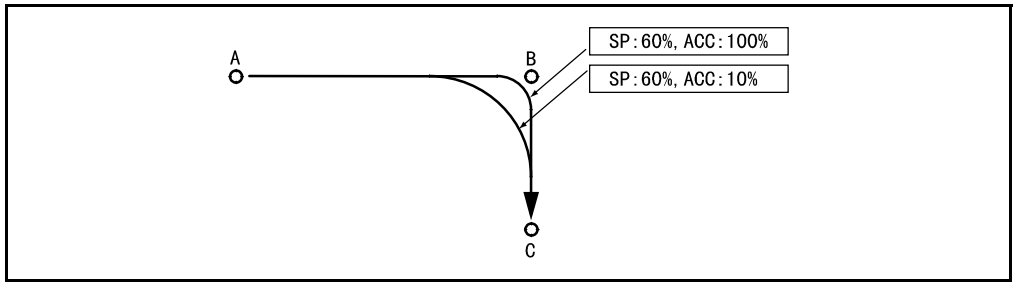


图3-16 任意设定加速度的示例

⚠注意：通过 (Pass) 动作的减速途中进行瞬时停止，若重新开始其次的动作，则向下一步骤的指令值变动。速度和加速度及减速度低时，请予注意。

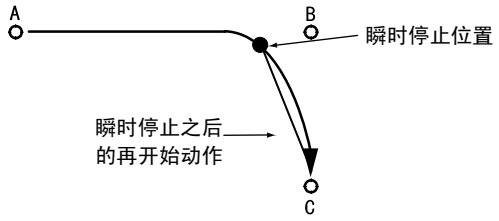


图3-17 瞬时停止后的重新开始动作

3.2.9 通过 (Pass) 开始位移

如果指定通过 (Pass) 开始位移为 "@P" 时，则在至此为止的动作的减速开始的同时开始下一个动作。因此，如图 3-18 中所示，通过 (Pass) 开始位置与通过 (Pass) 点 B 之间的距离 L 根据速度、加速度而发生变化。

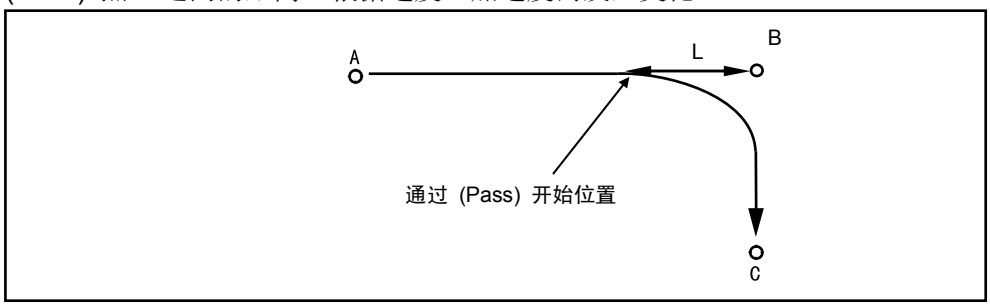


图 3-18 通过 (Pass) 开始位置

如果指定通过 (Pass) 开始位移为 "@数值" 时，则从数值 = L (mm) 的位置起开始通过 (Pass) 动作。

当 "@数值" 指定的值超过减速移动距离时，如图3-19所示，减速开始位置被修正。

因此，"@数值" 指定时，有减速度降低，移动时间变长的情况。

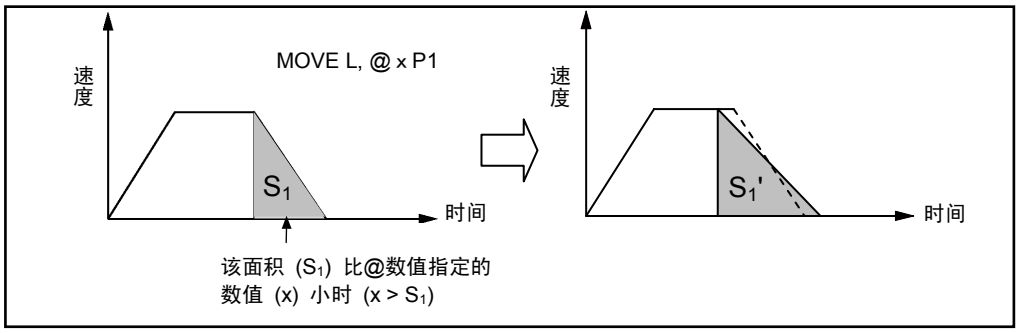


图 3-19 减速开始位置

此外，"@数值" 指定的值超过移动距离的 1 / 2 时，如图 3-20 所示减速开始位置被固定在移动距离的 1 / 2 处。

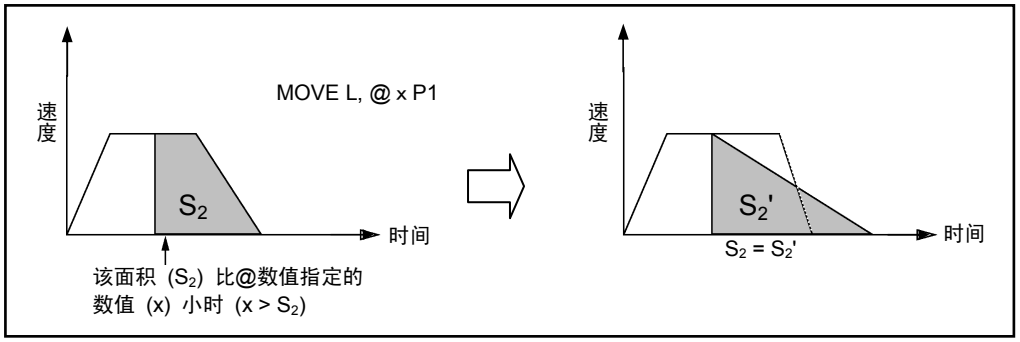


图 3-20 "@数值" 超过移动距离的 1 / 2 的情况

因此，X > S2 时，即使改变 X 通过 (Pass) 动作也不发生变化。
另外，当 X > S2 时，"请重新设定通过 (Pass) 开始位移量" 的警告讯息将被显示在多功能教导器上。

当减速开始位置变化时，以下的轨道的加速时间也被修正。
如图 3-21 所示，与@P 指定的情况相比，有动作时间变长的情况。

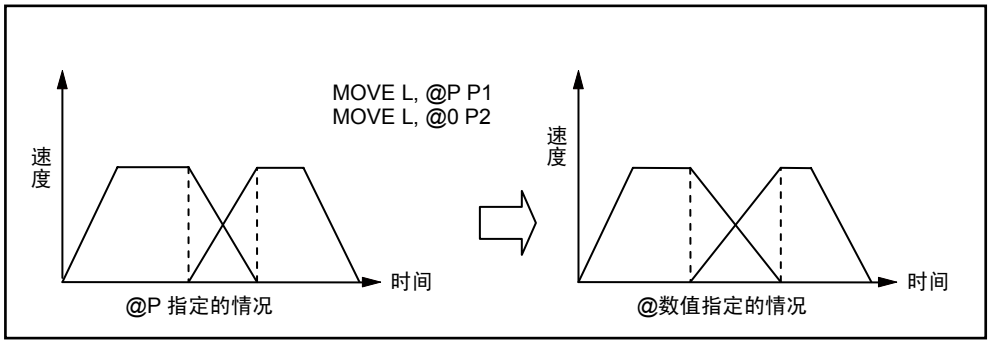


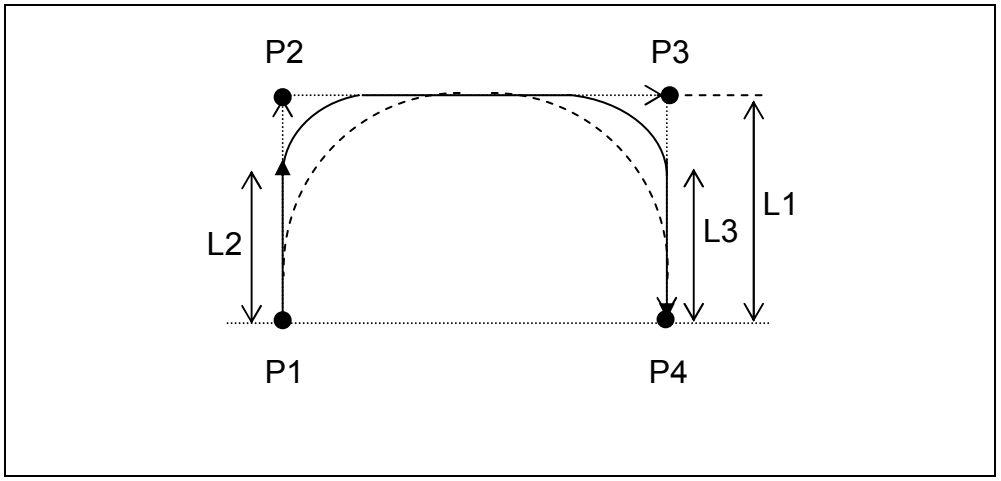
图 3-21 减速开始位置变化时的动作时间

注意： 以下情况，即使 "@数值" 进行指定，也将从比数值 = L (mm) 位置更短的位置开始进行通过 (Pass) 动作。
(1) 通过 (Pass) 后轨道短且速度模式为三角模式的情况
与 "3.2.7通过 (Pass) 动作的效果减弱的情况" 相同，通过 (Pass) 动作开始位置将变动。特别是在从低速动作变为高速动作时执行通过 (Pass) 动作，动作开始位置容易发生变动。通过 (Pass) 动作时，请在通过 (Pass) 动作前后尽可能使内部速度、加速度恒定来使用。

补充说明
在指定通过 (PASS) 开始变位的数值时，可对变量进行指定 (Ver.3.0之后的版本)。
指定I型变量。如将变量指定为-1，则进行与 "@P" 相同的通过 (PASS) 动作。
如将变量指定为-2，则进行编码器数值确认动作 "@E"。
如指定为其他负值，则显示错误。

3.2.10 弧形运转功能 "仅适用 Ver.1.9 以上版本、4 轴机械手"

如果使用弧形运行功能，可以使其进行高效率的Pick＆Place (P＆P) 动作。



如上图所示从当前位置 (P1) 向目标位置 (P4) 动作的P＆P动作，通常通过 (Pass) 指定经由的P2、P3点来实现。使其在P2、P3附近进行通过 (Pass) 动作，可以缩短动作时间。
在弧形运行中，与通过 (Pass) 动作不同的Z方向的动作中的任何位置都能够移向下一个动作，所以可以更高效率的动作。

还有，只是指定相对于P4的Z轴的动作量 (L1)、通过 (Pass) 的开始位置（距P1的距离L2）、通过 (Pass) 的结束位置（距P4的距离L3）即可，因此程序创建也比较简单。

但是，在弧形运行中不进行位置控制。动作速度的变动会引起动作路径的变化，动作时请注意与周围设备的干扰。另外，弧形运行中的动作全部成为PTP插补。

弧形运行功能所使用的共用程序库 (Library)：
ArchMove, SetArchParam

3.3 插补控制

当机械手手臂的前端移动时，其路径不止一个。各轴的运转相互结合，可以实现多种路径。还可以控制使其为直线或圆弧。下面对与动作路径的种类相应的控制方法进行说明。

以下指令指定插补方法：

APPROACH、DEPART、DRAW、MOVE

3.3.1 PTP 控制

所谓PTP (Point to Point) 是指点向点的移动。移动的路径依存于机械手的姿势，不只限于直线运动。

在图3-22中表示了基于PTP控制的动作示例。

以动作控制指令指定插补方法时，如果指定 "P" 则进行PTP动作。

在PTP动作目标位置上指定P型、T型变量时，如指定机械手形态，则会向所指定的机械手形态进行移动。不指定机械手形态的情况下，会与现在的机械手形态成为同一形态。

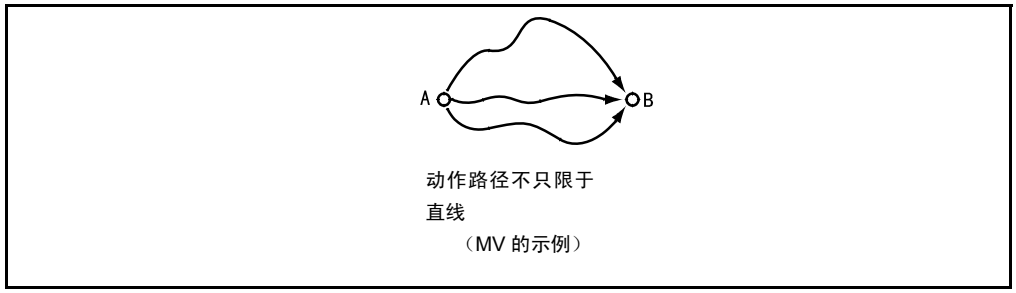


图3-22 PTP控制动作

3.3.2 CP 控制

CP控制是使到达动作目标位置的路径成为直线而进行的插补控制。
在图3-23中表示了基于CP控制的动作示例。
以动作控制指令指定插补方法时，如指定 "L" 则进行CP动作。

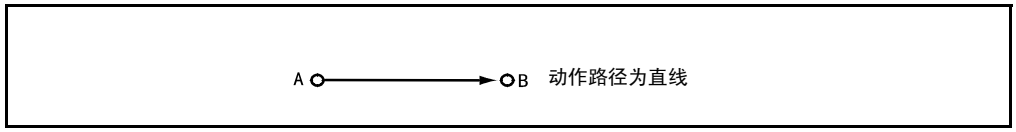


图3-23 CP控制动作

- CP控制时作为原则，不能向与当前的机械手形态不同的形态移动。如果指定不同形态时，则会发生 "607F 机械手形态不一致" 的错误。但是，可以移动时，则不发生错误。
- CP控制时的机械手形态选择与当前的姿势相近的形态。因此，即使以P型、T型变量指定机械手形态，也不限于变为所指定的形态。如果与指定的形态发生不同时，则会发生 "601C 请更改形态" 的警告。
- 以CP控制来执行程序中的最初的动作指令时，视机械手的位置有时可能无法动作。推荐对程序中的最初动作指令使用PTP控制。

3.3.3 圆弧插补控制

圆弧插补控制是将到达动作目标位置的路径如同描绘圆弧一样进行插补控制。
在图3-24中表示了基于圆弧插补控制的动作示例。
在通过 (Pass) 动作控制指令指定插补方法时，如指定 "C" 则进行圆弧插补动作。

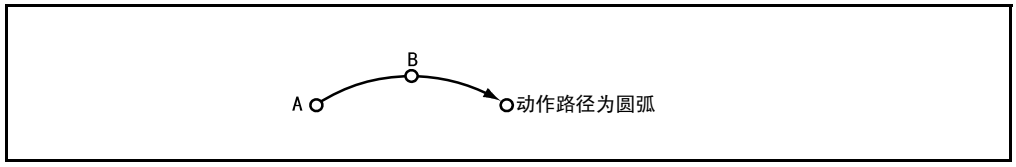


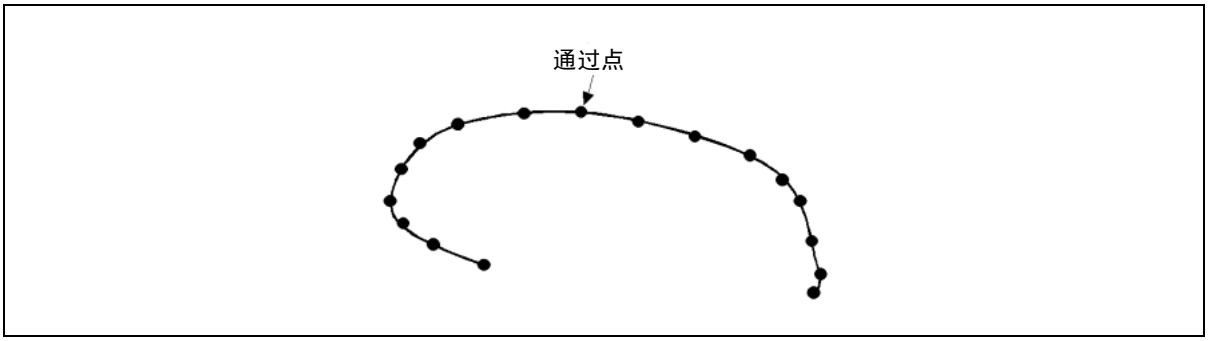
图3-24 圆弧插补控制动作

- 圆弧插补控制时，与CP控制时相同,作为原则，不能向与当前的机械手形态不同的形态移动。如果指定为不同形态时，则会发生 "607F 机械手形态不一致" 的错误。但是，可以移动时，则不发生错误。
- 圆弧插补控制时的机械手形态选择与当前的姿势相近的形态。因此，即使以P型、T型变量指定机械手形态，也不限于变为所指定的形态。如和指定形态发生不同时，则会发生 "601C 请更改形态" 的警告。
- 以圆弧插补来控制执行程序中的最初的动作指令时，因机械手的位置有时可能无法动作。推荐对程序中的最初动作指令使用PTP控制。

3.3.4 自由曲线插补［Ver2.3 以上版本］

3.3.4.1 概要

如下图所示，在通过指定的点队列（通过点）的自由曲线上以一定速度动作。动作开始时加速，动作结束时减速。
进行密封涂敷、粘贴盘根、去除毛刺等操作时，若使用自由曲线插补则可进行稳定的操作。

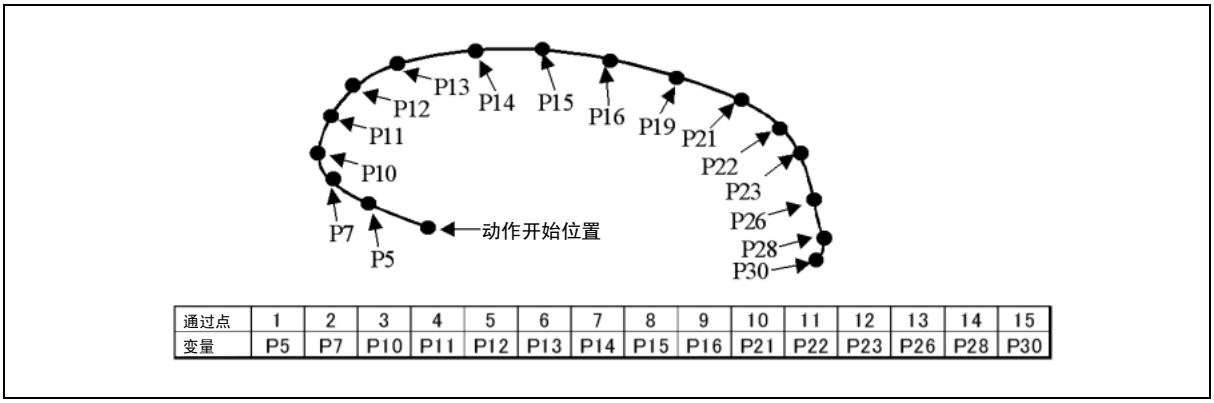


自由曲线插补

3.3.4.2 操作步骤

(1) 教导通过点

教导通过点。教导通过点为P型、J型其任意一种变量。T型变量所指教的位置数据，在 "(2) 通过点登录" 时需要从T2P指令转换为P型。
为了便于了解通过点的编号和P型、J型变量编号的对应情况，请创建下图所示的对应表。



通过点	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
变量	P5	P7	P10	P11	P12	P13	P14	P15	P16	P21	P22	P23	P26	P28	P30

通过点的对应表示例

(2) 通过点登录

指定自由曲线轨道编号，登录通过点。通过点用专用指令 (SETSPLINEPOINT) 在每一个自由曲线轨道编号上进行登录。

如例1所示根据通过的顺序编号请编写登录通过点的程序。详细情况请参照SETSPLINEPOINT指令、CLRSPLINEPOINT指令。自由曲线的轨道编号设为1。

```
PROGRAM  PASSPOINT1
CLRSPLINEPOINT 1
SETSPLINEPOINT  1, P5
SETSPLINEPOINT  1, P7
SETSPLINEPOINT  1, P10
        ⋮
SETSPLINEPOINT  1, P28
SETSPLINEPOINT  1, P30
END
```

例1 通过点登录程序

(3) 自由曲线插补的执行

指定轨道编号执行自由曲线插补。自由曲线的动作指令为MOVE S <自由曲线的轨道编号>。详细情况请参照MOVE指令。通过点请在自由曲线插补指令之前进行登录。在动作开始之前登录时，如例2所示。

```
PROGRAM  PRO1
TakeArm
    ⋮
CALL PASSPOINT1  '通过点登录
MOVE S, 1 '执行轨道编号 1 的自由曲线动作
    ⋮
END
```

例2 在自由曲线动作开始之前登录通过时

执行TakeArm指令时，用SETSPLINEPOINT所登录的通过点全部被取消。用初始化程序等其他的程序登录通过点时，为了不取消用自由曲线动作程序的TakeArm指令所登录的通过点，请执行xdSPLClrTakeArm (0)。例3是在初始化程序中登录通过点的程序示例。详细情况请参照共用程序库 (Library) xdSPLClrTakeArm。

```
PROGRAM  INITIAL
    ⋮
CALL PASSPOINT1            '对通过点进行登录。
CALL xdSPLClrTakeArm(0)  '将 TakeArm 时的通过点清零无效化
    ⋮
END
```

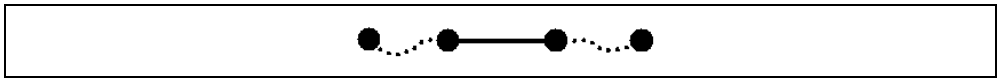
例3　用初始化程序登录通过点时

(4) 通过点的调整

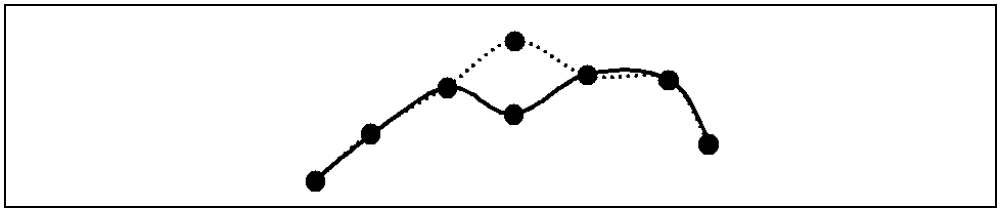
请用教导检查模式确认自由曲线的路径。自由曲线动作也可以使用单步倒退功能。请通过瞬时停止、单步倒退、单步运行确认路径。路径与理想曲线相偏离时，为了缩短通过点间隔请插入通过点。
提高速度时，轨道有向曲线内侧偏离的情况。这种情况，请将通过点向理想曲线的外侧指教、调整。

3.3.4.3 自由曲线的特征

(1) 通过点为1点时，为直线动作。
(2) 如下图所示在直线上指教通过点4点时，中间的2点之间为直线。

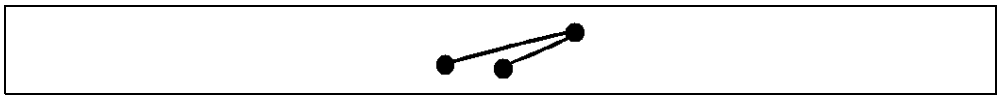


(3) 如下图所示变更了通过点时，被变更的点的前后2点之间的路径改变。



(4) 自由曲线的路径与理想曲线偏离时，请缩短通过点的间隔。

(5) 如下图所示在同一点往返，或在锐角指定弯曲的通过点时，可能会发生通过点速度降低、发生错误 "676∗指令加速度限制超程"、"60D0最佳可搬运质量执行异常"。这种情况，请降低速度进行调整。

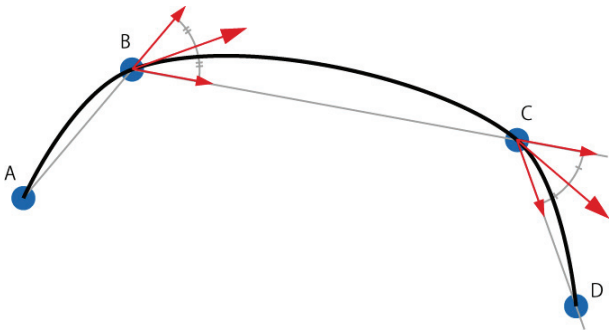


(6) 与动作同步执行I / O输出入等非动作指令时，请如下例所示使用通过点等待指令（共用程序库 (Library)）。在图2中通过P11（第4号通过点）时输出I / O，会成为以下的程序。详细情况请参照共用程序库 (Library) xdWAITSPLINE。

```
PROGRAM  PRO1
TakeArm
      ⋮
CALL PASSPOINT1   '通过点进行登录。
MOVE S, 1, NEXT   '执行轨道编号 1 的自由曲线动作。
CALL xdWAITSPLINE(4,1) '等待通过第 4 号通过点
SET IO[240]
      ⋮
END
```

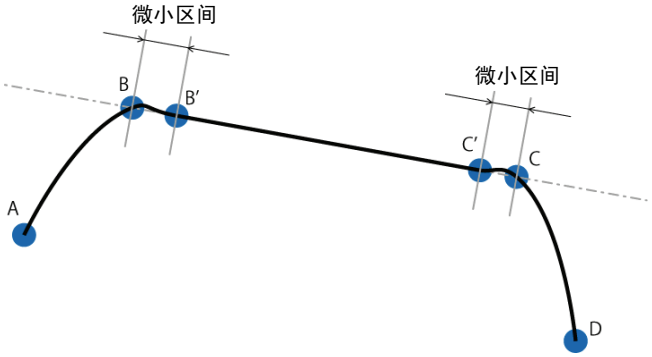
■自由曲线插补的特性（轨迹生成概略）

自由曲线插补以一条在连接区间前2点的方向与连接下一点的方向的中间方向拥有斜率的曲线开始，以一条在连接区间前2点的方向与连接后2点的方向的中间方向拥有斜率的曲线结束。



■为直线区间时

必须在直线上配置4点。
请将直线开始位置和直线区间结束位置的直线区间（右图蓝点的小区间的间隔）作为微小区间进行教导。



<要点>

将4点放在直线上，中间的2点之间（B′-C′点之间）将完全变成直线，但因B-B′之间、C′-C之间受到前后的点的影响，将偏差直线。
前后的点的影响可以通过缩短区间距离来降低，因此，如果变成微小区间，就可以减少偏差。

3.3.4.4 使用自由曲线插补时的注意事项

(1) 每个轨道编号，通过点最大可以登录到200点。登录超过200点时，会发生"685A自由曲线登录数据数量超过" 的错误。

(2) 轨道编号，最大可以指定到20。指定超过20时，会发生 "685B自由曲线编号指定异常" 的错误。

(3) 自由曲线动作后变更了通过点时，不能进行通过点变更前的自由曲线的步骤返回动作。发生 "737D 到此不能返回" 的错误。

(4) 通过点之间的移动距离缩短姿势变化大时，会发生 "608* 指令速度限制超程"、"60D0 执行最佳可搬运质量异常" 的错误。这种情况，请降低速度。

(5) 在传送跟踪中，不能执行自由曲线动作。

(6) 使用自由曲线时，不能使用复电功能。

(7) 当希望利用通过（PASS）动作反复在一个自由曲线上进行动作时，请将此自由曲线登录到2个轨道编号上，并根据动作切换轨道编号。如让轨道编号相同的自由曲线进行通过（PASS）动作，有可能会产生错误。

3.3.5 高轨迹控制功能 [Ver.2.61 以后]

3.3.5.1 概要

高轨迹控制功能是指能够提高机械手动作轨迹精度的功能。尤其是提高高速时的圆弧动作和自由曲线插补动作的动作轨迹精度。进行密封涂敷等作业时，使用高轨迹控制功能，能够进行稳定的作业。

3.3.5.2 高轨迹控制功能的使用方法

通过使用程序库功能并执行高轨迹动作程序库（SetHighPathAccuracy、ResetHighPathAccuracy），将功能设定为有效或无效。程序库的详情，请参照PAC程序库项目。

```
PROGRAM PRO1                    程序说明
TAKEARM                           '获得臂组控制权。
MOVE   P, @E P1                   '通过PTP动作向P1移动。
CALL   SetHighPathAccuracy        '使高轨迹控制功能有效。
MOVE S, 5                         '通过自由曲线动作，使轨迹编号5进行
                                  高轨迹移动。
CALL   ResetHighPathAccuracy      '使高轨迹控制功能无效。
MOVE   P, @0 P5                   '通过PTP动作，从P4向P5移动。
END                               '程序结束。
```

'高轨迹动作的程序示例

3.3.5.3 使用高轨迹控制功能时的注意事项

(1) 请只在想要提高动作轨迹的动作时，才使高轨迹控制功能有效。
(2) 因动作条件、负荷条件，可能会一边振动一边动作。
届时，请降低速度或加速度，或者使功能无效。
(3) 关闭控制器电源或电机，高轨迹控制功能即无效。
(4) 不适用附加轴选件的轴（7、8轴）。
(5) 力限制功能有效时，不能使高轨迹控制功能有效。否则会发生错误。
(6) 在动作中（通过动作中等）不能使高轨迹控制功能有效。
(7) 请正确进行负荷设定（前端负荷质量、前端负荷重心位置等）。未正确设定时，会影响精度。

3.3.6 特异点回避功能 [Ver.2.61 以后，6 轴专用]

3.3.6.1 功能的概要

特异点是指在底座坐标表示的位置中存在无穷多个机械手采取姿势的点。臂、肘、手腕的形态在各自的边界存在特异点。就手腕的形态而言，在5轴为0度时，4轴和6轴将平行，姿势将不确定为一个值。（参照操作指南 4.坐标系与形态）
在CP动作过程中通过特异点或其附近时，为了保持形态，特定的轴将试图瞬时地快速动作。因此会引起超速或软件限位超程错误。
回避这种由特异点带来的错误的功能叫做 "特异点回避功能"。但是，本功能只回避出现最频繁的手腕的特异点（5轴为0度时）。如果是 "Ver.3.2之前的版本"，曾发生过在通过（PASS）动作中无法回避特异点的情况。如果是 "Ver.3.2" 之后的版本，在通过（PASS）动作中也可以成功回避特异点。由于本功能使原来的保持着动作开始时的形态进行的动作，变成在动作过程中变更手腕形态进行的动作，因此，其间的机械手轨道将偏差原来的轨道动作。机械手的姿势也可能会因动作条件而大为改变。

3.3.6.2 功能的使用方法

使特异点回避功能有效，有下列2种方法。
此外，如果是 "Ver.3.2之后的版本"，可通过使 "特异点回避扩展设定" 生效，来成功回避通过（PASS）动作中的特异点。但是，要想使此功能生效，需要事先使特异点回避功能有效设定生效。这个功能仅可利用多功能教导器进行设定。

通过程序指令来设定

通过在程序内记述 "SetSingularAvoid"，设定特异点回避功能的有效/无效（参照编程手册Ⅱ SetSingularAvoid）。通过关闭控制器电源或关闭电机进行重置，使其变为 "无效"。

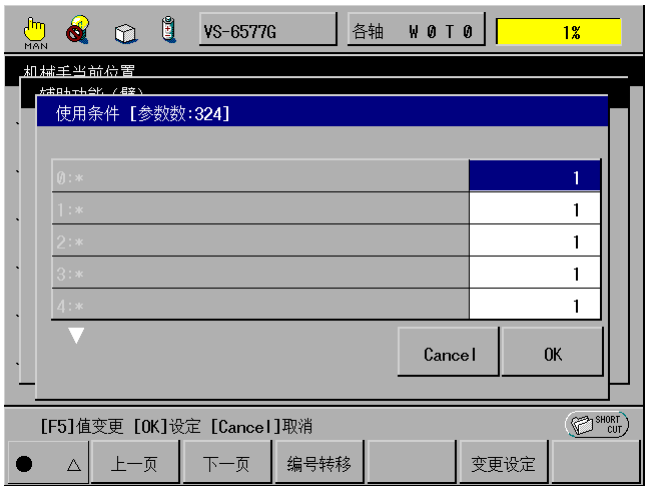
利用多功能教导器来设定

可以对所有CP动作，使特异点回避功能以及特异点回避扩展设定（Ver.3.2之后版本）生效/无效。在关闭控制器电源之后，也能保持设定。

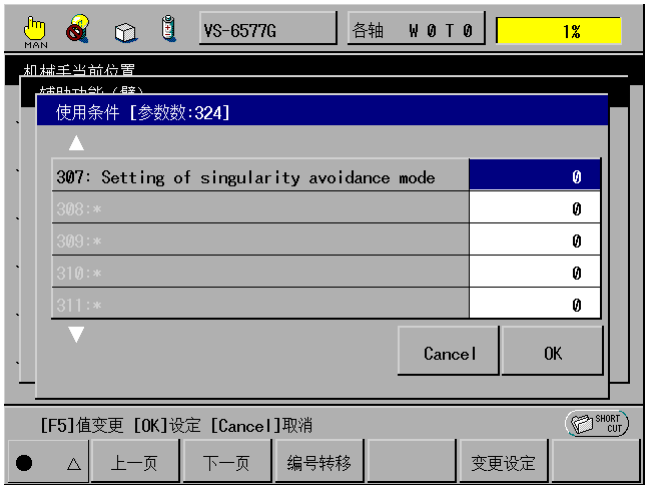
3.3.6.3 利用多功能教导器的设定方法

操作路径：基本画面－［F2 臂］－［F6 辅助功能］－［F7 使用条件］

(1) 按上述路径操作多功能教导器，即显示［使用条件］画面



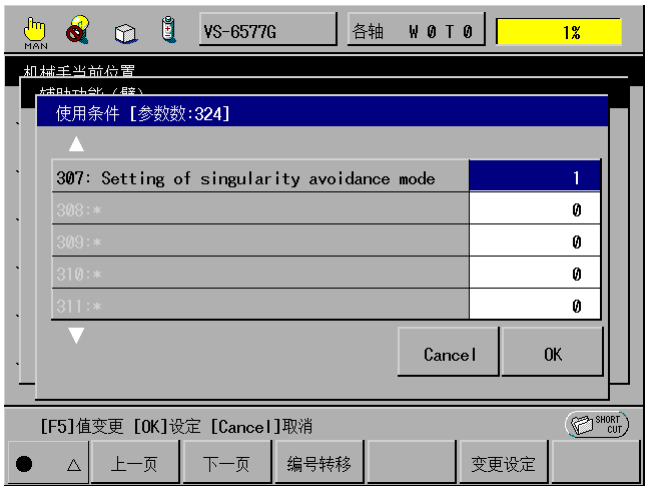
(2) 用［F3编号跳跃］跳到307页，即显示［特异点回避功能有效设定］。



(3) 按［F5 设定变更］，在十键画面上变更设定，然后按［OK］。
有效：1　　无效：0



(4) 确认使用条件的参数已经改变，然后按［OK］结束。



⚠注意： 通过多功能教导器使功能有效时，所有CP动作就将成为特异点回避的对象。本功能可能会因动作条件而偏差原来的轨道，机械手的姿势会大为改变。请采用通过程序库来设定，确认各个动作与周围设备不会发生干扰之后再使用。

3.3.6.4 特异点附近警告

特异点回避功能在实际动作中起作用时，如果在下列操作过程中通过特异点附近，会出现警告声音和警告讯息。

- 手动模式、教导检查模式下的变量移动（仅CP动作）
- 教导检查模式下的程序动作（仅CP动作）

自动模式下不进行警告。

3.3.6.5 使用特异点回避功能时的注意事项

(1) 本功能不能用于手动模式下的各轴动作、XY动作及工具动作。虽然可以用于程序中的动作指令及变量移动，但是在PTP动作中无效。

(2) 本功能在通过手腕形态的特异点（5轴为0度）附近时进行回避动作。在回避动作中，可能会出现与原来应该动作的轨道不同、机械手的姿势大为改变的情况。因为特异点的性质，不能在手指位置和姿势不偏差的情况下进行控制，所以，不允许偏差轨道的动作不能使用本功能。此外，使用本功能时，请注意因偏差轨道而与周围设备的干扰。

(3) 本功能不适用臂、肘的特异点。只对手腕形态的特异点（5轴为0度）有效。

(4) 对于4轴和5轴机械手，本功能无效。

(5) CP插补时不能向5轴为0度的位置移动。CP插补时向5轴为0度的位置附近的移动，只在通过特异点时做回避动作。

(6) 在动作过程中，可能会因动作条件（工具长度、动作速度、动作方向及动作位置和姿势）而发生错误，不能动作。此时，通过降低速度或改变动作位置和姿势，有时也可以动作。

3.4 在动作指令之后有输出指令时

在执行动作指令时，针对机械手控制器指令的位置，在机械手的当前位置有延迟。结束动作时，机械手控制器一旦当指令位置到达动作指令的目标位置后，则转移至下一指令的执行。为此，继动作指令后有输出指令时，机械手在到达目标位置之前，有时输出指令会被执行。在到达目标位置之后，为了执行下一指令，有如下所示的方法。

●设置基于DELAY指令的等待时间。（图3-25中的程序示例）
●执行编码器数值确认动作而不是结束动作。（图3-26中的程序示例）

注意：在编码器数值确认动作中，重负荷的情况下，有时编码器数值与目标位置无法完全一致。

```
PROGRAM PRO13
    :
    :
  MOVE L, @0 P3     '结束动作指令
  DELAY 500         '等待0.5秒
  SET IO1           '将I / O1置于ON
  RESET IO2         '将I / O2置于OFF
    :
    :
END
```

图3-25 设置基于DELAY指令的等待时间的程序示例

```
PROGRAM PRO13
    :
    :
  MOVE L, @E P3     '编码器数值确认动作指令
  SET IO1           '将I / O1置于ON
  RESET IO2         '将I / O2置于OFF
    :
    :
END
```

图3-26 使用编码器数值确认动作的程序示例

3.5 力限制功能

3.5.1 概要

所谓力限制功能，是利用软件在机械手上进行柔性设定的功能。
该功能可以吸收机械夹治具在夹持时的位置偏离、防止在机械手和工件上施加过大的力。本功能，具有在每个关节进行柔度设定的功能（电流限制功能）和在每个机械手前端的坐标系各个要素进行柔度设定的功能（前端力限制功能）。

> 注意：本功能以使用 PAC 共用程序库 (Library) 的实例进行了说明，而在 Ver. 1.9 以上版本也可以使用 "12.10 特殊控制" 指令。

3.5.2 各个轴的电流限制功能［V1.2 以上版本］

以机械手的各轴为单位设定柔度的功能。以限制各轴电机的转矩（电流值）实现柔度。主要是防止对机械手或工件施加过大的力，或防止由于过负荷或过电流错误导致的停止使用。

3.5.2.1 电流限制功能使用方法

使用共用程序库 (Library) 功能，以执行电流限制共用程序库 (Library) (SetCurLmt, ResetCurLmt) 将功能设定为有效或无效。但是，请进行以下设定。
关于共用程序库 (Library) 的详细内容，请参照PAC共用程序库 (Library) 项。

(1) 前端负荷设定（V*-D系列必须）

请正确设定前端部负荷质量和负荷重心位置。请输入使用条件中的前端部负荷质量 "g"、负荷重心位置X6 "mm"、负荷重心位置Y6 "mm"、负荷重心位置Z6 "mm" 中使用的机械夹治具或工件所对应的值。但是，在工件卡盘、松开卡盘等中质量有变化时，请使用aspACLD改变设定。
详细内容请参照 "4.7 '使用条件' 中最佳可搬运重量设定功能"。

注意： 当没有正确设定时，有时在电流限制设定时机械手可能向重力方向掉落。

(2) 重力补偿设定（V*-D系列必须）

力限制时，必须将重力补偿置于有效。在无效的情况下，会发生错误 "665a 不能进行电流限制设定"。为了将重力补偿有效化，要用教导器将使用条件的重力补偿有效无效设定设定为1。设定的方法如下所示。

①在教导器的初始画面上，请按压［F2 臂］→［F6 辅助功能］→［F7 使用条件］。使用条件参数设定画面被显示。

②请按压［F3 编号跳跃］、指定24号。显示重力补偿有效无效设定行。

③请在重力补偿有效无效设定上输入1，按压OK。

将重力补偿有效无效设定设定为1时，在校准完成的状态下，重力补偿为常时有效。

另外，用SetGravity、ResetGravity可以将重力补偿设定为有效或者无效，但在进行有效和无效切换时，机械手有时会有微小的动作。用教导器将使用条件的重力补偿的有效无效设定设定为1，建议在初始化程序中将SetGravity进行一次CALL。

(3) 重力补偿设定（仅限在V*-D系列必要时）

由于前端部负荷质量设定值与实际的负荷质量的偏差，有时会产生由于重力补偿值导致的重力平衡偏差。此时，将承受重力的轴的电流限制设定值设定得偏小时，有时机械手可能向重力方向掉落。此时，为了防止机械手的掉落，将机械手停止状态的力矩与重力补偿力矩进行比较，对误差部分进行修正的功能是重力补偿修正功能。当将承受重力的轴的电流限制值设定在30%以下时，请务必实施重力补偿修正。

在机械手停止状态下，如果执行SetGrvOffset，则计算修正值。

但是，重力补偿修正值是**执行了SetGrvOffset的姿势的修正值。自修正姿势有较大姿势变化时，请再次执行SetGrvOffset**。

此外，**如果在接触机械手等承受外力的状态下执行SetGrvOffset，则修正值中会有误差。请在不受外力（重力除外）的状态下，执行SetGrvOffset**。

(4) 偏差允许值设定（需要时）

在电流限制中如果受到外力，则关节将沿力的方向旋转。由此，造成关节角度增加，有时会发生错误 "612* J*偏差过大"。在此情况下，请用SetEralw设定偏差允许值。**偏差检查功能是安全功能。请不要超出需要扩大范围**。

(5) 共用程序库 (Library) 执行步骤

■ 电流限制有效设定时

①用教导器将使用条件中的重力补偿有效无效设定为1。（仅限V*-D系列）

② 利用初始化程序进行以下追加 （仅限V*-D系列）

```
CALL SetGravity
```

③ 电流限制有效化 ( ) 内为需要时

等待动作结束。

```
(CALL SetGrvOffset )        '计算重力补偿修正值。
                                （仅限V*-D系列）
 CALL SetCurLmt(n1,m1)    '将n1轴的电流值设定为额定的m1 (%)。
( CALL SetEralw(n2,m2) )   '将n2轴的偏差允许值设定为m2（度）。
```

■ 电流限制无效设定时

```
CALL ResetCurLmt(0)       '将所有轴的电流限制复位，偏差允许值初始化
( CALL ResetGrvOffset )   '将重力补偿修正值初始化
```

3.5.3 前端力限制功能［V1.4 以上版本］

以机械手前端坐标系的各元件为单位设定柔度的功能（适应性功能）。从前端的力限制值，控制各个轴的电机转矩，以此实现前端的柔度。坐标系可以选择底座坐标、工具坐标、工件坐标。机械手通过外力方向动作或通过接触进行高度检测时使用该功能。

3.5.3.1 前端力限制功能使用可能设定方法

本功能为扩展功能。由此，**需要用教导器设定为可以使用力限制功能**。扩展功能一旦设定为可以使用，因为在控制器电源置于OFF后仍将保持可以使用的状态，所以不需要再次设定。可以使用力限制功能的设定方法如下所示。

(1) 显示教导器的扩展功能的密码编号输入画面。

操作路径：基本画面 -［F6 设定］-［F7 选项］-［F8 扩展功能］-［F5 追加功能］



(2) 输入密码编号 "6519"，并按下OK。对显示的系统讯息进行确认，按下OK后，将显示被追加的功能（依从功能）。



3.5.3.2 前端力限制功能使用方法

使用共用程序库 (Library) 功能，通过 (Pass) 执行力限制共用程序库 (Library) (SetCompControl, ResetCompControl) 将功能设定为有效、无效。但是，请进行以下设定。关于共用程序库 (Library) 的详细内容，请参照PAC共用程序库 (Library) 项。

(1) 前端负荷设定（必须）

3.5.2.1 请参照电流限制功能使用方法

(2) 重力补偿设定（必须）

3.5.2.1 请参照电流限制功能使用方法

(3) 力限制参数的设定（必须）

力限制参数有以下 5 种。

① 力限制坐标系选择参数

从底座坐标、工具坐标、工件坐标中选择力限制坐标系。用力限制共用程序库 (Library) (SetFrcCood) 进行设定。

② 力限制比例

对用①所设定的坐标系的各个成分（X方向、Y方向、Z方向、X旋转、Y旋转、Z旋转）的力限制比例进行设定。用力限制共用程序库 (Library) (SetFrcLimit)进行设定。进行SetFrcLimit设定时，③柔度比例和④粘性比例也均为同一个比例。

③ 柔度比例（需要时）

对用①所设定的坐标系的各个成分（X方向、Y方向、Z方向、X旋转、Y旋转、Z旋转）的柔度进行设定。用力限制共用程序库 (Library) (SetCompRate) 进行设定。进行SetCompRate设定时，④粘性比例也为同一个比例。

④ 粘性比例（需要时）

对用①所设定的坐标系的各个成分（X方向、Y方向、Z方向、X旋转、Y旋转、Z旋转）的粘性比例进行设定。使用力限制共用程序库 (Library) (SetDampRate)进行设定。前端力限制特殊功能的速度控制模式之外，几乎没有使用的必要。但是，在设定时，其值请不要小于用SetFrcLimit,SetCompRate设定的柔度比例值。

⑤ 力限制时，位置、姿势偏差允许值（需要时）

对用①所设定的坐标系的各个成分（X方向、Y方向、Z方向、X旋转、Y旋转、Z旋转）的位置、姿势偏差允许值进行设定。使用力限制共用程序库 (Library) (SetCompEralw) 进行设定。初始值的位置偏差是100 (mm) 姿势偏差是30（度）。**偏差检查功能是安全功能。请不要超出需要扩大范围。**

(4) 重力补偿补正（需要时）

在SetCompControl时，对机械手停止状态的转矩和重力补偿转矩进行比较，修正误差部分。但是，**如果在接触机械手等承受外力的状态下执行SetCompControl，则修正值中会有误差。请在不受外力（重力除外）的状态下执行SetCompControl。在受到外力的状态下力限制有效的情况下，使用SetGrvOffset和SetCompFControl，按以下要求执行共用程序库 (Library)。**

(SetCompControl = SetGrvOffset + SetCompFControl)
机械手停止状态下，与外部没有接触（干扰）的状态下，
CALL SetGrvOffset
将机械手移向外部接触（干扰）状态
CALL CompFControl

(1) 共用程序库 (Library) 执行步骤
■力限制有效设定时
① 用教导器将使用条件中的重力补偿有效无效设定为1。
② 用初始化程序进行以下追加
CALL SetGravity
③ 力限制参数设定和有效化 ( ) 内为需要时
CALL SetFrcCoord(n) '将力限制坐标系设定为n。
CALL SetFrcLimit(fx,fy,fz,mx,my,mz)'力限制 (fx、fy、fz、mx、my、mz) 设定 [%]。
(CALL SetCompRate (cx,cy,cz,crx,cry,crz))
'将柔度设定值设定为 (cx、cy、cz、crx、cry、crz) [%]。
(CALL SetDumpRate(dx,dy,dz,drx,dry,drz))
'将粘性设定值设定为 (dx、dy、dz、drx、dry、drz) [%]。
CALL SetCompControl '将力限制置于有效。
(CALL SetCompEralw (ex,ey,rz,erx,ery,erz)) '将位置偏差设定值设为 (ex,ey,ez) (mm)、将姿势偏差设定值设为(erx,ery,erz) （度)。
(CALL SetEralw(n2,m2)) '将n1轴的偏差允许值设定为m1（度)。

■限制无效化设定时 ( ) 内为需要时
CALL ResetCompControl '力限制无效化设定
CALL ResetCompEralw '将位置偏差设定值初始化
(CALL ResetEralw(0)) '将角度偏差设定值初始化

3.5.3.3 前端力限制功能特殊共用程序库 (Library) 使用方法

以使用力限制特殊共用程序库 (Library)，可以更加有效地使用力限制功能。但是，**该功能因为直接更改机械手的控制数据，有时视设定值的情况，机械手可能动作异常。请将偏差容许值设定得小些并进行调整**。利用力限制特殊共用程序库 (Library) 可以设定以下的参数。

(1) 力限制时的偏移力
力限制时，由于摩擦等即使施加了力，机械手也不模仿动作时，**通过在模仿方向上施加一定的补偿力可以使得易于模仿**。利用力限制特殊功能共用程序库 (Library) (SetFrcAssis)，设定各方向的补偿量。电源投入时的初始值为0。

(2) 力限制时的电流限制设定值

在力限制时，将与机械手前端的位置偏差对应的转矩施加给电机，由此限制前端力。因此，不进行与以前各轴单位的角度偏差相应的控制（电流限制值 = 0），有时机械手动作会发生振动。此时，**取代调整力限制比例，通过调整力限制时的电流限制设定值，可以实现更加平稳的限制动作**。当不是力限制特定方向的前端力限制，而是用电流限制设定值进行调整时，将特定方向的力限制设定值置于0，在限制方向动作时调整旋转轴的电流限制值。利用力限制特殊功能共用程序库 (Library)，(SetCompJLimit)，设定各轴的电流限制值。电源起动时的初始值为0。

(3) 力限制方法选择参数

通过在力限制时实施与速度偏差而不是前端的位置偏差相应的控制，**提高碰撞动作时的接触动作的稳定性**。请利用力限制特殊共用程序库 (Library) (SetCompVMode) 设定速度控制模式，以SetFrcLimit与SetDampRate设定控制方向。

（例）Z方向的速度控制模式的设定

```
CALL SetFrcCoord(0)
CALL SetCompVMode
CALL SetFrcLimit(100,100,0,100,100,100)
CALL SetDampRate(100,100,100,100,100,100)
CALL SetCompControl
```

3.5.3.4 与安全相关的注意事项

(1) 异常检测功能

利用力限制功能对力进行限制时，机械手有时会受外力作用沿力限制方向动作。还有在施加修正方向以外的外力时，有时也会根据限制方向的分力而动作。

因此，要考虑机械手调整时的安全，追加以下的异常检测功能。

① 力限制时位置偏差过大异常 (60F8)

机械手工具端的位置偏差超过允许值的情况下发生。偏差允许值的初始值，X、Y、Z方向为100 (mm) X、Y、Z周围为30（度）。可以用SetCompEralw变更偏差允许值。但是在扩大偏差允许值时，由于不能检测异常，所以不要在需要异常上扩大允许值。

② 力限制指令值异常 (60FB)

机械手工具端检测到异常的力时发生。在对力限制比例进行高设定的方向，如果让机械手进行对接动作，则会发生错误。发生错误时，请确认力限制方向和力限制参数是否设定正确。

③ 超过了力限制时的指令速度限制值 (60FC)

在力限制过程中，不能进行高速动作。外部速度和内部速度的积超过50%的状态下，如果执行动作命令，则会发生该错误。发生错误时，请下调内部速度的设定值。

④ 力限制时，不能进行PTP动作 (60FD)

在限制过程中，禁止进行PTP动作（参照3.3.1 PTP控制）。在力限制过程中执行PTO动作的情况下，发生该错误。在CP动作上进行变更或将力限制进行无效化之后，请执行动作命令。在力限制过程中启动连续进行时，以及用重新开始模式进行重新启动的情况下，也会发生该错误。

⑤ 力限制过程中的当前位置，J*软件限位超程 (66D*)

在力限制过程中，有时会由于外力作用而造成机械手动作，向超出软件限位的位置移动。在此种情况下发生该错误。在手动模式下发生软件限位超程时，电机电源没有OFF，而在力限制过程中发生该错误时，电机电源OFF。

(2) 为了安全地进行调整

由于错误设定力限制功能的参数等原因,有时会发生机械手沿重力方向掉落，受外力的作用，沿与设想方向不同的方向动作。

为了确保调整时的安全性，请不要将角度偏差允许值、位置偏差允许值扩大出必要范围。还有，力限制方向以外的偏差允许值，用SetCompEralw进行设定，设定为比初始值小的值（X、Y、Z方向10 (mm)，X、Y、Z周围5（度）左右），并实施调整。

(3) 力限制功能的参数调整

力限制时的限制力，请用力限制比例的值进行调整。但是，力限制值和力限制比例的关系，会根据机械手的姿势、方向而变化。要在每一个力限制姿势、方向进行调整。

机械手会根据各个关节的静止摩擦力，在外力引起的关节转矩增大时，就会沿力的方向动作。静止摩擦力大于因外力所引起的关节转矩时，机械手就会在力平衡的状态下停止。

在力限制过程中让机械手动作时，由于姿势变化而造成摩擦力变化，摩擦力和限制力的平衡状态变化，机械手有时会在力限制方向（指令值方向）动作。在力限制过程中让机械手动作时，请注意机械手在力限制方向上的动作。

(4) 位置教导

机械手的位置，利用变量的位置输入功能进行教导。变量位置输入时的位置就是机械手当前的指令位置。由于力限制功能，直接按压机械手进行运行，在进行位置输入（直接教导）的情况下，因为机械手的指令位置和实际位置不一致，教导位置和机械手的实际位置发生偏离。使用力限制进行位置教导时，请按以下要领进行操作。

① 决定教导编号用的I型变量。假定使用I10。

② 编写以下的程序。

在P型变量上进行教导时

```
Program DirectTeach
    P(I10) = CURPOS
end
```

在J型变量上进行教导时

```
Program DirectTeach
    J(I10) = CURJNT
end
```

③ 将教导位置编号输入为I10，教导位置移动之后，执行DirectTeach。

(5) 工具坐标、工件坐标设定

用SetFrcCoord将力限制坐标设定为工具坐标，执行力限制时，在力限制过程中即使用Changetool**变更工具坐标，力限制方向也不会变化。但是，与机械手动作有关的工具坐标发生变化**。由此，变为力限制坐标系和动作坐标系不一致的状态。设定工件坐标的情况也是一样的。

将工具坐标的X、Y、Z值加大设定，执行力限制时，对于机械手的姿势变化的控制感度提高，有时机械手会发生振动错误而停止。此时，用SetFrcLimit、SetCompRate将设定为100的参数从100缓慢下调进行调整。

(6) 连续功能

在力限制过程中，不能连续进行动作。在力限制过程中电机OFF，如果在等待连续进行启动的状态下电机ON，则会发生错误 "66EF力限制已经被解除"。还有，在力限制过程中瞬时停止后，如果连续进行启动，则会发生错误 "66FD力限制过程中不能进行PTP动作"。

(7) 复电功能

在力限制过程中，控制器电源断开时，复电功能无效。

(8) 力限制过程中的机械手的动作精度

在力限制过程中执行动作命令时，机械手的动作轨迹精度降低。特别是有时会发生力限制方向产生大的位置偏离。精度所需要的动作，请将力限制功能置于无效。

力限制过程中，机械手的停止精度恶化。在力限制过程中，执行编码器数值确认动作（参照3.2.3编码器确认动作）时，有时会发生长时间停止或是错误"6651检查命令超时"。停止精度所需要的动作，请将力限制功能置于无效。

3.5.4　碰撞检测功能

功能的概要

此功能无需使用特殊的传感器，即可检测机械手和手指的工具与周围设备的碰撞，使机械手紧急停止，以降低碰撞造成的损害。手动操作、自动运转均可使用。

> ⚠注意：本功能不能完全回避碰撞的危害。并且也并不保证人身安全。

对应版本

各机械手在以下各控制器版本下能够发挥功能。

当使用低于以下控制器版本的项目数据时，请参照 "关于旧版项目数据"。

VS-G系列：Ver.2.61之后版本

VS-***系列：Ver.3.20之后版本

VM-G、VP-G系列：Ver.2.70之后版本

HM-G、HS-G系列：Ver.2.80之后版本

注意： 碰撞检测功能在旧机件板上无法发挥功能。

3.5.4.1　功能的使用方法

有效或无效设定

可以利用程序库或多功能教导器，按每根轴进行碰撞检测功能的有效或无效设定。但是，不能使附加轴有效。利用多功能教导器进行的设定，在关闭控制器电源之后也能保持。利用程序库进行的设定，在关闭控制器电源之后即返回原来的设定。初始值为“无效”。

[1]利用程序库设定

可以通过在PAC程序上记述程序库SetCollisionJnt、ResetCollisionJnt，按程序单位、动作单位来设定功能的有效或无效。详情请参照编程手册Ⅱ PAC程序库。

[2]利用多功能教导器设定

可以通过下列多功能教导器操作，设定功能的有效或无效。在机械手操作过程中不能设定。

操作路径：基本画面－［F2 臂］－［F6 辅助功能］－［F2 碰撞检测］

(1) 如果在多功能教导器的画面上触摸想要进行碰撞检测的轴的有效栏，功能即变为有效，显示色从黑色变为绿色。如果再次触摸，即变为无效，显示色返回到黑色。



(2) 在以上画面中，如果按［F6 选择全轴］，则所有轴有效。
并且，如果按［F4 不选择全轴］，则变为所有轴无效。

外力最大值

表示在使机械手手动或自动动作时，机械手检测出的外力的最大值。请以该值为参考来设定检测等级。只有电机为ON状态的碰撞检测有效设定轴才计算外力最大值，无效设定的轴显示为0。并且，要想按程序单位、动作单位确认最大值时，将在动作前清除当前的外力最大值。

[1]外力最大值的显示方法

可以通过下列多功能教导器操作显示外力最大值。将显示功能有效的轴的外力最大值。

操作路径：基本画面－［F2 臂］－［F6 辅助功能］－［F2 碰撞检测］



[2]外力最大值的清除方法

在碰撞检测设定画面上按［F3 清除外力］，即显示确认讯息，然后按［OK］。手动和自动的外力最大值都将被清除。并且，即使关闭控制器电源，外力最大值也将被清除。



检测等级的设定

可以利用程序库或多功能教导器，按每根轴设定碰撞检测的灵敏度（检测等级）。输入范围为1～500，值越大灵敏度就越迟钝，值越小就越灵敏。
默认值为100，是机械手即使按额定负荷最大限度地动作也可以不出错地检测碰撞的值。
利用多功能教导器进行的设定，在关闭控制器电源之后也能保持。利用程序库进行的设定，在关闭控制器电源之后即返回原来的设定。

> ⚠注意： 为了防止误检，请在设定检测等级时，对显示的外力最大值留有余地。请以外力最大值的1.2倍的值为标准进行设定。

[1]利用程序库设定

可以通过在PAC程序上记述程序库SetCollisionLevel，按程序单位、动作单位来设定检测等级。详情请参照编程手册Ⅱ PAC程序库。

[2]利用多功能教导器设定

可以通过下列多功能教导器操作，设定检测等级。在机械手操作过程中不能设定检测等级。

操作路径：基本画面－［F2 臂］－［F6 辅助功能］－［F2 碰撞检测］

(1) 在多功能教导器的画面上触摸手动时等级、自动时等级的相应栏。



(2) 按［F5 变更值］，用十键输入数值，然后按［OK］。



(3) 显示已变更的值。



碰撞检测警告

在碰撞检测功能有效的状态下，机械手检测出的外力最大值超过设定的检测等级时，机械手将紧急停止，并在多功能教导器上显示通知碰撞的警告。
将对检测出碰撞的轴显示警告。按［OK］即可清除警告。



显示碰撞检测设定画面，即可知道外力最大值已超过检测等级。



注意：	机械手或工具在接触设备等的状态下紧急停止时，如果清除警告显示之后，原封不动地将电机电源置于ON，则会重新检测出碰撞，电机电源可能不能置于ON。为了让机械手退避，想要将电机电源置于ON时，请使碰撞检测功能“无效”或增大检测等级之后再将电机电源置于ON。

使用碰撞检测功能时的注意事项

(1) 如果没有正确设定前端负荷，就可能会误检碰撞，对机械手的外力检测灵敏度也有影响，因此，请正确设定前端负荷之后再使用。
(2) 本功能不能与力限制功能同时使用。请在力限制功能为OFF的状态下使用。
(3) 本功能对附加轴无效。
(4) 因为机械的老化（润滑脂量的变化、齿轮的磨损等），机械手检测的外力可能会变大或发生误检的现象。届时请提高检测等级。
(5) 在机械手程序中指定切换本功能的有效或无效时，等到机械手轴的工作结束之后再进行切换。机械手轴动作期间不能切换。
(6) 请正确调整搭载了气动平衡功能的机械手的气动平衡。如果气动平衡调整不当，有可能会影响检测值，有时会导致检测出错。

关于旧版的项目数据

当使用低于以下版本的旧项目数据时，需要按照以下程序，重新构建项目数据。

VS-G系列：Ver.2.61之后的版本
VS-***系列：Ver.3.20之后的版本
VM-G、VP-G系列：Ver.2.70之后的版本
HM-G、HS-G系列：Ver.2.80之后的版本

(1) 将旧的项目数据转发到控制器上（上述版本之后的版本）。
(2) 事先记下使用条件参数、附加轴设定、软件限位、传送带跟踪的各设定值。
(3) 利用WINCAPSⅢ（上述版本之后的版本）创建新项目。
(4) 仅将新项目的臂参数发送至控制器。注意不要将取消值也发送过去。
(5) 将在(2)中记下的设定反映到控制器上。
(6) 将所有控制器数据接收到新项目中。

第 4 章

速度、加速度、减速度的指定

为使机械手动作，有必要设定相对于机械手的最大能力的速度、加速度、减速度的比例。
此章就速度、加速度、减速度的含义和设定进行说明。

第 4 章 速度、加速度、减速度的指定

4.1 外部速度、内部速度

在机械手的速度中，有外部速度和内部速度。
所谓外部速度是指在程序执行之前，通过多功能教导器或外部设备设定的速度。
所谓内部速度是指在程序中通过指令设定的速度。

4.2 速度指定

机械手动作时的实际速度是由外部速度和内部速度的乘积所决定的
例如：
外部速度：70%
内部速度：30%
这样设定的情况下，
实际的速度 = 最高速度 × 0.7 × 0.3
所以，最高速度的21%成为自动模式下的实际速度。

⚠ 注：在手动模式、教导检查模式中，以自动模式时的10%的速度动作。

在程序中，如果通过SPEED指令变更内部速度，则内部加速度、内部减速度也被自动设定。用100去除内部速度的平方得到的值，为内部加速度和内部减速度。JSPEED指令变更内部轴速度。如果变更内部轴速度、则内部轴加速度、内部轴减速度也被自动设定。用100去除内部轴速度的平方得到的值，为内部轴加速度和内部轴减速度。

4.3 外部加速度、外部减速度、内部加速度、内部减速度

在加速度与减速度中，分别有外部加速度、外部减速度及内部加速度和内部减速度。

所谓外部加速度、外部减速度是指在程序执行之前，通过多功能教导器或外部设备设定的加速度。

所谓内部加速度、内部减速度是指在程序中通过指令设定的加速度和减速度。

4.4 加速度、减速度的设定

机械手动作时的实际加速度，由外部加速度和内部加速度的积所决定。减速度也一样。

例如：

外部减速度：70%

内部减速度：30%

这样设定的情况下，

实际的减速度 = 最高减速度 × 0.7 × 0.3

所以，最高减速度的21%成为自动模式下的实际减速度。

设定加速度和减速度的指令如表4-1所示。

⚠ 注：在手动模式、教导检查模式下，以自动模式时的10%的加速度、减速度动作。

表4-1 加速度、减速度的设定指令

指令	功能
SPEED	内部速度设定
ACCEL	内部加速度设定
DECEL	内部减速度设定
JSPEED	内部轴速度设定
JACCEL	内部轴加速度设定
JDECEL	内部轴减速度设定

4.5 速度、加速度设定示例

图4-1中表示的程序没有进行内部速度的设定。
如果以外部速度80%的设定执行这样的程序，则为同图中所示的值。

```
PROGRAM PRO1
TAKEARM
    MOVE P, P1
END

实际速度    = 外部速度 × 内部速度
            = 80% × 100%
            = 80%

实际加速度 = 外部加速度 × 内部加速度
            = 64% × 100%
            = 64%

实际减速度 = 外部减速度 × 内部减速度
            = 64% × 100%
            = 64%
```

图4-1 不进行内部速度设定的示例

图4-2中表示的程序有内部速度的设定。
如果以外部速度80%的设定执行这样的程序，则为同图中所示的值。

```
PROGRAM PRO2
TAKEARM
    SPEED 50              '内部速度50%
    MOVE P, P1
END

实际速度    = 外部速度 × 内部速度
            = 80% × 50%
            = 40%

实际加速度 = 外部加速度 × 内部加速度
            = 64% × 25%
            = 16%

实际减速度 = 外部减速度 × 内部减速度
            = 64% × 25%
            = 16%
```

图4-2 进行内部速度设定的示例

图4-3中表示的程序有内部加速度和内部减速度的设定。
如果以外部速度80%的设定执行这样的程序，则为同图中所示的值。

```
PROGRAM PRO1
TAKEARM
    ACCEL 50                    '内部加速度50%
    DECEL 25                    '内部减速度25%
    MOVE P, P1
END

实际速度    = 外部速度 × 内部速度
            = 80% × 100%
            = 80%

实际加速度  = 外部加速度 × 内部加速度
            = 64% × 50%
            = 32%

实际减速度  = 外部减速度 × 内部减速度
            = 64% × 25%
            = 16%
```

图4-3 进行内部加速度与内部减速度设定的示例

⚠ 注意： 在设定加速度、减速度时，请确认随通过 (Pass) 动作的路径变化无碰撞等危险。
当在通过 (Pass) 动作中基于动作选项或指令更改了速度、加速度时，因为路径发生变化，所以存在碰撞等危险性，请加以注意。

4.6 最佳可搬运质量设定功能

是与机械手的前端负荷或姿势相应，设定最佳速度、加速度的功能。可以选择表 4-2 所示的 4 个模式。

表 4-2 最佳可搬运质量的模式

模式	设定条件	设定内容	
		PTP 动作	CP 动作
0	负荷条件值	最高加速度	最高加速度
1	负荷条件值 机械手的姿势	最高速度、加速度	与模式 0 相同
2		与模式 0 相同	最高速度、加速度
3		与模式 1 相同	与模式 2 相同

4.6.1 模式 0

是出厂时的默认模式。通过机械手的负荷条件值，设定 PTP 动作、CP 动作时的最高加速度。

设计时循环时间的计算，请参照《机械手概述》的 "定位时间" 的资料。

4.6.2 模式 1

通过机械手的负荷条件值与动作中的机械手的姿势，设定 PTP 动作时的 1、2、3 轴的最高速度与加速度。PTP 动作时的 4、5、6 轴和 CP 动作与模式 0 相同。

4.6.2.1 使用模式 1 的情况下

需要缩短 PTP 动作的动作时间的时候，请设定为模式 1。
模式 1 设定时的动作时间，请在锁定运行状态下进行确认。

4.6.2.2 使用时的注意事项

在动作过程中，有时会发生过载错误和偏差过大错误。关于负荷率，请在调整时利用多功能教导器的过载预定值（参照操作指南 "5.3 机械手的当前位置显示、［F2 臂］-［F6 辅助功能］-［F10 过载预测］"），或者利用 WINCAPSⅢ的 LOG 功能进行确认。
发生过载错误时，请设定计时器和内部速度、加速度，调整电机负荷。
发生偏差过大错误时，请调整速度、加速度。

由于动作速度，有时通过 (Pass) 轨道会有 20mm 左右的变化。在障碍物附近进行通过 (Pass) 动作，有可能与障碍物产生干扰，因此要在模式 0 下执行。

4.6.3 模式 2

通过机械手的负荷条件值与动作中的机械手的姿势，设定 CP 动作时的最高速度与加速度。PTP 动作时与模式 0 相同。

4.6.3.1 使用模式 2 的情况下

以下的 2 种情况，请设定为模式 2。

・需要缩短 CP 动作时的动作时间时
有时在动作中速度会自动降低，所以请在锁定运行状态下进行动作时间确认。

・如果使用模式 0 或模式 1，则会在 CP 动作中发生指令速度限制溢出错误 (6081～6086)。在模式 0 或模式 1 设定时，当 CP 动作时的轨道通过特异点附近（操作手册 "4.1.3 关于手腕、肘、臂的形态 [2] 形态的边界"）或动作范围限制附近时，有时会发生指令速度限制溢出错误而停止。此时，如果设定为模式 2，则在指令速度限制内自动降低速度，并且可以在不发生错码的情况下动作。

4.6.3.2 模式 2 使用时的注意事项

・有时会在动作中发生过负荷错误。在调整时，请使用多功能教导器的过负荷预想值（参照操作指南 "5.3 机械手的当前位置的表示，［F2 臂］-［F6 辅助功能］-［F10 过载预测］"），或是 WINCAPSⅢ的 LOG 功能，确认负荷率（WINCAPSⅢ指南 "LOG 功能"）。发生过载错误时，请对定时器和内部速度、加速度进行设定，从而调整电机负荷。

・由于动作速度，有时通过 (Pass) 轨道会有 20mm 左右的变化。在障碍物附近进行通过 (Pass) 动作，有可能与障碍物产生干扰，因此要在模式 0 下执行。

・在 CP 动作时，在等速移动区间中，因为速度有可能变化，所以需要等速移动的作业，请在模式 0 或模式 1 状态下执行。

・在 CP 动作时，有时会发生指令加速度限制溢出错误 (6761～6766)、偏差过大错误 (6111～6116)。当发生错误时，请通过内部速度、内部加速度对速度、加速度进行调整。此外，有时在高速动作时会产生最大 5mm 左右的轨迹错位。当在动作附近有障碍物时，请降低速度使用。

・当在特异点附近（参照操作向导 "4.1.3 关于手腕、肘、臂的形态 [2]形态的边界"）在动作速度正在降低时使其瞬时停止时，有时瞬时停止时间会延长。但是，瞬时停止距离没有变化。

4.6.4 模式 3

PTP 动作时与模式 1 同样运行，CP 动作时与模式 2 同样动作。请参照 "4.6.2 模式 1"、"4.6.3 模式 2"。

4.6.5 设定所需要的使用条件

明确安装在机械手第 6 轴上的负荷（工具和工件）质量和重心位置之后，请设定前端负荷质量和重心位置。前端负荷质量和重心位置，在多功能教导器或者WINCAPSⅢ使用条件上进行设定（参照 "4.6 最佳可搬运质量设定功能"）。为了适应操作过程中的工具、工件变更，也可以在程序中进行变更。通过程序进行变更的方法，请参照 "4.7.2 内部负荷条件值（前端负荷质量、负荷重心位置）和内部模式设定方法"。

4.6.6 在设定时

用该功能所设定的负荷条件值（前端负荷质量、负荷重心位置）和模式中，有从多功能教导器和 WINCAPSⅢ设定的外部负荷条件值、外部模式和在程序内部设定的内部负荷条件值、内部模式。

如果设定外部负荷条件值、外部模式，则内部负荷条件值、内部模式等也成为相同的值。

机械手从内部负荷条件和内部模式设定最高速度、加速度，并执行动作。

当前的内部负荷条件值和内部模式，可以在多功能教导器上进行确认（参照 "4.7 "使用条件" 中的最佳可搬运质量设定功能"）。

注意①：负荷条件值必须设定与负荷要求相适应的正确值。
如果设定值不正确，则在动作过程中会发生过电流错误、偏差过大错误和过载错误。还会造成机械手故障，在异常发生时、机械手停止输入时停止距离延长，有与周围设备发生碰撞的危险。

②：请正确设定机械手的安装条件。请使用多功能教导器、WINCAPSⅢ进行设定。
如果设定值不正确，则在动作过程中会发生过电流错误、偏差过大错误和过载错误。还会造成机械手故障，在异常发生时、机械手停止输入时停止距离延长，有与周围设备发生碰撞的危险。

■ 关于控制器电源起动时的模式

是可以选择将控制器电源启动时的设定初始化为最佳可搬运质量设定模式的"模式0"、还是保持当前的设定模式。
（参照 "4.7.4 最佳可搬运质量初始化设定的设定方法"）

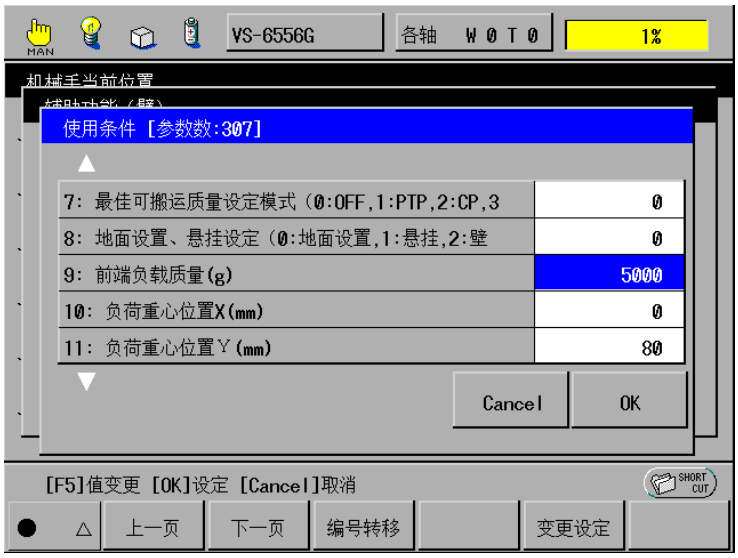
4.7 "有关使用条件" 的最佳可搬质量设定功能

4.7.1 外部负荷条件值（前端负荷质量、负荷重心位置）与外部模式的设定方法

由多功能教导器进行的设定

操作路径：基本画面 –［F2 臂］–［F6 辅助功能.］–［F7 使用条件.］

如果操作多功能教导器、经由上述的操作路径，则出现 "使用条件[参数数:]" 画面。当前的内部负荷条件值和内部模式等被显示。（请参照操作指南 "2.8 负荷质量、负荷重心、最佳可搬运质量有关的基本参数的设定 (TP / WC) ■使用多功能教导器"）



在该 "使用条件［参数数:］" 画面中，选择以下项目，按下 "F5 变更设定"，则数字键盘形式的 "更改参数" 画面被显示，可以在此更改各个参数。

设定项目：
［最佳可搬运质量设定模式］
［前端负载质量 (g)］
［负荷重心位置X (mm)］
［负荷重心位置Y (mm)］

［负荷重心位置Z (mm)］、
但是，<4轴机械手的Ver.1.9以后版本> 为［负荷惯量 ($kgcm^2$)］

最佳可搬运质量初始化设定是 0～3，如果输入以外的设置，则会发生 "输入的数值超过范围"。

前端负荷质量，如果输入的数值超过每个机械手所设定的允许值，则会发生 "输入的数值超过范围"。

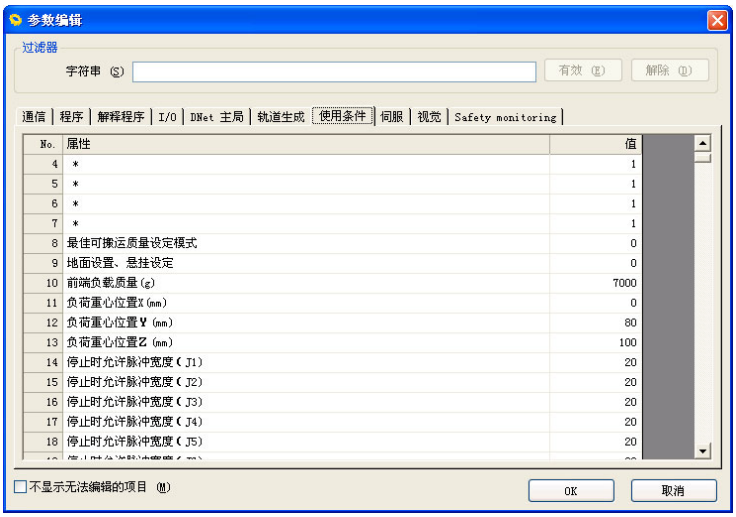
负荷重心位置的输入要满足允许范围。在不满足每个机械手所设定的允许范围时，则会发生 "输入的数值超过范围"。



由 WINCAPSⅢ进行的设定

通过使用程序支持工具WINCAPSⅢ，对外部负荷条件值（前端负荷质量、重心位置）和外部模式的设定方法进行说明。（请参照操作指南 "2.8负荷质量、负荷重心、最佳可搬运质量有关的基本参数的设定 (TP / WC) ■使用WINCAPSⅢ", 以及WINCAPSⅢ指南 "参数的确认"。）

如果从WINCAPSⅢ的［项目(P)］菜单中选择［参数(T)］，则［参数编辑］的视窗会显示出来。
如果点击［参数编辑］视窗中的［使用条件］，则使用条件参数的一览会显示出来。



在该画面中，双击以下项目中的设定值栏，则可进行各参数值的输入及更改。

设定项目：
［最佳可搬质量设定模式］
［前端负载质量 (g)］
［负荷重心位置X (mm)］
［负荷重心位置Y (mm)］

［负荷重心位置Z (mm)］
但是，<4轴机械手的Ver.1.9以后版本> 为［负荷惯量 ($kgcm^2$)］

设定各参数值之后，将数据传送给机械手控制器。
首先用多功能教导器的 [MOTOR]，将电机电源置于OFF。
点击WINCAPSⅢ［通信(N)］菜单中的［发送接收数据(T)…］，弹出［发送接收数据］对话框，进行发送操作。
关于发送数据，请参照WINCAPSⅢ指南中的 "4.7 与机械手控制器的通信"。



注意①：点击 "发送接收数据" 对话框中的 "← 接收(R)"，则可以接收控制器的数据。在这种情况下，外部负荷条件值和外部模式被接收。在程序中变更的内部负荷条件值和内部模式，不能被转发到WINCAPSⅢ一方。

②：负荷重心位置用工具0坐标系进行输入（参照图4-9）。单位是mm。工具0坐标系的原点是6轴法兰中心，Y成分为从法兰中心开始至ϕ6H7销钉孔方向（定向矢量方向），Z成分是通过法兰中心与法兰面垂直的方向（近似矢量方向），X成分是以定向矢量为Y轴、近似矢量为Z轴时的右手坐标系中的X轴方向（常规矢量方向）（参照图4-10）。

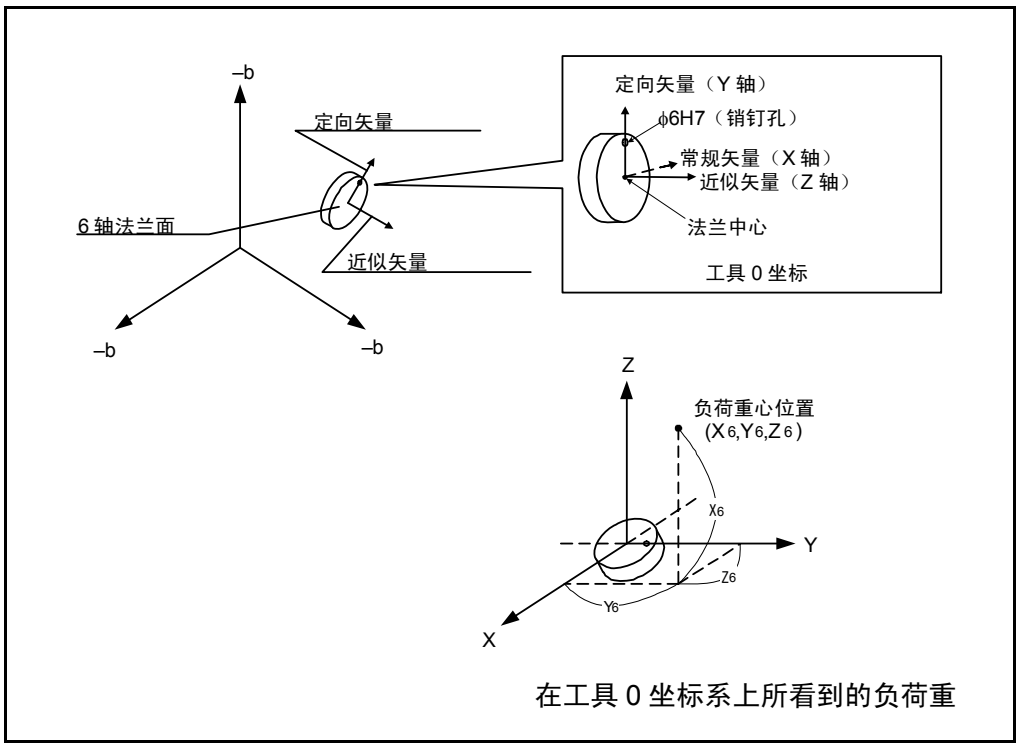


图4-9 负荷重心位置

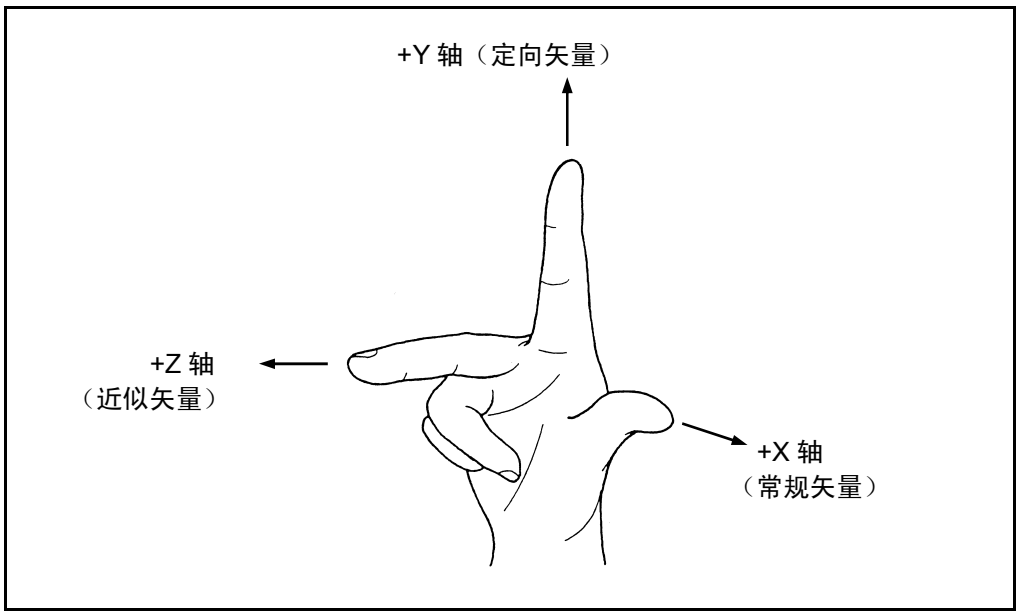


图4-10 右手坐标系

4.7.2 内部负荷条件值（前端负荷质量、负荷重心位置）与内部模式的设定方法

4.7.2.1 内部负荷条件值的设定方法

通过执行以前的语言共用程序库 (Library) aspACLD 进行设定。详细内容请参照 "22.1 以前的语言、aspACLD"。

4.7.2.2 内部模式的设定方法

通过执行以前的语言共用程序库 (Library) aspChange 进行设定。详细内容请参照 "22.1 以前的语言、aspChange"。

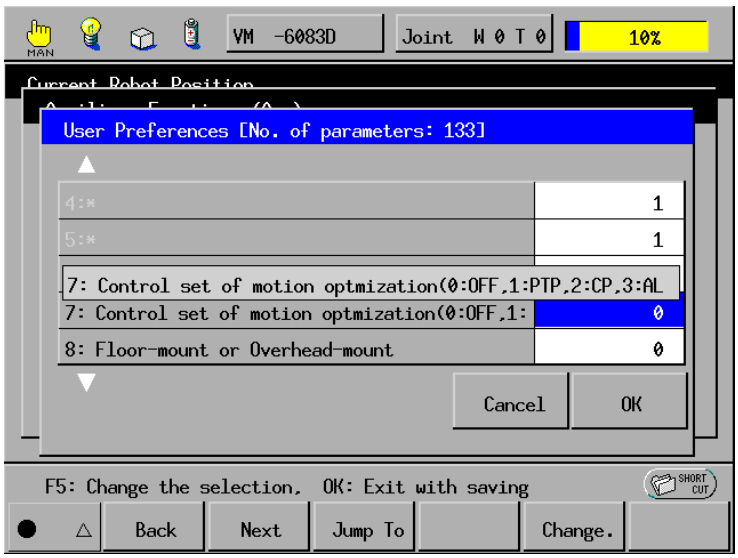
4.7.3 机械手的安装条件设定方法

当机械手为6轴机械手，进行地面安装、悬挂安装使用时的设定。地面安装设定为 "0"， 悬挂设定为 "1"。出厂时设定为 "0"（地面安装）。当悬挂使用时，请变更设定。

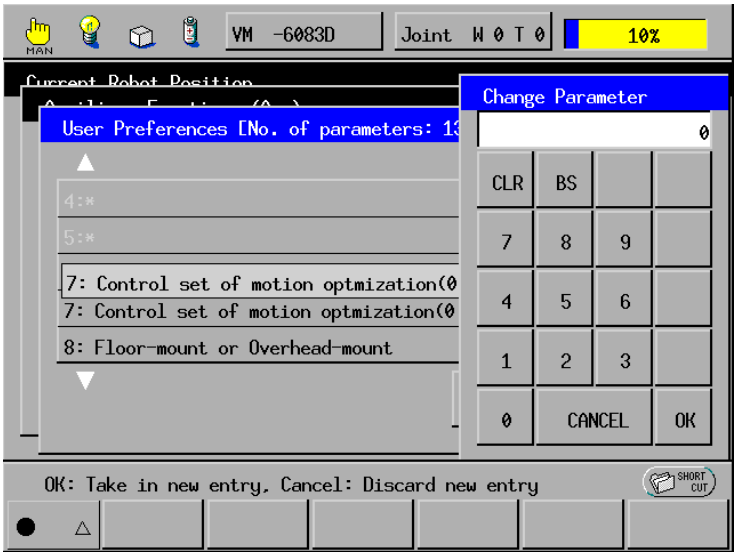
由多功能教导器进行的设定

操作路径：基本画面 –［F2 臂］–［F6 辅助功能.］–［F7 使用条件.］

如果操作多功能教导器、经由上述的操作路径，则出现 "使用条件[参数数:]" 画面。当前的内部负荷条件值和内部模式等被显示。（请参照操作指南 "2.10 机械手安装条件的设定 (TP / WC) ■使用多功能教导器"。）



在该 "使用条件［参数数:］" 画面中，选择［地面设置、悬挂设定］的项目，按下［F5 变更设定］，则数字键盘形式的［变更参数］画面被显示，可以在此更改各个参数。
请将设置条件设定为 "0" 以及 "1"。如果输入此外的值，就会显示“输入的值超出范围”。



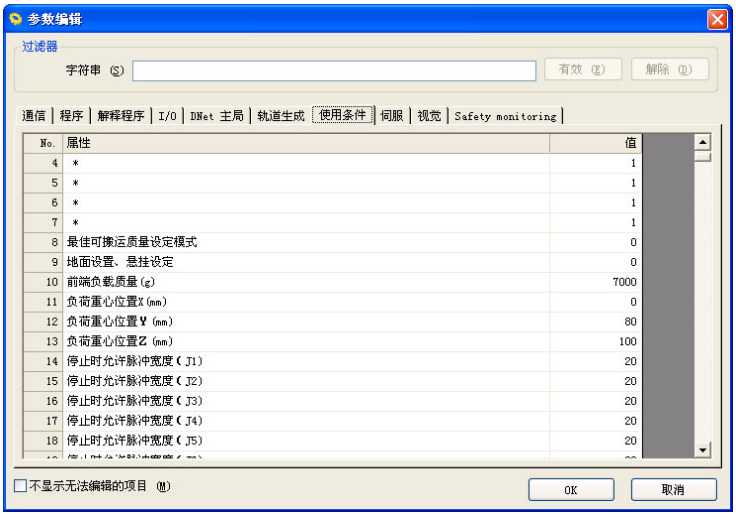
注意：请将通过多功能教导器设定的安装条件传送给WINCAPSⅢ。
传送方法请参照p.4-18的注意①。

由 WINCAPSⅢ进行的设定

通过使用程序支持工具WINCAPSⅢ，对地面设置和悬挂设置状态下使用机械手的设定方法进行说明。（请参照操作指南 "2.9 机械手安装条件的设定 (TP/WC) ■使用WINCAPSⅢ"，以及WINCAPSⅢ指南 "参数的确认"。）

从WINCAPSⅢ的［项目(P)］菜单中选择［参数(T)］，［参数编辑］视窗就会显示出来。

点击［参数编辑］视窗中的［使用条件］图标 ，使用条件参数的一览就会显示出来。



在该画面中，双击［地面设置、悬挂设定］的设定值栏，则可以进行参数值的输入及更改。

设定参数值之后，将数据传送给机械手控制器。
首先，用多功能教导器的［MOTOR］，将电机电源置于OFF。
点击WINCAPSⅢ的［通信(N)］菜单中的［发送接收数据(T)…］，打开 "发送接收数据" 对话框，然后进行发送操作。
关于发送数据，请参照WINCAPSⅢ指南的 "4.7 与机械手控制器的通信"。



注意：点击［发送接收数据］对话框中的［← 接收］，则可以接收控制器中的数据。在这种情况下，外部负荷条件值和外部模式被接收。在程序中变更的内部负荷条件值和内部模式，不能传送到WINCAPSⅢ一方。

4.7.4　最佳可搬运质量初始化设定的设定方法［V1.4 以后版本］

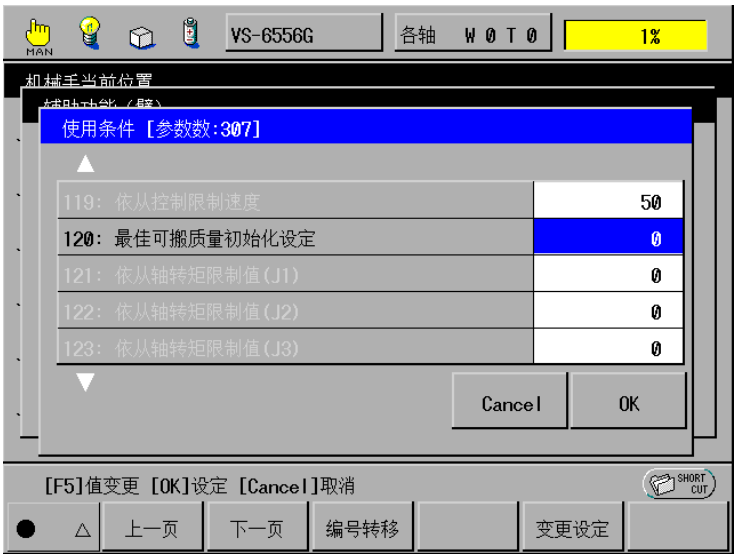
控制器电源ON时，若想将最佳可搬运质量设定模式初始化为 "模式0" 或保持当前设定值，可以进行选择。

设定值	设定值的含义
0	控制器电源接通时，将最佳可搬运质量设定模式初始化为模式 0。 （出厂时）
1	即使接通控制器电源，也不将最佳可搬运质量初始化。 （保持当前的值）

由多功能教导器进行的设定

操作路径：基本画面 –［F2 臂］–［F6 辅助功能］–［F7 使用条件］

如果操作多功能教导器、经由上述的操作路径，则出现 "使用条件［参数数:］" 画面。当前的内部负荷条件值和内部模式等被显示。



在该 "使用条件［参数数:］" 画面中，选择 "最佳可搬质量初始化设定" 项目，按下［F5 变更设定］，则数字键盘形式的［变更参数］画面被显示，可以在此更改各个参数。

0：无效 → 控制器电源接通时初始化

1：有效 → 即使接通控制器电源也不初始化。
（保持当前的值）

最佳可搬运质量初始化设定是0～1，如果输入以外的设置，则发生 "输入的值在范围之外" 的情况。

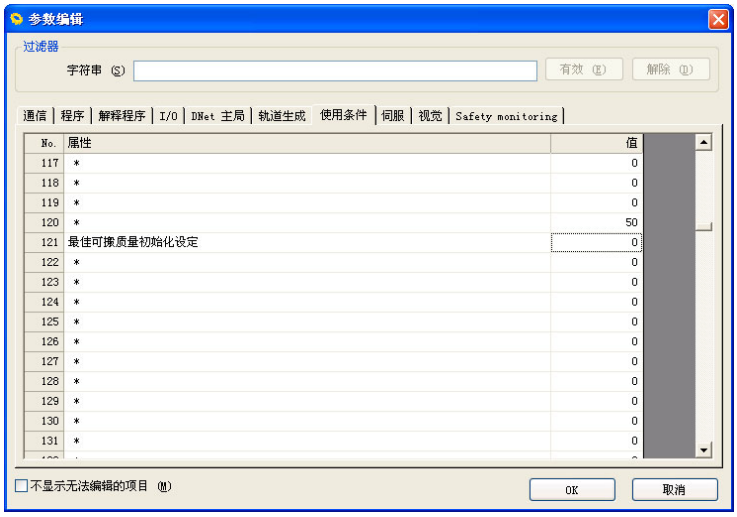


注意：请将通过多功能教导器设定的最佳可搬质量初始化设定转送给WINCAPSⅢ。关于发送方法，请参照P4-18的注意事项②。

由 WINCAPSIII进行的设定

通过使用程序支持工具WINCAPSIII, 对最佳可搬运质量进行初始化设定的设定方法进行说明。

从WINCAPSIII的［项目(P)］菜单中选择［参数(T)］,［参数编辑］视窗即会显示出来。
点击［参数编辑］视窗的［使用条件］，使用条件参数的一览即会显示出来。

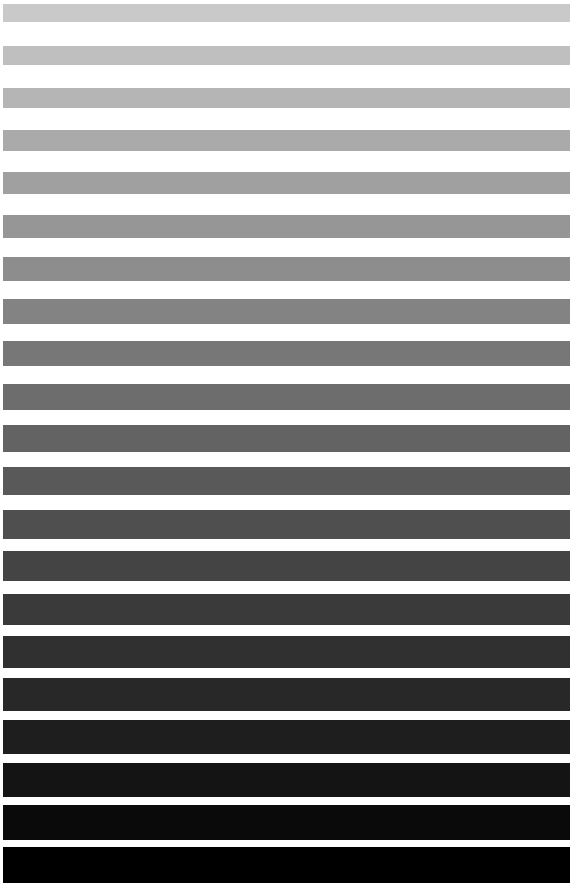


在该画面中，双击最佳可搬质量初始化设定项目，则可以进行参数值的输入及更改。

设定各参数值之后，将数据传送至机械手控制器。
首先用多功能教导器的［MOTOR］，将电机电源置于OFF。
点击WINCAPSIII的［通信(N)］菜单中的［发送接收数据(T)...］，打开 "发送接收数据" 对话框，然后进行发送操作。
关于发送数据，请参照WINCAPSIII指南中的 "4.7 与机械手控制器的通信"。

注意：点击［发送接收数据］对话框中的［← 接收］，则可以接收控制器中的数据。在程序中变更的最佳可搬运质量初始化设定不能发送到WINCAPSIII一方。

第 5 章
视觉控制



对编写程序时所必需的与视觉相关的用语进行说明。

第 5 章 视觉控制

5.1 视觉控制

对作为选件内置于机械手控制器中的μVision板指令进行相关说明。
在编写、输入使用视觉装置的程序时，请予阅读。

5.1.1 视觉控制的用语

5.1.1.1 像素

在μVision卡的内部，将各个点的集合体作为像素数据使用。将每个点称为 "像素(PIXEL)"。
μVision卡所处理的像素数，在存放存储器（处理画面）中为横512像素 × 纵480像素、在描绘专用存储器（描绘画面）中为横624像素 × 纵480像素。

5.1.1.2 辉度

在μVision卡处理的图像数据的各个像素，具有表示亮度的数值。将该数值称为辉度。辉度的取值为0～255之间的256级数值。越接近于0表示越暗，越接近于255则表示越亮。

5.1.1.3 视窗

如图5-1中所示，将μVision卡作为图像处理的范围设定为视窗。视窗在μVision板内部以视窗编号为单位存储其大小。编集视窗的方法有2种，分别是基于WINCAPSⅢ视觉功能的方法，和基于用户程序的方法。如果电源切断，通过用户程序编辑的视窗讯息会丢失，请予注意。

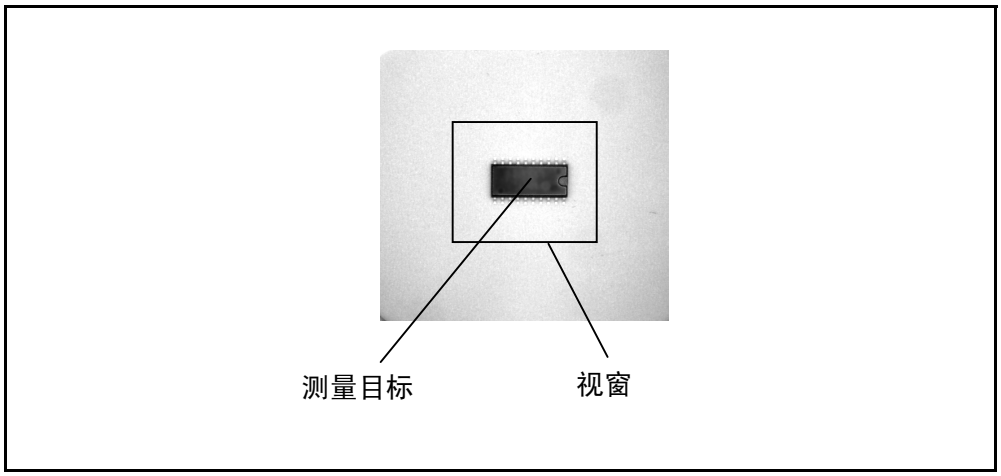


图5-1 图像处理视窗

5.1.1.4 面积、重心、主轴角

[1] 面积

将照相机拍取的图像数据进行2值化处理，并在视窗中指定的范围内的白 (1)与黑 (0) 的各自像素数作为面积进行计数。因为在μVision卡中，不改变视窗内的各个像素的辉度值，实时地进行2值化处理来计数面积，因此无需事先对图像数据进行2值化处理。

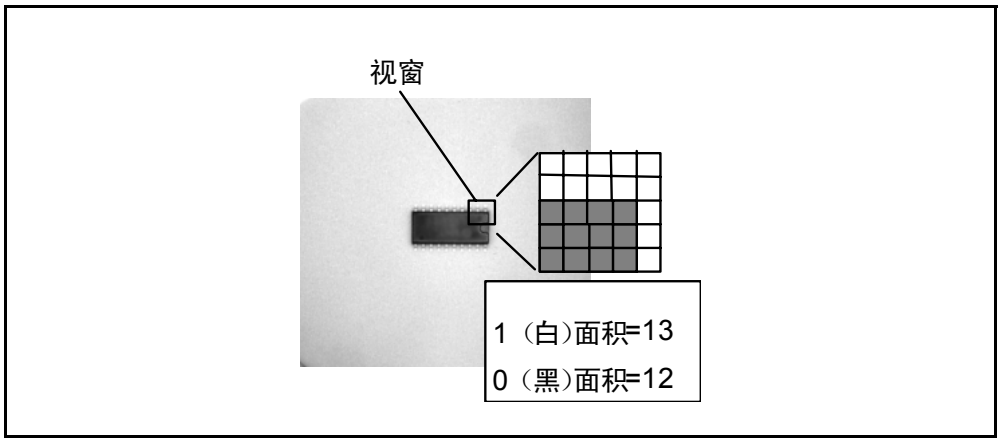


图5-2 面积

[2] 重心

照相机拍取到的图像数据中的目标物变为平面。在平面上，目标物的重量平衡点称为重心。

在μVision卡上，从在视窗上指定的范围内的白 (1) 和黑 (0) 的像素上求出重心。重心用X坐标值和Y坐标值表示。

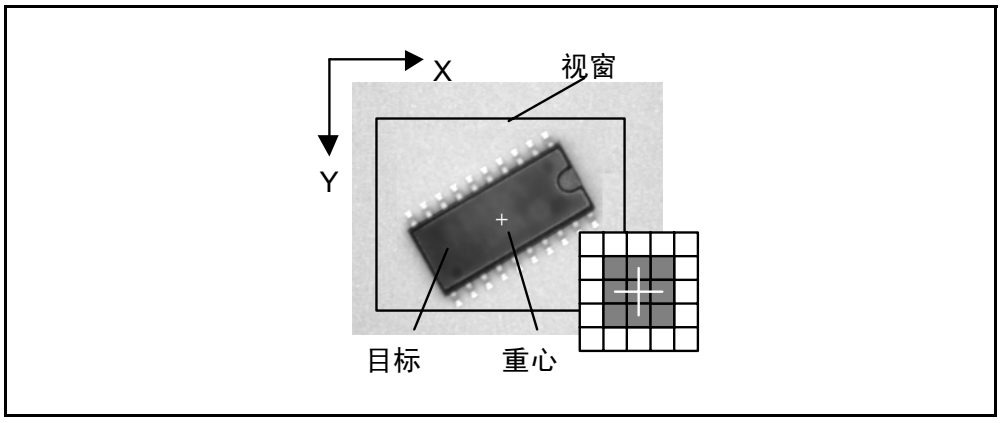


图5-3 重心

[3] 主轴角

照相机拍取到的图像数据中的目标物变为平面。使该平面的目标物旋转时，将最经常旋转的较长方向的轴称为主轴的长轴，将与该轴正交的轴称为主轴的短轴。在μVision卡中，将从水平轴（X轴）到主轴的长轴为止的角度（Θ）定义为主轴角。在将图像数据进行2值化处理之后，针对在视窗中指定的范围内的白 (1) 或黑 (0) 的目标物求得主轴角。

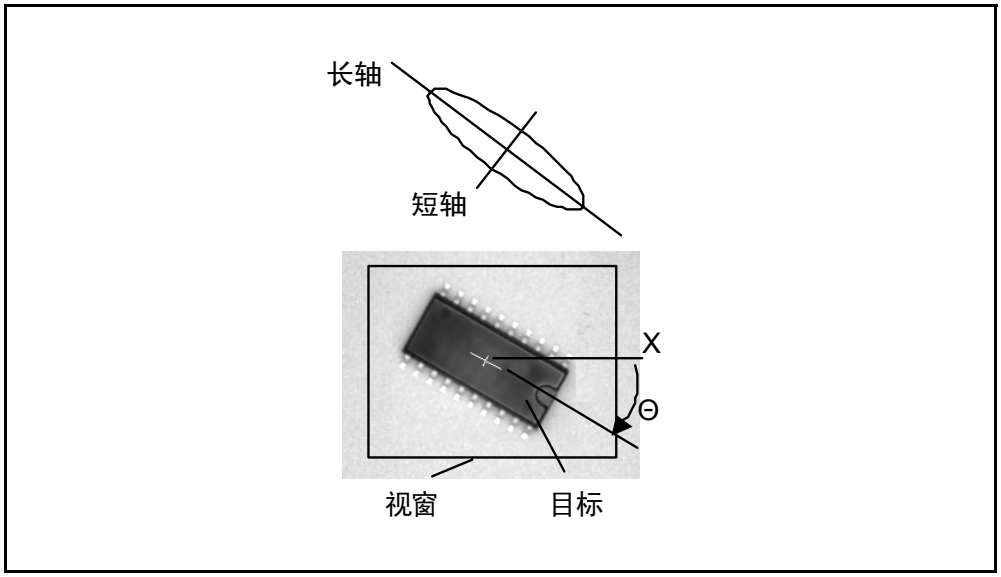


图5-4 主轴角

注意：因为正方形及正圆形的主轴角无法确定，请予注意。

5.1.1.5 2 值化

[1] 2 值化

从照相机输入到μVision卡的图像数据，每个像素都有256级辉度。所谓2值化，是指以界值为界将各个像素的辉度改写为白 (1) 和黑 (0)。该界值称为2值化基准值。μVision卡用2值下限和2值上限两个数值指定2值化基准值。比2值下限大而不超过2值上限的数值被2值化为白 (1)，将以外的数值2值化为黑 (0)。

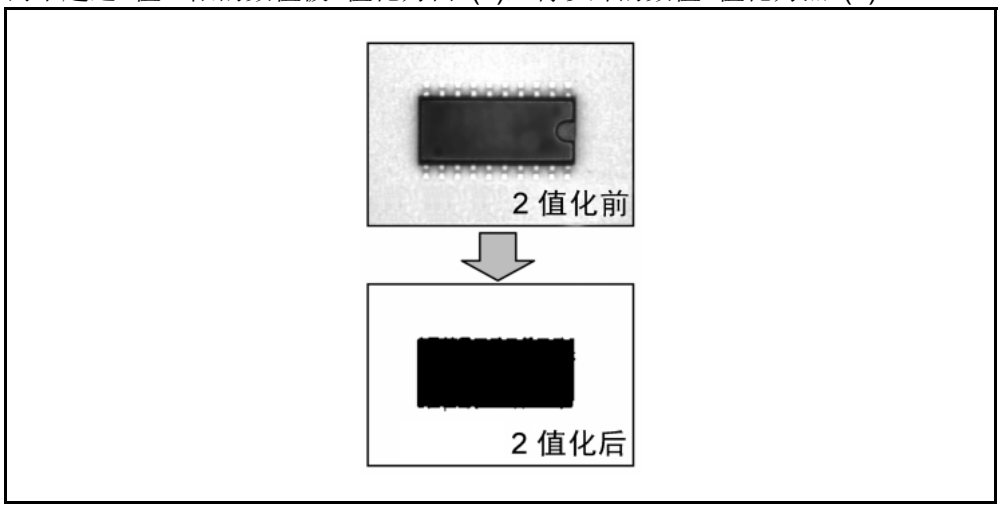


图5-5 2值化

[2] 直方柱状体

直方柱状体是在照相机拍取到的图像数据中的视窗指定范围内，对辉度值的出现频率进行计数的结果。将直方柱状体图像化显示后，辉度值的分布状况变得易懂，且在对图像数据进行2值化处理时2值化基准值的决定也变得容易了。此外，虽然在μVision卡中提供有自动决定2值化基准值的功能，但是这些仍使用直方柱状体决定2值化基准值。图5-6是进行2值化处理前的图像数据的直方柱状体，由其可见对象与背景区分为2的样子。图5-7是以不同的2值化基准值进行2值化处理的结果。通过调整下限（2值下限）与上限（2值上限），2值化处理的结果不同。

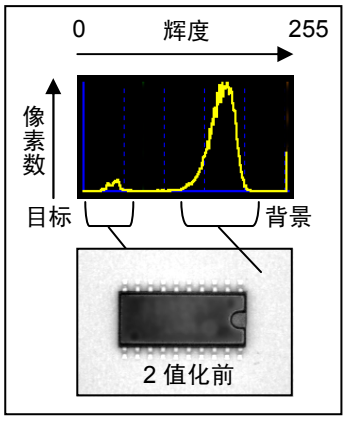


图5-6 直方柱状体（2值化之前）

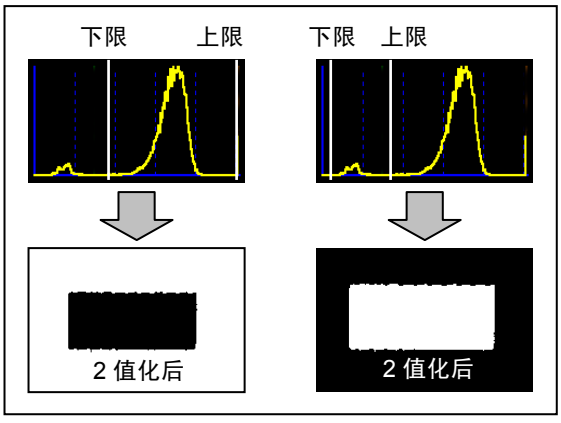


图5-7 直方柱状体（2种2值化）

[3] 2 值化基准值检测

模式法

图像的直方柱状体具有2个高峰（对应于对称图形和背景）的双峰性分布的情况下，在这2个峰之间的谷部分设定2值化基准值 (t) 的方法称为模式法。

在μVision卡中，为了从辉度值0到255的方向查出直方柱状体的峰、谷，在图5-8的情况下，将t点视为谷，设为2值化基准值。

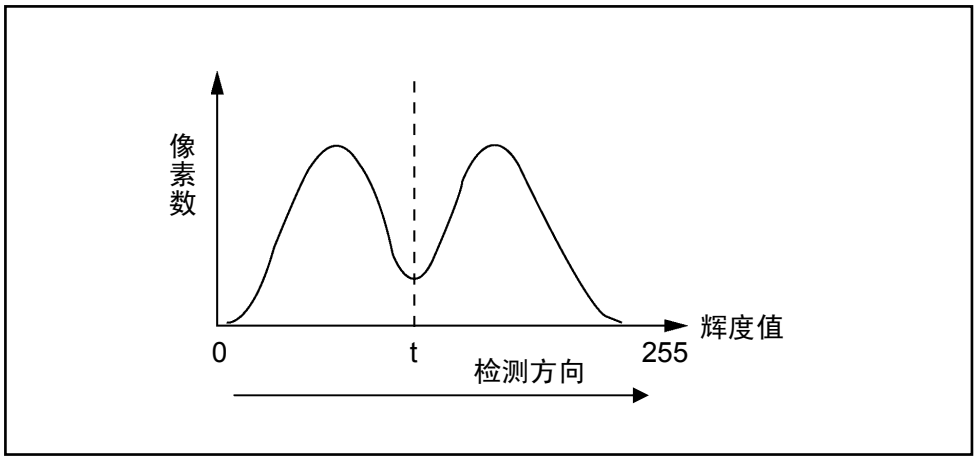


图5-8 通过模式法检测出2值化基准值

判别分析法

在图像直方柱状体中，以2值化基准值K拆分为黑部分和白部分2个等级时，为了使这2个等级之间的分离达到最好，利用统计方法来决定2值化基准值K的方法称为判别分析法。当直方柱状体的图像不具有2个峰的双峰性时，前述的模式法不适用。此时，通过判别分析法可以得到最佳的界值。

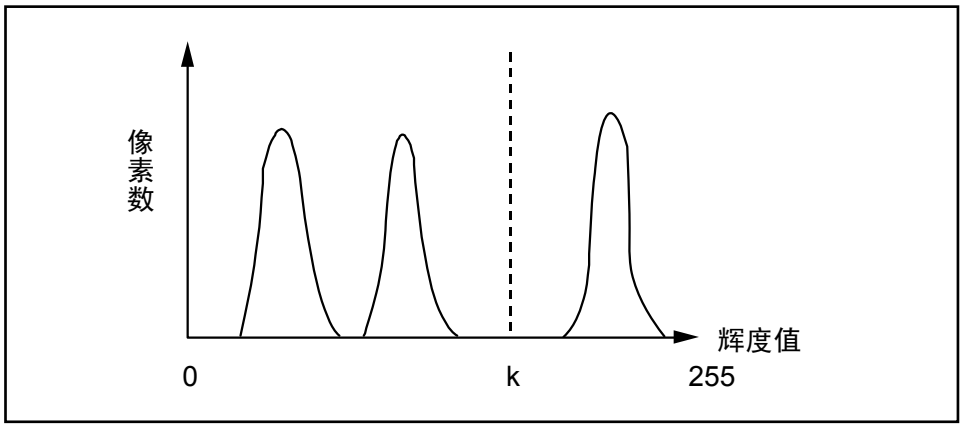


图5-9 由判别分析法检测出2值化基准值

P花砖法

是在预先知道目标的白或者黑的面积为Sx的情况下，利用直方柱状体查出与目标面积Sx相一致的2值化基准值的方法。

目标为白的情况下，从直方柱状体的明亮辉度值的像素进行加算求出面积，恰好为Sx的辉度称为2值化基准值。

目标为黑的情况下，从暗辉度值的像素上加算，以同样方式求出2值化基准值。

在预先知道目标面积的情况下，这是有效的方法。

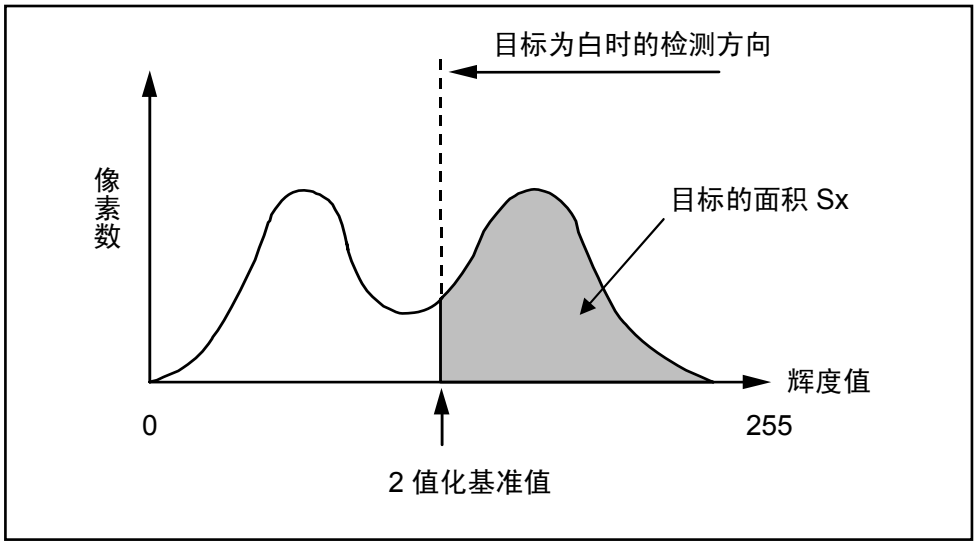


图5-10 由P花砖法检测出2值化基准值

5.1.1.6 辉度积分值

从照相机中取得的图像中，对在视窗上指定的范围内在所有像素的辉度值进行求和称为辉度积分。将求和的结果数值称为辉度积分值。

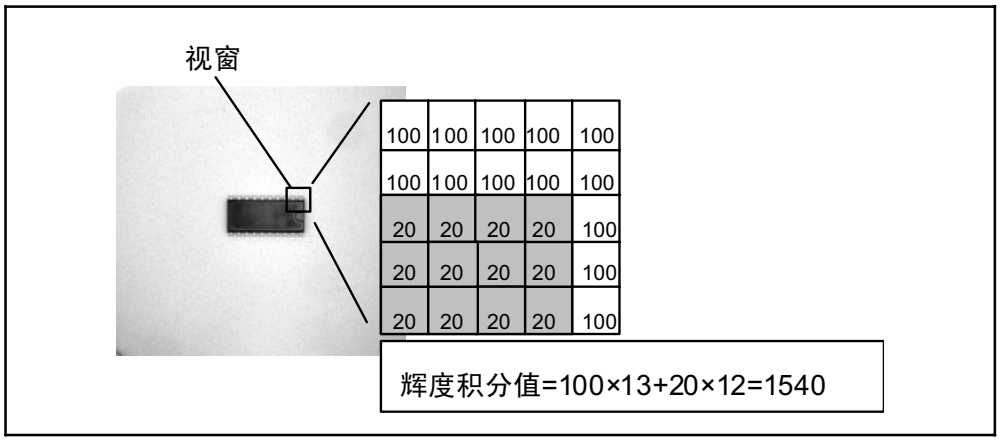


图5-11 辉度积分值

5.1.1.7 边缘

关于所指定的视窗内的目标物，从暗到亮（黑 → 白）、从亮到暗（白 → 黑）进行变化的点，即辉度的变化点称为边缘。

μVision 卡检测到的边缘位置是指视窗内的辉度值或者面积值所满足指定的基准值的部位。在水平值的指定中有绝对值和差分值，根据目标物的状况分别使用。绝对值能够对指定的部位进行检测辉度值或面积值所通过的位置。差分值能够对辉度值或面积值的变化量大于指定值时的位置进行检测。

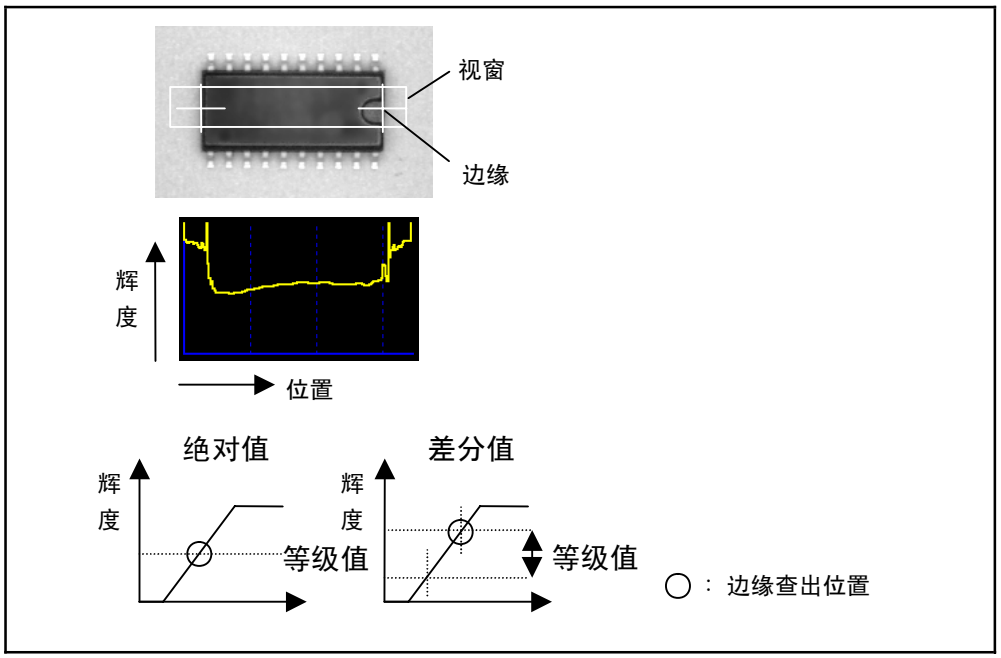


图5-12 边缘

5.1.1.8 标注

将从照相机中取得的图像数据2值化基准值，在白 (1) 或者黑 (0) 的像素的连接空间，按顺序进行编号的处理称为标注。

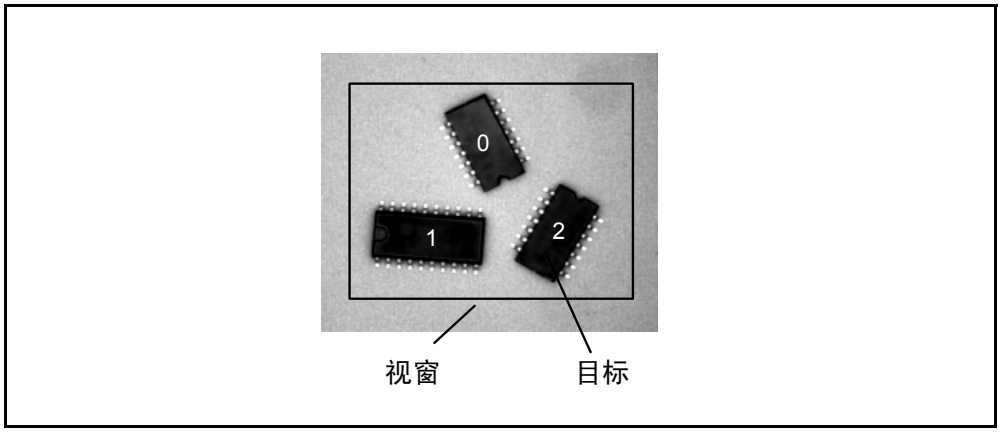


图5-13 标注

通过标注，可以对视窗上指定的范围内所存在的多个目标物逐个进行区别处理。如果在 µVision 卡中进行标注，就可以求出各个目标物的特征即面积、重心、主轴角、脊柱形状、周长。所谓脊柱形状，是指与目标物外接的长方形。

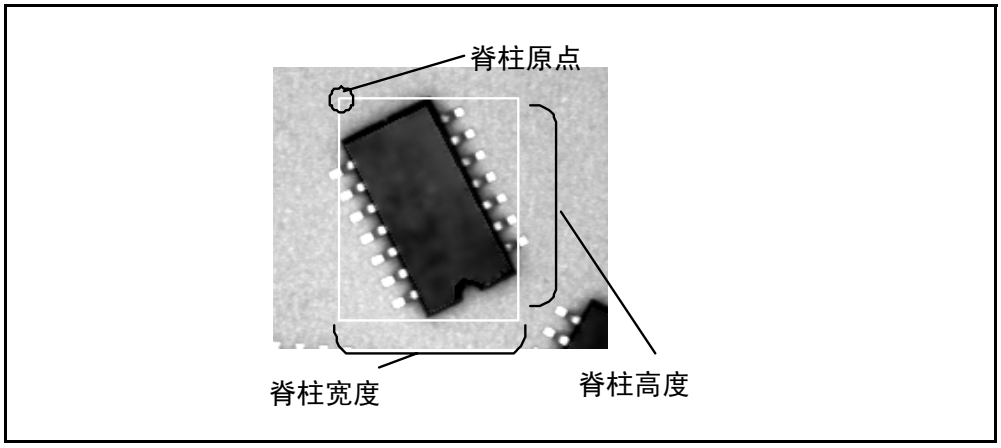


图5-14 脊柱形状

所谓周长，就是构成目标物外形的像素。

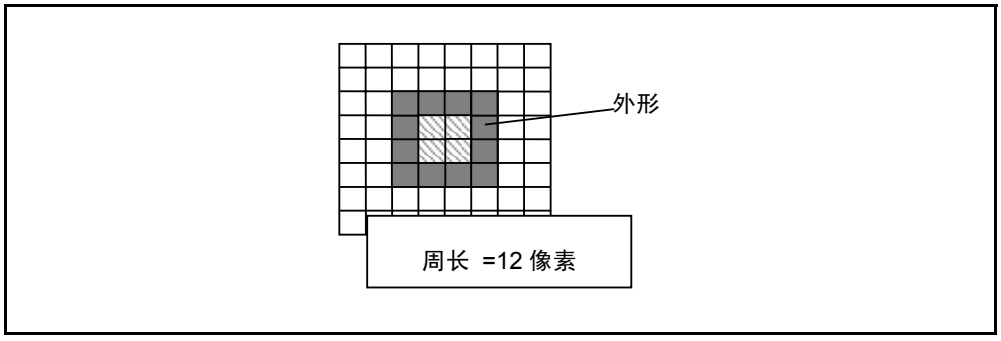


图5-15 周长

5.1.1.9 检索

所谓检索，就是指在测量目标图像的搜索范围内（视窗范围内），运行预先登录的标准的图像数据（检索模型），搜索一致位置的过程。

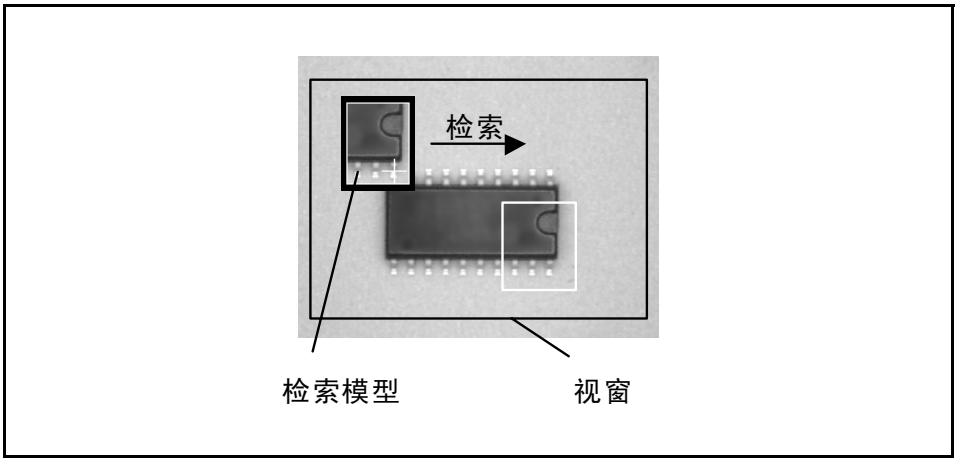


图5-16 检索

在μVision卡中，将标准的图像数据称为检索模型，它由图像数据和基准坐标(OX、OY) 构成。

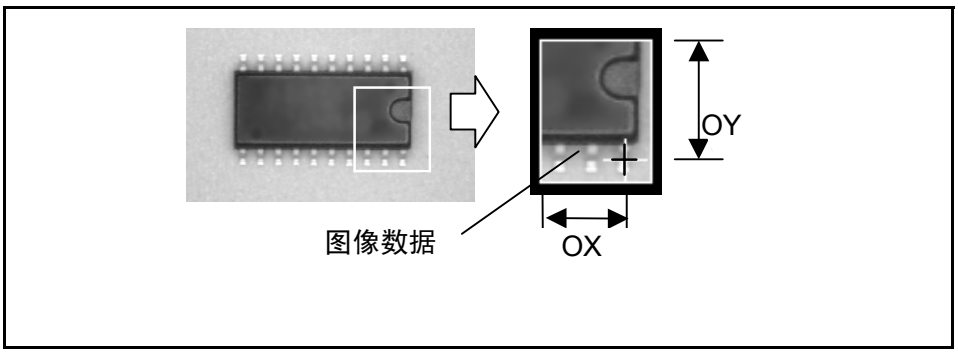


图5-17 检索模型

表示检索模型和测量目标图像相一致程度的数值，称为 "一致度"。得到的一致度大于所指定的值的情况下，可以得到检索模型和测量目标图像一致位置的坐标。坐标的检测精度，可以用像素以最小单位为1像素，辅助像素以1像素以下的精度得到结果。通过辅助像素进行测量的情况下，测量时间要比像素进行的测量时间要长。

5.1.2 在使用视觉命令时的注意事项（视觉卡）

在机械手控制器中使用视觉命令时，需要μVision卡（选件）。
视觉命令需要首先执行视觉处理权取得获取命令 (TAKEVIS)。在μVision-21上不需要。

（例如） TAKEVIS
CAMIN 1
VISPLNOUT 0
:

如果在执行命令时发生了错误，那么用照相机输入命令之外的命令在其次执行视觉命令时会显示错误讯息。

（例如） WINDMAKE R,1,100,10,0,2
VISPRJ 1,100,100,1,128 ← 在该时刻 "视窗形状异常" 错误
WINDDISP 1 ← 在其次的命令执行时显示错误讯息。
:

在通过WINCAPSⅢ与视觉板进行通信时，如执行视觉处理权取得命令（TAKEVIS），会出现错误。

5.1.3 与原有的μVision-15 间的互换性

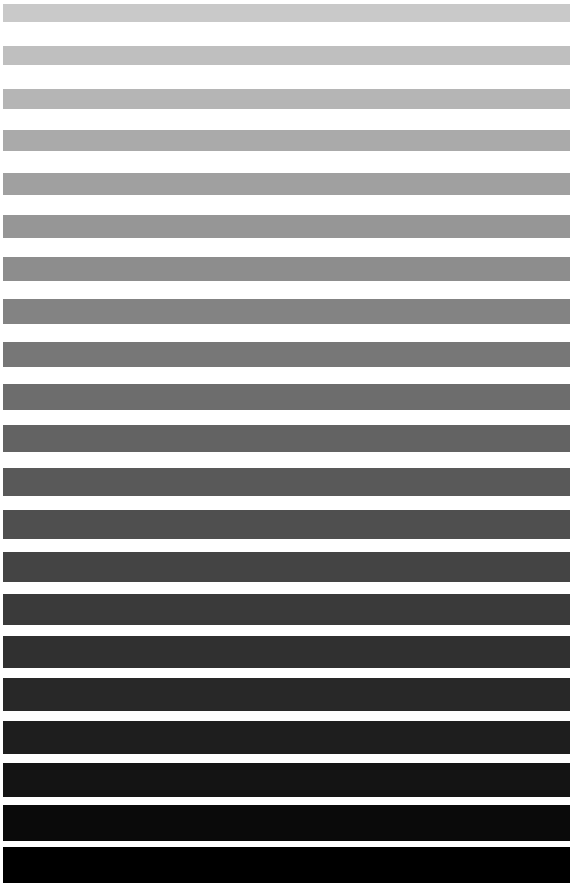
原有的μVision-15的功能继承了外部输出（并行I / O、RS232C）相关之外的功能。但是，因为命令的书写格式等有所变更，在参考原有的程序重新编辑时，请参照下述的对应表。

	μVision-15	μVision卡	说明
图像输出入	VIN	CAMIN	照相机输入
	AOUT	VISCAMOUT	照相机图像的显示
	VOUT	VISPLNOUT	处理画面的显示
		VISOVERLAY	覆盖设定
视窗设定	WNDALN	WINDMAKE	视窗的设定
	WNDCIR		
	WNDDPT		
	WNDELP		
	WNDRCT		
	WNDSCT		
	HWIND		
	CLRWND	WINDCLR	视窗设定数据的消除
	WNDCPY	WINDCOPY	视窗设定数据的复制
	WDISP	WINDDISP	视窗的显示
描绘	SCREEN	VISSCREEN	描绘条件的设定
	CLS	VISCLS	画面的消除
	PUTP	VISPUTP	点的描绘
	LINE	VISLINE	直线（长度、角度指定）的描绘
	LNDPT	VISPTP	直线（2点指定）的描绘
	RECT	VISRECT	矩形的描绘
	CIRCLE	VISCIRCLE	圆的描绘
	ELIPSE	VISELLIPSE	椭圆的描绘
	SECT	VISSECT	扇形的描绘
	PAINT	VISRECT	矩形的描绘
		VISCIRCLE	圆的描绘
		VISELLIPSE	椭圆的描绘
	CROSS	VISCROSS	十字标记的描绘
	CRSALN		
	LOC	VISLOC	字符显示位置指定
	PRINT	VISDEFCHAR	描绘字符的设定
		VISPRINT	字符的描绘
图像处理	GETP	VISGETP	获取指定坐标的辉度
	HIST	VISHIST	执行直方柱状体
	HCLR		
		VISREFHIST	直方柱状体结果的获取
	LEVEL	VISLEVEL	计算2值化标准值
	BINA	VISBINA	2值化基准值处理
	RBINA	VISBINAR	2值化基准值显示
	AREA	VISMEASURE	特征（面积、重心、主轴角等）计算
	BIGT		
	CENTR		
	MOMENT		
	STRT		
	PROJ	VISPROJ	投影处理
	EDGE	VISEDGE	边缘测量
检索功能	CORN	SHCORNER	棱角的检索
	CIRC	SHCIRCLE	圆的检索

第2部

指令参考

第 6 章
指令的表示方法与分类



记述了DENSO机械手所使用的机械手语言PAC的指令的表示方法与指令一览。
要迅速找出有关个别指令的信息，请利用指令一览表。

第 6 章 指令的表示方法与分类

6.1 指令解说的表示方法

第9章以后，就PAC的各个指令进行解说。在此，就其表示方法进行说明。

9.4 初始坐标

(1) (2)

HOME（语句）【SLIM 依据】

(3) 功能　将任意的坐标作为初始位置定义。

(4) 书写格式　HOME <位置型>

(5) 说明　定义以 <位置型> 指定的任意的坐标为初始位置。
在每个程序分别定义初始位置。
HOME 文被多次定义时，以最新的定义作为初始位置。

(6) 关联项目　GOHOME

(7) 应用示例

6 轴

```
DIM lp1 As position
HOME (350, 0, 450, 0, 0, 180 )  '将 (350, 0, 450, 0, 0, 180 ) 坐标定义为初始位置。
HOME lp1                        '将 lp1 坐标定义为初始位置。
```

指令解说的表示方法

(1) 指令的名称：指令的名称在开始时有记述。

(2) 指令的类别：继指令的名称之后记述有指令的类别（语句、函数、系统变量、内置常量）。依据于SLIM规格的指令记述为【SLIM依据】。

(3) 功能：就指令的功能进行说明。

(4) 书写格式：就语法上的记述方法进行说明。指令中有自变量 (argument) 时将自变量 (argument) 用 "<"和 ">" 括起来表示。有多个自变量 (argument) 时，可以用段落记号 ","。可以省略的要素用 "[" 和 "]" 括起来表示。
象 "[<a> [,<b>]]" 这样被 "[" 和 "]" 括起来的 "[,<b>]"，当 <a> 被省略时就表示有省略的必要。
多个要素中，只需要其中任意一个时，将多个要素用 "{" 和 "}" 括起来表示，每个自变量 (argument) 用 "|" 断开表示。

(5) 说明：就指令的使用方法进行说明。

(6) 关联项目：记述相关的其他指令及说明。

(7) 应用示例：就指令的使用示例进行说明。

6.2 指令一览表

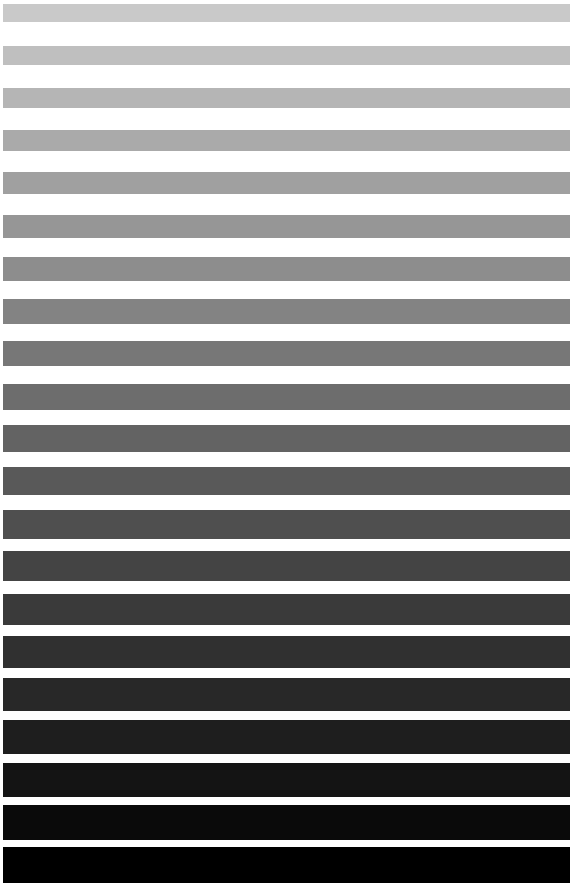
6.2.1 按字母顺序排列的指令一览

请参照目录的 "按字母顺序排列的指令一览"。

6.2.2 按功能单位排列的指令一览

请参照目录的 "按功能单位排列的指令一览"。

第 7 章
PAC 语言的构成要素



在本章中，就构成PAC语言的要素进行说明。

第 7 章　PAC 语言的构成要素

7.1　新机械手语言PAC

记述机械手的动作和操作的编程语言，称为机械手语言。
DENSO机械手所使用的机械手语言，被命名为PAC (Programming language for Assembly Cell)。PAC是为了将机械手控制程序的开发和维护比以往更具高效率而新开发的语言。主要的特征如下所示。

- 它是JIS所规定的、产业用机械手语言SLIM的高度互换语言。
- 由于它是结构化的编程语言所以易读、易开发和维护。
- 不仅是机械手，视觉装置的控制也能用PAC进行统一的记述。
- 通过多项任务功能，能够进行高效的程序处理。
- 通过中断处理功能，可以高效地记述错误及时刻等所采取的例外处理。

7.2　机械手语言PAC与原来的语言的关系

以前，机械手语言，因机械手或制造商不同，大多没有兼容性。对每个机械手使用不同的语言会给用户造成负担，为了使其标准化，JIS制定了产业用机械手程序语言SLIM。

SLIM，主要是以记述装配操作为目的而进行规格化 (JIS B 8439) 的语言，它以BASIC语言为基础，追加了机械手特有的数据类型及机械手动作命令等。因此，它的优点是即使是已经习惯了SLIM以前的机械手语言的人，也能够比较熟练地学习SLIM语言。

DENSO机械手所使用的机械手语言PAC，以具有向上兼容BASIC语言为基础，不仅继承了SLIM的优点，而且具备了结构化语言及多项任务功能等许多特征。关于机械手特有的命令，因规格与SLIM相同，所以PAC也成为SLIM语言的向上兼容。从而，PAC在发挥与SLIM语言同样的便于掌握的特点同时，成为了适应高度要求的机械手语言。

7.3 语言要素

构成PAC语言的要素有以下几种。

●标识符.....为了识别构成要素所用的名称
●变　量.....用于暂时存储数据
●常　量.....具有固定数值的数据
●运算符.....在2个值之间进行计算的符号
●式.....求取数值所需的构成要素的组合
●指　令.....为了执行处理而编入PAC语言的命令

同时，变量及常量的数据有若干种类型。
在本章中，就构成要素以及数据型进行说明。
关于指令，在第9章～第22章进行详细说明。

7.4 名称

PAC语言里有识别程序中各种要素的规则。在本章中，就该规则进行说明。而且，表示指令、变量、函数、标签、程序的名称遵守以下的规则。

●名称必须以字符（半角字母。无大小写区分）或所规定的记号开始。在名称中可以使用字符、数字和下划线。
●在名称中可以使用字符、数字和下划线。
●名称的开头字符必须是字母。
●不能使用句号、斜线、反斜线、空格、冒号、分号、单引号、双引号和星号。
●不能使用+、-、*、/、(、) 等作为运算符使用的字符。
●为了将名称与其他的语言识别开，在名称和其他的语言之间插入空格。
●名称所能使用的字符数最大为64字符。

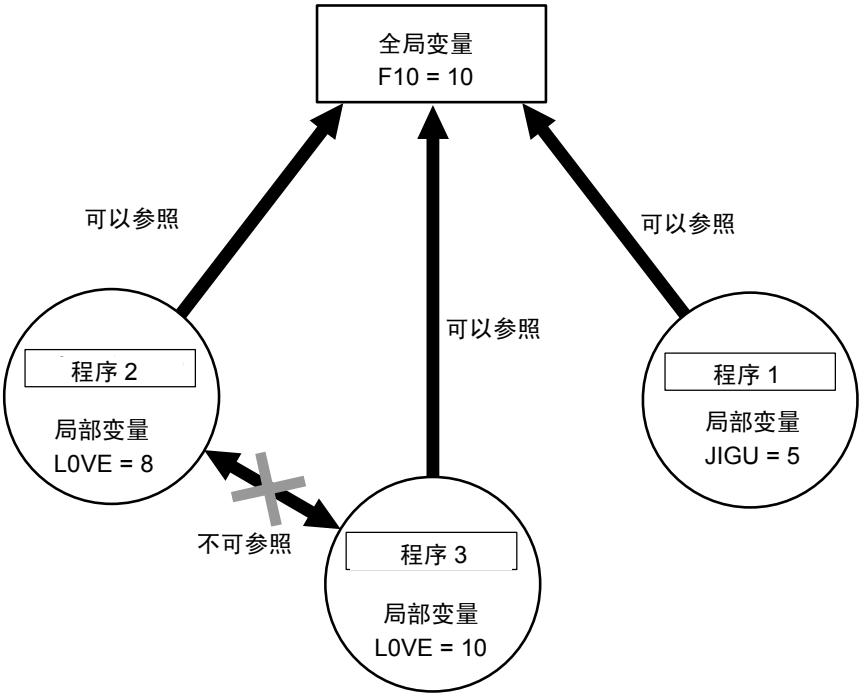
7.5　标识符

7.5.1　变量

变量是指，将程序中使用的数据暂时存放。有全局变量和局部 (Local) 变量、系统变量。

全局变量是从任何一个程序（任务）都能共同使用的变量。

局部 (Local) 变量是只在一个程序中有效的变量。同时被执行的其他程序中，即使有相同名称的局部 (Local) 变量，也只在其所属的程序中运行，相互之间没有影响。



因为局部 (Local) 变量不与其他程序中的变量相干扰，所以可以避免出现误动作。

全局变量和局部 (Local) 变量

［１］ 全局变量

全局变量的名称，在表示类型的字母 (I、F、D、S、V、P、J、T、IO) 的后面加上整数式来表示。只有I / O变量为2个字母 (IO)。
例如，F0001、F1、F [1] 均表示相同单精度实数型变量。
因变量名在系统中被预约，所以无需进行定义即可使用。
在全局变量中，可以使用以下所示的类型。

- I型： 整数型（范围：-2147483648～+2147483647）
  例）I0001、I1、I [1]
- F型：单精度实数型 (-3.402823E + 38～3.402823E + 38)
  例）F0001、F1、F [1]
- D型：双精度实数型 (-1.7976931348623157D + 308～1.7976931348623157D + 308)
  例）D0001、D1、D [1]
- S型：字符串型（最大为243个字符）
  例）S0001、S1、S [1]
- V型：矢量型(X, Y, Z)
  例）V0001、V1、V [1]
- P型：位置型(X,Y,Z,RX,RY,RZ,FIG)（6轴）
  例）P0001、P1、P [1]
- J型：关节型(J1,J2,J3,J4,J5,J6)（6轴）
  例）J0001、J1、J [1]
- T型：齐次变换型(Px,Py,Pz,Ox,Oy,Oz,Ax,Ay,Az,FIG)
  例）T0001、T1、T [1]
- IO型：I / O型
  例）IO0001、IO1、IO [1]

注意：在视觉装置μVision-21中不能使用V型、P型、J型、T型。

全局变量的间接参照

在指定全局变量时，用算式指定变量的编号，叫做变量的间接参照。
在进行间接参照时，将变量的编号用 [ ] 括起来的算式表示。

应用示例：

```
I1 = I [5*3]        '将I15的数值赋值I1。
F1 = F [I1]         '将以I1的数值作为编号的F型变量的数值赋值F1。
D1 = D [I1+1]       '将在I1数值上加上1的数值作为编号的D型变量的数值赋值
                    D1。
                    '代入。
S1 = S [I7]         '将以I7的数值作为编号的S型变量的数值赋值S1。
V1 = V [I1]         '将以I1的数值作为编号的V型变量的数值赋值V1。
MOVE L, P [I5]      '向以I5的数值作为编号的P型变量的位置移动。
J1 = J [I5*3]       '将以I5的3倍的数值作为编号的J型变量的数值赋值J1。
T1 = T [I1*3]       '将以I1的3倍的数值作为编号的T型变量的数值赋值T1。
SET   IO [I1*5]     '将以I1的5倍的数值作为编号的比特型端口ON。
```

[2] 局部（Local）变量

在局部 (Local) 变量中，和全局变量一样，可以使用以下类型的变量。

- I型：整数型（范围：-2147483648～+2147483647）
- F型：单精度实数型 (-3.402823E + 38～3.402823E + 38)
- D型：双精度实数型 (-1.7976931348623157D + 308～1.7976931348623157D + 308)
- S型：字符串型（最大为243个字符）
- V型：矢量型(X,Y,Z)
- P型：位置型(X,Y,Z,RX,RY,RZ,FIG) （6轴）
- J型：关节型(J1,J2,J3,J4,J5,J6) （6轴）
- T型：齐次变换型(Px,Py,Pz,Ox,Oy,Oz,Ax,Ay,Az,FIG)
- IO型：I / O型

局部 (Local) 变量使用类型定义命令，在进行类型定义之后可以使用。
数值型和字符串型的局部 (Local) 变量，使用类型定义字符也能进行类型定义。
关于类型定义命令以及类型定义字符，请参照P8-6 "8.6 定义语句"。

注意①：如果不进行类型定义而使用变量，作为单精度实数型变量能够行使功能，但使用未进行类型定义的变量，也可能导致程序发生错误。请尽可能地进行类型定义。

②：在进行程序支持工具WINCAPSⅢ的默认设定中，做了 "必须要进行明确的类型定义" 的设定。在该设定中，如果省略类型定义，则在编译时就会出现错误。如果无特殊理由，请按该设定使用。

③：在程序中，如果参照I / O型以外的局部 (Local) 变量时，需要在参照之前将一些数值赋值局部 (Local) 变量（变量的初始化）。如果不经过初始化而参照局部 (Local) 变量，在执行时会发生错误。在这种情况下，要预先在程序中记述变量的赋值语、或排列变量以外在程序的变量定义部分，通过记述DEFINT A=1等，可以将变量初始化。

④：具有局部 (Local) 变量的程序，通过 CALL 指令被调出，在处于执行中、待机中、暂时停止中的状态时，如果重新启动其程序，会存在最初所调出的任务和重新启动时所调出的任务这 2 个任务。即使是其他的任务，在相同的程序中因不能确保局部 (Local) 变量的独立，所以会有不可预测的误动作。
为了防止这种误动作，通过 CALL 指令调出程序时，请不要用其单独程序进行启动。

⑤：在视觉装置 μVision-21 中，不能使用 V 型、P 型、J 型、T 型。

[3]　系统变量

系统变量是为调查系统状况的变量。变量名是系统的保留字，所以不需要定义变量名。
系统变量有以下类型。

CURJNT	CURPOS	CURTRN	CURFIG	CURACC	CURDEC	CURJDEC
CURJSPD	CURSPD	DESTJNT	DESTPOS	DESTTRN	DATE$	TIME$
TIMER	ERL	ERR				

7.5.2　函数

函数是用预先确定的运算方法对指定的数值（自变量 (argument)）求出的运算结果。根据函数的不同也存在没有自变量 (argument) 的数值。
在PAC语言的函数中，有处理姿势 (Pose) 数据以及矢量数据的函数、处理数值数据的算数函数、处理字符串数据的函数、用户定义函数的4种。
关于函数指令，请参照第15章 "函数"。

7.5.3　标签

标签是表示程序中的文的位置。可以用标签表示分支目的地位置。
如果将标签的名称命名为有意义的名称，则程序会变得易懂。

7.5.4 程序

可以使用程序的名称指定程序，从程序中调出其他的程序。
程序的名称在程序的前头用PROGRAM指令进行定义。
关于程序的名称，在 "8.2 程序名称和定义" 中进行详细说明。

程序示例：将使用该程序名称的程序调出

程序调出一侧的程序

```
PROGRAM PRO1
    IF IO [138] = ON THEN
        CALL MOTION
    ELSE
        DELAY 200
    END IF
END
```

被调出一侧的 "MOTION" 程序

```
PROGRAM MOTION
    TAKEARM
    SPEED 100
    MOVE P, P1
    DELAY 200
    MOVE P, P2
END
```

7.6　数据型

在用PAC语言处理的数据中，有字符串、数值、位置、矢量、I / O的类型。以下就这些数据类型进行说明。

7.6.1　字符串

字符串 (String) 型数据也叫做S型。
最大可以包含到243个字符的字符串。

7.6.2　数值

在数值型数据中有以下3种类型。

I型 ：整数型（范围：-2147483648～+2147483647）
F型 ：单精度实数型 (-3.402823E + 38～3.402823E + 38)
D型 ：双精度实数型 (-1.79769313486231D + 308～
1.79769313486231D + 308)

7.6.3　矢量

矢量型数据也叫做V型。
由X、Y、Z成分组成的3个单精度实数参数构成。

V型：矢量型 (X、Y、Z)

7.6.4　姿势 (Pose)

姿势 (Pose) 型数据中有以下3种类型。

P型　：位置型 (X, Y, Z, RX, RY, RZ, FIG)
J型　：关节型 (J1, J2, J3, J4, J5, J6)
T型　：齐次变换型 (Px, Py, Pz, Ox, Oy, Oz, Ax, Ay, Az, FIG)

7.6.5　I / O (ON / OFF)

I / O型的数据，是将I / O端口的状态（ON或OFF）作为数值持有。

7.7 数据型的转换

在不同的数据型之间，可以转换类型。

7.7.1 数值

数值数据可以按照以下的规则进行类型转换。

●如果将数值数据赋值不同类型的数值变量，数值会顺着变量的类型被转换。

●类型不同的数值如何进行运算，会顺着精度高的类型进行运算。

●在逻辑运算中，数值转换为整数进行运算，结果为整数。

●实数被转换为整数的情况下，被取整为不超过该实数值的最大整数。

●双精度实数被赋值单精度实数的情况下，其结果被取为7位有效数字。

7.7.2 字符和数值

在PAC语言中，有以下指令用于转换字符、字符串、字符编码、数值。

字符和数值的转换指令

转换前 \ 转换后	向字符编码转换	向字符串转换	向数值转换	2进制标示转换为字符串	16进制标示转换为字符串
字符编码		CHR$	-	-	-
字符	ASC		VAL$	-	-
数值	-	STR$		BIN$	HEX$

7.7.3 姿势 (Pose) 型数据

姿势 (Pose) 型数据可以用后面的函数进行型转换。

在姿势 (Pose) 型数据中，反映类型转换执行时的工具、工件坐标系。

姿势 (Pose) 型数据的转换指令

转换前 \ 转换后	转换为P型	转换为J型	转换为T型
P型		P2J	P2T
J型	J2P		J2T
T型	T2P	T2J	

应用示例：J0 = P2J (P0)

T0 = P2T (P0)

P0 = J2P (J0)

T0 = J2T (J0)

P0 = T2P (T0)

J0 = T2J (T0)

7.8　常量

常量是具有固定数值的公式。
PAC语言的常量可以进行以下分类。

1 数值数据	I型：	整数型常量
	F型：	单精度实数型常量
	D型：	双精度实数型常量
2 字符串数据	S型：	字符串型常量（最大243个字符）
3 矢量数据	V型：	矢量型常量 (X, Y, Z)
4 姿势(Pose)数据	P型：	位置型常量(X, Y, Z, RX, RY, RZ, FIG) （为6轴时）
	J型：	关节型常量 (J1, J2, J3, J4, J5, J6) （为6轴时）
	T型：	齐次变换型常量 (Px, Py, Pz, Ox, Oy, Oz, Ax, Ay, Az, FIG)
		（V型、P型、J型、T型的各个要素是单精度实数）

以下就各型的常量进行说明。

7.8.1　数值常量

[1]　整数型常量

从-2147483648到 +2147483647范围内的整数的常量。
在整数型常量中，有10进制、2进制、16进制的3种标示形式。

10进制形式

是用10进制数表示的整数型常量。
对-2147483648到+2147483647范围内的实数一旦加上整数型后缀符 "%"，小数点以下被舍去，被视为整数型的常量。

例：32767
　　-125
　　+10
　　3256.21% → 因有整数型后缀符，被视为3256。

2进制形式

是用2进制数表示的整数型常量。
在数值的前头加上 "&B"，用0或1的排列表示。
如果用2进制形式标示整数型常量的数值范围，
会是&B0～&B11111111111111111111111111111111
（32比特的范围内）。

例：&B110
　　&B0011

16进制形式

是用16进制数表示的整数型常量。
在数值的前头加上 "&H"，用0到F排列表示。
如果用16进制形式标示整数型常量的数值范围，
会是&H0～&HFFFFFFFF（32比特的范围内）。
例：&H110
　　&H3D5A

[2] 单精度实数型常量

是具有7位有效位精度的实数型的常量。
数值的范围是-3.402823E + 38～3.402823E + 38。
在单精度实数型常量中，有以下3种标示形式。
●最后带有单精度实数型后缀符 "！" 的数
●使用E的指数形式
●没有上述指定，7位以下的实数
例：1256.3
　　35.78!
　　-9.345E-06

[3] 双精度实数型常量

是具有15位有效位精度的实数型的常量。
数值的范围是 -1.79769313486231D + 308～1.79769313486231D + 308。
在双精度实数型常量中，有以下3种标示形式。
●在最后带有双精度实数型后缀符 "#" 的数
●使用D的指数形式
●没有上述指定，超过7位且15位以下的实数
例：1256.325468
　　35.78#
　　-9.345D-06

注意：根据单精度实数型常量和双精度实数型常量的有效位数和内部表现，有时会产生误差。

例

F1 = 10.0025
D1 = 10.0025
这种情况下，在内部被表现为1.0002499580383…E+1，往单精度实数型进行输入时，通过将有效位数的下一位（第8位）四舍五入，10.00250可正确进入。往双精度实数型进行输入时，通过将有效位数的下一位（第16位）四舍五入，10.002499580383…那样的数值可进入。

7.8.2　字符串常量

字符串型常量是表示字符串的常量。
字符串用双引号 " " 将前后括起来表示。
字符串的长度必须在243个字符以内。

例："PAC"

7.8.3　矢量型常量

矢量型常量，是由X、Y、Z成分组成,矢量表示的常量。
X、Y、Z的各个成分用单精度实数表示。

例：将矢量型常量赋值矢量型变量V1中。
V [1] = (1,0,0)

7.8.4　姿势 (Pose) 常量

[1]　位置型常量

位置型常量，是由位置 (X, Y, Z)、姿势 (Pose) (RX, RY, RZ)、形态 (FIG) 3个构成，由共计7个单精度实数参数构成。位置X、Y、Z用mm单位表示，另外，姿势 (Pose) (RX, RY, RZ) 用旋转角度 (°) 表示，形态 (FIG) 用 (不定时为-1) 编号表示。关于形态 (FIG) 的编号，请参照操作指南的 "关于手腕、肘、臂的形态" 与 "关于臂的形态"。
姿势 (Pose) 表示相对于X轴、Y轴、Z轴的旋转角度，将各自右旋转方向作为正向转动，其范围被限定为-180°～+180°。还有，因为暂停取决于轴的旋转顺序，所以其旋转顺序被规定为沿X轴、Y轴、Z轴的顺序旋转。

例：将位置型常量赋值位置型变量P1中。
P [1] = (100, 200, 300, 10, 20, 30, 0)
‘X = 100, Y = 200, Z = 300, RX = 10, RY = 20, RZ = 30, FIG= 0

[2] 关节型常量

关节型常量，是由从1轴到6轴的各轴数值构成。各轴数值用单精度实数表示。单位是角度 (°)。

例：将关节型常量赋值轴型变量J1中。

```
J [0] = (10、20、30、40、50、60)  '1～6轴：10°、20°、30°、40°、50°、60°
```

[3] 齐次变换型常量

齐次变换型常量是由位置矢量、定向矢量、近似矢量的3个矢量以及形态所表示的姿势 (Pose) 常量。

3个矢量分别用3个单精度实数型参数表示。因此，齐次变换型常量用10个单精度实数型参数表示。

位置矢量的3个要素单位是mm。

> 注意：定向矢量、近似矢量都是大小为1单位的矢量，且相互必须是垂直正交的。如果指定不为上述的矢量，通过动作命令使用时，系统就会优先近似矢量，将定向矢量自动垂直正交之后，再进行动作。

例：将齐次变换型常量赋值齐次变换型变量T1中。

```
T [1] = (10, 20, 30, 1, 0, 0, 0, 1, 0, 4)
```

7.9　式和运算符

将返回值结果值的表达式称为式。式有2种。1种是如常量及变量一样具有数据的式，另1种是通过运算符结合成多个要素而构成的式。PAC语言的所有数据型的数值通过式表示。
如果使用本节以下所说明的运算符，则可以在式中进行运算。

7.9.1　赋值运算符

向变量赋值，要在赋值语句使用赋值运算符 "=" 进行。赋值运算符的右侧的数值被赋值左侧。
在赋值语句中，根据数据的种类而使用以下的指令。

LET、LETA、LETF、LETJ、LETO、LETP、LETR、LETRX、LETRY、LETRZ、LETX、LETY、LETZ

只有用于数值变量赋值的LET指令，才能够省略指令进行记述，所以以下的2个式有相同的意义。

```
例：LET I1 = 5        '将5赋值I1中。
    I1 = 5            '将5赋值I1中。
```

7.9.2　算术运算符

在算术运算上，使用下表所示的运算符。
运算按照表中所示的优先顺序进行。

算术运算符

算术运算符	运算内容	运算的优先顺序
^	指数（幂乘）运算	高
-	负符号	↕
*、/	乘法、除法	↕
MOD	余数	↕
+、-	加法、减法	低

注意①：如果用0进行除法运算，则在执行时会发生错误。
②：在整数的加法、乘法运算时如发生溢位，溢出的部分被忽略。此时不会发生错误请予注意。

```
例：I1 = 2147483647 + 1    '2147483647为整数型变量所能取得的最大值
    如果执行上面的公式，结果会是-2147483648。
    -2147483648是整数型变量所能取得的最小值。
```

③：实数的加法、乘法运算时发生溢位的情况及想要从实数型变量赋值用整数型变量不能记录的数值时，会发生错误。
④：整数之间的除法运算，会被取整为不超过其结果的数值的最大整数。
⑤：整数的余数符号，如下表所决定。

算术运算符

除　数 被除数	+	0	-
+	+	错误	+
0	0	错误	0
-	-	错误	+

7.9.3　关系运算符

比较2个数值时，使用关系运算符。结果由布尔值（真为 "1"、伪为 "0"）取得，用于流程控制语句的条件公式等。

关系运算符

关系运算符	运算内容
=	相等
=.	几乎相等（近似比较）
<>	不相等
<	小
>	大
<=	小于或等于
>=	大于或等于

备注：近似比较运算符 (=.) 之比较精度的设定，可按照如下程序进行。
TP：［F1 程序］-［F6 辅助功能］-［PRJ设定.］ No.12
WC：［项目］-［参数］ 程序图标No.13

7.9.4　逻辑运算符

逻辑运算符进行比特运算。
整数型之外的数值，要先转换成整数型后被运算。

注意：如果指定的值超出整数型变量的有效范围时，会发生错误。

逻辑运算符

逻辑运算符	运算内容
NOT	否定
AND	逻辑积
OR	逻辑和
XOR	排他的逻辑和

例：比特运算
I1 = &B1100　XOR　&B0101
这种情况的结果为&B1001。

注意：关于 NOT 运算符

NOT 运算符因进行比特运算所以在否定公式的评价结果的情况下不能使用。

不能否定公式的评价结果的示例：

```
if NOT (I1=I2) then ...
```

I1 = I2 为（真）时 NOT 不能为（伪）。（为真。）

要否定公式的评价结果时请按以下的书写格式进行记述。

可以否定公式的评价结果的示例：

```
if (I1 = I2) = FALSE then ...
```

7.9.5　字符串运算符

字符串通过字符串运算符 "+" 可以连接。

例：A$ = "ABC" + "DEF"　　' A$为 "ABCDEF"。

7.9.6 矢量运算

在矢量运算时，使用如下表所示的运算符。
运算按照表中所示的优先顺序进行。

矢量运算符

矢量运算符	运算内容	运算的优先顺序
.、*、x	内积、纯量倍数、外积	高
+、-	加法、减法	低

例1：V1和V2的内积的计算
F1 = V1 . V2

例2：计算V1的2倍。
V2 = V1 * 2

例3：V1和V2的外积V3的计算
V3 = V1 x V2

注意：外积运算符的 x 是字母的艾克斯 (X)，其前后必须加入空格。

7.9.7 位置运算

是对于位置型的数据所进行的运算。
在进行平移、旋转偏差的运算时，只能使用 "+"。

例 1：计算从偏差的基准点 PO 开始，在机械手坐标系只偏离 (dx, dy, dz) 的点，P1。（如果是无须运算的要素，请记述为 0。）
P1 = P0 + (dx, dy, dz, 0, 0, 0)

例 2：计算从现在位置开始，在工具坐标系只偏离 (df, dg, dh) 的点，P2。（如果是无须运算的要素，请记述为 0。）
P2 = * + (df, dg, dh, 0, 0, 0) H

例 3：计算将基准点 PO 的姿势 (Pose)，围绕机械手坐标系的 X 轴、Y 轴、Z 轴分别只旋转 Rx（度）、Ry（度）、Rz（度）的点，P3。（如果是无须运算的要素，请记述为 0。）
P3 = P0 + (0, 0, 0, Rx, Ry, Rz)

例 4：计算从现在位置开始，在工具坐标系只偏离 (df, dg, dh)，围绕着常规矢量转动 Rx（度）、定向矢量周围 Ry（度）、近似矢量周围 Rz（度）的点，P4。（如果是无须运算的要素，请记述为 0。）
P4 = * + (df, dg, dh, Rx, Ry, Rz) H

注意：在偏差上不能使用位置型变量。

7.9.8　关节运算

是对关节型的数据进行的运算。
在平移偏差的运算上，只能使用 "+"。

例：计算从偏差的基准点J0开始，只在机械手坐标系 (d1, d2, d3, d4, d5, d6) 偏离的点，J1。(无须运算的轴，请记述为0。)

```
J1 = J0 + (d1, d2, d3, d4, d5, d6)
```

注意：在偏差上不能使用关节型变量。

7.9.9　齐次变换矩阵运算

是对齐次变换矩阵进行的运算。
齐次变换矩阵的积，使用运算符 "*"。

例：计算T1和T2的积，T3。

```
T3 = T1 * T2
```

此时，T3的形态被设定为T2的形态。

7.9.10　算术运算符、逻辑运算符、关系运算符的优先顺序［Ver.1.5 版本以上］

可在1行上进行算术运算符、逻辑运算符、关系运算符等复数个记述。

运算符之间的优先顺序请参照如下。

运算符	运算内容	优先顺序
^	指数（幂乘）	高
-	负符号	↑
*、/、.、x	乘法、除法、内积、外积	
MOD	余数	
+、-	加法、减法	
NOT	否定	
AND	逻辑积	
OR	逻辑和	
XOR	排他的逻辑和	↓
=、=.、<>、<、>、<=、>=	关系运算符	低

优先顺序相同的情况下从公式的左向右进行评价。用（ ）括起来的情况下先处理括弧内的运算。(优先顺序提高)

优先顺序的示例

1. if IO128 = ON and IO129 = OFF then 是按

(1)

(2)

(3)

的顺序被进行处理。

2. if (IO128 = ON) and (IO129 = OFF) then 是按

(1) (2)

(3)

的顺序被进行处理。

格式的示例

```
if (IO128 = ON) and (IO129 = OFF) then
        move   p,p1
endif
```

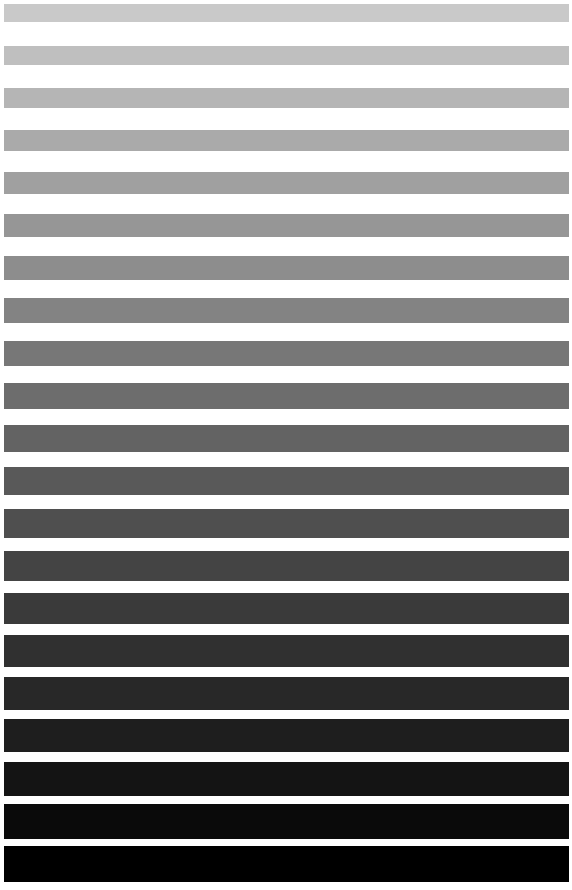
7.10 PAC语言中单位的处理

PAC语言中的各物理量的表示单位如下表所示。

PAC语言中的物理量的表示单位

物理量	单位
长　度	毫米(mm)
角　度	度(DEGREE)
时　间	毫秒(msec.)
速度	%（相对于最高速度的百分比）
加速度	%（相对于最大加速度的百分比）
减速度	%（相对于最大减速度的百分比）

第 8 章
PAC 语言的语法



对基于PAC语言进行编程时的规定进行说明。

第 8 章　PAC 语言的语法

8.1　语句与行

PAC语言的程序由若干行构成。
在任意一行都可以记述一个语句。
1行的长度最大为255字节 (Byte)。
语句是记述PAC语言处理的最小单位，由一个指令构成。
指令由指令的名称和给予指令的信息（参数）构成。

8.2　程序名称与定义

程序名称和变量等程序执行时所需要的项目，在执行之前要先进行定义。
特别是进行程序名称定义时，必须在程序的最初的有效行进行定义。将该语句叫做PROGRAM定义语句。
PROGRAM定义语句也可以省略。如果省略，则程序的文件名作为程序名使用。
例如：文件名为PRO1.PAC时，如果不用PROGRAM定义语句命名其他的名称，则程序名成为PRO1。
如果省略PROGRAM定义语句，则调出程序时，不能够提交自变量 (argument)。
另外，在编集程序过程中，难于识别程序名。若无特殊理由，请不要省略PROGRAM定义语句。
关于程序的调出，请参照 "2.1 程序的调出与子程序"。

注意①：可从多功能教导器操作启动的程序，只限于不具有自变量 (argument) 的程序。另外，可从外部机器调出启动的程序，只限于程序名称为 "PRO <编号>" 的形式的程序。不能从多功能教导器和外部机器调出启动的程序，可以从其他的程序调出。
②：调出具有 "PRO <编号>" 形式程序名称的程序时，不能具有自变量 (argument)。
③：把程序名的前面 3 个字符设为 "PRO" 时，必须要追加编号。
<编号> 是作为数字解释的，所以，即使记述为 PRO01 和 PRO001，也会作为 PRO1 进行处理。

8.3 标签

为了表示程序的分支目的地和程序中的语句位置，可以使用标签。
在使用标签时有以下的规则。

●标签的名称要以星号 (*) 开始。
●标签的名称的第2个字符，必须是任意的一个字母。
●标签的名称的第3个字符之后，可以使用英文字母和数字的任意组合。
●标签名称的最后一个字符，必须是冒号 (：)。
●不能在标签名称上使用保留字。
●被参照的标签名称必须在行的最初位置。
●被参照的标签名称有相同的若干个时，就会发生错误。
●只有在标签所在的程序中可参照标签范围。

```
PROGRAM WITH_LABEL
   IF IO138 = 1  THEN  *ACTION  ELSE  *NOACTION
   *ACTION:          SPEED 100
                     MOVE P, P1
                     DELAY 200
                     MOVE P, P2
   *NOACTION:        DELAY 200
END
```

使用标签的程序示例

8.4　字符组

在PAC语言上能够使用的字符是英文字母（字母）、数字、特殊记号。英文字母不分大写和小写。
特殊记号除了算术运算符（+, -, *, /）之外，有以下种类。

●逗号（，）
　在参数并列时，用于对参数进行区分。
●分号（；）
　在指令的自变量 (argument) 上有参数并列时，用于区分。
●单引号（'）
　作为REM指令的代用指令使用。
●双引号（"）
　将任意字符串的前后括起来，用于字符串数据的指定。
●星号（＊）
　在标签名称的前面使用。
　表示界面坐标系的当前位置。
●空格
　在命令名称前后是必须的。

注意：REM语句的备注和字符串常量上，可以包含汉字编码 (Shift JIS.)

8.5 保留字

PAC语言在对指令的名称和运算符的等进行处理的基础上，对使用方法进行确定的字句，叫做保留字。

为了在使用保留字时对保留字进行识别，在语句的前、后或两侧需要插入语句上所规定的特殊字符（空格字符、"、#、:）。

PAC语言的保留字，在附录5 "保留字一览" 上记述。

变量名称和标签上不能使用保留字本身，而包含保留字的名称可以使用。例如："HOME" 这个词是保留字，但 "HOME1" 这个名称可以用于变量名称或标签等。

8.6　定义语句

需要确定变量和常量、函数等名称和类型的，在程序使用之前，要用定义语句进行定义。
定义语句大致可以分为以下3种。

●类型定义：定义变量以及常量的类型。
●函数 / 程序定义：定义函数和程序。
●用户坐标系定义：对用户坐标系进行定义。

8.6.1　类型定义

[1]　类型定义字符（后缀符）

使用确定的后缀符，可以进行变量的类型定义。后缀符如下所示。

类型定义字符（后缀符）

种类	后缀符	使用示例
整数型后缀符	%	A%
单精度实数型后缀符	!	A!
双精度实数型后缀符	#	A#
字符串型后缀符	$	A$

注意：在程序中，变量的类型按照最初定义的类型进行处理。在程序的中途，即使使用与定义不同类型的后缀符，也不进行类型的识别。
A、A%、A!、A#、A$都作为同一个变量进行识别，按照最初使用的类型进行登录。因此，如果使用与最初使用的类型不同的类型后缀符，就会发生错误。

[2] 类型定义命令

使用以下的类型定义指令，可以定义变量的类型。

类型定义命令

种类	指令	使用示例
整数型	DEFINT	DEFINT AA,AB
单精度实数型	DEFSNG	DEFSNG BA,BB
双精度实数型	DEFDBL	DEFDBL CA,CB
字符串型	DEFSTR	DEFSTR DA,DB
矢量型	DEFVEC	DEFVEC EA,EB
位置型	DEFPOS	DEFPOS FA,FB
关节型	DEFJNT	DEFJNT GA,GB
齐次变换型	DEFTRN	DEFTRN HA,HB

在这些指令中进行类型定义的同时，可以进行变量的初始化。
在由这些指令所进行的定义语句中，不能在变量名称上追加后缀符。

使用示例：

```
DEFINT AA = 1              '将AA作为整数型赋值1。
DEFSNG BB (10)     '将BB设为要素数10的单精度实数型。
```

错误的示例：

```
DEFINT AB%          '由于使用后缀符，所以发生错误。
```

[3]　排列定义

在排列定义的定义语句中，通过在变量名称上追加后缀符，可以对I / O变量以外的所有类型进行排列。
但是，在定义时不能进行排列的初始化。
排列的追加字必须是0以上。排列的维数最大是3维。
排列的要素总数以32767为上限。

排列定义

种类	指令	使用示例
排列定义	DIM	DIM AA (10,10)

排列定义示例：

```
DIM CC (3,3,3) AS INTEGER     '将CC设为整数型的3维排列。
```

上述示例使用DEFINT，也可以进行如下表示。

```
DEFINT CC (3,3,3)             '将CC进行整数型的3维排列。
```

注意：DEF???语句和DIM语句的不同

DIM语句在不进行排列时也能使用。例如：

```
DEFINT  CC
```

变为

```
DIM  CC  AS  INTEGER
```

DIM语句比DEF???语句更通用，但不能在类型定义的同时对变量进行初始化。例如：以下的表示不能用DIM语句。

```
DEFINT  CC = 1      '将CC定义为整数型，并将初始值设为1。
```

如果在定义的同时不进行初始化，则所有的类型和排列都进行同样的处理，所以建议使用DIM语句。

[4]　I ／ 0 变量定义

将变量名称和特定的I / O端口相对应。

I ／ 0变量定义

种类	指令	使用示例
I / O变量定义	DEFIO	DEFIO PORT = BYTE,104

8.6.2 函数 / 程序定义

定义函数或程序名称的指令如下。

函数 / 程序定义

种类	指令	使用示例
函数定义	DEF FN	DEF FN AREA (R) = PI∗R∗R
程序定义	PROGRAM	PROGRAM PRO1

8.6.3 用户坐标系的定义

以下所示的指令定义用户坐标系。

用户坐标系的定义

种类	指令	使用示例
干扰区域的定义	AREA	AREA 1,P0,V0,15,6
工具坐标系定义	TOOL	TOOL 1,P0
工件坐标系定义	WORK	WORK 1,P0

8.6.4 文件夹变量定义 / EXTERN 变量定义［RC7_Ver. 2.2 以上版本］

如果用FOLDER语句定义局部 (Local) 变量，则从外部程序通过使用EXTERN语句可对局部 (Local) 变量进行参照。
对文件夹变量和EXTERN变量进行定义的指令如下所示。

文件夹功能的定义

种类	指令
文件夹变量定义	FOLDER DEFINT
	FOLDER DEFSNG
	FOLDER DEFDBL
	FOLDER DEFSTR
	FOLDER DEFVEC
	FOLDER DEFPOS
	FOLDER DEFJNT
	FOLDER DEFTRN
	FOLDER DIM
EXTERN变量定义	EXTERN DEFINT
	EXTERN DEFSNG
	EXTERN DEFDBL
	EXTERN DEFSTR
	EXTERN DEFVEC
	EXTERN DEFPOS
	EXTERN DEFJNT
	EXTERN DEFTRN
	EXTERN DIM

8.7　赋值语句

赋值语句在各个类型变量上对值进行设定。
大致分为数值赋值语句、字符串赋值语句、矢量赋值语句、姿势 (Pose) 赋值语句4种。

注意：在所有的赋值语句上都可以省略LET指令。

8.7.1　数值赋值语句

用数值赋值语句，将值赋值数值变量。

```
例：LET I [1] =10        '将10赋值I [1]。
    D [2] = 3.14         '将3.14赋值D [2]。
```

8.7.2　字符串赋值语句

字符串赋值语句是将字符串赋值字符串型变量。

```
例：S [2] ="DENSO"           '将 "DENSO" 赋值S [2]。
    LET SS$ = S [2]          '将S [2] 的值赋值SS$。
```

8.7.3　矢量赋值语句

矢量赋值语句是将值赋值矢量型变量。可以使用以下的4个指令。

矢量赋值语句

种类	指令	使用示例
矢量型赋值	LET	LET V [2] = (1,2,3)
X成分赋值	LETX	LETX V [2] = F1
Y成分赋值	LETY	LETY V [2] = F1
Z成分赋值	LETZ	LETZ V [2] = F1

8.7.4 姿势 (Pose) 赋值语句

在姿势 (Pose) 赋值语句中，有位置赋值语句、关节赋值语句、齐次变换赋值语句、初始位置赋值语句4种。

[1] 位置赋值语句

位置赋值语句是将值赋值位置型变量。可以使用以下的指令。

位置赋值语句

种类	指令	使用示例
位置型赋值	LET	LET P [3] = (10,10,10,0,0,0)
位置矢量赋值	LETP	LETP P3 = V1
旋转矢量赋值	LETR	LETR P3 = V2
X分量赋值	LETX	LETX P [3] = F1
Y分量赋值	LETY	LETY P [3] = F1
Z分量赋值	LETZ	LETZ P [3] = F1
RX分量赋值	LETRX	LETRX P [3] = F1
RY分量赋值	LETRY	LETRY P [3] = F1
RZ分量赋值	LETRZ	LETRZ P [3] = F1
形态分量赋值	LETF	LETF P [3] = I1

[2] 关节赋值语句

关节赋值语句将值赋值关节型变量。可以使用以下的指令。

关节赋值语句

种类	指令	使用示例
关节型赋值	LET	LET J [4] = (1,2,3,4,5,6) 6轴
轴分量赋值	LETJ	LETJ 1,J [4] = F1

[3]　齐次变换赋值语句

齐次变换赋值语句将值赋值齐次变换型变量。可以使用以下的指令。

齐次变换赋值语句

种类	指令	使用示例
齐次变换型赋值	LET	LET T [x] = (1,2,3,4,5,6,7,8,9,4)
位置矢量赋值	LETP	LETP T [x] = V1
定位置矢量赋值	LETO	LETO T [x] = V [1]
近似矢量赋值	LETA	LETA T [x] = V2
形态分量赋值	LETF	LETF T [x] = I1

[4]　初始位置赋值语句

初始位置赋值语句将任意的坐标值设定为初始位置。

初始位置赋值语句

种类	指令	使用示例
初始位置赋值	HOME	HOME ＊或HOME CURPOS 将当前位置设为初始位置。

8.8 流程控制语句

为了控制程序各语句的执行顺序，使用流程控制语句。
在流程控制上，为了指示程序中的语句位置，使用标签。
流程控制大致可以分为无条件分支、条件分支、选择、重复、定义完毕处理调出等5种。

8.8.1 无条件分支

用于将程序的执行移向任意一个位置时。记述GOTO指令之后继续下一个所要执行的语句标签。

例：GOTO　*LABEL1

8.8.2 条件分支

使用IF～THEN～ELSE语句或IF～ENDIF语句，根据条件是否成立，决定分支目的地。
在IF之后所记述的关系式如果是真 (TRUE (1))，则执行THEN之后的程序，如不是，则执行ELSE以后的程序。

8.8.3 选择

根据所指定式的值，选择要执行的处理。有3个指令。

在SELECT CASE语句中，在SELECT行的CASE之后加入算式，从满足该算式值的CASE行之后开始执行到下一个CASE行或END SELECT行为止，当所有的SELECT行都不相等时，将从CASE ELSE以后开始执行到END SELECT行为止。

在ON～GOSUB语句上，在ON之后设定算式，根据算式的值，移向子程序执行。

在ON～GOTO语句上，在ON之后设定算式，根据算式的值，移向标签名称执行。

8.8.4　重复

根据指定条件，控制是否进行重复。有4个指令。

在FOR～NEXT语句上，在FOR行的FOR之后加入重复的条件，直到满足该条件为止，在相应的NEXT行之间进行处理。

在DO～LOOP语句上，在WHILE或UNTIL以后设置关系式，在满足该关系式之间（WHILE时），或直到满足该关系式止，（UNTIL时），反复在DO行和LOOP行之间进行处理。

在WHILE～WEND语句上，在WHILE以后加入关系式，在满足该关系式的过程中，重复在WHILE行和WEND行之间进行处理。与DO WHILE～LOOP语句具有同样的效果。

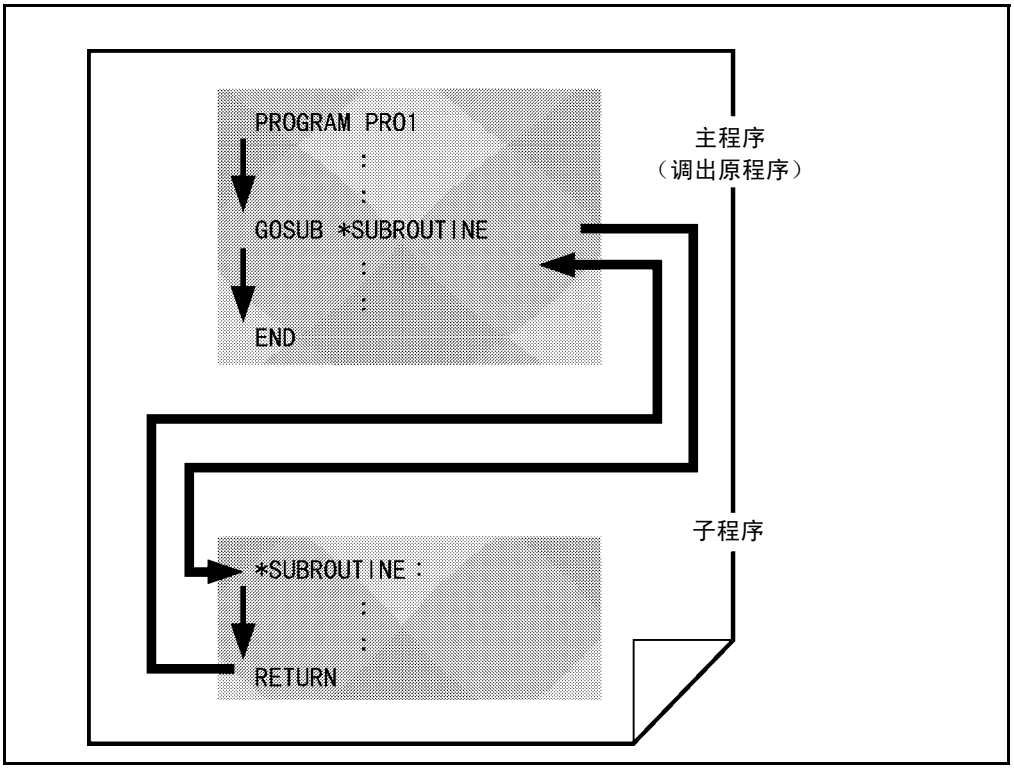
在REPEAT～UNTIL语句上，在UNTIL之后加入关系式，直到满足该关系式为止，重复REPEAT行和UNTIL行之间的处理。与DO～LOOP UNTIL语句具有相同的效果。

8.8.5　定义完成处理的调出

可以在程序中另行写出循环执行特定的动作的部分，视必要性进行调出。将该过程称之为定义完成处理的调出。
定义完成处理的调出中有子程序调出和程序调出2种。

[1]　辅助程序

要调出子程序时，在GOSUB语句使用标签指定目的文件。
子程序必须与调出的程序在相同的文件中。子程序的最后一行有RETURN语句，在执行一系列的处理之后，通过执行RETURN语句，将控制返回到调出子程序的原程序行的下一行。



子程序的调出构造

[2]　程序

在调用程序时，通过CALL语句指定程序名来执行。也可以进行复原调出。
详细内容请参照 "2.1程序的调出与子程序"。

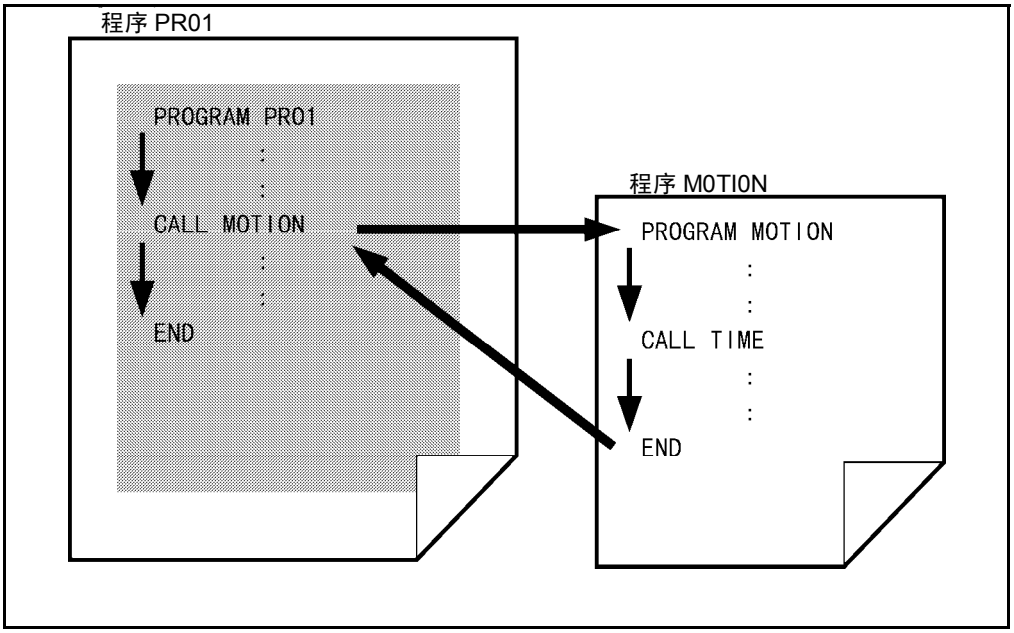


图8-1 程序的调出构造

8.9 机械手控制语句

机械手控制语句大致可以分为：动作控制语句、形态控制语句、停止控制语句、速度控制语句、时间控制语句和坐标转换语句。

8.9.1 动作控制语句

对机械手的动作进行控制的语句，称为动作控制语句。
机械手手臂的动作控制语句、夹治具控制语句和电机控制语句。

[1] 机械手手臂的控制

控制机械手手臂的动作语句的目标姿势 (Pose) 有4种情况：即P型情况、J型情况、T型情况以及P / J / T型以外数据的情况。

目标姿势 (Pose) 为P、J、T型数据

动作控制指令（目标姿势（Pose）为P、J、T型数据）

动作的类型	指令
一般移动	MOVE
接近移动	APPROACH
起程移动	DEPART

目标姿势 (Pose) 为P、J、T型以外的数据

动作控制指令（目标姿势（Pose）为P、J、T型以外的数据）

动作的类型	指令
旋转移动	ROTATE
接近方向旋转移动	ROTATEH
平移移动	DRAW
各轴相对动作	DRIVE
各轴绝对动作	DRIVEA
初始位置移动	GOHOME

8.9.2 停止控制语句

停止控制语句可使程序的执行停止或结束。

停止控制指令

动作的类型	指令	备考
执行暂时停止	HOLD	步骤停止
执行停止	HALT	瞬时停止。将自任务进行SUSPEND
执行结束	STOP	将自任务进行KILL
动作的中断	INTERRUPT	通过中断而瞬时停止

8.9.3　速度控制语句

速度控制语句设定臂的移动速度和加速度和减速度。

速度控制指令

动作的类型	指令
夹治具前端移动速度指定	SPEED
轴移动速度指定	JSPEED
夹治具前端加速度指定	ACCEL
轴加速度指定	JACCEL
夹治具前端减速度指定	DECEL
轴减速度指定	JDECEL

8.9.4　时间控制语句

时间控制语句通过动作时间对机械手进行控制。

时间控制指令

动作的类型	指令
指定时间停止	DELAY
条件停止	WAIT

8.9.5　坐标转换语句

坐标转换语句进行坐标系的切换。

坐标转换指令

动作的类型	指令
工具坐标系切换	CHANGETOOL
工件坐标系切换	CHANGEWORK
干扰检查区域的有效化	SETAREA
干扰检查区域的无效化	RESETAREA

8.10 输入输出控制语句

输入输出控制语句分为3种：DI / DO控制语句、RS232C控制语句和多功能教导器控制语句。

8.10.1 DI / DO 控制语句

DI / DO控制语句控制I / O端口的输入输出。

DI / DO控制指令

动作的类型	指令
数据读取	IN
数据写入	OUT
I / O端口ON	SET
I / O端口OFF	RESET
非动作命令并行处理	IOBLOCK

8.10.2 RS232C 控制语句

RS232C控制语句控制RS232C以及Ethernet端口的输入输出。

RS232C控制指令

动作的类型	指令
数据写入	PRINT
数据读取	INPUT
数据写入	WRITE
缓冲区清空	FLUSH

8.10.3 多功能教导器控制语句

多功能教导器控制语句对多功能教导器的输入输出进行设定。

多功能教导器控制指令

动作的类型	指令
调试画面输出	PRINTDBG
各个操作按钮定义	PRINTLBL
讯息画面输出	PRINTMSG
蜂鸣器鸣响	BUZZER
TP简易操作盘定义［Ver. 1.5以上版本］	set_button set_page change_bCap change_pCap disp_page

8.11 多项任务控制语句

多项任务控制语句包括任务控制语句和信号 (semaphore) 控制语句。

8.11.1 任务控制语句

任务控制语句控制其任务控制语句所属的任务以外的任务。

任务控制指令

动作的类型	指令
任务的生成、启动	RUN
任务的中断	SUSPEND
任务的删除	KILL
任务的保护	DEFEND

8.11.2 信号 (Semaphore) 控制语句

信号 (Semaphore) 控制语句进行信号 (semaphore) 有关的控制。

信号 (Semaphore) 控制指令

动作的类型	指令
信号 (Semaphore) 的生成	CREATESEM
信号 (Semaphore) 的删除	DELETESEM
信号 (Semaphore) 的释放	GIVESEM
信号 (Semaphore) 的获取	TAKESEM
信号 (Semaphore) 等待任务的解除	FLUSHSEM

8.11.3 专用信号 (Semaphore) 控制语句

专用信号 (semaphore) 控制语句

动作的类型	指令
臂信号 (Semaphore) 的解除	GIVEARM
臂信号 (Semaphore) 的获取	TAKEARM
视觉信号 (Semaphore) 的释放	GIVEVIS
视觉信号 (Semaphore) 的获取	TAKEVIS

8.12 时刻、日期控制

时刻日期控制语句对时刻和日期以及获取经过时间、时刻进行的中止进行控制。

时刻日期控制指令

动作的类型	指令
获取当前的日期	DATE$
获取当前的时刻	TIME$
获取经过的时间	TIMER

8.13 错误控制

8.13.1 错误控制指令

错误控制语句对错误所造成的中止进行控制。

错误控制指令

动作的类型	指令
错误中止定义	ON ERROR GOTO
错误编码	ERR
错误行	ERL
错误复位	RESUME

8.13.2 错误存放功能［Ver. 1.98 以上版本］

8.13.2.1　所谓错误存放功能

是指发生错误时，将该错误存放在I型变量空间（作为环形缓冲区使用）的功能。通过使用Ver. 1.98上所追加的SETERR命令和GETERR命令，可以定义写入环形缓冲区上的错误以及从环形缓冲区上读出的错误。
为了使用错误存放功能，需要从多功能教导器上使用扩展功能进行功能追加、设定。

8.13.2.2　关于错误存放功能的设定项目和环形缓冲区

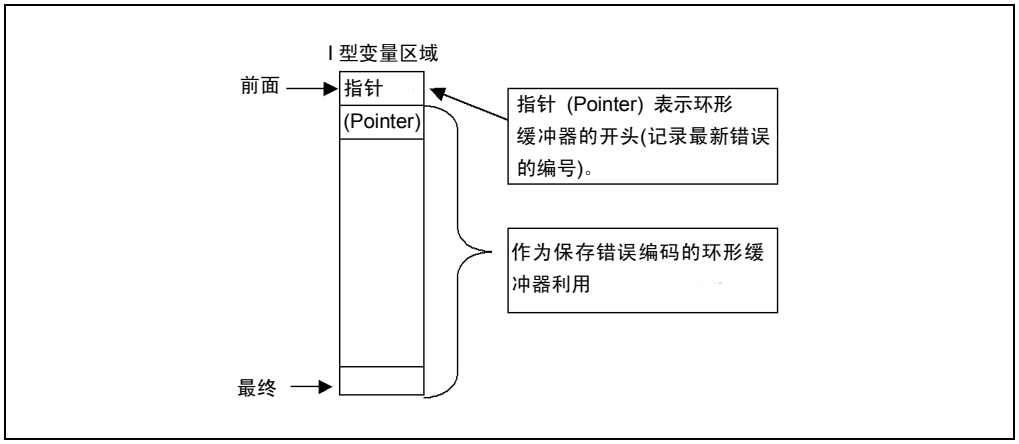
设定项目

- 功能使用 / 未使用的设定
- 设定I型变量的开头编号和最末尾编号。I型变量作为错误存放空间使用

关于环形缓冲区

关于环形缓冲区的指针 (Pointer) 位置、I型变量的开头编号和最末尾编号，请参照下图。

注：因为I型变量空间的开头存放有指针 (Pointer)，所以实际的环形缓冲区是从I型变量空间的开头编号 +1开始的。



8.13.2.3 错误存放功能的设定方法

(1) 显示多功能教导器的功能扩展画面。
操作路径：[F6 设定] — [F7 选项] — [F8 扩展功能]

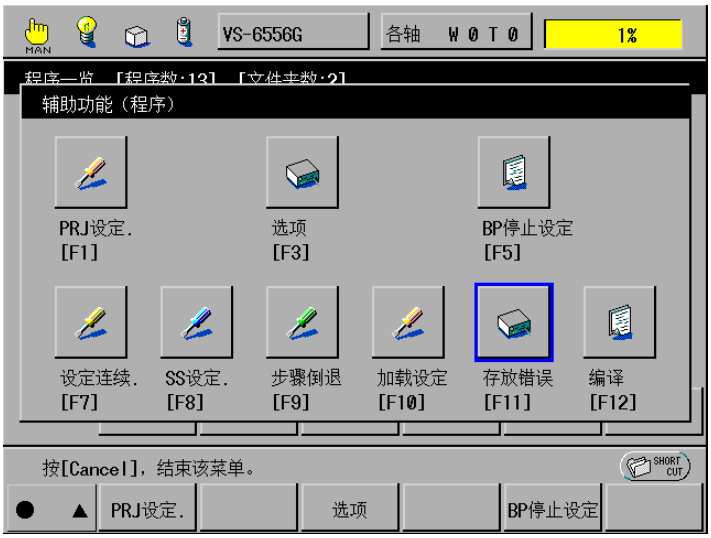
(2) 按压 [F5 追加功能]，在密码输入画面上将 "3237" 输入。



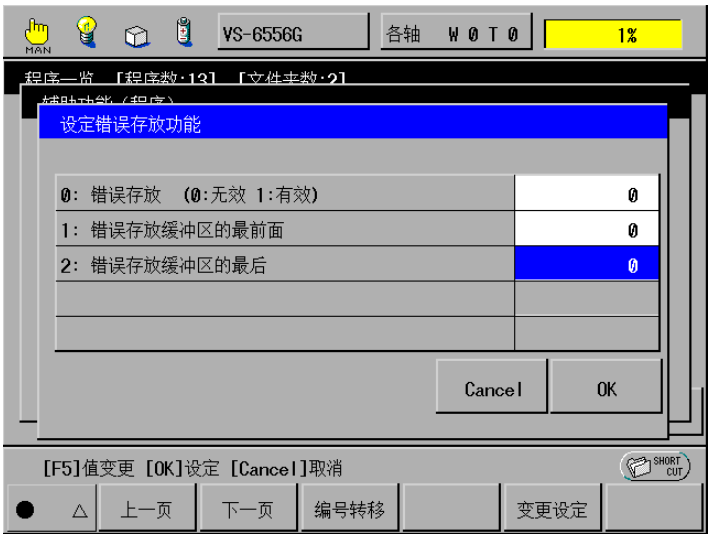
(3) 如果按压 [OK]，则追加存放错误功能。



(4) 让辅助设定画面显示，按压［F11 存放错误］按钮。
操作路径：［F1 程序］－［F6 辅助功能］－［F11 存放错误］



(5) 设定错误存放功能画面被显示。



在此，对错误存放功能进行设定。

项目	设定内容
0：错误存放	如果使用错误存放功能，则设定 "1"
1：错误存放缓冲区的最前面	设定成为错误存放空间开头的I型变量编号
2：错误存放缓冲区的最后	设定成为错误存放空间最末尾的I型变量编号

8.13.2.4　错误存放功能相关的指令

关于SETERR、GETERR、CLRERR、GETERRLVL，请参照 "第18章 错误控制"。

8.14 系统信息

系统信息通过以下所示的指令获得。

系统信息指令

动作的类型	指令
系统的环境设定值的获取	GETENV
系统的环境设定值的赋值	LETENV
ROM版本信息的获取	VER$
获取程序的状态	STATUS

8.15 预处理程序

预处理器语句将程序翻译（编译）成执行形式时，控制字符串的置换或文件的获取。

预处理器指令

动作的类型	指令
将常量、宏名称置换为字符串	#define
#define语句的取消	#undef
文件的获取	#include
模拟的错误的发生	#error
指定程序的最佳化	#pragma optimize

8.16 根据值进行调出与根据参照进行调出

从程序上调出其他的程序时，有时需要将数据提交给程序。在这种情况下，通过在调出的程序名上添加自变量 (argument) 列表，可以将数据提交。自变量 (argument) 的提交方法有两种，一种是直接将值提交的 "根据值进行调出"，另一种是提交参数的 "根据参照进行调出"。

在调出程序的命令中，有CALL和RUN。在CALL指令中可以使用两种调出方法，而在RUN指令中，只能使用 "根据值进行的调出"。

提交给程序的自变量 (argument) 的类型，必须与被调程序上所记述的自变量 (argument) 类型相一致。因此，在提交常量时，需要添加类型定义字符。还有，在被调程序中，自变量 (argument) 是作为局部 (Local) 变量被定义的，所以需要在自变量 (argument) 名上添加类型定义字符。

8.16.1 根据值进行调出

常量和式和字符式必须用数值提交。如果变量也用括弧（ ）括起来，则被视为式，可以用值提交。

例如：调出程序PROGRAM SUB1 (AA#) 的方法

- ●用数值提交变量时　　CALL SUB1 ((D1))
- ●提交常量时　　CALL SUB1 (10#)
- ●提交算式时　　CALL SUB1 (D1 + D2)

注意：① 用值提交变量时，不能根据排列全体的值进行调出。根据排列全体的值进行调出时，作为整个排列的参照进行调出。
② 在提交算式时，请注意式结果的类型。必须与被调程序的自变量 (argument) 类型相同。

8.16.2 根据参照进行调出

将全局变量以及局部 (Local) 变量作为自变量 (argument) 提交。
作为自变量 (argument) 指定全体排列时，要在排列名上添加括弧（）。
全局变量、局部排列的1要素，不能用参照进行提交。用参照提交全局变量、局部排列的1要素时，先将值赋值到局部 (Local) 变量，通过其局部 (Local) 变量进行参照调出。

例1：调出程序PROGRAM SUB1 (AA#) 时

●提交局部 (Local) 变量时1（用DEFDBL对DD已经定义时）

```
CALL SUB1 (DD#)
```

●提交局部 (Local) 变量时2（用DEFDBL对DA已经定义时）

```
CALL SUB1 (DA)
```

注意：局部 (Local) 变量需要预先赋值。在以下情况时，用值进行提交。

```
CALL SUB1 (D1)
```

例2：调出程序PROGRAM SUB2 (BB% (10)) 的情况下

●提交整个排列时（用DIM对AB% (10) 已经定义时）

```
CALL SUB2 (AB%())
```

8.17 视觉控制

8.17.1 图像输入输出

以下所示的指令控制照相机的影像和存储器内的图像数据的输入输出。

图像输入输出控制指令

动作的类型	指令
将照相机存放在存储器	CAMIN
设定照相机影像存放的功能	CAMMODE
设定所要输入的照相机影像的界限值	CAMLEVEL
将照相机影像显示在监视器上	VISCAMOUT
在监视器上显示存储器的图像	VISPLNOUT
将描绘画面信息在监视器上显示	VISOVERLAY
一览表的数据设定	VISDEFTABLE
一览表的数据参照	VISREFTABLE

8.17.2 视窗设定

以下所示的指令进行视窗（进行图像处理的范围）的编集。

视窗设定指令

动作的类型	指令
视窗信息的设定	WINDMAKE
视窗信息的清空	WINDCLR
视窗信息的复制	WINDCOPY
视窗信息的参照	WIMDREF
视窗的描绘	WINDDISP

8.17.3 描绘

以下所示的指令控制存放存储器（处理画面）、覆盖存储器（描绘专用画面）的描绘动作。

描绘指令

动作的类型	指令
描绘目标画面的指定	VISSCREEN
描绘时的辉度指定	VISBRIGHT
画面的取消	VISCLS
点的描绘	VISPUTP
由长度、角度指定进行的直线描绘	VISLINE
连结2点的直线的描绘	VISPTP
矩形的描绘	VISRECT
圆的描绘	VISCIRCLE
椭圆的描绘	VISELLIPSE
扇形的描绘	VISSECT
十字标签的描绘	VISCROSS
字符的显示位置指定	VISLOC
描绘字符的设定	VISDEFCHAR
字符的显示	VISPRINT

8.17.4 图像处理

以下所示的指令进行图像数据的处理。

图像处理指令

动作的类型	指令
处理对象画面的指定	VISWORKPLN
指定坐标的辉度的获取	VISGETP
直方柱状体的测量	VISHIST
直方柱状体结果的参照	VISREFHIST
2值化基准值的计算	VISLEVEL
2值化处理	VISBINA
2值化显示	VISBINAR
过滤器处理	VISFILTER
图像间运算	VISMASK
画面的复制	VISCOPY
面积、重心、主轴角的测量	VISMEASURE
投影数据的测量	VISPROJ
边缘的测量	VISEDGE

8.17.5 编码识别

以下所示的指令进行QR编码的读取。

编码识别指令

动作的类型	指令
'QR编码的读取	VISREADQR

8.17.6 标注

以下所示的指令进行标签的处理。

标注指令

动作的类型	指令
标注的执行	BLOB
不同标签编号的特征测量	BLOBMEASURE
标签编号的获取	BLOBLABEL
标签图像的复制	BLOBCOPY

8.17.7 检索功能

以下所示的指令进行画像模型的登录和检索。

检索功能指令

动作的类型	指令
检索模型的登录	SHDEFMODEL
登录模型的信息获取	SHREFMODEL
登录模型的复制	SHCOPYMODEL
登录模型的取消	SHCLRMODEL
登录模型的显示	SHDISPMODEL
模型的检索	SHMODEL
棱角检索条件设定	SHDEFCORNER
棱角的检索	SHCORNER
圆检索条件设定	SHDEFCIRCLE
圆的检索	SHCIRCLE

8.17.8 结果获取

以下所示的指令，从图像处理结果内容获取各种信息。

结果获取指令

动作的类型	指令
图像处理结果的获取	VISGETNUM
编码识别结果的获取	VISGETSTR
图像处理结果X坐标的获取	VISPOSX
图像处理结果Y坐标的获取	VISPOSY
处理结果状态的获取	VISSTATUS

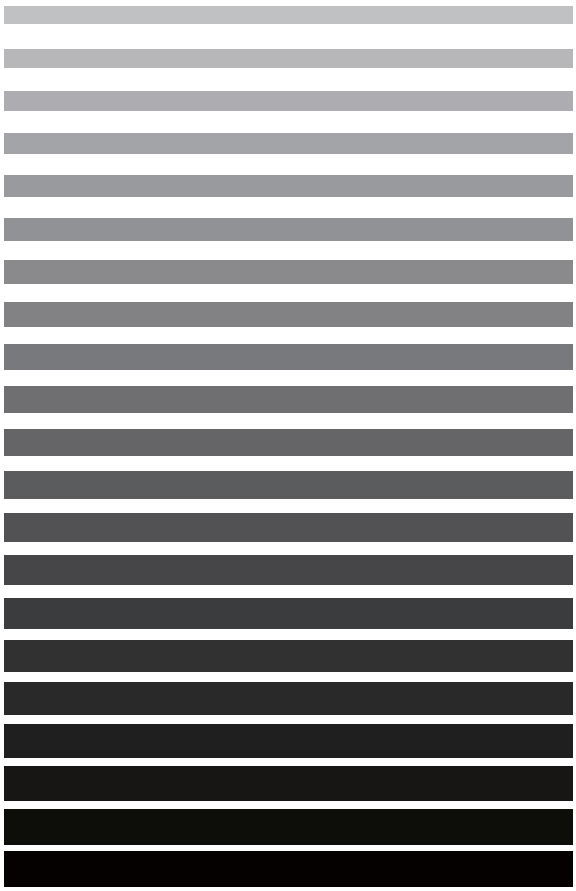
8.17.9 视觉 CAL

以下所示的指令获取CAL（视觉－机械手坐标转换）数据。

视觉CAL指令

动作的类型	指令
坐标转换数据的获取	VISREFCAL

第 9 章
定义语句



在程序中使用变量和函数的情况下，需要预先用定义语句进行定义。在本章所介绍的定义语句指令就是为了定义程序中的变量和函数。
系统变量与内置函数无需经过定义即可在程序中使用。

第 9 章　定义语句

9.1　程序名

PROGRAM（语句）

功　能

定义程序名。

格　式

PROGRAM< 程序名 >[(< 自变量 (argument)>[,< 自变量 (argument)>…])]

说　明

将在 < 程序名 > 上指定的字符串作为程序名进行定义。

程序名的定义在程序的开始行进行。若前面无程序名的定义，文件名将成为程序名。

程序名若为 PRO < 编号 > 形式的名称，则不能添加自变量 (argument)。

程序名前面的字符必须是字母。程序名最多为 64 个字符。

< 自变量 (argument)> 是调出侧提交的自变量 (argument) 数据。变量作为自变量 (argument) 的形式接受来自于 CALL 语句的提交时，如果变更在 PROGRAM 语句内用 < 自变量 (argument)> 所指定的变量值，则调出侧指定为 < 自变量 (argument)> 的变量值也随之变化。

< 自变量 (argument)> 的类型必须与 CALL 语句的自变量 (argument) 类型相同。

类型采用后缀符或 AS 表达式如下所示进行。

```
PROGRAM  SUB0(aa%,bb!)
PROGRAM  SUB0(aa AS INTEGER,bb AS SINGLE)
```

在 < 自变量 (argument)> 中使用排列变量时，输入调出侧 DIM 语句定义的排列下标字符的最大值和数量。

< 自变量 (argument)> 的最大个数为 32 个。

只有 PRO< 编号 > 的程序可以从外部启动。

相关项目

CALL

应用示例

```
PROGRAM PRO1                        ' 将 PRO1 作为程序名进行定义。
PROGRAM SUB2( la, lb%, lc# )        ' 将具有 la、 lb%、 lc# 的自变量 (argument)
                                    'SUB2 作为程序进行定义。
PROGRAM SUB3( lb%( 12, 5 ) )        ' 将 SUB3 作为程序进行定义，以 2 维排列 lb%
                                    ' 接受自变量 (argument)。在该示例中，排列的下标字符为
                                    12,5。

PROGRAM SUB0( la%, lb! )
PROGRAM SUB0( la As Integer, lb As Single )
```

9.2 干扰区域坐标

AREA（语句）

功　能

定义进行干扰检查的区域。

格　式

AREA < 区域编号 >、< 位置 >、< 矢量 >、<I / O 编号 >、< 干扰位置存放位置型变量编号 >[,< 错误查出设定 >]

说　明

定义干扰检查区域。

在 < 区域编号 > 中能够定义的区域编号是 0 ～ 31 的 32 个位置。

< 位置 > 是指定干扰检查区域的中心位置和角度的位置。

< 矢量 > 是指定干扰检查区域的范围的矢量。

区域的一边的长度为 < 矢量 > 的各个成分的 2 倍的长度。

在干扰检查区域发生干扰时，将设定的 I / O 编号赋予 <I / O 编号 >。I / O 的状态一直保持在执行 RESETAREA 或将其 I / O 进行 RESET 为止。

在［Ver.1.8 以上版本］中，可以做到 10104、 IO [104] 和 IO 变量的表述。另外，通过设定 -1 可以做到不进行 IO 输出的设定。

将存放干扰位置的位置型变量的编号赋予 < 干扰位置存放位置型变量编号 >。

在［Ver.1.8 以上版本］中，可以做到 P55、 P [55] 和 P 型变量的表述。另外，如果设定 -1 可以做到不向 P 型变量进行位置输入的设定。

在 < 错误查出设定 > 中，设定位置干扰查出的错误条件。干扰时的错误设定如下所示。

0: 在空间内部干扰时，没有错误输出
1: 在空间内部干扰时，输出错误
2: 在空间内部干扰时，输出错误 （切换为手动后可以操作）
3: 在空间外部干扰时，没有错误输出
4: 在空间外部干扰时，输出错误
5: 在空间外部干扰时，输出错误 （切换为手动后可以操作）

干扰检查对指定为工具坐标系的原点和干扰检查区域的长方体进行比较。如果在干扰检查区域的内侧查出到工具坐标原点，则被判定为干扰。

在干扰位置存放方位型变量编号中，查出到干扰时 （I / O 被 SET 时），存放用户坐标系中的工具坐标原点位置。一般情况下为长方体面上的点，而在执行 SETAREA 时工具坐标原点位于长方体内部时，则就是该位置。

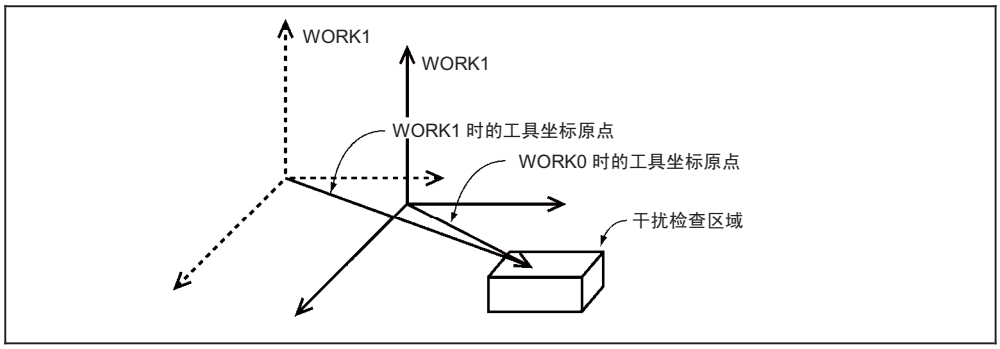
使用 WINCAPS Ⅲ、多功能教导器也可以进行设定。

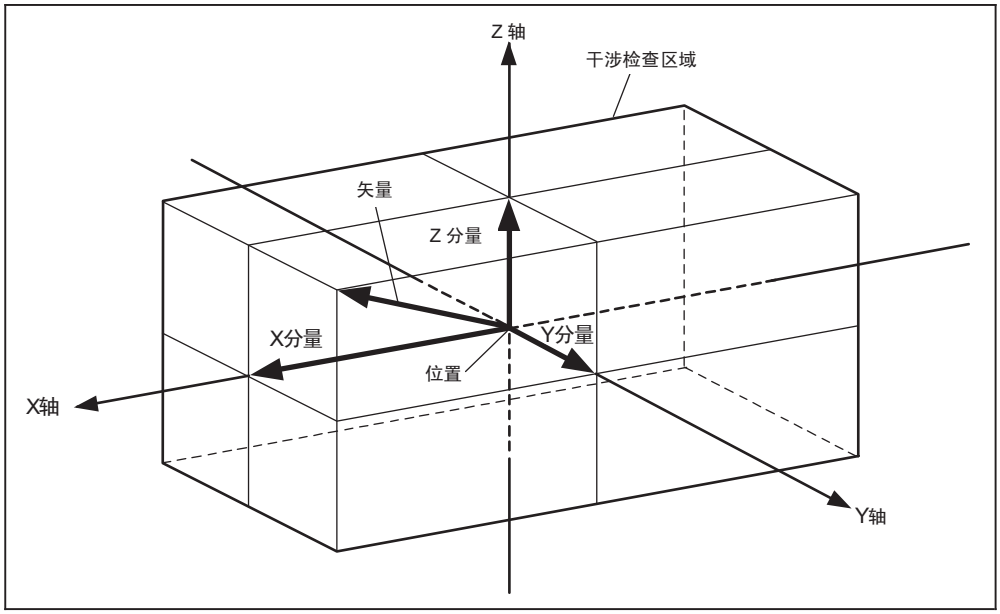
相关项目

SETAREA、 RESETAREA、 AREAPOS、 AREASIZE

注意事项

- 区域的中心位置总是以 WORK0 为基准。
- 即使变更用户坐标系，干扰检查区域的位置也不会改变。





应用示例

共用

```
AREA 2, P50, V10, 104, 55          ' 在 2 号上用 P50、 V10 规定的区域进行定义。
SETAREA 2                          ' 将 2 号的区域检查置为有效。
RESETAREA 2                        ' 将 2 号的区域检查置为无效。
```

6 轴

```
AREA 2, P50+(100, 100, 0, 10, 0, 0 ), V10, 104, 55
                                   ' 在 2 号上用 P50 + (100、 100、 0、 10、 0、 0)、
                                   'V50 所规定的区域进行定义。
SETAREA 2                          ' 将 2 号的区域检查置为有效。
RESETAREA 2                        ' 将 2 号的区域检查置为无效。
```

4 轴

```
AREA 2, P50+(100, 100, 0, 10 ), V10, 104, 55
                                   ' 在 2 号上用 P50 + (100、 100、 0、 10)、
                                   'V50 所规定的区域进行定义。
SETAREA 2                          ' 将 2 号的区域检查置为有效。
RESETAREA 2                        ' 将 2 号的区域检查置为无效。
```

POSTUREAREA（语句）【Ver.3.0 以上版本】

功　能

定义高感度位置姿势检测功能的设定。

格　式

POSTUREAREA< 位置 >，< 范围（矢量）>，<I/O 编号 >，< 检测位置存放位置型变量编号 >[,< 检测时的错误设定 >][,TAR_POSTURE(< 姿势 >，< 误差范围 >)][,TAR_EXJOINT(< 轴编号 >，< 位置 >，< 误差范围 >)]

当指定附加轴有 2 轴以上时，

POSTUREAREA< 位置 >，< 范围（矢量）>，<I/O 编号 >，< 检测位置存放位置型变量编号 >[,< 检测时的错误设定 >][,TAR_POSTURE(< 姿势 >，< 误差范围 >)][,TAR_EXJOINT((< 轴编号 >，< 位置 >，< 误差范围 >),（< 轴编号 >，< 位置 >，< 误差范围 >）[,（< 轴编号 >，< 位置 >，< 误差范围 >）…])

说　明

定义高感度位置姿势检测功能的位置 （范围）、姿势、附加轴角度。

当指定的位置、姿势、附加轴角度全部在范围内时，可同时检测位置与姿势，因此，可作为照相机快门触发器使用。

高感度位置姿势检测功能可间隔 1ms 检测一次。

< 位置 >：用位置型变量指定位置。详情请参照干涉区域指令 （AREA）。

< 范围 （矢量） > 指定检测与判断的范围。利用从中心至端点的矢量指定区域的大小。详情请参照干涉区域指令 （AREA）。

<I/O 编号 >：检测时，指定 ON 状态的 I/O 编号。如不使用 I/O，请输入 -1。

< 检测位置存放位置型变量编号 >: 在检测时，可将现在的位置存放为位置型变量。请指定位置型变量。如不存放为变量，请输入 -1。

< 检测时的错误设定 >: 设定检测时提示错误的方法。

0：当进入检测区域时，没有错误输出。
1：当进入检测区域时，输出错误。
2：当进入检测区域时，没有错误输出。（可切换成手动操作）
3：当离开检测区域时，没有错误输出。
4：当离开检测区域时，输出错误。
5：当离开检测区域时，没有错误输出。（可切换成手动操作）

TAR_POSTURE （< 姿势 >，< 误差范围 >）：指定检测姿势 （可省略）。

< 姿势 >：指定检测姿势。当利用位置型变量指定时，位置的要素将被忽视。当利用数值指定时，用（Rx，Ry，Rz）指定。如果是 4 轴机械手，则利用 （Rz）或 （0,0,Rz）指定。

< 误差范围 >: 指定检测姿势的范围。利用角度指定其与 < 姿势 > 的误差量。如指定为 -1，则不检测姿势。

TAR_EXJOINT （< 轴编号 >，< 位置 >，< 误差范围 >）: 指定所检测的附加轴角度 （可省略）。

< 轴编号 >：指定所检测附加轴的轴编号。

< 位置 >：指定附加轴的检测位置。

< 误差范围 >：指定检测位置的容许误差。如指定为 -1，则不检查附加轴位置。

相关项目

SETPOSTUREAREA、 RESETPOSTUREAREA

注意事项

- 高感度位置姿势检测功能是一个可选功能。请购买功能许可证。
- 可设定的区域只有一个。
- 仅可利用程序进行设定。无法用 TP 以及 WINCAPS 进行设定。
- 一旦将电源置于 OFF，设定将无法保持。
- 当电机置于 OFF 时，此功能无法发挥作用，但在锁定时，利用 8ms 即可检测到。

应用示例

```
POSTUREAREA, P1, V0, 24, -1, 0, TAR_POSTURE(V2, 1), TAR_EXJOINT((7, 30, 4),
(8, 90, 1))
```

9.3 用户函数

DEF FN（语句）

功　能

定义用户定义函数。

格　式

DEF FN < 函数名 >[< 后缀符 >] = < 常量 >

DEF FN < 函数名 >[< 后缀符 >](< 自变量 (argument)>[,< 自变量 (argument)>…]) = < 算式 >

说　明

将以 FN 开始的 < 函数名 > 作为用户定义函数进行定义。

将 < 算式 > 中使用的变量名在 < 自变量 (argument)> 上进行指定。

< 后缀符 > 可以省略，但通过添加 < 后缀符 > 可以同时定义变量的类型。后缀符可以使用以下的字符。

整数型后缀符 : %
单精度型后缀符 : !
双精度型后缀符 : #
字符串型后缀符 : $

如果省略后缀符，则被视为单精度实数型。

对变量名相同，类型不同的变量不能定义。

应用示例

```
DEF FND$ = "DENSO"                          ' 将 FND$ 作为用户定义函数进行定义。
DEF FNLAP#(radius) = 2 * PI * radius        ' 将 FNLAP# (radius) 作为双精度实数型的
                                            ' 用户定义函数进行定义。
DEF FNAREA(radius) = PI * POW(radius, 2)    ' 将 FNAREA (radius) 作为单精度实数型的
                                            ' 用户定义函数进行定义。
PRINT #1, FND$                              ' 从 ch1 输出 "DENSO"。
PRINT #2, HANKEI
PRINT #1, FNLAP#(HANKEI)                    ' 从 ch1 输出 (2 * PI * HANKEI) 的值。
PRINT #2, FNAREA(HANKEI)                    ' 从 ch2 输出 (PI * POW(HANKEI, 2) 的值。
```

9.4 初始坐标

HOME（语句）

功　能

将任意的坐标作为初始位置定义。

格　式

HOME < 位置型 >

说　明

定义以 < 位置型 > 指定的任意坐标为初始位置。

初始位置在各个程序中定义。

HOME 文被多次定义时 , 以最新的定义作为初始位置。

相关项目

GOHOME

应用示例

6 轴

```
DIM lp1 As position
HOME (350, 0, 450, 0, 0, 180 )      ' 将 (350, 0, 450, 0, 0, 180) 的坐标
                                    ' 定义为初始位置。
HOME lp1                            ' 将 lp1 的坐标定义为初始位置。
```

4 轴

```
DIM lp1 As position
HOME (200, 300, 300, 45, 0 )        ' 将 (200, 300, 300, 45, 0) 的坐标
                                    ' 定义为初始位置。
HOME lp1                            ' 将 lp1 的坐标定义为初始位置。
```

9.5 工具坐标

TOOL（语句）

功　能

定义工具坐标系。

格　式

TOOL < 工具坐标系编号 >， < 位置型 >

说　明

将 < 位置型 > 中所表示的位置作为 < 工具坐标系编号 > 中所表示的工具坐标系进行定义。

在工具坐标系编号中可以指定的范围为 1 ～ 63。

相关项目

CHANGETOOL、 TOOLPOS

注意事项

只要电源处于接通状态，用该指令所变更的值就会被保持。即使切断电源也需要保持该值时，请进行系统参数的保存 （参照操作指南 P5-159）。

应用示例

6 轴

```
DIM lp1 As position
TOOL 1, lp1                          ' 将 lp1 作为工具坐标系编号 1 的工具坐标系进行定义。

TOOL 2, (100, 100, 50, 0, 90, 0 )
```

4 轴

```
DIM lp1 As position
TOOL 1, lp1                          ' 将 lp1 作为工具坐标系编号 1 的工具坐标系进行定义。

TOOL 2, (100, 100, 50, 0 )
```

9.6 工件坐标

WORK（语句）

功　能

定义用户坐标系。

格　式

WORK < 用户坐标系编号 >， < 位置型 >[,< 属性 >]

说　明

将赋值到 < 位置型 > 中所表示的位置型变量的用户坐标作为 < 用户坐标系编号 > 中所表示的用户坐标系进行定义。

在用户坐标系编号中，可使用 1 ～ 7。

在 < 属性 > 中给工件坐标附加属性。（0：标准、 1：固定工具）省略时，将输入 0。（Ver.3.0 以上版本）

相关项目

CHANGEWORK、 WORKPOS

注意事项

- 在工件坐标的 X、Y、Z 要素上设定的数值超过机械手的作业区域时，会因数值处理的位数丢失而发生机械手的位置偏离。工件坐标的 X、 Y、 Z 要素请使用机械手的作业区域内的值。
- 只要电源处于接通状态，用该指令所变更的值就会被保持。如果希望电源切断时也能保持数值，请设定变更使用条件中的 "TOOL·WORK 返回条件 "。
- 即使利用 "WORK" 指令变更现在所选择编号的工件坐标系值，机械手的工件坐标系也不会改变。请重新选择 "CHANGEWORK"。

应用示例

6 轴

```
DEFINT li1, li2
WORK 1, P1                          ' 将赋值到用户坐标系编号 1、位置型变量编号 2 的
                                    ' 坐标作为用户坐标系进行定义。
WORK li1, P[li2]                    ' 将赋值到 li1 所表示的用户坐标系编号、
                                    'li2 所表示的位置型变量的坐标作为用户坐标系
                                    ' 进行定义。
WORK li1, (100, 100, 50, 0, 0, 90)
WORK li1, P[li2], 1                 ' 在工件坐标系 li1 上，将 P[li2] 设定为固定工具的工件坐标。
WORK li1, (100, 100, 50, 0, 0, 90) , 1  ' 在工件坐标系 li1 上，设定固定工具的工件坐标。
```

4 轴

```
DEFINT li1, li2
WORK 1, P1                          ' 将赋值到用户坐标系编号 1、位置型变量编号 2 的
                                    ' 坐标作为用户坐标系进行定义。
WORK li1, P[li2]                    ' 将赋值到 li1 所表示的用户坐标系编号、
                                    'li2 所表示的位置型变量的坐标作为用户坐标系
                                    ' 进行定义。
WORK li1, (100, 100, 50, 0)
WORK li1, P[li2], 1                 ' 在工件坐标系 li1 上，将 P[li2] 设定为固定工具的工件坐标。
WORK li1, (100, 100, 50, 90) , 1    ' 在工件坐标系 li1 上，设定固定工具的工件坐标。
```

9.7 局部 (Local) 变量

DEFINT（语句）

功　能

定义 I 型变量 （整数型变量）。整数的范围是 –2147483648 ～ 2147483647。

格　式

DEFINT < 变量名 >[=< 常量 >][,< 变量名 >[=< 常量 >]…]

说　明

将在 < 变量名 > 上指定的变量作为整数型的变量进行定义。通过在 < 变量名 > 之后接着写等号和常量，可以在定义的同时进行初始化。

通过用 "," 断开，可以一次定义多个变量名。

相关项目

DEFDBL、 DEFSNG、 DEFSTR

应用示例

```
DEFINT lix, liy, liz          ' 将 lix, liy, liz 作为 I 型变量进行定义。
DEFINT lix = 1                ' 将 lix 作为 I 型变量进行定义，赋予初始值 = 1。
```

DEFSNG（语句）

功　能

定义 F 型变量 （单精度实数型变量）。单精度实数的范围是 -3.4E + 38 ～ 3.4E + 38。

格　式

DEFSNG < 变量名 >[=< 常量 >][,< 变量名 >[=< 常量 >]…]

说　明

将在 < 变量名 > 上指定的变量作为单精度实数型进行定义。通过在 < 变量名 > 之后接着写等号和常量，可以在定义的同时进行初始化。

通过用 "," 断开，可以一次指定多个变量名。

相关项目

DEFDBL、 DEFINT、 DEFSTR

应用示例

```
DEFSNG lfx, lfy, lfz          ' 将 lfx, lfy, lfz 作为 F 型变量进行定义。
DEFSNG lfx = 1.0              ' 将 lfx 作为 F 型变量进行定义，赋予初始值 = 1.0。
```

DEFDBL（语句）

功　能

定义 D 型变量 （双精度实数型变量）。双精度实数的范围是 -1.7D + 308 ～ 1.7D + 308。

格　式

DEFDBL < 变量名 >[=< 常量 >][,< 变量名 >[=< 常量 >]…]

说　明

将在 < 变量名 > 上指定的变量作为双精度实数型进行定义。通过在 < 变量名 > 之后接着写等号和常量，可以在定义的同时进行初始化。

通过用 "," 断开，可以一次指定多个变量名。

相关项目

DEFINT、 DEFSNG、 DEFSTR

应用示例

```
DEFDBL ldx, ldy, ldz        ' 将 ldx, ldy, ldz 作为 D 型变量进行定义。
DEFDBL ldx = 1.0            ' 将 ldx 作为 D 型变量进行定义，赋予初始值 = 1.0。
```

DEFSTR（语句）

功　能

定义 S 型变量 （字符串型变量）。字符串的长度为最大 243 个字符。

格　式

DEFSTR < 变量名 >[=< 常量 >][,< 变量名 >[=< 常量 >]…]

说　明

将在 < 变量名 > 上指定的变量作为字符串型进行定义。通过在 < 变量名 > 之后接着写等号和常量，可以在定义的同时进行初始化。

通过用 "," 断开，可以一次指定多个变量名。

相关项目

DEFDBL、 DEFINT、 DEFSNG

应用示例

```
DEFSTR lsx, lsy, lsz        ' 将 lsx, lsy, lsz 作为 S 型变量进行定义。
DEFSTR lsx = "DENSO"        ' 将 lsx 作为 S 型变量进行定义，赋予初始值 = "DENSO"。
```

DEFVEC（语句）

功　　能

定义 V 型变量（矢量型变量）。

格　　式

DEFVEC < 变量名 >[=< 矢量型常量 >][,< 变量名 >[=< 矢量型常量 >]…]

说　　明

将在 < 变量名 > 上指定的变量作为矢量型进行定义。通过在 < 变量名 > 之后接着写等号和矢量型常量，可以在定义的同时进行初始化。

通过用 "," 断开，可以一次指定多个变量名。

相关项目

DEFJNT、 DEFPOS、 DEFTRN

应用示例

```
DEFVEC lvx, lvy, lvz          ' 将 lvx, lvy, lvz 作为 V 型变量进行定义。
DEFVEC lvx = (10, 10, 5)      ' 将 lvx 作为 V 型变量进行定义，赋予初始值 = (10, 10, 5)。
```

DEFPOS（语句）

功　能

定义 P 型变量 （位置型变量）。

格　式

DEFPOS < 变量名 >[=< 位置型常量 >][,< 变量名 >[=< 位置型常量 >]…]

说　明

将在 < 变量名 > 上指定的变量作为位置型进行定义。通过在 < 变量名 > 之后接着写等号和位置型常量，可以在定义的同时进行初始化。

通过用 "," 断开，可以一次指定多个变量名。

相关项目

DEFJNT、 DEFTRN、 DEFVEC

应用示例

6 轴

```
DEFPOS lpx, lpy, lpz                    ' 将 lpx, lpy, lpz 作为 P 型变量进行定义。
DEFPOS lpx = (10, 10, 5, 0, 9, 0, 1)    ' 将 lpx 作为 P 型变量进行定义，赋予初始值 =
                                        '(10, 10, 5, 0, 9, 0, 1)。
```

4 轴

```
DEFPOS lpx, lpy, lpz                    ' 将 lpx, lpy, lpz 作为 P 型变量进行定义。
DEFPOS lpx = (100, 100, 300, 45, 0)     ' 将 lpx 作为 P 型变量进行定义，赋予初始值 =
                                        '(100, 100, 300, 45, 0)。
```

DEFJNT（语句）

功　能

定义 J 型变量（关节型变量）。

格　式

DEFJNT < 变量名 >[=< 关节型常量 >][,< 变量名 >[=< 关节型常量 >]…]

说　明

将在 < 变量名 > 上指定的变量作为关节型进行定义。通过在 < 变量名 > 之后接着写等号和关节型常量，可以在定义的同时进行初始化。

通过用 "," 断开，可以一次指定多个变量名。

相关项目

DEFPOS、 DEFTRN、 DEFVEC

应用示例

6 轴

```
DEFJNT ljx, ljy, ljz                ' 将 ljx, ljy, ljz 作为 J 型变量进行定义。
DEFJNT ljx = (10, 10, 5, 0, 9, 0)   ' 将 ljx 作为 J 型变量进行定义，赋予初始值 =
                                    '(10, 10, 5, 0, 9, 0)。
```

4 轴

```
DEFJNT ljx, ljy, ljz                ' 将 ljx, ljy, ljz 作为 J 型变量进行定义。
DEFJNT ljx = (10, 10, 5, 0)         ' 将 ljx 作为 J 型变量进行定义，赋予初始值 =
                                    '(10, 10, 5, 0)。
```

DEFTRN（语句）

功　　能

定义 T 型变量 （齐次变换型变量）。

格　　式

DEFTRN < 变量名 >[=< 齐次变换型常量 >][,< 变量名 >[=< 齐次变换型常量 >]…]

说　　明

将在 < 变量名 > 上指定的变量作为齐次变换型进行定义。通过在 < 变量名 > 之后接着写等号和齐次变换型常量，可以在定义的同时进行初始化。

通过用 "," 断开，可以一次指定多个变量名。

相关项目

DEFJNT、 DEFPOS、 DEFVEC

应用示例

```
DEFTRN ltx, lty, ltz                ' 将 ltx, lty, ltz 作为 T 型变量进行定义。
DEFTRN ltx = (10, 10, 5, 20, 20, 10, 30, 30, 15)
                                    ' 将 ltx 作为 T 型变量进行定义，赋予初始值 =
                                    '(10, 10, 5, 20, 20, 10, 30, 30, 15)。
```

DEFIO（语句）

功　能

定义对应输出入端口的 IO 变量 （I / O 变量）。

格　式

DEFIO < 变量名 >=<I / O 变量的类型 >， < 端口地址 >[,< 掩码信息 >]

说　明

将 < 变量名 > 上所指定变量作为 I / O 变量进行定义。

<I / O 变量的类型 >

选择 I / O 变量的类型。I / O 变量的类型有 BIT、BYTE、WORD、 INTEGER、 SINGLE。 BIT 型指定 1 比特、BYTE 型指定 8 比特、 WORD 型指定 16 比特、 INTEGER 型指定 32 比特、 SINGLE 型指定 32 比特的范围。

（SINGLE 型无法在低于 Ver.3.2 的版本上使用）

< 端口地址 >

指定输出入端口上的开始编号。

利用整数型常量指定。

也可利用 2 个整数型常量的加减计算公式进行指定。

< 掩码信息 >

输入端口的情况下，取输入数据和掩码信息的 AND。

输出端口的情况下，取输出数据和掩码信息的 AND 进行输出，但掩码未被设定的比特的输出状态不变化。

在 SINGLE 中，掩码信息无效。如果记述掩码信息，则会出现编译错误。

相关项目

IN、 OUT、 SET、 RESET

注意事项

- WORD、 INTEGER 的情况下，相当于最高位比特的端口为符号比特。
  SINGLE 的表示方法为 IEEE754 形式。
  数值的范围和相当于各种比特的端口编号如下所示。

WORD	数值范围　　: -32768 ～ 32767 最高位比特端口编号: 开头端口地址 +15
INTEGER	数值范围　　: -2147483648 ～ 2147483647 最高位比特端口编号: 开头端口地址 +31
SINGLE	数值范围　　: -3.4E+38~3.4E+38 符号部的端口编号　: 开头端口地址 +31 指数部的端口编号　: 开头端口地址 +30~ 开头端口地址 +23 假数部的端口编号　: 开头端口地址 +22~ 开头端口地址

- 在 WINCAPSIII 中，无法显示 SINGLE 型的值 （查看等）。关于 SINGLE 型的值，请参照多功能教导器。
- WORD 型的数值范围为利用 IN 指令获取数值时的范围。当利用 OUT 指令输出时，且指定超出此范围的数值时，会输出整数部分的下位 16 比特。

应用示例

```
#DEFINE IO_OFFSET 256                    ' 将 ' "IO_OFFSET" 定义为 256
DEFIO samp1 = BIT, 1                     ' 将 samp1 作为从端口 1 开始的 BIT 型 I / O 变量进行定义。
                                         '  samp1 的反馈值表示端口 1 的状态，
                                         ' 为 0 或 1 的 1 比特整数。
DEFIO samp2 = BYTE, 10, &B00010000       ' 将 samp2 添加从端口 10 开始的 BYTE 型 I / O
                                         ' 变量屏蔽信息之后进行定义。 samp2 的反馈值表示端口
                                         '10 的状态，为 0 或 16 的 8 比特整数。
DEFIO samp3 = WORD, 15                   ' 将 samp3 作为从端口 15 开始的 WORD 型 I / O 变量进行定
                                         义。
                                         '  samp3 的反馈值表示从端口 15 到 30 的状态，
                                         ' 为 0 ～ &Hffff 的 16 比特整数。
DEFIO samp4 = INTEGER, 1                 ' 将 samp4 作为从端口 1 开始的 INTEGER 型 I / O 变量进行定
                                         义。
                                         '  samp4 的反馈值表示从端口 1 到 32 的状态，
                                         ' 为 0 ～ &Hffffffff 的 32 比特整数。
DEFIO samp5 = BYTE, IO_OFFSET + 10       ' 将 samp5 作为从 10+IO_OFFSET 计算结果 （在应用示例中
                                          为 266）的端口编号开始的 BYTE 型 I / O 变量
                                         ' 进行定义。samp5 的反馈值在应用示例中表示从端口 266 到
                                          273 的状态， ' 为 0 ～ &Hff 的 8 比特整数。
DEFIO samp6 = SINGLE, 128                ' 将 samp6 作为从端口 128 开始的 SINGLE 型 I/O 变量
                                         ' 进行定义。
                                         'samp6 的反馈值表示从端口 128 到 159 的状态，
                                         ' 为 -3.4E+38~3.4E+38 的 32 比特单精度实数。
```

9.8 排列

DIM（语句）

功　能

定义排列。

格　式

DIM < 变量名 >[< 后缀符 >][(< 要素数 >[,< 要素数 >[,< 要素数 >]])][AS< 变量类型 >][,< 变量名 >[< 后缀符 >]…]

说　明

将 < 变量名 > 上所指定的变量作为排列变量进行定义。

通过添加 < 后缀符 >，可以同时定义变量的类型。

后缀符可以使用以下的字符。

整数型后缀符 : %

单精度实数型后缀符 : !

双精度实数型后缀符 : #

字符串型后缀符 : $

如果省略后缀符，则被视为单精度实数型。

在 < 要素数 > 上指定排列的要素的最大值。要素数必须是 1 以上。

要素的总数必须是 32767 以下。

排列的维数最大到 3 维。

排列可以使用的下标的字符范围是 0 ～ （要素数 -1）。

通过添加 AS < 变量类型 > 可以定义变量的类型。

AS 表达式和后缀符不能同时使用。

在 AS 表达式中的 < 变量类型 > 里可以使用以下的类型。

变量类型	标识符	变量类型	标识符
长整数型	INTEGER	矢量型	VECTOR
单精度实数型	SINGLE	位置型	POSITION
双精度实数型	DOUBLE	关节型	JOINT
字符串型	STRING	齐次变换型	TRANS

应用示例

```
DIM samp1(5)                ' 将 samp1 作为大小为 (5) 的单精度实数型的排列变量进行定义。
DIM samp2(10, 10)           ' 将 samp2 作为大小为 (10, 10) 的单精度实数型的排列变量进行定义。
DIM samp3(20, 5, 10)        ' 将 samp3 作为大小为 (20, 5, 10) 的单精度实数型的排列变量
                            ' 进行定义。
DIM samp4%(3, 3, 3)         ' 将 samp4 作为大小为 (3, 3, 3) 的整数型的排列变量进行定义。
DIM samp5!(4, 3)            ' 将 samp5 作为大小为 (4, 3) 的单精度实数型的排列变量进行定义。
```

```
DIM samp6#(3)                    ' 将 samp6 作为大小为 (3) 的双精度实数型的排列变量进行定义。
```

9.9 文件夹功能

FOLDER（语句）【RC7_Ver.2.2 以上版本】

功　能

定义从外部程序可访问的局部 (Local) 变量。

格　式

FOLDER DEFINT < 变量名 >[=< 常量 >][,< 变量名 >[=< 常量 >]…]
FOLDER DEFSNG < 变量名 >[=< 常量 >][,< 变量名 >[=< 常量 >]…]
FOLDER DEFDBL < 变量名 >[=< 常量 >][,< 变量名 >[=< 常量 >]…]
FOLDER DEFSTR < 变量名 >[=< 常量 >][,< 变量名 >[=< 常量 >]…]
FOLDER DEFVEC < 变量名 >[=< 矢量常量 >] [,< 变量名 > [=< 矢量常量 >]…]
FOLDER DEFPOS < 变量名 >[=< 位置常量 >] [,< 变量名 > [=< 位置常量 >]…]
FOLDER DEFJNT < 变量名 >[=< 连接常量 >] [,< 变量名 > [=< 关节常量 >]…]
FOLDER DEFTRN < 变量名 >[=< 齐次变换型常量 >][,< 变量名 >[=< 齐次变换型常量 >]…]
FOLDER DIM < 变量名 > < 后缀符 >[(< 要素数 >[,< 要素数 >[,< 要素数 >])][AS< 变量类型 >][< 变量名 >…]

说　明

如果用 FOLDER 语句定义局部 (Local) 变量，从外部程序使用 EXTERN 语句可以参照局部 (Local) 变量。

- 在程序内不能定义同一个变量名

列表 1	说明
PRO1 　FOLDER DEFINT AA 　FOLDER DEFSNG AA 　DEFINT AA END	在列表 1 中， AA 的 3 行定义均成为变量 2 重定义错误。

- 在文件夹内不能定义同一个文件夹变量名

列表 2 （PRO1 和 PRO2 在同一个文件夹内）	说明
PRO1 　FOLDER DEFINT AA END PRO2 　FOLDER DEFSNG AA END	列表 2 因在同一个文件夹内用其他程序对 AA 文件夹变量进行定义所以成为错误。

列表 3 （PRO1 和 PRO2 在不同文件夹）	说明
PRO1 　FOLDER DEFINT AA END	列表 3 用分别的文件夹可以对相同的 AA 文件夹变量进行定义 （为不同的变量）
PRO2 　FOLDER DEFINT AA END	

- 在其他程序中文件夹变量名和局部 (Local) 变量名是同一个时可以进行定义

列表 4 （PRO1 和 PRO2 在同一个文件夹内）	说明
PRO1 FOLDER DEFINT AA END PRO2 DEFINT AA END	PRO1 的 AA 和 PRO2 的 AA 为分别变量。

如果用教导器编辑程序或从 WINCAPS Ⅲ将 < 管理表 / 执行程序 > 发送给控制器，则 FOLDER 变量被清空。

相关项目

EXTERN

应用示例

1) 在同一个文件夹内存在以下 3 个程序的情况下
   - 如果在执行 PRO1 之后执行 PRO3，就 "STOP" （程序结束）。
   - 如果在执行 PRO2 之后执行 PRO3，就 "HOLD "STOP"" （暂停程序）。
   - 在 3 个程序中所被定义的 AA 变量 3 个均为相同值。

```
PRO1
  FOLDER DEFINT AA     '定义 FOLDER 变量
  AA=1                 '将 1 赋值 AA 变量
END

PRO2
  EXTERN DEFINT AA     '定义 EXTERN 变量
  AA=2                 '将 2 赋值 AA 变量
END

PRO3
  EXTERN DEFINT AA     '定义 EXTERN 变量
  SELECT CASE AA       'AA 的值与 CASE 语句一致的情况下，
                       '该指令被执行。
  CASE 1               'AA 为 1 的情况下
    STOP               '结束程序。
  CASE 2               'AA 为 2 的情况下
    HOLD "STOP"        '暂停程序执行。
  END SELECT           '定义多个条件判断语句的结束。
END
```

2) 在以下文件夹结构中存在程序
注 1: 33 被赋值 I100。
注 2: 55 被赋值 I100。

```
PRO1
  FOLDER DEFINT AA          '定义 FOLDER 变量
  CALL LIB1.PRO1            '将 LIB1 文件夹中的 PRO1 进行 CALL
  I100=AA                   '将 AA 赋值 I100  （注 1）
  CALL  LIB2.PRO1           '将 LIB2 文件夹中的 PRO1 进行 CALL
  I100=AA                   '将 AA 赋值 I100  （注 2）
END
```

LIB1 文件夹

```
PRO1
  EXTERN DEFINT AA    '定义 EXTERN 变量
  AA = 33             '将 33 赋值 AA
END
```

LIB2 文件夹

```
PRO1
  EXTERN DEFINT AA    '定义 EXTERN 变量
  AA = 55             '将 55 赋值 AA
END
```

3) 用 FOLDER 变量定义了初始值
注 : 如果不执行 PRO1 则初始值不被赋值。

- 如果在执行 PRO1 之后执行 PRO3 的话 3 进入 I1，也进入 I2。
- 如果在执行 PRO2 之后执行 PRO1 的话 3 进入 I1， 0 进入 I2。

```
PRO1
  FOLDER DEFINT AA=3
  I1=AA
END

PRO2
  EXTERN DEFINT AA
  I2=AA
END
```

EXTERN（语句）【RC7_Ver.2.2 以上版本】

功　能

定义参照用其他的程序所定义的 FOLDER 变量的行为。

格　式

EXTERN DEFINT < 变量名 >[,< 变量名 >…]
EXTERN DEFSNG < 变量名 >[,< 变量名 >…]
EXTERN DEFDBL < 变量名 >[,< 变量名 >…]
EXTERN DEFSTR < 变量名 >[,< 变量名 >…]
EXTERN DEFVEC < 变量名 >[,< 变量名 >…]
EXTERN DEFPOS < 变量名 >[,< 变量名 >…]
EXTERN DEFJNT < 变量名 >[,< 变量名 >…]
EXTERN DEFTRN < 变量名 >[,< 变量名 >…]
EXTERN DIM < 变量名 > < 后缀符 >[(< 要素数 >[,< 要素数 >[,< 要素数 >])][AS< 变量类型 >][,< 变量名 >…]

说　明

定义参照用其他的程序所定义的 FOLDER 变量的行为。

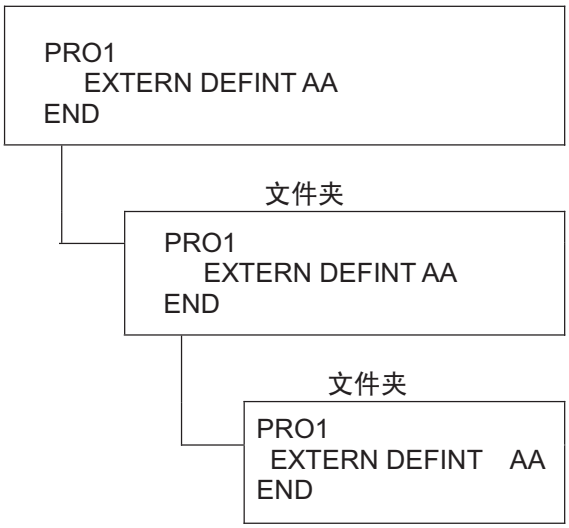
FOLDER 变量搜索方法

1) 搜索 EXTERN 所定义的 PAC 存在的文件夹中是否存在 FOLDER 变量。
   - FOLDER 变量存在的情况下搜索结束。
   - 不存在时前进至 2)。
2) 移向上 1 个的文件夹搜索 FOLDER 变量是否存在。
   - FOLDER 变量存在的情况下搜索结束。
   - 不存在时反复 (2)。
   - 即使搜索文件夹的最前面也不存在 FOLDER 变量时说明发生编译错误。

列表 1	说明
PRO1 　EXTERN DEFINT AA 　DEFINT AA END	列表 1 因相同变量名在 PAC 程序中被定义所以发生编译错误。

列表 2	说明
PRO1 　EXTERN DEFINT AA 　EXTERN DEFINT AA END	列表 2 因相同 EXTERN 变量名在同一个 PAC 文件中被定义所以发生编译错误。

列表 3	说明
PRO1 　EXTERN DEFINT AA END PRO2 　EXTERN DEFINT AA END	用列表 3 在不同的 PAC 文件中相同的 EXTERN 变量已被定义时是可以定义的。

列表 4	说明
	用列表 4 在不同的文件夹中相同的 EXTERN 被定义时是可以定义的。

如果用教导器编辑程序或从 WINCAPS Ⅲ将 < 管理表 / 执行程序 > 发送给控制器，则 EXTERN 变量被清空。

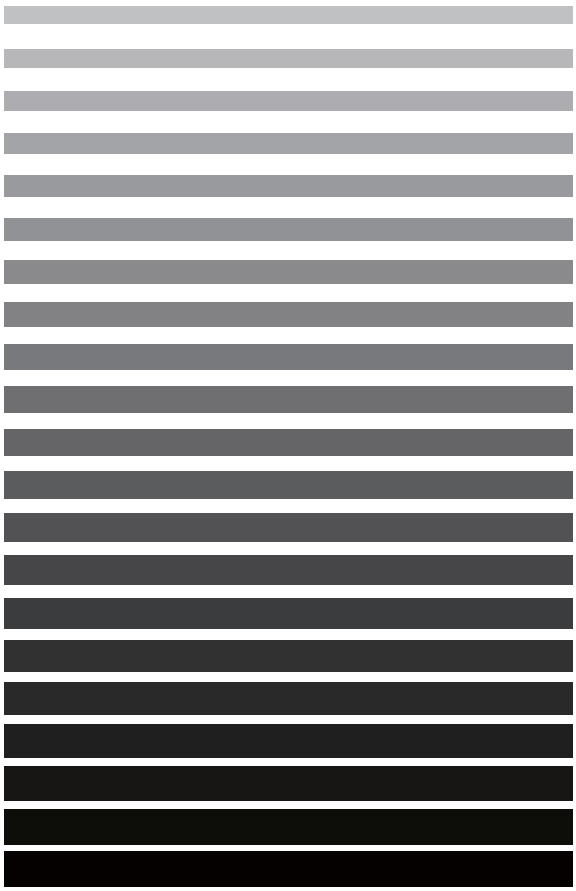
相关项目

FOLDER

应用示例

参照 FOLDER 上所记述的应用示例。

第 10 章
赋值语句



使用赋值语句指令，对变量进行赋值。根据所处理的数值类型的不同，使用不同的指令。根据处理数值的类型，分别使用指令。

第 10 章　赋值语句

10.1　变量

LET（语句）【依据 SLIM】

功　能

对变量赋值。

格　式

[LET] < 变量名 > = < 运算式 >

说　明

[LET] 可以省略。

一般情况下 < 变量名 > 和 < 运算式 > 类型必须是相同的。

但是，变量名和运算式的类型不同时进行以下转换。

- 转换为 < 变量名 > 的变量类型。
  实数转换为整数时，小数点以下被舍去。
  双精度实数变量被转换为单精度实数变量时，数值被取为 7 位有效数字。
- 运算是按照精度高的类型的精度进行运算。

相关项目

应用示例

```
DEFINT li1, li2
LET li1 = li2 + 1                    ' 将 (li2+1) 值赋值 li1。
li1 = li2 + 1                        ' 将 (li2+1) 值赋值 li1 （与上述命令语句意义相同）。
```

6 轴

```
DEFPOS lp1, lp2
DEFJNT lj1, lj2
lp1 = lp2 + (10, 10, 10, 0, 0, 0)    ' 将 (lp2 + (10,10,10,0,0,0) ) 的值赋值 lp1。

lj1 = lj2 + (10, 20, 30, 40, 0, 0)   ' 将 (lj2 + (10,20,30,40,0,0) ) 的值赋值 lj1。
```

4 轴

```
DEFPOS lp1, lp2
DEFJNT lj1, lj2
lp1 = lp2 + (10, 10, 10, 20)         ' 将 (lp2 + (10, 10, 10, 20) ) 的值赋值 lp1。

lj1 = lj2 + (10, 20, 30, 40)         ' 将 (lj2 + (10, 20, 30, 40) ) 的值赋值 lj1。
```

10.2 矢量

LETA（语句）

功　能

赋值于齐次变换型的近似矢量中。

格　式

LETA < 齐次变换型变量 > = < 矢量型 >

说　明

在以 LETA 开头的赋值语句中，式子的右边是矢量。由于该赋值的执行，< 齐次变换型变量 > 内的近似矢量被变更为 < 矢量型 >。

相关项目

LETO、 LETP

应用示例

```
DEFTRN lt1, lt2, lt3
DEFVEC lv1, lv2
LETA lt1 = AVEC(lt3)          ' 将 lt3 的近似矢量赋值 lt1 的近似矢量。

LETA lt2 = lv1 x lv2          ' 将 (lv1 x lv2) 的值赋值 lt2 的近似矢量。
```

LETO（语句）

功　能

赋值于齐次变换型的定向矢量中。

格　式

LETO < 齐次变换型变量 > = < 矢量型 >

说　明

在以 LETO 开头的赋值语句中，式子的右边是矢量。由于该赋值的执行，< 齐次变换型变量 > 内的定向矢量被变更为 < 矢量型 >。

相关项目

LETA、 LETP

应用示例

```
DEFTRN lt1, lt2, lt3
DEFVEC lv1, lv2
LETO lt1 = OVEC(lt3)          ' 将 lt3 的定向矢量赋值 lt1 的定向矢量。

LETO lt2 = lv1 x lv2          ' 将 (lv1 x lv2) 的值赋值 lt2 的定向矢量。
```

LETP（语句）

功　能

赋值于位置型或齐次变换型的位置矢量中。

格　式

LETP { < 位置型变量 > | < 齐次变换型变量 > } = < 矢量型 >

说　明

在以 LETP 开头的赋值语句中，式子的右边是矢量。由于该赋值的执行， < 位置型变量 > 或 < 齐次变换型变量 > 内的位置矢量被变更为 < 矢量型 >。

相关项目

LETO、 LETA

应用示例

```
DEFTRN lt1, lt2
LETP lt1 = PVEC(lt2)                    ' 将 lt2 的位置矢量赋值 lt1 的位置矢量。
```

10.3 形态

LETF（语句）

功　能

赋值于位置型或齐次变换型的形态分量中。

格　式

LETF { < 位置型变量 > | < 齐次变换型变量 >} = < 形态 >

说　明

在以 LETF 开头的赋值语句中，式子的右边是 < 形态 >。由于该赋值的执行，< 位置型变量或齐次变换型变量 > 内的形态成分被变更为 < 形态 > 的值。

相关项目

CURFIG、 FIG、机械手形态 （附录 2）

应用示例

```
DEFPOS lp1, lp2
LETF lp1 = 1                    ' 将 lp1 的形态设定为 LEFTY-ABOVE-FLIP。
LETF lp2 = CURFIG               ' 将 lp2 的形态设定为当前的姿势。
```

10.4 连接角

LETJ（语句）

功　能

赋值于关节型的指定连接角中。

格　式

LETJ < 轴编号 >,< 关节型变量 > = < 算式 >

说　明

以 LETJ < 轴编号 > 开头的赋值语句中，式子的右边是 < 算式 >。根据该赋值的执行，由 < 关节型变量 > 中的 < 轴编号 > 指定的连接角被变更为 < 算式 > 的值。

相关项目

JOINT

应用示例

```
DEFJNT lj1, lj2, lj3
DEFSNG lf1, lf2
LETJ 1, lj1 = JOINT(2, lj3)        ' 将 lj3 的 2 轴的连接角度赋值给 lj1 的 1 轴。
LETJ 3, lj2 = lf1 - lf2            ' 将 (lf1 - lf2) 的值赋值给 lj2 的 3 轴。
```

10.5 姿勢

LETR（语句）

功　能

赋值于位置型的 3 个旋转分量中。

格　式

LETR < 位置型变量 > = < 矢量型 >

说　明

以 LETR 开头的赋值语句中，式子的右边是矢量。由于该赋值的执行，< 位置型变量 > 内姿势被变更为 < 矢量型 >。

相关项目

LETP、 LETRX、 LETRY、 LETRZ

应用示例

```
DEFPOS lp1, lp2
LETR lp1 = RVEC(lp2)                    ' 将 lp2 的姿势赋值给 lp1 的姿势。
```

10.6 旋转分量

LETRX（语句）

功　能

赋值于位置型的 X 轴旋转分量中。

格　式

LETRX < 位置型变量 > = <X 轴旋转角 >

说　明

以 LETRX 开头的赋值语句中，式子的右边是 X 轴旋转角。由于该赋值的执行，< 位置型变量 > 内的 X 轴旋转成分被变更为 <X 轴旋转角 > 的值。

相关项目

LETRY、 LETRZ、 LETR

应用示例

```
DEFPOS lp1, lp2, lp3
DEFSNG lf1, lf2
LETRX lp1 = POSRX(lp3)          ' 将 lp3 的 X 轴旋转成分赋值给 lp1 的 X 轴旋转分量。

LETRX lp2 = lf1 - lf2           ' 将 (lf1 - lf2) 的值赋值给 lp2 的 X 轴旋转分量。
```

LETRY（语句）

功　能

赋值于位置型的 Y 轴旋转分量中。

格　式

LETRY < 位置型变量 > = <Y 轴旋转角 >

说　明

以 LETRY 开头的赋值语句中，式子的右边是 Y 轴旋转角。由于该赋值的执行，< 位置型变量 > 内的 Y 轴旋转分量被变更为 <Y 轴旋转角 > 的值。

相关项目

LETRX、 LETRZ、 LETR

应用示例

```
DEFPOS lp1, lp2, lp3
DEFSNG lf1, lf2
LETRY lp1 = POSRY(lp3)          ' 将 lp3 的 Y 轴旋转成分赋值给 lp1 的 Y 轴旋转分量。

LETRY lp2 = lf1 - lf2           ' 将 (lf1 - lf2) 的值赋值给 lp2 的 Y 轴旋转分量。
```

LETRZ（语句）

功　能

赋值于位置型的 Z 轴旋转分量中。

格　式

LETRZ < 位置型变量 > = <Z 轴旋转角 >

说　明

以 LETRZ 开头的赋值语句中，式子的右边是 Z 轴旋转角。由于该赋值的执行，
< 位置型变量 > 内的 Z 轴旋转成分被变更为 <Z 轴旋转角 > 的值。

相关项目

LETRX、 LETRY、 LETR

应用示例

```
DEFPOS lp1, lp2, lp3
DEFSNG lf1, lf2
LETRZ lp1 = POSRZ(lp3)          ' 将 lp3 的 Z 轴旋转分量赋值给 lp1 的 Z 轴旋转分量。

LETRZ lp2 = lf1 - lf2           ' 将 (lf1 - lf2) 的值赋值给 lp2 的 Z 轴旋转分量。
```

LETT（语句）

功　　能

赋值于位置型的 T 分量中。

格　　式

LETT < 位置型变量 > = <T 轴旋转角 >

说　　明

以 LETT 开头的赋值语句中，式子的右边是 T 轴旋转角。由于该赋值的执行，
< 位置型变量 > 内的 T 轴旋转成分被变更为 <T 轴旋转角 > 的值。

相关项目

应用示例

```
DEFPOS lp1
DEFSNG lf1, lf2
LETT lp1 = lf1 - lf2                    ' 将 (lf1 - lf2) 的值赋值给 lp1 的 T 轴旋转分量。
```

10.7 轴分量

LETX（语句）【依据 SLIM】

功　能

对矢量型 / 赋值于位置型 / 齐次变换型的 X 轴分量中。

格　式

LETX { < 矢量型变量 > | < 位置型变量 > | < 齐次变换型变量 > } = <X 轴分量 >

说　明

以 LETX 开头的赋值语句中，式子的右边是 X 轴分量。由于该赋值的执行，
< 矢量型变量 > 或 < 位置型变量 > 或 < 齐次变换型变量 > 内的 X 轴分量被变更为 <X 轴分量 > 的值。

相关项目

LETP、 LETY、 LETZ

应用示例

```
DEFPOS lp1, lp2
DEFVEC lv1, lv2
LETX lv1 = POSX(lv2)          ' 将 lv2 的 X 轴分量赋值给 lv1 的 X 轴分量。
LETX lp2 = POSX(lp2)          ' 将 lp2 的 X 轴分量赋值给 lp1 的 X 轴分量。
```

LETY（语句）【依据 SLIM】

功　　能

赋值于矢量型 / 位置型 / 齐次变换型的 Y 轴分量中。

格　　式

LETY { < 矢量型变量 > | < 位置型变量 > | < 齐次变换型变量 > } = <Y 轴分量 >

说　　明

以 LETY 开头的赋值语句中，式子的右边是 Y 轴分量。由于该赋值的执行，
< 矢量型变量 > 或 < 位置型变量 > 或 < 齐次变换型变量 > 内的 Y 轴分量被变更为 <Y 轴分量 > 的值。

相关项目

LETP、 LETX、 LETZ

应用示例

```
DEFPOS lp1, lp2
DEFVEC lv1, lv2
LETY lv1 = POSY(lv2)          ' 将 lv2 的 Y 轴分量赋值给 lv1 的 Y 轴分量。
LETY lp2 = POSY(lp2)          ' 将 lp2 的 Y 轴分量赋值给 lp1 的 Y 轴分量。
```

LETZ（语句）【依据 SLIM】

功　能

赋值于矢量型 / 位置型 / 齐次变换型的 Z 轴分量中。

格　式

LETZ { < 矢量型变量 > | < 位置型变量 > | < 齐次变换型变量 > } = <Z 轴分量 >

说　明

以 LETZ 开头的赋值语句中，式子的右边是 Z 轴分量。由于该赋值的执行，
< 矢量型变量 > 或 < 位置型变量 > 或 < 齐次变换型变量 > 内的 Z 轴分量被变更为 <Z 轴分量 > 的值。

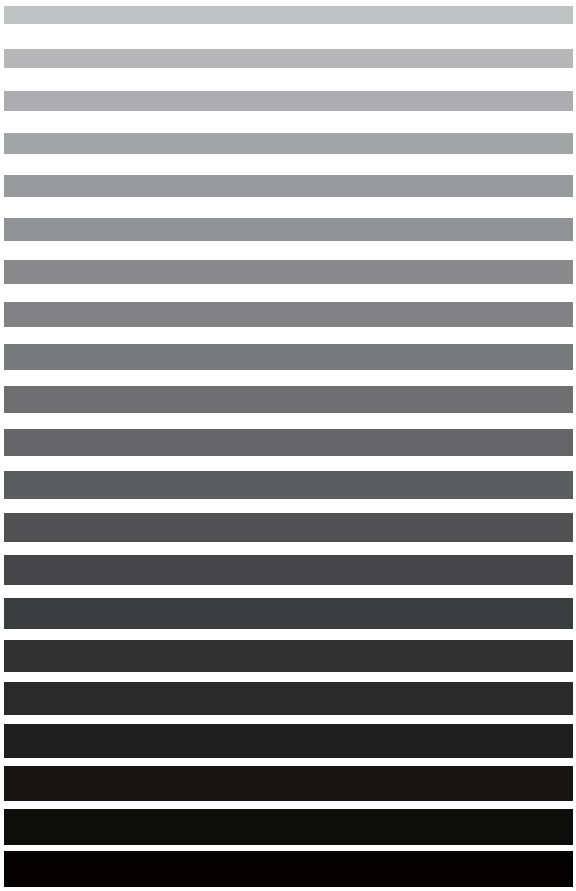
相关项目

LETP、 LETX、 LETY

应用示例

```
DEFPOS lp1, lp2
DEFVEC lv1, lv2
LETZ lv1 = POSZ(lv2)          ' 将 lv2 的 Z 轴分量赋值给 lv1 的 Z 轴分量。
LETZ lp2 = POSZ(lp2)          ' 将 lp2 的 Z 轴分量赋值给 lp1 的 Z 轴分量。
```

第 11 章
流程控制语句



根据条件对程序流程进行变更时使用流程控制语句。如判断条件之后，选择下一个步骤等。判断条件的方法和下一个步骤地进展方法有若干种。

第 11 章　流程控制语句

11.1　程序的停止

END（语句）【依据 SLIM】

功　能

定义基于程序的动作结束。

格　式

END

说　明

如果执行该指令，则结束程序所控制的动作。

但并不意味着程序文件的结束。

使用子程序时，有时将子程序集中书写在主程序之后，而主程序和子程序的分隔，则由该 END 命令进行。

不使用子程序时，如果省略 END 命令，则把最终行视为有 END 命令进行处理。

相关项目

PROGRAM

应用示例

```
DEFSNG lf1, lf2, lf3
PROGRAM prog1(lf1, lf2, lf3)
END                                    ' 结束 prog1。
```

STOP（语句）【依据 SLIM】

功　能

结束程序的执行。

格　式

STOP

说　明

在程序执行过程中，如果遇到 STOP 语句，则在该位置结束程序的执行。

相关项目

DELAY、 HALT、 HOLD、 STATUS

应用示例

REM 执行多个条件判断。

```
SELECT CASE Index          'Index 的值与 CASE 语句的值相一致时，执行该指令。

  CASE 0                   'Index 为 0 的情况。
    STOP                   ' 程序结束。
  CASE 1                   'Index 为 1 的情况。
    HALT "STOP"            ' 暂停程序执行。
  CASE 2                   'Index 为 2 的情况。
    HOLD "STOP"            ' 暂停程序执行。
  CASE 3                   'Index 为 3 的情况。
    STOPEND                ' 循环停止连续启动的程序。
  CASE 4                   'Index 为 4 的情况。
    ON li1 + li2 GOSUB *samp1, *samp2, *samp3
                                                  ' 调出与 li1 + li2 的值相同顺序上
                                                  ' 所书写的标签名的子程序。
  CASE 5                   'Index 为 5 的情况。
    ON li1 + li2 GOTO *samp1, *samp2, *samp3
                                                  ' 转移到与 li1 + li2 的值相同顺序上
                                                  ' 所书写的标签。
  CASE 6                   'Index 为 6 的情况。
    END                    ' 定义基于程序的动作结束。
END SELECT                 ' 定义多个条件判断语句的结束。
```

STOPEND（语句）

功　能

是对连续启动或随附 CYCLE 选项启动的程序进行循环停止的语句。循环启动包含该语句的程序时，即使执行该语句也不会影响动作。

格　式

STOPEND

说　明

如果在程序的执行过程中遇到 STOPEND 语句，则对执行中的程序进行循环停止。

相关项目

STOP、 HALT、 END

应用示例

REM 执行多个条件判断。

```
SELECT CASE Index          'Index 的值与 CASE 语句的值相一致时，执行该指令。

  CASE 0                   'Index 为 0 的情况。
    STOP                   ' 程序结束。
  CASE 1                   'Index 为 1 的情况。
    HALT "STOP"            ' 暂停程序执行。
  CASE 2                   'Index 为 2 的情况。
    HOLD "STOP"            ' 暂停程序执行。
  CASE 3                   'Index 为 3 的情况。
    STOPEND                ' 循环停止连续启动的程序。
  CASE 4                   'Index 为 4 的情况。
    ON li1 + li2 GOSUB *samp1, *samp2, *samp3
                                                ' 调出与 li1 + li2 的值相同顺序上
                                                ' 所书写的标签名的子程序。
  CASE 5                   'Index 为 5 的情况。
    ON li1 + li2 GOTO *samp1, *samp2, *samp3
                                                ' 转移到与 li1 + li2 的值相同顺序上
                                                ' 所书写的标签。
  CASE 6                   'Index 为 6 的情况。
    END                    ' 定义基于程序的动作结束。
END SELECT                 ' 定义多个条件判断语句的结束。
```

11.2 调出

CALL（语句）

功　能

调出程序并执行。

格　式

CALL < 程序名 > ［（< 自变量 (argument)> ［,< 自变量 (argument)>…］）］

说　明

CALL 语句调出在 < 程序名 > 所指定的程序，并转移控制。

CALL 的程序内的 END 语句与子程序内的 RETURN 语句具有相同的含义，并将控制归还至其调出源。

< 自变量 (argument)> 指定提交至用 < 程序名 > 指定的程序的自变量 (argument)。

< 自变量 (argument)> 常量用以下任意一种方法进行传递。一种为 " 通过值进行调出 "，另一种为 " 通过传址调出 "。详情请参照 "8.16 通过值进行的调出和通过传址进行的调出 "。

1．通过值进行的调出

常量、算式和字符式用数值提交。如果变量也用括弧 （ ）括起来，则被视为式，可以用数值提交。

附有编号的变量中，整数型、单精度实数型、双精度实数型和字符串型变量若以 （ ）括起来，则可以数值提交。

例 : 调出程序 PROGRAM SUB1 (AA#) 的方法

- 用数值提交变量时 CALL SUB1 ((D1))
- 提交常量时 CALL SUB1 (10#)
- 提交算式时 CALL SUB1 (D1 + D2)

注意

在提交算式时，请注意式的结果类型。必须与被调出侧的程序的自变量 (argument) 类型相同。

2．根据传址进行的调出

在根据传址进行的调出中，可作为自变量 (argument) 提交变量。

需要作为 < 自变量 (argument)> = 指定全体排列，要在排列名上添加括弧 （ ）。

全局变量、独立排列的 1 要素，不能用参照进行提交。

需要用参照提交全局变量、独立排列的 1 要素时，将值暂时赋值给局部 (Local) 变量，通过其局部 (Local) 变量进行参照调出。

根据传址进行调出提交的变量在调出侧程序内，若变更其内容，即使返回到原来的程序，其变更也有效。

例 1: 调出程序 PROGRAM SUB1 (AA#) 时

- 提交局部 (Local) 变量时 1 CALL SUB1 (DD#)
- 提交局部 (Local) 变量时 2 CALL SUB1 (DA)

（用 DEFDBL 对 DA 已经定义时）

注意

局部 (Local) 变量需要预先赋值。

例 2: 调出程序 PROGRAM SUB2 (BB% (10)) 的情况下

- 提交全体排列时 CALL SUB2 (AB% ())

（用 DIM 对 AB% (10) 已经定义时）

注意

- 若未在变量中赋值数值，执行时将出现错误。
- 程序名称为 PRO < 数字 > 时，不能提交自变量 (argument)。
- 请使调出侧与被调出侧的自变量 (argument) 数值相同。

相关项目

PROGRAM

应用示例

```
DEFDBL ld1, ld2
CALL SUB1((ld1))              ' 用数值提交变量时。
CALL SUB1(10#)                ' 提交常量时。
CALL SUB1(ld1 + ld2)          ' 提交算式时。
```

GOSUB（语句）【依据 SLIM】

功　能

调用子程序。

格　式

GOSUB < 标签名 >

说　明

调出指定的 < 标签名 > 所示的子程序。

从一个子程序中也可多次调出其他子程序（子程序的多重化）。

相关项目

GOTO、 RETURN

注意事项

若从通过 GOSUB (ON ～ GOSUB) 调出的子程序返回到调出源时，需要使用 RETURN 语句。

应用示例

```
DIM li1 As Integer
IF li1 = 0 THEN                 'Ii1 为 0 的情况。
  STOP                          ' 结束程序的执行。
ELSEIF li1 = 1 THEN             'Ii1 为 1 的情况。
  GOTO *samp1                   ' 转移到 *samp1 的标签。
  GO TO *samp2                  ' 转移到 *samp2 的标签。
ELSEIF li1 = 2 THEN             'Ii1 为 2 的情况。
  GOSUB *samp3                  ' 调出标签名 *samp3 的子程序。
ELSE                            'Ii1 为其他的情况。
  RETURN                        ' 返回到原来的主程序。
END IF                          ' 定义 IF 语句的结束。
```

ON ～ GOSUB（语句）【依据 SLIM】

功　能

根据式的值，调出相应的子程序的调出。

格　式

ON < 式 > GOSUB < 标签名 > ［，< 标签名 >］…

说　明

转移到与 < 式 > 的值相同顺序上所书写的标签，然后继续执行程序。此时的顺序从左数为 1、 2、 …。

< 式 > 的值为实数时，将其取整为不超过该值的最大整数之后，再进行处理。

注意

式的结构超过顺序时，不执行任何程序。

相关项目

ON ～ GOTO、 RETURN、 SELECT CASE

注意事项

若从通过 GOSUB (ON ～ GOSUB) 调出的子程序返回到调出源时，需要使用 RETURN 语句。

应用示例

REM 执行多个条件判断。

```
SELECT CASE Index        'Index 的值与 CASE 语句的值相一致时，执行该指令。

  CASE 0                 'Index 为 0 的情况。
    STOP                 ' 程序结束。
  CASE 1                 'Index 为 1 的情况。
    HALT "STOP"          ' 暂停程序执行。
  CASE 2                 'Index 为 2 的情况。
    HOLD "STOP"          ' 暂停程序执行。
  CASE 3                 'Index 为 3 的情况。
    STOPEND              ' 循环停止连续启动的程序。
  CASE 4                 'Index 为 4 的情况。
    ON li1 + li2 GOSUB *samp1, *samp2, *samp3
                                    ' 调出到与 li1 + li2 的值相同顺序上
                                    ' 所书写的标签名的子程序。
  CASE 5                 'Index 为 5 的情况。
    ON li1 + li2 GOTO *samp1, *samp2, *samp3
                                    ' 转移到与 li1 + li2 的值相同顺序上
                                    ' 所书写的标签。
  CASE 6                 'Index 为 6 的情况。
    END                  ' 定义基于程序的动作结束。
END SELECT               ' 定义多个条件判断语句的结束。
```

RETURN（语句）【依据 SLIM】

功　能

从子程序中返回。

格　式

RETURN

说　明

由 GOSUB 语句将流程控制转移到子程序，碰到 RETURN 语句时，结束子程序的执行并返回到原来的主程序中。

相关项目

GOSUB

应用示例

```
DIM li1 As Integer
IF li1 = 0 THEN                 'Ii1 为 0 的情况。
  STOP                          ' 结束程序的执行。
ELSEIF li1 = 1 THEN             'Ii1 为 1 的情况。
  GOTO *samp1                   ' 转移到 *samp1 的标签。
  GO TO *samp2                  ' 转移到 *samp2 的标签。
ELSEIF li1 = 2 THEN             'Ii1 为 2 的情况。
  GOSUB *samp3                  ' 调出标签名 *samp3 的子程序。
ELSE                            'Ii1 为其他的情况。
  RETURN                        ' 返回到原来的主程序。
END IF                          ' 定义 IF 语句的结束。
```

11.3 重复

DO ～ LOOP（语句）

功　能

进行迭代（循环）。

格　式

DO [{WHILE|UNTIL} ［< 条件式 >］]
　:
LOOP
或
DO
　:
LOOP [{WHILE|UNTIL} ［< 条件式 >］]

说　明

DO WHILE 和 DO UNTIL 是前迭代。

LOOP WHILE 和 LOOP UNTIL 是后迭代。

WHILE 语句在条件式被满足的时（真（0 以外的数））反复；UNTIL 语句反复直至满足条件式。

如果省略 < 条件式 > 的右边，则判定值是否为真（0 以外的数）。

如果省略 < 条件式 >，则将条件式视为真（0 以外的数），在 WHILE 语句中为无限循环，而在 UNTIL 语句中，在前判定时是未处理，而在后判定时，只执行一次循环。

与 DO WHILE ～ LOOP 具有相同功能的还有 WHILE ～ WEND 这种结构的语句，可以用 DO WHILE ～ LOOP 进行统一。

与 DO ～ LOOP UNTIL 具有相同功能的还有 REPEAT ～ UNTIL 这种结构的语句，可以用 DO ～ LOOP UNTIL 进行统一。

可以将 WHILE 以及 UNTIL 省略，但在这种情况下为无限循环。

相关项目

EXIT DO、 WHILE ～ WEND、 REPEAT ～ UNTIL、 FOR ～ NEXT

应用示例

```
DEFINT li1, li2, li3, li4, li5, li6, li7, li8, li9
DO WHILE li1 > li2                          ' 执行前迭代。
  IF li1 = 4 THEN EXIT DO                   'li1 = 4 时，从 DO ～ LOOP 中跳出。
  FOR li3 = 0 TO 5                          ' 反复进行 5 次 FOR ～ NEXT 的处理。
    FOR li4 = li5 TO li6                    ' 将 li5 的值逐次加 1，反复处理 FOR ～ NEXT，
                                            ' 直至达到 li6。
      FOR li7 = 1 TO li8 STEP 2             ' 逐次加 1 至 2，反复处理 FOR ～ NEXT，直至达到 li8。

        IF li2 = 2 THEN EXIT FOR            'li2 = 0 时，从 FOR ～ NEXT 中跳出。
        DO WHILE li2 < li9                  ' 执行前迭代。
          GOSUB *samp2
          li9 = li9 + 1
        LOOP                                ' 直至 li2 < li9 时，调出 GOSUB *samp2 语句。
      NEXT li7                              ' 反复进行。
    NEXT                                    ' 反复进行。
  NEXT                                      ' 反复进行。
  li9 = 0
  DO                                        ' 执行后迭代。
    GOSUB *samp2
    li9 = li9 + 1
  LOOP UNTIL li9 < 5                        ' 直至 li9 < 5 时，调出 GOSUB *samp2 语句。
LOOP                                        ' 反复进行。
```

EXIT DO（语句）

功　能

执行从 DO ～ LOOP 中的强制退出。

格　式

EXIT DO

说　明

执行从 DO ～ LOOP 中的强制退出，移动至 DO ～ LOOP 语句下一个命令。

相关项目

DO ～ LOOP

应用示例

```
DEFINT li1, li2, li3, li4, li5, li6, li7, li8, li9
DO WHILE li1 > li2                          ' 执行前迭代。
  IF li1 = 4 THEN EXIT DO                   'li1 = 4 时，从 DO ～ LOOP 中跳出。
  FOR li3 = 0 TO 5                          ' 反复进行 5 次 FOR ～ NEXT 的处理。
    FOR li4 = li5 TO li6                    ' 将 li5 的值逐次加 1，反复处理 FOR ～ NEXT，
                                            ' 直至达到 li6。
      FOR li7 = 1 TO li8 STEP 2             ' 逐次加 1 至 2，反复处理 FOR ～ NEXT，直至达到 li8。

        IF li2 = 2 THEN EXIT FOR            'li2 = 0 时，从 FOR ～ NEXT 中跳出。
        DO WHILE li2 < li9                  ' 执行前迭代。
          GOSUB *samp2
          li9 = li9 + 1
        LOOP                                ' 直至 li2 < li9 时，调出 GOSUB *samp2 语句。
      NEXT li7                              ' 反复进行。
    NEXT                                    ' 反复进行。
  NEXT                                      ' 反复进行。
  li9 = 0
  DO                                        ' 执行后迭代。
    GOSUB *samp2
    li9 = li9 + 1
  LOOP UNTIL li9 < 5                        ' 直至 li9 < 5 时，调出 GOSUB *samp2 语句。
LOOP                                        ' 反复进行。
```

FOR ～ NEXT （语句）【依据 SLIM】

功　能

循环执行位于 FOR ～ NEXT 之间的一系列的指令。

格　式

FOR < 变量名 > = < 初始值 > TO < 最终值 > ［STEP < 增量 >］

：

NEXT ［< 变量名 >］

说　明

根据 FOR 行指定的条件，反复执行位于 FOR ～ NEXT 区间内的一系列命令。

< 初始值 > 设定 < 变量名 > 中指定的变量的初始值。

< 最终值 > 设定 < 变量名 > 中指定的变量的最终值。

< 增量 > 设定初始值和最终值之间的增量。如果省略 STEP，则增量被视为 1。

在以下情况下，不执行 FOR ～ NEXT，转移到执行 NEXT 的下一个。

- < 增量 > 是正值， < 初始值 > 比 < 终值 > 大时。
- < 增量 > 是负值， < 初始值 > 比 < 终值 > 小时。

但是，在 < 变量 > 中赋值 < 初始值 >。

在 1 个 FOR ～ NEXT 中，可以再放置 1 个 FOR ～ NEXT （称为程序套结构、嵌套结构）。

在这种情况下，在各个 < 变量名 > 中必须使用不同的。

此时， 1 个 FOR ～ NEXT 必须完全在其他的 FOR ～ NEXT 的内部。

注意

FOR 和 NEXT 必须是 1 对 1 对应的。

从外部以 GOTO 转移到 FOR ～ NEXT 循环内的程序或相反从循环内部转移到外部的程序，其运行都不能得到保证。增量为 0 时为无限循环。

相关项目

DO ～ LOOP、 EXITFOR

应用示例

```
DEFINT li1, li2, li3, li4, li5, li6, li7, li8, li9
DO WHILE li1 > li2                          ' 执行前迭代。
  IF li1 = 4 THEN EXIT DO                   'li1 = 4 时，从 DO ～ LOOP 中跳出。
  FOR li3 = 0 TO 5                          ' 反复进行 5 次 FOR ～ NEXT 的处理。
    FOR li4 = li5 TO li6                    ' 将 li5 的值逐次加 1，反复处理 FOR ～ NEXT，
                                            ' 直至达到 li6。
      FOR li7 = 1 TO li8 STEP 2             ' 逐次加 1 至 2，反复处理 FOR ～ NEXT，直至达到 li8。

        IF li2 = 2 THEN EXIT FOR            'li2 = 0 时，从 FOR ～ NEXT 中跳出。
        DO WHILE li2 < li9                  ' 执行前迭代。
          GOSUB *samp2
          li9 = li9 + 1
        LOOP                                ' 直至 li2 < li9 时，调出 GOSUB *samp2 语句。
      NEXT li7                              ' 反复进行。
    NEXT                                    ' 反复进行。
  NEXT                                      ' 反复进行。
  li9 = 0
  DO                                        ' 执行后迭代。
    GOSUB *samp2
    li9 = li9 + 1
  LOOP UNTIL li9 < 5                        ' 直至 li9 < 5 时，调出 GOSUB *samp2 语句。
LOOP                                        ' 反复进行。
```

EXIT FOR（语句）

功　能

执行从 FOR ～ NEXT 中的强制退出。

格　式

EXIT FOR

说　明

执行从 FOR ～ NEXT 中的强制退出，移动至 FOR ～ NEXT 语句下一个命令。

相关项目

FOR ～ NEXT

应用示例

```
DEFINT li1, li2, li3, li4, li5, li6, li7, li8, li9
DO WHILE li1 > li2                      ' 执行前迭代。
  IF li1 = 4 THEN EXIT DO               'li1 = 4 时，从 DO ～ LOOP 中跳出。
  FOR li3 = 0 TO 5                      ' 反复进行 5 次 FOR ～ NEXT 的处理。
    FOR li4 = li5 TO li6                ' 将 li5 的值逐次加 1，反复处理 FOR ～ NEXT，
                                        ' 直至达到 li6。
      FOR li7 = 1 TO li8 STEP 2         ' 逐次加 1 至 2，反复处理 FOR ～ NEXT，直至达到 li8。

        IF li2 = 2 THEN EXIT FOR        'li2 = 0 时，从 FOR ～ NEXT 中跳出。
        DO WHILE li2 < li9              ' 执行前迭代。
          GOSUB *samp2
          li9 = li9 + 1
        LOOP                            ' 直至 li2 < li9 时，调出 GOSUB *samp2 语句。
      NEXT li7                          ' 反复进行。
    NEXT                                ' 反复进行。
  NEXT                                  ' 反复进行。
  li9 = 0
  DO                                    ' 执行后迭代。
    GOSUB *samp2
    li9 = li9 + 1
  LOOP UNTIL li9 < 5                    ' 直至 li9 < 5 时，调出 GOSUB *samp2 语句。
LOOP                                    ' 反复进行。
```

REPEAT ～ UNTIL（语句）

功　能

执行后迭代。

格　式

REPEAT
　:
UNTIL ［< 条件式 >］

说　明

直至 < 条件式 > 被满足为止，在 REPEAT 语句和 UNTIL 语句之间进行反复处理。

如果省略 < 条件式 >，则被设为真 （0 以外的数），只执行一次循环。

与 REPEAT ～ UNTIL 功能相同的，由 DO ～ LOOP UNTIL 这样的结构语句，可以统一在 DO ～ LOOP UNTIL 上。

相关项目

DO ～ LOOP、 WHILE ～ WEND

应用示例

```
DEFINT li1, li2, li3, li4, li5, li6, li7, li8, li9
DO WHILE li1 > li2                      ' 执行前迭代。
  IF li1 = 4 THEN EXIT DO               'li1 = 4 时，从 DO ～ LOOP 中跳出。
  FOR li3 = 0 TO 5                      ' 反复进行 5 次 FOR ～ NEXT 的处理。
    FOR li4 = li5 TO li6                ' 将 li5 的值逐次加 1，反复处理 FOR ～ NEXT，
                                        ' 直至达到 li6。
      FOR li7 = 1 TO li8 STEP 2         ' 逐次加 1 至 2，反复处理 FOR ～ NEXT，直至达到 li8。

        IF li2 = 2 THEN EXIT FOR        'li2 = 0 时，从 FOR ～ NEXT 中跳出。
        WHILE li2 < li9                 ' 执行前迭代。
          GOSUB *samp2
          li9 = li9 + 1
        WEND                            ' 直至 li2 < li9 时，调出 GOSUB *samp2 语句。
      NEXT li7                          ' 反复进行。
    NEXT                                ' 反复进行。
  NEXT                                  ' 反复进行。
  li9 = 0
  REPEAT                                ' 执行后迭代。
    GOSUB *samp2
    li9 = li9 + 1
  UNTIL li9 < 5                         ' 直至 li9 < 5 时，调出 GOSUB *samp2 语句。
LOOP                                    ' 反复进行。
```

WHILE ～ WEND（语句）

功　能

执行前迭代。

格　式

WHILE［< 条件式 >］
：
WEND

说　明

直至 < 条件式 > 被满足为止，在 WHILE 语句和 WEND 语句之间进行反复处理。

如果省略 < 条件式 >，则被设为真（0 以外的数），执行无限循环。

若用 GOTO 语句等在 WHILE 语句和 WEND 语句之间分支，至 WEND 语句进行一般执行，回复到 WHILE 语句，然后在该句内如一般插入执行。

与 WHILE ～ WEND 具有相同功能的还有 DO WHILE ～ LOOP 这种结构的语句，可以用 DO WHILE ～ LOOP 进行统一。

相关项目

DO ～ LOOP、REPEAT ～ UNTIL

应用示例

```
DEFINT li1, li2, li3, li4, li5, li6, li7, li8, li9
DO WHILE li1 > li2                    ' 执行前迭代。
  IF li1 = 4 THEN EXIT DO             'li1 = 4 时，从 DO ～ LOOP 中跳出。
  FOR li3 = 0 TO 5                    ' 反复进行 5 次 FOR ～ NEXT 的处理。
    FOR li4 = li5 TO li6              ' 将 li5 的值逐次加 1，反复处理 FOR ～ NEXT，
                                      ' 直至达到 li6。
      FOR li7 = 1 TO li8 STEP 2       ' 逐次加 1 至 2，反复处理 FOR ～ NEXT，直至达到 li8。

        IF li2 = 2 THEN EXIT FOR      'li2 = 0 时，从 FOR ～ NEXT 中跳出。
        WHILE li2 < li9               ' 执行前迭代。
          GOSUB *samp2
          li9 = li9 + 1
        WEND                          ' 直至 li2 < li9 时，调出 GOSUB *samp2 语句。
      NEXT li7                        ' 反复进行。
    NEXT                              ' 反复进行。
  NEXT                                ' 反复进行。
  li9 = 0
  REPEAT                              ' 执行后迭代。
    GOSUB *samp2
    li9 = li9 + 1
  UNTIL li9 < 5                       ' 直至 li9 < 5 时，调出 GOSUB *samp2 语句。
LOOP                                  ' 反复进行。
```

11.4　条件分支

IF ～ END IF（语句）

功　能

执行 IF ～ END IF 之间的条件式中的条件判断。

格　式

IF < 条件式 > THEN
　:
［ELSEIF < 条件式 > THEN］
　:
[ELSE]
　:
END IF

说　明

根据 < 条件式 > 的条件控制程序的执行。

如果 IF 语句的 < 条件式 > 为真 （0 以外的数），则执行 IF ～ ELSEIF 语句之间的语句，如果 < 条件式 > 为假 (0)，则判断 ELSEIF 语句的 < 条件式 >。同样，与 ELSEIF ～ ELSE， ELSE ～ END IF 执行。

相关项目

IF ～ THEN ～ ELSE

应用示例

```
DIM li1 As Integer
IF li1 = 0 THEN                 'Ii1 为 0 的情况。
  STOP                          ' 结束程序的执行。
ELSEIF li1 = 1 THEN             'Ii1 为 1 的情况。
  GOTO *samp1                   ' 转移到 *samp1 的标签。
  GO TO *samp2                  ' 转移到 *samp2 的标签。
ELSEIF li1 = 2 THEN             'Ii1 为 2 的情况。
  GOSUB *samp3                  ' 调出标签名 *samp3 的子程序。
ELSE                            'Ii1 为其他的情况。
  RETURN                        ' 返回到原来的主程序。
END IF                          ' 定义 IF 语句的结束。
```

IF ～ THEN ～ ELSE（语句）【依据 SLIM】

功　能

执行逻辑表达式的条件判断。

格　式

IF < 条件式 > THEN {< 语句 >|< 标签名 >} ［ELSE {< 语句 >|< 标签名 >}］

说　明

根据 < 条件式 > 的条件控制程序的执行。

如果 < 条件式 > 为真 （0 以外的数），则执行 THEN 之后，如果 < 条件式 > 为伪 (0)，则 ELSE 以后被执行。

相关项目

IF ～ END IF

应用示例

```
IF i1 = 0 THEN STOP ELSE GOSUB *samp1      'il 为 0 时，结束程序的执行， il 为其他时，
                                           ' 调出标签名为 * samp1 的子程序。

  il = il + 1                              ' 加上 il。
END                                        ' 定义程序的结束。
*samp1:                                    ' 定义子程序的标签。
  i0 = 0                                   ' 将 0 赋值 i0。
RETURN                                     ' 返回到原来的主程序。
```

SELECT CASE（语句）

功　能

执行多个条件判断。

格　式

SELECT CASE < 式 >
  CASE < 项目 > ［，< 项目 >...］
    :
  [CASE ELSE]
END SELECT

说　明

如果 < 式 > 的值与 CASE 语句的 < 项目 > 相一致，则执行该 CASE 以后的一系列命令。

在 < 式 > 中，可指定算式或字符串。

在 < 项目 > 中，可指定变量、常量、式及条件式。

条件式可如下所示进行指定。

- < 算式 1> TO < 运算式 2>
  调查 < 式 > 的结果在 < 算式 1> 以上还是在 < 算式 2> 以下。
  为字符串时不能使用。
- IS < 比较运算符 > < 算式 >
  比较 < 式 > 的结果和 < 算式 > 的数值。
  为字符串时，< 比较算符 > 仅为 "="。

CASE ELSE 语句与所有 CASE 语句不相符时被执行。

同时， CASE ELSE 语句必须放在 END SELECT 语句前。

相关项目

IF ～ END IF

应用示例

REM 执行多个条件判断。

```
SELECT CASE Index                    'Index 的值与 CASE 语句的值相一致时，执行该指令。

  CASE 0                             'Index 为 0 的情况。
    STOP                             ' 程序结束。
  CASE 1                             'Index 为 1 的情况。
    HALT "STOP"                      ' 暂停程序执行。
  CASE 2                             'Index 为 2 的情况。
    HOLD "STOP"                      ' 暂停程序执行。
  CASE 3                             'Index 为 3 的情况。
    STOPEND                          ' 循环停止连续启动的程序。
  CASE 4                             'Index 为 4 的情况。
    ON li1 + li2 GOSUB *samp1, *samp2, *samp3
                                                ' 调出与 li1 + li2 的值相同顺序上
                                                ' 所书写的标签名的子程序。
  CASE 5                             'Index 为 5 的情况。
    ON li1 + li2 GOTO *samp1, *samp2, *samp3
                                                ' 转移到与 li1 + li2 的值相同顺序上
                                                ' 所书写的标签。
  CASE 6 TO 8                        'Index 为 6 ～ 8 的情况。
    PRINTDBG " 预约 "                ' 向调试视窗输出讯息。
  CASE IS ≧ 9                        'Index 为 9 以上的情况。
    END                              ' 定义基于程序的动作结束。
END SELECT                           ' 定义多个条件判断语句的结束。
```

11.5 无条件分支

GOTO（语句）【依据 SLIM】

功　能

无条件执行程序的分支。

格　式

{GOTO|GO TO} < 标签名 >

说　明

转移到 < 标签名 > 中所指定的标签，从转移位置继续执行。

也可以使用 GO TO 代替 GOTO。

相关项目

GOSUB

应用示例

```
DIM li1 As Integer
IF li1 = 0 THEN                 'Ii1 为 0 的情况。
  STOP                          ' 结束程序的执行。
ELSEIF li1 = 1 THEN             'Ii1 为 1 的情况。
  GOTO *samp1                   ' 转移到 *samp1 的标签。
  GO TO *samp2                  ' 转移到 *samp2 的标签。
ELSEIF li1 = 2 THEN             'Ii1 为 2 的情况。
  GOSUB *samp3                  ' 调出标签名 *samp3 的子程序。
ELSE                            'Ii1 为其他的情况。
  RETURN                        ' 返回到原来的主程序。
END IF                          ' 定义 IF 语句的结束。
```

ON ～ GOTO（语句）【依据 SLIM】

功　能

执行根据表达式的值的无条件分支。

格　式

ON < 式 > GOTO < 标签名 > ［,< 标签名 >］ …

说　明

转移到与 < 式 > 的值相同顺序上所书写的 < 标签名 >，继续执行程序。此时的顺序从左数为 1、 2、 …。
< 式 > 的数值转换为整数后处理。

注意

式的结构超过顺序时，不执行任何程序。

相关项目

ON ～ GOSUB

应用示例

REM 执行多个条件判断。

```
SELECT CASE Index          'Index 的值与 CASE 语句的值相一致时，执行该指令。

  CASE 0                   'Index 为 0 的情况。
    STOP                   ' 程序结束。
  CASE 1                   'Index 为 1 的情况。
    HALT "STOP"            ' 暂停程序执行。
  CASE 2                   'Index 为 2 的情况。
    HOLD "STOP"            ' 暂停程序执行。
  CASE 3                   'Index 为 3 的情况。
    STOPEND                ' 循环停止连续启动的程序。
  CASE 4                   'Index 为 4 的情况。
    ON li1 + li2 GOSUB *samp1, *samp2, *samp3
                                        ' 调出与 li1 + li2 的值相同顺序上
                                        ' 所书写的标签名的子程序。
  CASE 5                   'Index 为 5 的情况。
    ON li1 + li2 GOTO *samp1, *samp2, *samp3
                                        ' 转移到与 li1 + li2 的值相同顺序上
                                        ' 所书写的标签。
  CASE 6                   'Index 为 6 的情况。
    END                    ' 定义基于程序的动作结束。
END SELECT                 ' 定义多个条件判断语句的结束。
```

11.6 备注

REM（语句）【依据 SLIM】

功　能

进行备注的记述。

格　式

{REM|'} ［< 备注 >］

说　明

在程序中加入 < 备注 > 时使用。

接在 REM 的字符串，对程序的执行没有任何影响。

也可以使用单引号 (') 代替 REM。

相关项目

应用示例

REM 执行多个条件判断。

```
SELECT CASE Index                ’Index 的值与 CASE 语句的值相一致时，执行该指令。

  CASE 0                         ’Index 为 0 的情况。
    STOP                         ’ 程序结束。
  CASE 1                         ’Index 为 1 的情况。
    HALT "STOP"                  ’ 暂停程序执行。
  CASE 2                         ’Index 为 2 的情况。
    HOLD "STOP"                  ’ 暂停程序执行。
  CASE 3                         ’Index 为 3 的情况。
    STOPEND                      ’ 循环停止连续启动的程序。
  CASE 4                         ’Index 为 4 的情况。
    END                          ’ 定义基于程序的动作结束。
END SELECT                       ’ 定义多个条件判断语句的结束。
```

第 12 章
机械手控制语句

本章介绍控制机械手动作的指令和机械手控制语句。

第 12 章　机械手控制语句

12.1　动作控制

APPROACH（语句）

功　能

执行工具坐标系指定的绝对动作。

格　式

APPROACH < 插补方法 >,< 基准位置 >,[@< 通过 (Pass) 开始位移 >]< 接近长度 >[,< 动作选择 >][,NEXT]

说　明

< 基准位置 > 可以使用位置型、关节型、齐次变换型。

6 轴

从 < 基准位置 > 沿工具坐标系的 -Z 轴方向、向 < 接近长度 > 分离的位置移动。

4 轴

从 < 基准位置 > 沿底座坐标系的 +Z 轴方向、向 < 接近长度 > 分离的位置移动。

在 < 插补方法 > 中可以在 P （标记为 PTP 也可）、 L 中进行选择。

插补方法	意思
P （或标记为 PTP）	用 PTP 控制移动。
L	用 CP 控制移动。

< 通过 (Pass) 开始位移 > 的值是以目标位置为中心的球的半径，如果动作指令值加入其中则移向下一个控制。用 mm 单位指定。是改变通过 (Pass) 开始计时的标准数值，臂前端进入其中时不会移向下一个控制。

如果省略则作为默认值 @0 处理。

如果设为 @0，则以末端动作运行。

如果设为 @P，则以通过 (Pass) 动作运行。

如果设为 @E，则根据编码器值确认是否到达目标位置，移向下一个动作。

在 < 动作选择 > 中有 SPEED、 ACCEL、 DECEL。

动作选择	意思
SPEED （或标记为 S）	指定移动速度。意思与 SPEED 语句相同。
ACCEL	指定加速度。意思与 ACCEL 语句相同。但是不能指定减速度。减速度的指定请使用 DECEL 语句。
DECEL	指定减速度。意思与 DECEL 语句相同。

如果使用 <NEXT 选择 >，则不等机械手的动作结束而继续下一个非动作命令。但是，以下的命令执行要一直等待机械手的动作结束 （通过 (Pass) 开始）。

机械手动作命令 (CHANGETOOL、CHANGEWORK、SPEED、JSPEED、ACCEL、JACCEL、DECEL、JDECEL)、最佳可搬运质量设定共用程序库 (Library) (aspACLD、 aspChange)、臂动作共用程序库 (Library)（mvSetPulseWidth 等）

此外，与动作选择并用时， NEXT 选择变为无效。使用了 <NEXT 选择 > 时，如果执行步骤停止，则在等待下一个动作命令执行时，在结束其动作之后停止。因此，请注意达到停止时的移动延长。

此外，在教导检查模式时， NEXT 选择变为无效。

补充说明

APPROACH 语句可以用 MOVE 语句改写。

APPROACH < 插补方法 >,[< 通过 (Pass) 开始位移 >]< 基准位置 >,< 接近长度 >[,< 动作选择 >][,NEXT]

如果将上述 APPROACH 语句用 MOVE 语句书写，则如下所示。

6 轴

MOVE < 插补方法 >,[< 通过 (Pass) 开始位移 >]< 基准位置 >+(0, 0, -< 接近长度 >)H[,< 动作选择 >][,NEXT]

例 : APPROACH P, P3, 100 ' 与 MOVE P， P3 + (0, 0, -100) H 相同

例 : APPROACH P, P3, 100 ' 与 MOVE P， P3 + (0, 0, -100) H 相同

4 轴

MOVE < 插补方法 >,[< 通过 (Pass) 开始位移 >]< 基准位置 >+(0, 0, -< 接近长度 >),[,< 动作选择 >][,NEXT]

例 : APPROACH P, P3, 100 ' 与 MOVE P， P3 + (0, 0, 100) 相同

相关项目

DEPART、 SPEED

注意事项

- 从基准位置所求得的接近位置有时会位于机械手的动作范围之外。在此种情况下，会发生 6070 号台错误（J* 软件限位超程、可运行范围之外、特异点）。
- 接近位置的形态为基准位置的形态。因此，接近位置在动作范围之外，有时会发生 " 错误 667* 软件限位超程、可运行范围之外 2"。此时，使用 FIGAPRL、 FIGAPRP （参照 "FIGAPRL （P.12-37） " 、 "FIGAPRP （P.12-39） "）计算接近位置的形态、或如备注所示置换为 MOVE 命令、并用 LETF （参照 10.3 项）对形态进行变更。
- 在 CP 动作中，当前所采取的形态 （参照操作指南 "4.1.3 关于手腕、肘、臂的形态 "）和基准位置的形态不同时，会发生错误 607F （机械手形态不一致）。但是，在可以进行机械手形态变化的动作时，不会发生错误。
- 在 CP 动作中，通过特异点 （参照操作指南 "4.1.3 关于手腕、肘、臂的形态 [2] 形态的边界 "）附近时，会发生 6080 号台错误（指令速度限制超程），停止运行。在这种情况下，请降低速度或将最佳可搬运质量设定模式 （参照 p.4-8 "4.6 最佳可搬运质量设定功能 "）设为 2 或 3 来使用。在仍然发生错误时，请回避特异点附近的轨道。
- 在 CP 动作中，接近位置的形态和基准位置的形态不一致、会发生警报 601C （请变更形态）。请将基准位置的形态变更为动作结束时的形态 （但是，即使发生警报对动作也没有影响）。

应用示例

```
DEFSNG lf1, lf2
DEFPOS lp1, lp2, lp3
```

6 轴

```
APPROACH P, (740, 0, 480, 180, 0, 180, 5), 70
                                    ' 以 (740, 0, 480, 180, 0, 180) 从机械手形态 5 的位置
                                    ' 沿 -Zm 方向向离开 70mm 的位置 （PTP 控制）移动。
APPROACH L, lp1, lf1, SPEED = 100
                                    ' 从 lp1 的位置沿 -Zm 方向向离开 lf1 的位置 （CP 控制，内部速度 =
                                    '100%）移动。
APPROACH P, lp2, @P lf2, S = 50
                                    ' 从 lp2 的位置沿 -Zm 方向向离开 lf2 的位置 （PTP 控制，内部速度 =
                                    '50%）通过 (Pass) 动作。
APPROACH L, lp3, 80                 ' 从 lp3 的位置沿 -Zm 方向向离开 80mm 的位置 （CP 控制）移动。
```

4 轴

```
APPROACH P, (100, 200, 300, 45, 1), 70
                                    ' 以 (100, 200, 300, 45, 1) 从机械手形态 1 的位置沿 +Zb
                                    ' 方向向离开 70mm 的位置 （PTP 控制）移动。
APPROACH L, lp1, lf1, SPEED = 100
                                    ' 从 lp1 的位置沿 +Zb 方向向离开 lf1 的位置 （CP 控制，内部速度 =
                                    '100%）移动。
APPROACH P, lp2, @P lf2, S = 50
                                    ' 从 lp2 的位置沿 +Zb 方向向离开 lf2 的位置 （PTP 控制，内部速度 =
                                    '50%）通过 (Pass) 动作。
APPROACH L, lp3, 80                 ' 从 lp3 的位置沿 +Zb 方向向离开 80mm 的位置 （CP 控制）移动。
```

DEPART（语句）

功　能

通过工具坐标系的指定执行相对动作。

格　式

DEPART < 插补方法 >,[@< 通过 (Pass) 开始位移 >]< 起程长度 >[,< 动作选择 >][,NEXT]

说　明

6 轴

从当前位置向工具坐标系的 -Z 轴方向移动 < 起程长度 > 量。

4 轴

从当前位置向底座坐标系的 +Z 轴方向移动 < 起程长度 > 量。

在 < 插补方法 > 中可以在 P （标记为 PTP 也可）、 L 中进行选择。

插补方法	意思
P （或标记为 PTP）	用 PTP 控制移动。
L	用 CP 控制移动。

< 通过 (Pass) 开始位移 > 的值是以目标位置为中心的球的半径，如果动作指令值加入其中则移向下一个控制。用 mm 单位指定。是改变通过 (Pass) 开始计时的标准数值，臂前端进入其中时不会移向下一个控制。

如果省略则作为默认值 @0 处理。

如果设为 @0，则以末端动作运行。

如果设为 @P，则以通过 (Pass) 动作运行。

如果设为 @E，则根据编码器值确认是否到达目标位置，移向下一个动作。

在 < 动作选择 > 中有 SPEED、 ACCEL、 DECEL。

动作选择	意思
SPEED （或标记为 S）	指定移动速度。意思与 SPEED 语句相同。
ACCEL	指定加速度。意思与 ACCEL 语句相同。但是不能指定减速度。减速度的指定请使用 DECEL 语句。
DECEL	指定减速度。意思与 DECEL 语句相同。

如果使用 <NEXT 选择 >，则不等机械手的动作结束而继续下一个非动作命令。但是，以下的命令执行要一直等待机械手的动作结束 （通过 (Pass) 开始）。

机械手动作命令 (CHANGETOOL、CHANGEWORK、SPEED、JSPEED、ACCEL、JACCEL、DECEL、JDECEL)、最佳可搬运质量设定共用程序库 (Library) (aspACLD、 aspChange)、臂动作共用程序库 (Library)（mvSetPulseWidth 等）

此外，与动作选择并用时， NEXT 选择变为无效。使用了 <NEXT 选择 > 时，如果执行步骤停止，则在等待下一个动作命令执行时，在结束其动作之后停止。因此，请注意达到停止时的移动延长。

此外，在教导检查模式下， NEXT 选择变为无效。

补充说明

DEPART 语句可以用 MOVE 语句改写。

DEPART < 插补方法 >,[< 通过 (Pass) 开始位移 >]< 起程长度 >[,< 动作选择 >][,NEXT]

如果将上述 DEPART 语句用 MOVE 语句书写，则如下所示。

6 轴

MOVE < 插补方法 >,[< 通过 (Pass) 开始位移量 >]< 当前位置 >+(0, 0, -< 起程长度 >)H[,< 动作选择 >][,NEXT]

例 1: DEPART P, 70 ' 与 MOVE P， P0 + (0, 0, -70) H 相同 ； P0 为当前位置

例 2: DEPART P, @P 70 ' 与 MOVE P， @P P0 + (0, 0, -70) H 相同 ； P0 为当前位置

4 轴

MOVE < 插补方法 >,[< 通过 (Pass) 开始位移量 >]< 当前位置 >+(0, 0, < 起程长度 >)[,< 动作选择 >][,NEXT]

例 1: DEPART P, 70 ' 与 MOVE P， P0 + (0, 0, 70) 相同 P0 为当前位置

例 2: DEPART P, @P 70 ' 与 MOVE P， @P P0 + (0, 0, 70) 相同； P0 为当前位置

相关项目

APPROACH、 SPEED

注意事项

DEPART 动作位置的形态，为动作起始时的形态。因此， DEPART 动作位置有时会在动作范围之外，发生 " 错误 667* 软件限位超程、可运行范围之外 2"。在这种情况下，要按备注所示置换为 MOVE 命令，并用 LETF（参照 "10.3 形态 "）对形态进行变更。

应用示例

DEFSNG lf1, lf2

6 轴

```
DEPART P, 70                ' 从当前位置沿 -Zm 方向向离开 70mm 的位置 （PTP 控制）移动。

DEPART L, lf1, SPEED = 100  ' 从当前位置沿 -Zm 方向向离开 lf1 的位置 （CP 控制，
                            'S = 100）移动。
DEPART P, lf2, S = 50       ' 从当前位置沿 -Zm 方向向离开 lf2 的位置 （PTP 控制，
                            'S = 50）移动。
DEPART L, 80                ' 从当前位置沿 -Zm 方向向离开 80mm 的位置 （CP 控制）移动。
```

4 轴

```
DEPART P, 70                ' 从当前位置沿 Zb 方向向离开 70mm 的位置 （PTP 控制）移动。

DEPART L, lf1, SPEED = 100  ' 从当前位置沿 Zb 方向向离开 lf1 的位置 （CP 控制，
                            'S = 100）移动。
DEPART P, lf2, S = 50       ' 从当前位置沿 Zb 方向向离开 lf2 的位置 （PTP 控制，
                            'S = 50）移动。
DEPART L, 80                ' 从当前位置沿 Zb 方向向离开 80mm 的位置 （CP 控制）移动。
```

DRAW（语句）

功　能

通过工件作业坐标系的指定执行相对动作。

格　式

DRAW < 插补方法 >, [@< 通过 (Pass) 开始位移量 >]< 并进移动量 >[,< 动作选择 >][,NEXT]

说　明

从当前位置移动 < 并进移动量 > 的距离。

在 < 插补方法 > 中可以在 P （标记为 PTP 也可）、 L 中进行选择。

插补方法	意思
P （或标记为 PTP）	用 PTP 控制移动。
L	用 CP 控制移动。

< 通过 (Pass) 开始位移量 > 的值是以目标位置为中心的球的半径，如果运行指令值加入其中则移向下一个控制。用 mm 单位指定。是改变通过 (Pass) 开始计时的标准数值，臂前端进入其中时不会移向下一个控制。

如果省略则作为默认值 @0 处理。

如果设为 @0，则以末端动作运行。

如果设为 @P，则以通过 (Pass) 动作运行。

如果设为 @E，则根据编码器值确认是否到达目标位置，移向下一个动作。

在 < 动作选择 > 中有 SPEED、 ACCEL、 DECEL。

动作选择	意思
SPEED （或标记为 S）	指定移动速度。意思与 SPEED 语句相同。
ACCEL	指定加速度。意思与 ACCEL 语句相同。但是不能指定减速度。减速度的指定请使用 DECEL 语句。
DECEL	指定减速度。意思与 DECEL 语句相同。

如果使用 <NEXT 选择 >，则不等机械手的动作结束而继续下一个非动作命令。但是，以下的命令执行要一直等待机械手的动作结束 （通过 (Pass) 开始）。

机械手动作命令 (CHANGETOOL、CHANGEWORK、SPEED、JSPEED、ACCEL、JACCEL、DECEL、JDECEL)、最佳可搬运质量设定共用程序库 (Library) (aspACLD、 aspChange)、臂动作共用程序库 (Library) （mvSetPulseWidth 等）

此外，与动作选择并用时， NEXT 选择变为无效。使用了 <NEXT 选择 > 时，如果执行步骤停止，则在等待下一个动作命令执行时，在结束其动作之后停止。因此，请注意达到停止时的移动延长。

此外，在教导检查模式下， NEXT 选择变为无效。

补充说明

DRAW 语句可以用 MOVE 语句改写。

DRAW < 插补方法 >,[< 通过 (Pass) 开始位移 >]< 并进移动量 >[,< 动作选择 >][,NEXT]

如果将上述 DRAW 语句用 MOVE 语句书写，则如下所示。

MOVE < 插补方法 >,[< 通过 (Pass) 开始位移 >]< 当前位置 >+< 并进移动量 >

例 : DRAW L, (50, 10, 50) ’ 与 MOVE L, P0 + (50, 10, 50) 相同

相关项目

SPEED、 TOOL、 CHANGEWORK

注意事项

DRAW 动作位置的形态，为动作起始时的形态。因此， DRAW 动作位置有时会在动作范围之外，发生 " 错误 667* 软件限位超程、可运行范围之外 2"。在这种情况下，要按备注所示置换为 MOVE 命令，并用 LETF （参照 "10.3 项 "）对形态进行变更。

应用示例

```
DEFVEC lv1, lv2
DRAW L, (50, 10, 50)          ' 从当前位置向离开 (X = 50,Y = 10, Z = 50) 的位置
                              ' （CP 控制）移动。
DRAW L, lv1, SPEED = 90       ' 从当前位置向离开 lv1 的位置 （CP 控制， S = 90）移动。

DRAW L, lv2, S = 50           ' 从当前位置向离开 lv2 的位置 （CP 控制， S = 50）移动。
```

DRIVE（语句）【依据 SLIM】

功　能

进行各轴的相对动作。

格　式

DRIVE[<@ 通过 (Pass) 开始位移 >](< 轴编号 >,< 相对移动量 >)[,(< 轴编号 >,< 相对移动量 >)…][,< 动作选择 >][,NEXT]

说　明

让 < 轴编号 > 上所指定的轴向 < 相对移动量 > 所指定的角度 (DEG) 移动。< 相对移动量 > 为正时，让指定轴向 + 方向，为负时，向 - 方向动作。

为了执行该指令，需要获取包括要动作的轴在内的臂组。将相同的轴进行多个指定时，后面的指定为有效。

<@ 通过 (Pass) 开始位移 > 的值是以目标位置为中心的球的半径，如果动作指令值加入其中则移向下一个控制。用 mm 单位指定。是改变通过 (Pass) 开始计时的标准数值，臂前端进入其中时不会移向下一个控制。

- 如果省略则作为默认值 @0 处理。
- 如果设为 @0，则以末端动作运行。
- 如果设为 @P，则以通过 (Pass) 动作运行。
- 如果设为 @E，则根据编码器值确认是否到达目标位置，移向下一个动作。

在 < 动作选择 > 中有 SPEED、 ACCEL、 DECEL。

动作选择	意思
SPEED （或标记为 S）	指定移动速度。意思与 SPEED 语句相同。
ACCEL	指定加速度。意思与 ACCEL 语句相同。但是不能指定减速度。减速度的指定请使用 DECEL 语句。
DECEL	指定减速度。意思与 DECEL 语句相同。

如果使用 <NEXT 选择 >，则不等机械手的动作结束而继续下一个非动作命令。但是，以下的命令执行要一直等待机械手的动作结束 （通过 (Pass) 开始）。

机械手动作命令 (CHANGETOOL、CHANGEWORK、SPEED、JSPEED、ACCEL、JACCEL、DECEL、JDECEL)、最佳可搬运质量设定共用程序库 (Library) (aspACLD、 aspChange)、臂动作共用程序库 (Library) （mvSetPulseWidth 等）

此外，与动作选择并用时， NEXT 选择变为无效。使用了 <NEXT 选择 > 时，如果执行步骤停止，则在等待下一个动作命令执行时，在结束其动作之后停止。因此，请注意达到停止时的移动延长。

此外，在教导检查模式下， NEXT 选择变为无效。

相关项目

DRIVEA、 MOVE 的 EX 或 EXA 选择

（让附加轴运行的指令只有 MOVE 的 EX 或 EXA 选择、 DRIVE、 DRIVEA。)

注意事项

指定附加轴，在 <@ 通过 (Pass) 开始位移 > 选择上输入数值时，与数值无关进行通过 (Pass) 动作。

应用示例

例 1:

```
DEFINT li1, li2, li3
DEFSNG lf1, lf2, lf3
DRIVE (li1, 30)                          ' 从当前位置将 li1 移动 30 度 (deg)。
DRIVE (li1, lf1)                         ' 从当前位置将值为 li1 的轴移动 lf1 值的量。
DRIVE (li1, 0.78RAD), (li2, lf2), (li3, lf3)
                                         ' 从当前位置将 li1 移动 0.78(rad) 值的量、
                                         ' 将 li2 移动 lf2 所示值的量、将 li3 移动 lf3 值的量。
```

例 2 （附加轴的示例）:

	J1	J2	J3	J4	J5	J6	J7	J8
Group 0	○	○	○	○	×	×	×	×
Group 1	×	×	×	×	×	×	○	○
Group 2	○	○	○	○	×	×	○	○
Group 3	×	×	×	×	×	×	×	×
Group 4	×	×	×	×	×	×	×	×

```
PROGRAM PRO1
TAKEARM 1                                ' 臂组 1 包括 7 轴 8 轴。
DRIVE (7,30),(8,50)                      ' 让 7 轴 8 轴动作。
END
```

为了让 7 轴 8 轴动作，在上例中，需要获取臂组 1 或臂组 2。

DRIVEA（语句）

功　能

进行各轴的绝对动作。

格　式

DRIVEA[<@ 通过 (Pass) 开始位移 >](< 轴编号 >,< 轴坐标 >)[,(< 轴编号 >,< 轴坐标 >)…][,< 动作选择 >][,NEXT]

说　明

让 < 轴编号 > 上所指定的轴向 < 轴坐标 > 所指定的角度 (DEG) 移动。

为了执行该指令，需要获取包括要动作的轴在内的臂组。将相同的轴进行多个指定时，后面的指定为有效。

< 通过 (Pass) 开始位移 > 的值是以目标位置为中心的球的半径，如果动作指令值加入其中则移向下一个控制。用 mm 单位指定。是改变通过 (Pass) 开始计时的标准数值，臂前端进入其中时不会移向下一个控制。

- 如果省略则作为默认值 @0 处理。
- 如果设为 @0，则以末端动作运行。
- 如果设为 @P，则以通过 (Pass) 动作运行。
- 如果设为 @E，则根据编码器值确认是否到达目标位置，移向下一个动作。

在 < 动作选择 > 中有 SPEED、 ACCEL、 DECEL。

动作选择	意思
SPEED （或标记为 S）	指定移动速度。意思与 SPEED 语句相同。
ACCEL	指定加速度。意思与 ACCEL 语句相同。但是不能指定减速度。减速度的指定请使用 DECEL 语句。
DECEL	指定减速度。意思与 DECEL 语句相同。

如果使用 <NEXT 选择 >，则不等机械手的动作结束而继续下一个非动作命令。但是，以下的命令执行要一直等待机械手的动作结束 （通过 (Pass) 开始）。

机械手动作命令 (CHANGETOOL、CHANGEWORK、SPEED、JSPEED、ACCEL、JACCEL、DECEL、JDECEL)、最佳可搬运质量设定共用程序库 (Library) (aspACLD、 aspChange)、臂动作共用程序库 (Library)（mvSetPulseWidth 等）

此外，与动作选择并用时， NEXT 选择变为无效。使用了 <NEXT 选择 > 时，如果执行步骤停止，则在等待下一个动作命令执行时，在结束其动作之后停止。因此，请注意达到停止时的移动延长。

此外，在教导检查模式下， NEXT 选择变为无效。

相关项目

DRIVE、 MOVE 的 EX 或 EXA 选择

（让附加轴运行的指令只有 MOVE 的 EX 或 EXA 选择、 DRIVE、 DRIVEA。）

注意事项

指定附加轴，在 <@ 通过 (Pass) 开始位移 > 选择上输入数值时，与数值无关进行通过 (Pass) 动作。

应用示例

例 1:

```
DEFINT li1, li2, li3
DEFSNG lf1, lf2, lf3
DRIVEA (li1, 30)                                    ' 将 li1 移动 30 度 (deg)。
DRIVEA (li1, lf1)                                   ' 从当前位置将值为 li1 的轴移动至 lf1 值。
DRIVEA @P (li1, 0.78RAD), (li2, lf2), (li3, lf3)
                                                    ' 从当前位置将 li1 移动 0.78(rad) 值的量、
                                                    ' 将 li2 移动 lf2 所示值的量、将 li3 移动 lf3 值的量。
```

例 2 （附加轴的示例）:

	J1	J2	J3	J4	J5	J6	J7	J8
Group 0	○	○	○	○	×	×	×	×
Group 1	×	×	×	×	×	×	○	○
Group 2	○	○	○	○	×	×	○	○
Group 3	×	×	×	×	×	×	×	×
Group 4	×	×	×	×	×	×	×	×

```
PROGRAM PRO1
TAKEARM 1                                           ' 臂组 1 包括 7 轴 8 轴。
DRIVE (7,30),(8,50)                                 ' 让 7 轴 8 轴动作。
END
```

为了让 7 轴 8 轴动作，在上例中，需要获取臂组 1 或臂组 2。

GOHOME（语句）【依据 SLIM】

功　　能

向用 HOME 语句定义的位置点 （原点）移动。

格　　式

GOHOME

说　　明

将机械手用 PTP 控制从当前位置移向初始位置。

初始位置用 HOME 语句进行定义。

不设定 HOME 而执行时，会发生错误。

相关项目

HOME

应用示例

GOHOME ，从当前位置移向初始位置。

MOVE（语句）【依据 SLIM】

功　能

将机械手向指定坐标移动。

通过使用 EX 选择 （附加轴相对动作）或 EXA 选择 （附加轴绝对动作），可以使机械手和附加轴同步 （同时出发和到达）动作。

格　式

- < 插补方法 > 在自由曲线以外时
  MOVE < 插补方法 >,[@< 路径开始位移 >]< 姿势 (Pose)>[<EX 或 EXA 选择 >][,[@< 通过 (Pass) 开始位移 >]< 姿势 (Pose)>[<EX 或 EXA 选择 >]…][,< 动作选择 >][,NEXT]
- < 插补方法 > 在自由曲线时 【Ver.2.3 以上版本】
  MOVE S, [@< 通过 (Pass) 开始位移 >]< 轨道编号 >[<EX 或 EXA 选择 >][,< 动作选择 >][,NEXT]

说　明

从当前位置向指定坐标 < 姿势 (Pose)> 移动。

< 姿势 (Pose)> 可以使用位置型 （P 型）、关节型 （J 型）、齐次变换型 （T 型）。

- 位置型可以使用变量、带编号变量所组成的姿势 (Pose) 列、
  常量以及当前位置 (*,CURPOS)。
- 关节型可以使用变量、带编号变量所组成的姿势 (Pose) 列以及当前角度 (CURJNT)。
- 齐次变换型可以使用变量、带编号变量所组成的姿势 (Pose) 列。

姿势 (Pose) 列的表示方法 : P[3 TO 6] 从 P[3] 到 P[6]。

在 < 插补方法 > 中可以选择 P、 L、 C、 S 的 4 种。

插补方法	意思
P （或标记为 PTP）	用 PTP 控制从当前位置向指定坐标移动。
L	用 CP 控制从当前位置向指定坐标移动。
C	从当前位置经过路径姿势 (Pose) 向目的姿势 (Pose) 进行圆弧插补移动。 姿势从当前位置的姿势向目的姿势 (Pose) 的姿势进行插补动作。（路径姿势 (Pose) 的姿势被忽略） 在圆弧插补中需要路径姿势 (Pose)、目的姿势 (Pose) 的 2 点的指示。 （在 C 上不能使用姿势 (Pose) 列） 即使在路径姿势 (Pose) 上指定通过 (Pass) 开始位移动作也不会变化。 圆弧插补动作结束时的通过 (Pass)，按 MOVE C, P1, @P P2 进行指定。
S 自由曲线 【Ver.2.3 以上版本】	从当前位置，通过用 SETSPLINEPOINT 所登录的通过点，向最终的通过点移动。轨道为光滑的曲线。除了加速减速时，工具端的速度以一定速度在曲线上动作。 姿势以通过动作通过每个通过 (Pass) 点。

< 轨道编号 > 是自由曲线的轨道编号。 成为通过用 SETSPLINEPOINT 在每个 < 轴编号 > 所登录的通过点的自由曲线。最大可以指定 20。

< 通过 (Pass) 开始位移 > 的值是以指定坐标 （姿势 (Pose)）为中心的球的半径，如果动作指令值进入其中，则移向下一个控制。用 mm 单位指定。是改变通过 (Pass) 开始计时的标准数值，臂前端进入其中时不会移向下一个控制。

- 如果省略则作为默认值 @0 处理。
- 如果设为 @0，则以末端动作运行。
- 如果设为 @P，则以通过 (Pass) 动作运行。
- 如果设为 @E，则根据编码器值确认是否到达目标位置，移向下一个动作。

在 < 动作选择 > 中有 SPEED、 ACCEL、 DECEL。

动作选择	意思
SPEED （或标记为 S）	指定移动速度。意思与 SPEED 语句相同。
ACCEL	指定加速度。意思与 ACCEL 语句相同。但是不能指定减速度。减速度的指定请使用 DECEL 语句。
DECEL	指定减速度。意思与 DECEL 语句相同。

如果使用 <NEXT 选择 >，则不等机械手的动作结束而继续下一个非动作命令。但是，以下的命令执行要一直等待机械手的动作结束 （通过 (Pass) 开始）。 机械手动作命令 (CHANGETOOL、 CHANGEWORK、 SPEED、 JSPEED、 ACCEL、 JACCEL、 DECEL、 JDECEL)、最佳可搬运质量设定共用程序库 (Library) (aspACLD、 aspChange)、臂动作共用程序库 (Library) （mvSetPulseWidth 等）

此外，与动作选择并用时， NEXT 选择变为无效。使用了 <NEXT 选择 > 时，如果执行步骤停止，则在等待下一个动作命令执行时，在结束其动作之后停止。因此，请注意达到停止时的移动延长。

此外，在教导检查模式下， NEXT 选择变为无效。

像 MOVE P,P[3 TO 5] 或 MOVE P,P3,P6,P9 的表达，分别同

MOVE P,P[3 TO 5] →
MOVE P,P3,NEXT
MOVE P,P4,NEXT
MOVE P,P5

MOVE P,P3,P6,P9 →
MOVE P,P3,NEXT
MOVE P,P6,NEXT
MOVE P,P9

所表达的意思相同。（处理效率最好是左侧的表达）

因此，

- 在 IOBLOCK ON 和 OFF 之间，即使是有这些表达，在最终姿势 (Pose)（P5 或 P9）的动作开始之前，下一个非动作命令也不会执行。 "IOBLOCK ON/OFF （P.13-3） " 请参照。
- 使用了 NEXT 选择时，最终姿势 (Pose) (P5, P9) 的动作开始之前，下一个非动作命令也不会执行。

EX 或 EXA 选择用以下的格式指定。

<EX 或 EXA 选择 > 的格式

EX((< 轴编号 >,< 相对移动量 >)[,(< 轴编号 >,< 相对移动量 >)…])

EXA((< 轴编号 >,< 轴坐标 >)[,(< 轴编号 >,< 轴坐标 >)…])

在 < 轴编号 > 上只能指定附加轴，机械手轴不能指定。

相关项目

SPEED、 DRIVE、 DRIVEA

（让附加轴运行的指令只有 MOVE 的 EX 或 EXA 选择、 DRIVE、 DRIVEA。)

注意事项

- 用位置型、齐次变换型进行姿势 (Pose) 指定时，被指定的姿势 (Pose) 有时会在机械手的动作范围之外。在此种情况下，会发生 6070 号台错误 （J* 软件限位超程、可运行范围之外、特异点）。特别是在指定 16 ～ 31 形态时要注意。
- 在 CP 动作、圆弧插补动作中，当前所采取的形态 （参照操作指南 "4.1.3 关于手腕、肘、臂的形态 "）和指定坐标的形态不同时，会发生错误 607F （机械手形态不一致）。但是，在可以进行机械手形态变化的动作时，不会发生错误。

- 在 CP 动作、圆弧插补动作、自由曲线动作中，通过特异点 （参照操作指南 "4.1.3 关于手腕、肘、臂的形态 [2] 形态的边界 "）附近时，有时会发生 6080 号台错误 （指令速度限制超程），停止运行。在这种情况下，请降低速度或将最佳可搬运质量设定模式（参照 p.4-8 "4.6 最佳可搬运质量设定功能 "）设为 2 或 3 来使用。在仍然发生错误时，请回避特异点附近的轨道。
- 在 CP 动作中，动作结束时的形态和所指定姿势 (Pose) 的形态不一致、有时会发生警报 601C （请变更形态）。请求以动作结束时的形态再次教导 （但是，即使发生警报也不影响动作）。
- 圆弧插补动作时的通过动作、编码器值确认动作，请在目的姿势 (Pose) 上指定路径开始位移量。即使在通过 (Pass) 姿势 (Pose) 上指定通过 (Pass) 开始位移动作也不会变化。
- 圆弧插补动作，路径姿势 (Pose) 的姿势被忽略。因此，有时工具端会不通过路径姿势 (Pose)。要想让工具端通过路径姿势 (Pose)，请用工具定义在工具端设定工具坐标。
- 用圆弧插补动作，在上一个动作通过 (Pass) 时，如果在通过 (Pass) 动作过程中进行瞬时停止再启动，则会发生 " 错误 60de 进行了和指定的旋转动作不同的动作 "。
- 在圆弧插补动作中，如果当前的姿势 (Pose) 和目的姿势 (Pose) 一致时，则不动作。此外，如果当前的姿势 (Pose) 和路径姿势 (Pose) 一致时、路径姿势 (Pose) 和目的姿势 (Pose) 一致时，则会向目的姿势 (Pose) 进行 CP 动作。
- 在自由曲线动作中，会发生用 SETSPLINEPOINT 所登录的通过点的形态与当前的形态不同的错误 "607F 机械手形态不一致 "。
- 在自由曲线动作中，如果通过点之间的移动距离短、姿势变化大时，会发生 6080 号台错误 " 指令速度限制超程 " 并停止运行。在这种情况下，请降低速度、或将最佳可搬运质量设定模式设为 2 或 3。

附加轴的注意事项

- 被设定为无限旋转的轴不能执行 EXA 选择 （绝对动作命令）。
- 处理姿势 (Pose) 列时需要注意。
  例 : MOVE P,P[3 TO 5] EX((7,30))
  MOVE P,P3,NEXT
  MOVE P,P4,NEXT
  MOVE P,P5 EX((7,30))
  同所记述的相同。
  因此 7 轴和机械手向 P5 的坐标移动同步 （同时出发和到达）。

应用示例

例 1:

```
DIM li1 As Integer
6 轴
MOVE P, (740, 0, 480, 180, 0, 180, 5), NEXT      ' 以 (740, 0, 480, 180, 0, 180) 从机械手形态
                                                 '5 的位置向坐标 （PTP 控制）移动。
                                                 ' 动作开始之后，执行下一个命令。
4 轴
MOVE P, (100, 200, 300, 45, 1), NEXT             ' 以 (740, 0, 480, 180, 0, 180) 从机械手形态
                                                 '1 的位置向坐标 （PTP 控制）移动。
                                                 ' 动作开始之后，执行下一个命令。
MOVE L, lp1, SPEED = 100                         ' 向 lp1 的坐标
                                                 ' （CP 控制，内部速度 = 100%）移动。
MOVE P, @30 lp2, lp3, S = lil                    ' 向 lp2 (@30)、 lp3 的坐标顺次
                                                 ' （PTP 控制，内部速度 = li1）动作。
MOVE L, @20 lp4, @50 lp5, @100 lp6               ' 向 lp4(@20)、 lp5(@50)、 lp6(@100)
                                                 ' 的坐标顺次 （CP 控制）动作。
MOVE L, @P P[6 TO 15], lp7                       ' 从 P [6] 到 P [15] 以通过 (Pass) 动作顺次
                                                 ' 移动，向 lp7 的坐标 （CP 控制）动作。
MOVE C, lp1, @p lp2                              ' 进行通过 lp1 向 lp2 移动的圆弧插补
                                                 ' 动作。 用 lp2 进行通过 (Pass) 动作，
                                                 ' 移向下一个控制。
```

例 2 （附加轴的示例）：

	J1	J2	J3	J4	J5	J6	J7	J8
Group 0	○	○	○	○	×	×	×	×
Group 1	×	×	×	×	×	×	○	○
Group 2	○	○	○	○	×	×	○	○
Group 3	×	×	×	×	×	×	×	×
Group 4	×	×	×	×	×	×	×	×

```
PROGRAM PRO1
TAKEARM 2                        ' 获取包括机械手轴和附加轴在内的臂组 2
MOVE P,P0 EX((7,30),(8,10))      ' 向机械手的 P0 的移动和 7 轴、 8 轴的相对移动
                                 ' 同时出发和到达。
MOVE P,P1 EXA((7,30))            ' 向机械手的 P1 的移动和
                                 '7 轴的绝对移动同时出发和到达。
END
```

例 3 （自由曲线动作的示例）【Ver.2.3 以上版本】：

```
MOVE S, @P 2, S=10, NEXT         ' 以内部速度 10% 的速度在轨道编号 2 的
                                 ' 自由曲线上进行通过 (Pass) 动作。动作开始之后，
                                 ' 执行下一个命令。
```

ROTATE（语句）【依据 SLIM】

功　能

执行指定的旋转轴的旋转动作。

格　式

ROTATE < 旋转面 >,[@< 通过 (Pass) 开始位移 >]< 相对旋转角 > [,[< 旋转中心点 >] [,< 旋转选择 >] [,< 动作选择 >][,NEXT]]

旋转面的格式

6 轴

{{XY|YZ|ZX}|{XYH|YZH|ZXH}|(矢量型 , 矢量型 , 矢量型)}

4 轴

{{XY}|{XYH}}

说　明

以在 < 旋转面 > 上垂直所作的轴为中心进行 < 旋转角度 > 量、夹治具前端的转动。旋转轴对于旋转面有 2 种，同接近形成的角小的那种会被选择。

< 旋转面 > 可以从以下的 7 种中选择。

旋转面的格式

6 轴

XY|YZ|ZX 是在旋转面平行于工件坐标的 XY, YZ, ZX 平面时进行指定。

XYH|YZH|ZXH 是在旋转面平行于工具坐标的 XY, YZ, ZX 平面时进行指定。

（矢量型，矢量型，矢量型）制作从 3 点的 X, Y, Z 坐标向底座坐标基准的平面。

4 轴

XY 是在旋转面平行于工件坐标的 XY 平面时进行指定。

XYH 是在旋转面平行于工具坐标的 XY 平面时进行指定。

< 通过 (Pass) 开始位移 > 的值是以指定坐标 （姿势 (Pose)）为中心的球的半径，如果动作指令值进入其中，则移向下一个控制。用 mm 单位指定。是改变通过 (Pass) 开始计时的标准数值，臂前端进入其中时不会移向下一个控制。

如果省略则作为默认值 @0 处理。

如果设为 @0，则以末端动作运行。

如果设为 @P，则以通过 (Pass) 动作运行。

如果设为 @E，则根据编码器值确认是否到达目标位置，移向下一个动作。

< 相对旋转角 > 的单位是度 (DEG)，符号右旋转为正 (+)。

6 轴

< 旋转中心点 > 在当旋转面为 {XY|YZ|ZX} 的时候和（矢量型，矢量型，矢量型）的时候，指定工件坐标上的点，当旋转面为 {XYH|YZH|ZXH} 的时候，指定工具坐标上的点。

省略 < 旋转中心点 > 的情况下，当旋转面为 {XY|YZ|ZX} 的时候，(0, 0, 0) 成为旋转中心，当旋转面为 （矢量型，矢量型，矢量型）的时候，一个单位的矢量成为旋转中心。但是，如果指定旋转面 {XYH|YZH|ZXH} 时，不能以 (0, 0, 0) 为旋转中心动作。请务必指定旋转中心点。

4 轴

< 旋转中心点 > 在当旋转面为 {XY} 的时候，指定工件坐标上的点，当旋转面为 {XYH} 的时候，指定工具坐标上的点。

省略 < 旋转中心点 > 的情况下，当旋转面为 {XY} 的时候， (0, 0, 0) 成为旋转中心。但是，如果指定旋转面 {XYH} 时，不能以 (0, 0, 0) 为旋转中心动作。请务必指定旋转中心点。

< 旋转选择 > 用 pose = 1、 pose = 2 进行指定。

在 pose = 1 的情况下，机械手的姿势相对于旋转中心会恒定地变化。旋转角度为 ±540° 为止。在 pose = 2 的情况下，机械手的姿势保持当前的姿势。不指定选择时，会进行与 pose = 2 相同的动作。

在 < 动作选择 > 中有 SPEED、 ACCEL、 DECEL。

动作选择	意思
SPEED （或标记为 S）	指定移动速度。意思与 SPEED 语句相同。
ACCEL	指定加速度。意思与 ACCEL 语句相同。但是不能指定减速度。减速度的指定请使用 DECEL 语句。
DECEL	指定减速度。意思与 DECEL 语句相同。

如果使用 <NEXT 选择 >，则不等机械手的动作结束而继续下一个非动作命令。但是，以下的命令执行要一直等待机械手的动作结束 （通过 (Pass) 开始）。

机械手动作命令 (CHANGETOOL、CHANGEWORK、SPEED、JSPEED、ACCEL、JACCEL、DECEL、JDECEL)、最佳可搬运质量设定共用程序库 (Library) (aspACLD、 aspChange)、臂动作共用程序库 (Library) （mvSetPulseWidth 等）

此外，与动作选择并用时， NEXT 选择变为无效。使用了 <NEXT 选择 > 时，如果执行步骤停止，则在等待下一个动作命令执行时，在结束其动作之后停止。因此，请注意达到停止时的移动延长。

此外，在教导检查模式时， NEXT 选择变为无效。

相关项目

ROTATEH

注意事项

如果在通过 (Pass) 动作过程中进行瞬时停止再启动，则会发生 "错误 60de 进行了和指定的旋转动作不同的动作 "。

应用示例

```
        ROTATE XY, 45, V1                ' 在通过 V1 的点垂直于 XY 平面的轴旋转上以固定
                                         ' 姿势旋转 45 度。
只 6 轴  ROTATE (V2, V3, V4), F1          ' 在通过 V2 的点垂直于 (V2, V3, V4) 所做平面的轴
                                         ' 旋转上旋转 F1 的值量。
        ROTATE XYH, 43, V1, S = 100      ' 在通过工具坐标 V1 的接近矢量旋转上用 100%
                                         ' 的移动速度做 43 度的旋转动作。
```

ROTATEH（语句）

功　能

执行以近似矢量为轴的旋转动作。

格　式

ROTATEH [@< 通过 (Pass) 开始位移 >]< 近似矢量旋转的相对旋转角 >[,< 动作选择 >][,NEXT]

说　明

以近似矢量为轴旋转 < 近似矢量旋转的相对旋转角 > 量、夹治具前端。

< 近似矢量旋转的相对旋转角 > 的单位是度 (DEG)。

但是，旋转角请在 "-180° < 旋转角 <180°" 内指定。

< 通过 (Pass) 开始位移 > 的值是以指定坐标 （姿势 (Pose)）为中心的球的半径，如果动作指令值进入其中，则移向下一个控制。用 mm 单位指定。是改变通过 (Pass) 开始计时的标准数值，臂前端进入其中时不会移向下一个控制。

如果省略则作为默认值 @0 处理。

如果设为 @0，则以末端动作运行。

如果设为 @P，则以通过 (Pass) 动作运行。

如果设为 @E，则根据编码器值确认是否到达目标位置，移向下一个动作。

在 < 动作选择 > 中有 SPEED、 ACCEL、 DECEL。

动作选择	意思
SPEED （或标记为 S）	指定移动速度。意思与 SPEED 语句相同。
ACCEL	指定加速度。意思与 ACCEL 语句相同。但是不能指定减速度。减速度的指定请使用 DECEL 语句。
DECEL	指定减速度。意思与 DECEL 语句相同。

如果使用 <NEXT 选择 >，则不等机械手的动作结束而继续下一个非动作命令。但是，以下的命令执行要一直等待机械手的动作结束 （通过 (Pass) 开始）。

机械手动作命令 (CHANGETOOL、CHANGEWORK、SPEED、JSPEED、ACCEL、JACCEL、DECEL、JDECEL)、最佳可搬运质量设定共用程序库 (Library) (aspACLD、 aspChange)、臂动作共用程序库 (Library)（mvSetPulseWidth 等）

此外，与动作选择并用时， NEXT 选择变为无效。使用了 <NEXT 选择 > 时，如果执行步骤停止，则在等待下一个动作命令执行时，在结束其动作之后停止。因此，请注意达到停止时的移动延长。

此外，在教导检查模式时， NEXT 选择变为无效。

补充说明

ROTATEH 语句可以用 MOVE 语句改写。如果将 ROTATEH,[< 通过 (Pass) 开始位移 >]< 旋转角 >[,< 动作选择 >][,NEXT] 的 ROTATE 语句用 MOVE 语句书写 , 则如下所示。

MOVE L,[< 通过 (Pass) 开始位移 >]< 当前位置 >+(0, 0, 0, 0, 0, < 旋转角 >)H[,< 动作选择 >][,NEXT]

例 :

ROTATEH @P, F1’MOVE L, @P PO+(0, 0, 0, 0, 0, F1) H 相同 : PO 是当前位置

相关项目

ROTATE

应用示例

```
DEFSNG FF1=50                ' 将相对旋转角设为 50 度。
TAKEARM                      ' 获得机械手控制权。
MOVE P,P1                    ' 向 P1 点进行 PTP 移动。
CHANGETOOL 1                 ' 将 TOOL 的 1 设为有效。
ROTATEH @50 FF1              ' 将通过 (Pass) 开始位移置为 50，将相对旋转角置为 FF1。
CHANGETOOL 0                 ' 将 TOOL 置为 0。
```

CURJNT（系统变量）

功　　能

以 J 型取得机械手的当前角度。

格　　式

CURJNT

说　　明

机械手当前各轴编码器所查出的关节角度被以 J 型数据的形式存放。

机械手在动作时，获取执行该命令时刻的值。

获取目标姿势 (Pose) 时，请使用 DESTJNT。

相关项目

DESTJNT、 CURPOS、 CURTRN

注意事项

在机器锁定运行过程中不是各轴的编码器值所检测的关节角度，而是设想的关节角度 （指令角度）。

应用示例

```
DIM lj1 As Joint
lj1 = CURJNT                    ' 将当前的关节角度赋值 lj1。

6 轴
lj1 = CURJNT + (10, 10, 0, 0, 0, 10)
                                ' 将当前的关节角度 + (10, 10, 0, 0, 0, 10)
                                ' 赋值 lj1。

4 轴
lj1 = CURJNT + (10, 10, 0, 0)
                                ' 将当前的关节角度 + (10, 10, 0, 0) 赋值 lj1。
```

CURPOS（系统变量）【依据 SLIM】

功　　能

以 P 型数据得出工具坐标系中的当前位置。

格　　式

{CURPOS|*}

说　　明

从各轴编码器的检测值所检测到的当前工具坐标系上的位置被以 P 型数据的形式存放。

机械手在动作时，获取执行该命令时刻的值。

获取目标姿势 (Pose) 时，请使用 DESTPOS。

相关项目

DESTPOS、 CURJNT、 CURTRN

注意事项

在机器锁定运行过程中不是各轴的编码器值所检测的关节角度，而是设想的关节角度 （指令角度）。

应用示例

```
DEFPOS lp1, lp2
lp1 = CURPOS                    ' 将当前的工具坐标系上的位置赋值 lp1。
lp2 = *                         ' 将当前的工具坐标系上的位置赋值 lp2。

6 轴
lp1 = CURPOS + (100, 200, 0, 10, 10, 0)
                                ' 将当前的工具坐标系上的位置 + (100, 200, 0, 10, 10,
                                '0) 赋值 lp1。

4 轴
lp1 = CURPOS + (100, 200, 0, 10)
                                ' 将当前的工具坐标系上的位置 + (100, 200, 0, 10)
                                ' 赋值 lp1。
```

CURTRN（系统变量）【依据 SLIM】

功　　能

以 T 型数据得出工具坐标系中的当前位置。

格　　式

CURTRN

说　　明

从各轴编码器的检测值所检测到的当前工具坐标系上的位置以 T 型数据的形式存放。

机械手在动作时，获取执行该命令时刻的值。

在获取目标姿势 (Pose) 时，请使用 DESTTRN。

相关项目

DESTTRN、 CURJNT、 CURPOS

注意事项

在机器锁定运行过程中不是各轴的编码器值所检测的关节角度，而是设想的关节角度 （指令角度）。

应用示例

```
DEFTRN l+1, l+2
l+1 = CURTRN                    ' 将当前的工具坐标系上的位置赋值 l+1。
l+2 = *                         ' 将当前的工具坐标系上的位置赋值 l+2。
```

CUREXJ（语句）

功　　能

以 F 型获取附加轴的当前角度。

格　　式

CUREXJ(< 轴编号 >)

说　　明

将在 < 轴编号 > 上指定的附加轴的编码器上所检测到的角度写入 F 型变量。指定的附加轴在动作时，可以获取执行该命令时刻的值。

在获取目标位置时，请使用 DESTEXJ。

相关项目

CURJNT、 CURPOS、 CURTRN、 DESTEXJ

注意事项

在机器锁定运行过程中不是编码器值所检测的角度，而是设想的角度 （指令角度）。

应用示例

```
PROGRAM PRO1
DIM  lf1  AS  SINGLE
TAKEARM  1
DRIVEA(7,100)                    ' 将 7 轴运行至 100 度的位置。
lf1 = CUREXJ(7)                  ' 将 7 轴的当前位置赋值 lf1。
END
```

DESTJNT （系统变量）

功　　能

以 J 型获取当前的动作命令目标位置。

机械手停止时，获取当前的位置 （指令值）。

格　　式

DESTJNT

说　　明

之前的动作命令的目标位置被以J型数据的形式存放。因为可以用系统变量获取，所以就没有必要用局部 (Local) 变量及全局变量等进行同其他程序和数据的交接。

从各轴编码器的检测值获取检测到的位置时，请使用 CURJNT。

相关项目

CURJNT、 DESTPOS、 DESTTRN

注意事项

当输入动作停止命令 （参照 "INTERRUPT ON/OFF （P.12-43） "）动作停止时，停止位置就被输入。

例 :

```
INTERRUPT ON
MOVE P, J1 ←动作过程中中止信号 ON
INTERRUPT OFF
J2=DESTJNT
```

不会成为 J1 = J2。 J2 为停止位置。

应用示例

```
DEFJNT lj1, lj2
MOVE P, @P lj1, NEXT
```

6 轴

```
lj2 = DESTJNT+(100, 0, 0, 0, 0, 0)          ' 在这种情况下 DESTJNT = lj1。
```

4 轴

```
lj2 = DESTJNT+(100, 0, 0, 0)                ' 在这种情况下 DESTJNT = lj1。
```

DESTPOS（系统变量）

功　能

以 P 型获取当前的动作命令目标位置。

机械手停止时，获取当前的位置 （指令值）。

格　式

DESTPOS

说　明

之前的动作命令的目标位置以P型数据的形式被存放。因为可以用系统变量获取，所以就没有必要用局部 (Local) 变量及全局变量等进行同其他程序和数据的交接。

从各轴编码器的检测值获取检测到的位置时，请使用 CURPOS。

相关项目

CURPOS、 DESTJNT、 DESTTRN

注意事项

当输入动作停止命令 （参照 p.12-43 "INTERRUPT ON / OFF"）动作停止时，停止位置就被输入。

例 :

```
INTERRUPT ON
MOVE P, P1 ←动作过程中中止信号 ON
INTERRUPT OFF
P2=DESTPOS
```

不会成为 P1 = P2。 P2 为停止位置。

应用示例

```
DEFPOS lp1, lp2
MOVE P, @P lp1, NEXT

6 轴
lp2 = DESTPOS+(100, 10, 10, 0, 0, 0)          ’ 在这种情况下 DESTPOS = lp1。

4 轴
lp2 = DESTPOS+(100, 10, 10, 0)                ’ 在这种情况下 DESTPOS = lp1。
```

DESTTRN（系统变量）

功　　能

以 T 型获取当前的动作命令目标位置。

机械手停止时，获取当前的位置 （指令值）。

格　　式

DESTTRN

说　　明

之前的动作命令的目标位置被以 T 型数据的形式存放。因为可以用系统变量获取，所以就没有必要用局部 (Local) 变量及全局变量等进行同其他程序和数据的交接。

从各轴编码器的检测值获取检测到的位置时，请使用 CURTRN。

相关项目

CURTRN、 DESTJNT、 DESTPOS

注意事项

当输入动作停止命令 （参照 "INTERRUPT ON/OFF （P.12-43） "）动作停止时，停止位置就被输入。

例 :

```
INTERRUPT ON
MOVE P, T1 ←动作过程中中止信号 ON
INTERRUPT OFF
T2=DESTTRN
```

不会成为 T1 = T2。 T2 为停止位置。

应用示例

```
DEFPOS lp1, lp2
MOVE P, @P lp1, NEXT
lp2 = DESTTRN+(100, 10, 10)              ' 在这种情况下 DESTTRN = lp1。
```

DESTEXJ（语句）

功　能

以 F 型获取附加轴的以当前的动作命令所指定的目标位置。停止时，获取当前的位置 （指令值）。

格　式

DESTEXJ(< 轴编号 >)

说　明

将在 < 轴编号 > 上指定的附加轴的之前的动作命令所指定的目标位置写入 F 型变量。
从各轴编码器的检测值获取检测到的位置时，请使用 CUREXJ。

相关项目

CUREXJ、 DESTJNT、 DESTPOS、 DESTTRN

注意事项

当输入动作停止命令 （参照 "INTERRUPT ON/OFF （P.12-43） "）动作停止时，停止位置就被输入。
例 :

```
INTERRUPT ON
DRIVEA (7,100)          ' 动作过程中中止信号 ON， 7 轴停止进入下一步。
INTERRUPT OFF
F1= DESTEXJ(7)          'F1 不会成为 100。停止位置被赋值 F1。
```

应用示例

```
PROGRAM PRO1
DIM lf1 AS SINGLE
TAKEARM 1
DRIVEA(7,100),NEXT          ' 将 7 轴运行至 100 度的位置。
                            ' 用 NEXT 在动作结束前进入下一个运行。
lf1 = DESTEXJ(7)            ' 将是之前的动作命令的目标位置的 100
                            ' 赋值 lf1。
END
```

ARRIVE（语句）【Ver.1.2 以上版本】

功　能

通过设定相对于动作命令的全移动距离的动作比例，使程序待机直到到达机械手所设定的动作比例。

格　式

ARRIVE < 动作比例 >

说　明

以通常的动作命令是不能执行动作结束或到通过 (Pass) 开始为止的下一步骤的命令，但通过 IOBLOCK 指定及 NEXT 选择指定的动作命令就可以在动作中途执行下一步骤的命令。在这种情况下，通过 ARRIVE 命令可以使机械手所设定的到达动作比例为止，下一步骤的命令待命。

< 动作比例 > 表示相对于于动作命令的全移动距离的比例，在 1 ～ 99（单位是 %）的范围内以数值、或整数型（I 型）、实数型 （F 型）变量进行指定。

此功能只对机械手是有效的。因此用 DRIVE、 DRIVEA 只让附加轴动作时， ARRIVE 功能不工作。

注意事项

- ARRIVE 命令是在获取 (TAKEARM) 机械机械臂信号 (semaphore) 的任务，
  相对于之前的动作命令进行动作比例设定。
- 获取臂信号 (semaphore) 之后，如果在执行动作命令前执行 ARRIVE 命令时则会成为错误 648C。
- 如果没有获取臂信号 (semaphore) 而执行了 ARRIVE 命令时则会成为错误 21F7。
- ARRIVE 命令执行过程中如果瞬时停止，再启动时从瞬时停止的位置偏离了 10mm 以上则成为错误 6486。
- 如果在 ARRIVE 命令执行过程中瞬时停止，再启动时动作命令不被执行、达不到 < 动作比例 > 时则成为错误 6489。
- < 动作比例 > 表示相对于机械手移动距离的比例。可是，即便 < 动作比例 > 一定可由于速度、加速度等的动作条件 ARRIVE 所结束的机械手位置有时也会变动。此外，即便多少变动 < 动作比例 >， ARRIVE 所结束的机械手位置也有不变的情况。如果通过 ARRIVE 命令要取得机械手与周围机器的同步时，请确认时机。
- 与电流限制功能并用时，由于电流限制机械手位置有可能变得不能通过 < 动作比例 >。此时，因程序在无限地持续等待 ARRIVE 命令结束，所以请执行程序停止使其复位。
- 与 INTERRUPT 并用时，INTERRUPT ON 之后的动作命令由于中止跳跃信号而不被执行，但因 ARRIVE 命令被执行，根据中止跳跃信号的时机对于和有意图的动作命令不同的动作命令 ARRIVE 命令有被执行的时候。此外，根据中止跳跃信号的时机，机械手位置有可能变得不能通过 < 动作比例 >。此时，因程序在无限地持续等待 ARRIVE 命令结束，所以请执行程序停止使其复位。
- 因在教导检查上 NEXT 选择、 IOBLOCK 命令为无效，所以 ARRVIE 命令不具意义。

- 成为 ARRVIE 命令的目标的动作命令不能使用 < 动作选择 >。
  如果使用 < 动作选择 >，则 ARRVIE 命令为无效。
  如果使用成为 ARRVIE 命令的目标的动作命令的内部速度、外部加减速度时请依照如下示例。
  ＜例如＞

```
PROGRAM PRO1
TAKEARM
SPEED 10              ' 内部 SP 10
MOVE P,P1,NEXT
ARRIVE 50
SET IO [240]          ' 动作比例如果成为 50%，打开 I / O
SPEED 100             ' 将内部速度返回到原来的值
END
```

应用示例

例 1:

```
PROGRAM PRO1
TAKEARM
MOVE P, P1, NEXT
ARRIVE 50                    ' 动作比例如果成为 50%
SET IO [240]                 ' 就会打开 IO [240]。
ARRIVE I1                    ' 动作比例如果成为 I1%
RESET IO [240]               ' 就会关闭 IO [240]。
END
```

例 2 （附加轴的示例）：
用 4 轴机械手如下所示的臂组被设定时。

	J1	J2	J3	J4	J5	J6	J7	J8
Group 1	○	○	○	○	×	×	○	○

```
PROGRAM PRO1
TAKEARM 1
DRIVE(1,30),(7,30),NEXT
ARRIVE 50
SET IO [240]                 '打开 IO [240] 。
MOVE P,P0 EX((7,30)),NEXT
ARRIVE 50
RESET IO [240]               '关闭 IO [240] 。
DRIVE(7,30),NEXT
ARRIVE 50
SET IO [240]                 '打开 IO [240] 。
END
```

因与附加轴的同期动作也包括机械手轴，所以 ARRIVE 作用。

因在此之前的动作命令包括机械手轴，所以 ARRIVE 作用。

之前的动作命令只是附加轴，因此 ARRIVE 不发挥功能进入下一步。

POSCLR（语句）【Ver.1.5 以上版本】

功　能

将轴的当前位置强制性置为 0mm 或 0 度。

格　式

POSCLR < 轴编号 >

说　明

将 < 轴编号 > 所指定的轴的角度强制性置为 0mm 或 0 度。只有设为无限旋转的轴是能够执行的。

向相同方向使之连续旋转，而当前位置的数值变大难以处理的情况下、或当前值的数值跳向负的大数值的情况下 （向相同方向使之旋转时位置有时会跳）等使用。

为了执行该指令，需要获取包括要清空位置的轴在内的臂组。

相关项目

注意事项

- 机械手轴是不能执行的。（只附加轴）
- POSCLR 执行时要等机械手完全停止。因此，即使在 POSCLR 之前进行通过 (Pass) 动作指定，通过 (Pass) 动作也不会进行。
- 步骤返回功能在 POSCLR 执行以前是不能返回的。

应用示例

```
PROGRAM PRO1
TAKEARM 1
DRIVEA(7,100)                ' 将 7 轴运行至 100 度的位置。
POSCLR 7                     ' 将 7 轴的当前位置强制置为 0 度。
END
```

SETSPLINEPOINT（语句）【Ver.2.3 以上版本】

功　能

登录自由曲线动作的通过点。

格　式

SETSPLINEPOINT < 自由曲线的轨道编号 >、 < 通过点 >

说　明

作为在 < 自由曲线的轨道编号 > 上所指定的自由曲线轨道的通过点，在 < 通过点 > 上登录所指定的点。 < 通过点 > 为 P 型、 J 型变量。

相关项目

CLRSPLINEPOINT、 GETSPLINEPOINT、 MOVE

注意事项

- 在一个自由曲线上通过点最大可以登录到 200 点。登录超过那之上时，会发生错误 "685A 自由曲线登录数据数量超过 "。
- 如果执行了通过点未登录的自由曲线动作时，会发生错误 "685B 自由曲线编号指定异常 "。
- 轨道编号可以指定到 20。指定了超过 20 的编号时，会发生 "685B 自由曲线编号指定异常 " 的错误。
- 指定的通过点在动作范围之外时，会发生 6070 号台错误 （J* 软件限位超程、可运行范围之外）。
- 如果在通过点演示时设定工具坐标、工件坐标，请在执行本指令前先执行 CHANGETOOL 指令、CHANGEWORK 指令，并变更成同演示时相同的工具坐标、工件坐标。

应用示例

```
PROGRAM PRO1
TAKEARM
CLRSPLINEPOINT 5          ' 在轨道编号 5 号的自由曲线上将通过点全部清空。
SETSPLINEPOINT 5, P4      ' 在轨道编号 5 号的自由曲线上将 P4 作为第 1 号的通过。
SETSPLINEPOINT 5, P1      ' 在轨道编号 5 号的自由曲线上将 P1 作为第 2 号的通过。
SETSPLINEPOINT 5, J5      ' 在轨道编号 5 号的自由曲线上将 J5 作为第 3 号的通过。
MOVE S, 5                 ' 执行通过 P4、 P1 向 J5 移动的自由曲线动作。
```

CLRSPLINEPOINT（语句）【Ver.2.3 以上版本】

功　能

将自由曲线动作的通过点全部删除。

格　式

CLRSPLINEPOINT < 自由曲线的轨道编号 >

说　明

在 < 自由曲线的轨道编号 > 所指定的自由曲线轨道上，将用 SETSPLINEPOINT 所登录的通过点全部删除。

相关项目

SETSPLINEPOINT、 GETSPLINEPOINT

注意事项

如果执行 CLRSPLINEPOINT 会成为通过点未登录。如果执行了通过点未登录的自由曲线动作时，会发生错误 "685B 自由曲线编号指定异常 "。

应用示例

```
PROGRAM PRO1
TAKEARM
CLRSPLINEPOINT 5          ' 在轨道编号 5 号的自由曲线上将通过点全部清空。
SETSPLINEPOINT 5, P4      ' 在轨道编号 5 号的自由曲线上将 P4 作为第 1 号的通过。
SETSPLINEPOINT 5, P1      ' 在轨道编号 5 号的自由曲线上将 P1 作为第 2 号的通过。
SETSPLINEPOINT 5, J5      ' 在轨道编号 5 号的自由曲线上将 J5 作为第 3 号的通过。
MOVE S, 5                 ' 执行通过 P4、 P1 向 J5 移动的自由曲线动作。
```

GETSPLINEPOINT（语句）【Ver.2.3 以上版本】

功　能

取出已经登录的自由曲线动作的通过点。

格　式

< 通过点 >=GETSPLINEPOINT (< 自由曲线的轨道编号 >、 < 通过点编号 >)

说　明

在 < 自由曲线的轨道编号 > 所指定的自由曲线轨道上，将 < 通过点编号 > 所指定的通过点的点输出到 < 通过点 >。 < 通过点 > 为 P 型变量。

相关项目

CLRSPLINEPOINT、 SETSPLINEPOINT

注意事项

如果指定了未登录的通过点编号，会发生 "685C 自由曲线点指定异常 " 错误。

应用示例

```
PROGRAM PRO1
TAKEARM
CLRSPLINEPOINT 5              ' 在轨道编号 5 号的自由曲线上将通过点全部清空。
SETSPLINEPOINT 5, P4          ' 在轨道编号 5 号的自由曲线上将 P4 作为第 1 号的通过。
SETSPLINEPOINT 5, P1          ' 在轨道编号 5 号的自由曲线上将 P1 作为第 2 号的通过。
SETSPLINEPOINT 5, J5          ' 在轨道编号 5 号的自由曲线上将 J5 作为第 3 号的通过。
P10=GETSPLINEPOINT(5,2)       ' 将轨道编号 5 号的自由曲线的第 2 号的通过点设成 P10。
                              '（P1 的数据进入 P10）
```

12.2 形态控制

CURFIG（系统变量）

功　能

得出机械手形态的当前值。

格　式

CURFIG

说　明

当前的机械手的形态被用整数存放。整数值与形态的关系入下表所示。

6 轴

值	形 态
0	SINGLE4—SINGLE6—FLIP—ABOVE—RIGHTY
1	SINGLE4—SINGLE6—FLIP—ABOVE—LEFTY
2	SINGLE4—SINGLE6—FLIP—BELOW—RIGHTY
3	SINGLE4—SINGLE6—FLIP—BELOW—LEFTY
4	SINGLE4—SINGLE6—NONFLIP—ABOVE—RIGHTY
5	SINGLE4—SINGLE6—NONFLIP—ABOVE—LEFTY
6	SINGLE4—SINGLE6—NONFLIP—BELOW—RIGHTY
7	SINGLE4—SINGLE6—NONFLIP—BELOW—LEFTY
8	SINGLE4—DOUBLE6—FLIP—ABOVE—RIGHTY
9	SINGLE4—DOUBLE6—FLIP—ABOVE—LEFTY
10	SINGLE4—DOUBLE6—FLIP—BELOW—RIGHTY
11	SINGLE4—DOUBLE6—FLIP—BELOW—LEFTY
12	SINGLE4—DOUBLE6—NONFLIP—ABOVE—RIGHTY
13	SINGLE4—DOUBLE6—NONFLIP—ABOVE—LEFTY
14	SINGLE4—DOUBLE6—NONFLIP—BELOW—RIGHTY
15	SINGLE4—DOUBLE6—NONFLIP—BELOW—LEFTY
16	DOUBLE4—SINGLE6—FLIP—ABOVE—RIGHTY
17	DOUBLE4—SINGLE6—FLIP—ABOVE—LEFTY
18	DOUBLE4—SINGLE6—FLIP—BELOW—RIGHTY
19	DOUBLE4—SINGLE6—FLIP—BELOW—LEFTY
20	DOUBLE4—SINGLE6—NONFLIP—ABOVE—RIGHTY
21	DOUBLE4—SINGLE6—NONFLIP—ABOVE—LEFTY
22	DOUBLE4—SINGLE6—NONFLIP—BELOW—RIGHTY
23	DOUBLE4—SINGLE6—NONFLIP—BELOW—LEFTY
24	DOUBLE4—DOUBLE6—FLIP—ABOVE—RIGHTY
25	DOUBLE4—DOUBLE6—FLIP—ABOVE—LEFTY
26	DOUBLE4—DOUBLE6—FLIP—BELOW—RIGHTY
27	DOUBLE4—DOUBLE6—FLIP—BELOW—LEFTY
28	DOUBLE4—DOUBLE6—NONFLIP—ABOVE—RIGHTY
29	DOUBLE4—DOUBLE6—NONFLIP—ABOVE—LEFTY

值	形 态
30	DOUBLE4—DOUBLE6—NONFLIP—BELOW—RIGHTY
31	DOUBLE4—DOUBLE6—NONFLIP—BELOW—LEFTY

4 轴

值	形 态
0	SINGLE—RIGHTY
1	SINGLE—LEFTY
2	
3	
4	
5	
6	
7	
8	DOUBLE—RIGHTY
9	DOUBLE—LEFTY
10	
11	
12	
13	
14	
15	
16	
17	
18	
19	
20	
21	
22	
23	
24	
25	
26	
27	
28	
29	
30	
31	

相关项目

LETF、机械手形态 （附录 2）

应用示例

```
DIM li1 As Integer
li1 = CURFIG                    ’将当前机械手的形态赋值整数型变量 li1。
```

FIGAPRL（函数）

功　能

计算能够执行 CP 动作的接近位置和基准位置的形态。

格　式

FIGAPRL(< 基准位置 >, < 接近长度 >)

说　明

< 基准位置 > 只能使用位置型。

6 轴

6 轴计算从 < 基准位置 > 可以向夹治具坐标系的 -Z 轴方向和从 < 接近长度 > 分离的位置 （接近位置）可以向 < 基准位置 > 进行 CP 动作的 < 基准位置 > 的形态。

从接近位置向 < 基准位置 > 可以进行 CP 动作的形态，按照以下的顺序进行自动探查直至可以移动为止。

1) < 基准位置 > 的形态 （< 基准位置 > 的形态在不定时是当前的形态）
2) 替换手腕形态 （翻转－不翻转）。
3) 用手腕形态不翻转替换肘形态 （蜷起－向下）。
4) 将手腕形态置为翻转。
5) 用肘形态蜷起、手腕形态不翻转替换臂形态 （反手 - 正手）。
6) 将手腕形态置为翻转。
7) 置为肘形态向下、手腕形态不翻转。
8) 将手腕形态置为翻转。

第 8 号的搜索依然不可移动时，会发生 6070 号台 （J* 软件限位超程）错误。在这种情况下 , 请进行改变当前位置或接近长度等的处理。

4 轴

计算从 < 基准位置 > 可以向底座坐标系的 +Z 轴方向和从 < 接近长度 > 分离的位置（接近位置）可以向 < 基准位置 > 进行 CP 动作的 < 基准位置 > 的形态。

从接近位置向 < 基准位置 > 可以进行 CP 动作的形态，按照以下的顺序进行自动探查直至可以移动为止。

1) < 基准位置 > 的形态 （< 基准位置 > 的形态在不定时是当前的形态）

相关项目

FIGAPRP、 APPROACH

注意事项

- 在接近之后的移动动作为 CP 动作时，请使用 FIGAPRL。 PTP 动作时请使用 FIGAPRP。
- 请将 FIGAPRL 所使用的 < 基准位置 >、接近长度和 APPROACH 动作时所指定的 < 基准位置 >、接近长度设成一样的。< 基准位置 >、接近长度不同的情况下，接近之后的 CP 动作时有时会发生 6070 号台的错误。
- 即使用 FIGAPRL 求到了形态的情况下，在接近之后的 CP 动作也会发生 6080 号台 （指令速度限制超程）错误、停止运行。在这种情况下，请降低速度或将最佳可搬运质量设定模式 （参照 p.4-8 "4.6 最佳可搬运质量设定功能 "）设为 2 或 3。尽管如此仍然发生错误时，请调整接近长度。

应用示例

一般

```
I1=FIGAPRL(P1, 100.0)
LETF  P1=I1
APPROACH  P, P1, 100.0
MOVE  L, P1
```

6 轴

```
I1=FIGAPRL(P1 + (100, 200, 0, 0, 10, 0), 100)
LETF  P1=I1
APPROACH  P, P1, 100.0
MOVE  L, P1
```

4 轴

```
I1=FIGAPRL(P1 + (100, 200, 0, 0), 100)
LETF  P1=I1
APPROACH  P, P1, 100.0
MOVE  L, P1
```

FIGAPRP（函数）

功　能

计算能够执行 PTP 动作的接近位置与基准位置的形态。

格　式

FIGAPRP(< 基准位置 >, < 接近长度 >)

说　明

< 基准位置 > 只能使用位置型。

6 轴

计算从 < 基准位置 > 可以向夹治具坐标系的 -Z 轴方向和从 < 接近长度 > 分离的位置 （接近位置）可以向 < 基准位置 > 进行 PTP 动作的 < 基准位置 > 的形态。

从接近位置向 < 基准位置 > 可以进行 PTP 动作的形态，按照以下的顺序进行自动探查直至可以移动为止。

1) < 基准位置 > 的形态 （< 基准位置 > 的形态在不定时是当前的形态）
2) 替换手腕形态 （翻转－不翻转），从接近位置到
   < 基准位置 > 的 4 轴移动角度如果在 ±90 度以内，则可以动作。

4 轴的移动角度超过 ±90 度或不能移动时，发生错误 6070 号台（J* 软件限位超程）。在这种情况下 , 请进行改变当前位置或接近长度等的处理。

4 轴

计算从 < 基准位置 > 可以向底座坐标系的 +Z 轴方向和从 < 接近长度 > 分离的位置（接近位置）可以向 < 基准位置 > 进行 PTP 动作的 < 基准位置 > 的形态。

从接近位置向 < 基准位置 > 可以进行 PTP 动作的形态，按照以下的顺序进行自动探查直至可以移动为止。

1) < 基准位置 > 的形态 （< 基准位置 > 的形态在不定时是当前的形态）

相关项目

FIGAPRL、 APPROACH

注意事项

- 在接近之后的移动动作为 PTP 动作时，请使用 FIGAPRP。 CP 动作时请使用 FIGAPRL。
- 请将 FIGAPRP 所使用的 < 基准位置 >、接近长度和 APPROACH 动作时所指定的 < 基准位置 >、接近长度设成一样的。 < 基准位置 >、接近长度不同时，接近之后的 PTP 动作时有时会发生 6070 号台的错误。

应用示例

```
I1=FIGAPRL(P1, 100.0)
LETF  P1=I1
APPROACH  P, P1, 100.0
MOVE  L, P1
```

6 轴

```
I1 = FIGAPRP (P1 + (100, 200, 0, 0, 0, 10), 100, 0)
```

4 轴

```
I1 = FIGAPRP (P1 + (100, 200, 0, 0), 100, 0)
```

12.3 停止控制

HOLD（语句）【依据 SLIM】

功　能

暂时停止程序的执行。

格　式

HOLD < 讯息 >

说　明

暂时停止程序的执行，将步骤停止状态下在 < 讯息 > 上所指定的内容输出到多功能教导器。有关 < 讯息 > 的内容请参照 PRINTMSG 语句。

如果再启动，程序就会从 HOLD 语句的后面的语句开始执行。

讯息的字符数为半角 60 字符。

相关项目

DELAY、 HALT、 STOP

注意事项

作为讯息如果使用 F 型变量、 D 型变量时，只能表示到小数点后 4 位。

应用示例

```
DEFINT li1, li2
HOLD li1 + li2                ' 暂时停止程序的执行，置为步骤停止状态将
                              '(li1 + li2) 的数值输出到多功能教导器。

HOLD "stop"                   ' 暂时停止程序，置为步骤停止状态将字符串
                              '"stop" 输出到多功能教导器。
```

HALT （语句）【依据 SLIM】

功　　能

停止程序的执行。

格　　式

HALT < 讯息 >

说　　明

停止程序的执行，将 < 讯息 > 上所指定的内容输出到多功能教导器。有关 < 讯息 > 的内容请参照 PRINTMSG 语句。

如果再启动时，程序从它的开头执行。

讯息的字符数为半角 60 字符。

相关项目

DELAY、 HOLD、 STOP

注意事项

作为讯息如果使用 F 型变量、 D 型变量时，只能表示到小数点后 4 位。

应用示例

```
DEFINT li1, li2
HALT li1 + li2                  ' 停止程序，将 (li1 + li2) 的数值输出到
                                ' 多功能教导器。
HALT "end"                      ' 停止程序，将字符串 "end" 输出到多功能教导器。
```

INTERRUPT ON/OFF（语句）

功　能

中断机械手的动作。

格　式

INTERRUPT {ON|OFF}

说　明

INTERRUPT ON 和 INTERRUPT OFF 被成对使用，在被其包围的范围内，动作命令执行过程中中止跳跃信号 ON 时，中断动作命令之后，进入下一个步骤。

为了执行 INTERRUPT 指令，任务需要获取臂组。

如果执行 INTERRUPT ON 语句不置为可以中止的状态，则中止跳跃信号即使置为 ON，正在执行的动作命令也不会结束。

程序处于停止状态，或执行 GIVEARM 命令时， INTERRUPT 自动 OFF。

相关项目

注意事项

- 如果中止跳跃之后立即执行相对动作命令，则为从已停止位置的相对运行。由 INTERRUPT 所包围的范围内的动作命令请使用绝对动作命令。
- INTERRUPT ON 中不进行通过 (Pass) 动作。
  指定通过 (Pass) 动作时，全部以末端动作执行。
- 由 INTERRUPT 所包围的范围内的动作命令，通过中止跳跃信号的输入，停止所有的动作。在多项任务状态动作时需要注意。
  例

	J1	J2	J3	J4	J5	J6	J7	J8
Group 0	○	○	○	○	×	×	×	×
Group 1	×	×	×	×	×	×	○	○
Group 2	○	○	○	○	×	×	○	○
Group 3	×	×	×	×	×	×	×	×
Group 4	×	×	×	×	×	×	×	×

```
PRO1
TAKEARM   0
INTERRUPT ON
MOVE P,P0          '动作命令
INTERRUPT OFF
END
```

```
PRO2
TAKEARM   1
INTERRUPT ON
DRIVE(7,30)        '动作命令
INTERRUPT OFF
END
```

PRO1、PRO2 同时执行动作命令过程中，中止跳跃信号 ON 时，PRO1 的 MOVE 命令、PRO2 的 DRIVE 命令的任何一个均可中断动作，进入下一个步骤。

应用示例

例 1:

```
DIM lp1 As Position
INTERRUPT ON                    ' 专用 I / O 端口的中止信号被置为 ON 时 ,
                                ' 中断正在执行的运行动作之后，进入下一个步骤。
MOVE P, lp1
INTERRUPT OFF
```

例 2 （附加轴的示例）：

	J1	J2	J3	J4	J5	J6	J7	J8
Group 1	×	×	×	×	×	×	○	○

```
PROGRAM PRO1
TAKEARM 1                       ' 获取臂组 1 （包括 7 轴、 8 轴的臂组）
INTERRUPT ON
DRIVE (7,100),(8,30)            ' 在执行被 INTERRUPT 命令包围的范围的动作命令过程中，
                                ' 中止跳跃信号置为 ON 时，中断动作命令之后，
                                ' 进入下一个步骤。
INTERRUPT OFF
END
```

12.4 速度控制

SPEED（语句）【依据 SLIM】

功　能

指定当前所获取的臂组轴的内部移动速度。

格　式

SPEED < 移动速度 >

说　明

在 < 移动速度 > 中，是用 0.1 ～ 100 的实数指定当前所获取的臂组轴的最大内部移动速度的比率 (%) 的。

注意

如果指定了 0.1 以下的数值，比 0 大时虽然不会发生错误，但会有与实际的速度不同的情况。

实际的速度值是 （外部 × 内部 ÷ 100）。

如果设定 SPEED，则 JSPEED、 ACCEL、 DECEL、 JACCEL、 JDECEL 的值也会被自动设定。

	CP 控制	PTP 控制
设定速度	SPEED	JSPEED
设定加速度	ACCEL	JACCEL
设定减速度	DECEL	JDECEL
获取当前速度	CURSPD	CURJSPD
获取当前加速度	CURACC	CURJACC
获取当前减速度	CURDEC	CURJDEC

例 :SPEED 50 的时候被设定为以下的数值。

JSPEED	50	(与 SPEED 相同值)
ACCEL	25	(SPEED ＊ SPEED÷100)
JACCEL	25	(SPEED ＊ SPEED÷100)
DECEL	25	(SPEED ＊ SPEED÷100)
JDECEL	25	(SPEED ＊ SPEED÷100)

不设定臂组就设定速度的情况下，会发生错误。

相关项目

ACCEL、 DECEL、 JSPEED、 MPS

应用示例

例 1:

```
DIM li1 As Integer
SPEED 100                    ' 将夹治具前端的移动速度设定为 100。
SPEED li1/100                ' 将夹治具前端的移动速度设定为 (li1 / 100) 的数值。
```

例 2 （附加轴的示例）：

	J1	J2	J3	J4	J5	J6	J7	J8
Group 1	×	×	×	×	×	×	○	○

```
PROGRAM PRO1
TAKEARM 1                    ' 获取臂组 1 （包括 7 轴、 8 轴的臂组）
SPEED 100                    ' 臂组的轴 （7 轴、 8 轴）的速度被设定。
END
```

JSPEED（语句）

功　能

指定当前所获取的臂组轴的内部轴速度。

格　式

JSPEED < 轴速度 >

说　明

在 < 轴速度 > 中，是用 0.1 ～ 100 的实数指定当前所获取的臂组轴的最大内部轴速度的比率 (%) 的。

注意

如果指定了 0.1 以下的数值，比 0 大时虽然不会发生错误，但会有与实际的速度不同的情况。

实际的速度值是 （外部 × 内部 ÷ 100）。

如果设定 JSPEED，则 JACCEL、 JDECEL 的值也会被自动设定。

例 : JSPEED50 的时候被设定为以下的数值。

JACCEL 25 (JSPEED ＊ JSPEED÷100)
JDECEL 25 (JSPEED ＊ JSPEED÷100)

不设定臂组就设定速度的情况下，会发生错误。

相关项目

CURJSPD、 JACCEL、 JDECEL、 SPEED

应用示例

例 1:

```
DIM li1 As Integer
JSPEED 100                  ' 将轴的移动速度设定为 100。
JSPEED li1/100              ' 将轴的移动速度设定为 (li1 / 100) 的数值。
```

例 2 （附加轴的示例）：

	J1	J2	J3	J4	J5	J6	J7	J8
Group 1	×	×	×	×	×	×	○	○

```
PROGRAM PRO1
TAKEARM 1                   ' 获取臂组 1 （包括 7 轴、 8 轴的臂组）
JSPEED 100                  ' 只有臂组的轴 （7 轴、 8 轴）的轴速度被设定。
END
```

ACCEL （语句）【依据 SLIM】

功　能

指定当前所获取的臂组轴的内部加速度、内部减速度。

格　式

ACCEL < 加速度 > [,< 减速度 >]

说　明

在 < 加速度 >、< 减速度 > 中，是用 0.0001 ～ 100 的实数指定当前所获取的臂组轴的最大内部加速度、减速度的比率 (%) 的。

注意

如果指定了 0.0001 以下的数值，比 0 大时虽然不会发生错误，但会有与实际的加速度不同的情况。

实际的加减速度值是 （外部 × 内部 ÷ 100）。

加减速度值如果变更速度，就会自动变更为 (SPEED^2÷ 100) 的式的数值。

附加轴，如果设定 ACCEL 则 JACCEL （轴加速度）也被设定为相同值。

例 : 如果设定 ACCEL50，则 JACCEL50 （相同值）也被设定。

不设定臂组就设定加速度的情况下，会发生错误。

相关项目

DECEL、 SPEED

应用示例

例 1:

```
DEFINT li1, li2
ACCEL 100                    ' 将加速度设定为 100。
ACCEL 50, 25                 ' 将加速度设定为 50，减速度为 25。
ACCEL li1/100, li2/100       ' 将加速度设定为 (li1 / 100)、减速度为 (li2 / 100) 的数值。
```

例 2 （附加轴的示例）：

	J1	J2	J3	J4	J5	J6	J7	J8
Group 1	×	×	×	×	×	×	○	○

```
PROGRAM PRO1
TAKEARM 1                    ' 获取臂组 1 （包括 7 轴、 8 轴的臂组）
ACCEL 100                    ' 臂组的轴 （7 轴、 8 轴）的加速度被设定。
END
```

JACCEL（语句）【依据 SLIM】

功　能

指定当前所获取的臂组轴的内部轴加速度、内部轴减速度。

格　式

JACCEL < 轴加速度 > [,< 轴减速度 >]

说　明

在 < 轴加速度 >、 < 轴减速度 > 中，是用 0.0001 ～ 100 的实数指定当前所获取的臂组轴的最大内部轴加速度、轴减速度的比率 (%) 的。

注意

如果指定了 0.0001 以下的数值，比 0 大时虽然不会发生错误，但会有与实际的速度不同的情况。

实际的加减速度值是 （外部 × 内部 ÷ 100）。

轴加减速度值如果变更轴速度，就会自动变更为 (JSPEED^2÷ 100) 的式的数值。

不设定臂组就设定内部轴加速度的情况下，会发生错误。

相关项目

JDECEL、 JSPEED、 SPEED

应用示例

例 1:

```
DEFINT li1, li2
JACCEL 100                  ' 将轴加速度设定为 100。
JACCEL 50, 25               ' 将轴加速度设定为 50，轴减速度为 25。
JACCEL li1/100, li2/100     ' 将轴加速度设定为 (li1 / 100)、轴减速度为
                            ' (li2 / 100) 的数值。
```

例 2 （附加轴的示例）：

	J1	J2	J3	J4	J5	J6	J7	J8
Group 1	×	×	×	×	×	×	○	○

```
PROGRAM PRO1
TAKEARM 1                   ' 获取臂组 1 （包括 7 轴、 8 轴的臂组）
JACCEL 100                  ' 臂组的轴 （7 轴、 8 轴）的轴加速度被设定。
END
```

DECEL（语句）【依据 SLIM】

功　能

指定当前所获取的臂组轴的内部减速度。

格　式

DECEL < 减速度 >

说　明

在 < 减速度 > 中，是用 0.0001 ～ 100 的实数指定当前所获取的臂组轴的最大内部减速度的比率 (%) 的。

注意

如果指定了 0.0001 以下的数值，比 0 大时虽然不会发生错误，但会有与实际的减速度不同的情况。

实际的减速度值是 （外部 × 内部 ÷ 100）。

减速度值如果变更速度，就会自动变更为 (SPEED^2÷ 100) 的式的数值。

附加轴，如果设定 DECEL 则 JDECEL （轴加速度）也被设定为相同值。

例 : 如果设定 DECEL 50，则 JDECEL 50 （相同值）也被设定。

不设定臂组就设定减速度的情况下，会发生错误。

相关项目

ACCEL、 SPEED

应用示例

例 1:

```
DIM li1 As Integer
DECEL 100                    ' 将减速度设定为 100。
DECEL li1/100                ' 将减速度设定为 (li1 / 100) 的数值。
```

例 2 （附加轴的示例）：

	J1	J2	J3	J4	J5	J6	J7	J8
Group 1	×	×	×	×	×	×	○	○

```
PROGRAM PRO1
TAKEARM 1                    ' 获取臂组 1 （包括 7 轴、 8 轴的臂组）
DECEL 100                    ' 臂组的轴 （7 轴、 8 轴）的减速度被设定。
END
```

JDECEL（语句）【依据 SLIM】

功　能

指定当前所获取的臂组轴的内部轴减速度。

格　式

JDECEL < 轴减速度 >

说　明

在 < 轴减速度 > 中，是用 0.0001 ～ 100 的实数指定当前所获取的臂组轴的最大内部轴减速度的比率 (%) 的。

注意

如果指定了 0.0001 以下的数值，比 0 大时虽然不会发生错误，但会有与实际的速度不同的情况。

实际的减速度值是 （外部 × 内部 ÷ 100）。

轴减速度值如果变更轴速度，就会自动变更为 (JSPEED^2÷ 100) 的式的数值。

不设定臂组就设定内部轴减速度的情况下，会发生错误。

相关项目

JACCEL、 JSPEED、 SPEED

应用示例

例 1:

```
DIM li1 As Integer
JDECEL 100                  ' 将轴减速度设定为 100。
JDECEL li1/100              ' 将轴减速度设定为 (li1 / 100) 的数值。
```

例 2 （附加轴的示例）：

	J1	J2	J3	J4	J5	J6	J7	J8
Group 1	×	×	×	×	×	×	○	○

```
PROGRAM PRO1
TAKEARM 1                   ' 获取臂组 1 （包括 7 轴、 8 轴的臂组）
JDECEL 100                  ' 臂组的轴 （7 轴、 8 轴）的轴减速度被设定。
END
```

CURACC（系统变量）

功　能

获取当前所获取的臂组轴的内部加速度。

格　式

CURACC

说　明

在没有获取臂组的任务状态下执行 CURACC 指令时，返还 100。

相关项目

ACCEL、 DECEL、 CURDEC、 CURJACC、 CURJDEC、 CURJSPD、 CURSPD

应用示例

例 1:

```
DIM lf1 As Single
lf1 = CURACC                    ' 将加速度的当前值赋值 lf1。
ACCEL CURACC * 0.5              ' 将加速度设为当前的 50%。
```

例 2（附加轴的示例）:

```
PROGRAM PRO1
TAKEARM 1                       ' 获取臂组 1
ACCEL 70                        ' 将臂组 1 的加速度设定为 70
I0 = CURACC                     ' 将臂组 1 的加速度 70 赋值 I0。
END
```

CURJACC（系统变量）

功　能

功能获取当前所获取的臂组轴的内部轴加速度。

格　式

CURJACC

说　明

在没有获取臂组的任务状态下执行 CURJACC 指令时，返还 100。

相关项目

JACCEL、 JDECEL、 JSPEED、 SPEED、 CURDEC、 CURJDEC、 CURJSPD、 CURSPD

应用示例

例 1:

```
DIM lf1 As Single
lf1 = CURJACC                    ' 将轴加速度的当前值赋值 lf1。
JACCEL CURJACC * 0.5             ' 将轴加速度设为当前的 50%。
```

例 2（附加轴的示例）:

```
PROGRAM PRO1
TAKEARM 1                        ' 获取臂组 1
JACCEL 70                        ' 将臂组 1 的轴加速度设定为 70
I0 = CURJACC                     ' 将臂组 1 的轴加速度 70 赋值 I0。
END
```

CURDEC（系统变量）

功　能

获取当前所获取的臂组轴的内部减速度。

格　式

CURDEC

说　明

在没有获取臂组的任务状态下执行 CURDEC 指令时，返还 100。

相关项目

ACCEL、 DECEL、 CURACC、 CURJACC、 CURJDEC、 CURJSPD、 CURSPD

应用示例

例 1:

```
DIM lf1 As Single
lf1 = CURDEC                    ' 将减速度的当前值赋值 lf1。
DECEL CURDEC * 0.5              ' 将减速度设为当前的 50%。
```

例 2（附加轴的示例）:

```
PROGRAM PRO1
TAKEARM 1                       ' 获取臂组 1
DECEL 70                        ' 将臂组 1 的减速度设定为 70
I0 = CURDEC                     ' 将臂组 1 的减速度 70 赋值 I0。
END
```

CURJDEC（系统变量）

功　能

获取当前所获取的臂组轴的内部轴减速度。

格　式

CURJDEC

说　明

在没有获取臂组的任务状态下执行 CURJDEC 指令时，返还 100。

相关项目

JACCEL、 JDECEL、 JSPEED、 SPEED、 CURACC、 CURDEC、 CURJACC、 CURJSPD、 CURSPD

应用示例

例 1:

```
DIM lf1 As Single
lf1 = CURDEC                    ' 将减速度的当前值赋值 lf1。
DECEL CURDEC * 0.5              ' 将减速度设为当前的 50%。
```

例 2（附加轴的示例）:

```
PROGRAM PRO1
TAKEARM 1                       ' 获取臂组 1
JDECEL 70                       ' 将臂组 1 的轴减速度设定为 70
I0 = CURJDEC                    ' 将臂组 1 的轴减速度 70 赋值 I0。
END
```

CURJSPD（系统变量）

功　能

获取当前所获取的臂组轴的内部移动轴速度。

格　式

CURJSPD

说　明

在没有获取臂组的任务状态下执行 CURJSPD 指令时，返还 100。

相关项目

JSPEED、 CURACC、 CURDEC、 CURJACC、 CURJDEC、 CURSPD

应用示例

例 1:

```
DIM lf1 As Single
lf1 = CURJSPD                    ' 将轴的移动速度的当前值赋值 lf1。
JSPEED CURJSPD * 0.5             ' 将轴的移动速度设为当前的 50%。
```

例 2（附加轴的示例）:

```
PROGRAM PRO1
TAKEARM 1                        ' 获取臂组 1
JSPEED 70                        ' 将臂组 1 的轴速度设定为 70
I0 = CURJSPD                     ' 将臂组 1 的轴速度 70 赋值 I0。
END
```

CURSPD（系统变量）

功　能

获取当前所获取的臂组轴的内部移动速度。

格　式

CURSPD

说　明

在没有获取臂组的任务状态下执行 CURSPD 指令时，返还 100。

相关项目

CURACC、 CURDEC、 CURJSPD、 CURJACC、 CURJDEC

应用示例

例 1:

```
DIM lf1 As Single
lf1 = CURJSPD                    ' 将轴的移动速度的当前值赋值 lf1。
JSPEED CURJSPD * 0.5             ' 将轴的移动速度设为当前的 50%。
```

例 2（附加轴的示例）:

```
PROGRAM PRO1
TAKEARM 1                        ' 获取臂组 1
SPEED 70                         ' 将臂组 1 的速度设定为 70
I0 = CURSPD                      ' 将臂组 1 的速度 70 赋值 I0。
END
```

CUREXTACC（系统变量）【Ver.1.4 以上版本】

功　能

得出外部加速度的当前值。

格　式

CUREXTACC

说　明

当前所设定的外部加速度被存放。

相关项目

ACCEL、 DECEL、 CUREXTDEC

应用示例

```
DIM 1f1 As Single
1f1 = CUREXTACC
ACCEL CUREXTACC * 0.5
```

CUREXTDEC（系统变量）【Ver.1.4 以上版本】

功　能

得出外部减速度的当前值。

格　式

CUREXTDEC

说　明

当前所设定的外部减速度被存放。

相关项目

ACCEL、 DECEL、 CUREXTACC

应用示例

```
DIM 1f1 As Single
1f1 = CUREXTDEC
DECEL CUREXTDEC * 0.5
```

CUREXTSPD（系统变量）【Ver.1.4 以上版本】

功　能

得出外部速度的当前值。

格　式

CUREXTSPD

说　明

当前所设定的外部速度被存放。

相关项目

SPEED

应用示例

```
DIM 1f1 As Single
1f1 = CUREXTSPD
SPEED CUREXTSPD * 0.5
```

EXTSPEED（语句）【Ver.1.98 以上版本】

功　能

外部速度设定

格　式

EXTSPEED < 外部速度设定值 >

说　明

设定外部速度。

补充说明

通过用该指令也可以从程序中进行外部速度的设定。

相关项目

SPEED

应用示例

```
PROGRAM  PRO1
------------------------
EXTSPEED 100                    ' 将外部速度设定值设为 100
------------------------
END
```

12.5 时间控制

DELAY（语句）【依据 SLIM】

功　能

在指定的时间内，停止程序的执行。

格　式

DELAY < 延迟时间 >

说　明

等待 < 延迟时间 > 所指定的时间经过。

< 延迟时间 > 的单位是 ms，但实际的延迟时间是以 1 / 60s 为单位进行增减。此外，如果同时执行多个任务时，延迟时间有时会比指定值延长。

相关项目

HOLD、 WAIT

注意事项

- 在使用延迟时间时，如果在命令执行过程中进行瞬时停止而再启动，则要注意在暂时停止过程中也经过延迟时间。
- 实际的延迟时间有 16ms 左右的误差。

实际延迟时间的计算方法

x：延迟时间设定值、y：实际延迟时间

$y_1 = \dfrac{x \times 60 + 500}{1000}$ ，舍去小数点以下的。此外，当为 0 时，将其视为 1

$y = \dfrac{y_1 \times 1000}{60}$

实际的延迟时间为最小 $y - \dfrac{1000}{60}$ ms 到最大 y ms 之间。

例）在将延迟时间设定值设定为 30ms 时

$y_1 = \dfrac{30 \times 60 + 500}{1000} = 2.3$ ，舍去小数点以下的。因此为 2

$y = \dfrac{2 \times 1000}{60} = 33.33$

实际延迟时间为 $\dfrac{1000}{60}$ ms 到 $\dfrac{2000}{60}$ ms 之间。

- 当希望延迟至一定指定时间以上时，设定值的计算方法如下所示。

  x：希望必须延迟的时间、y：设定值

  $x_1 = \dfrac{x}{\left(\dfrac{1000}{60}\right)}$　　‘舍去小数点以下的。

  $y = x_1 \times \dfrac{1000}{60} + 25$　‘进上小数点以下的。

  例：需要延迟 30ms 以上时

  $x_1 = \dfrac{30}{\left(\dfrac{1000}{60}\right)} = 1.8$　　‘舍去小数点以下的，变为 1。

  $y = 1 \times \dfrac{1000}{60} + 25 = 41.666$　‘进上小数点以下的，变为 42。

  将设定值设置为 42（DELAY 42）时，则能够确保实际的延迟时间在 30ms 以上。

应用示例

```
DIM li1 As Integer
DELAY 100                    ’等待 100ms (0.1s) 的时间经过。
DELAY li1 + 10               ’等待 (li1 + 10) 的数值的时间经过。
```

WAIT （语句）【依据 SLIM】

功　能

执行条件程序的停止。

格　式

WAIT < 条件公式 > [,< 超时时间 >][,< 存放变量 >]

说　明

到 < 条件公式 > 成立为止停止程序的执行。

一旦设定 < 超时时间 >，在经过指定时间之后结束 WAIT 语句的执行，控制移到下一个指令。通过这样，可以防止无限期停止。

< 超时时间 > 的单位是 ms。

为了监督 < 条件公式 > 或 < 超时时间 > 的条件成立再评估的间隔取决于任务的优先度。

在［Ver.1.8 以上版本］中，一旦设定 < 存放变量 > 条件公式成立， TRUE (1) 进入遗漏 WAIT 语句时所指定的变量， FALSE (0) 进入在超时遗漏时所指定的变量。

相关项目

DELAY

注意事项

在使用超时时间时，如果在命令执行过程中进行瞬时停止而再启动，则要注意在暂时停止过程中也经过指定时间。

应用示例

```
DEFINT li1, li2, li3, li4, li5
WAIT li1 = 1                    ' 等待 li1 = 1 成立。
WAIT li2 = 0, 2000              ' 等待 li2 = 0 成立，经过 2 秒
                                ' 如果依然没有成立移到下一个语句。

WAIT li3 = li4, li5             ' 等待 li3 = li4 成立，经过 li5
                                ' 的数值的时间如果依然没有成立
                                ' 移到下一个语句。
WAIT IO[10] = ON                ' 等待 IO 的 10 号至 ON。

［Ver.1.8 以降］
WAIT li3 = li4, li5, li6        ' 等待 li3 = li4 成立，经过 li5
                                ' 的数值的时间如果依然没有成立
                                ' 移到下一个语句。这时，
                                ' 如果到 li5 的数值的时间为止
                                'li3 = li4 成立 li6 = TRUE 进入，
                                ' 如果条件公式没有成立而超时
                                'li6 = FALSE 进入。
```

12.6 坐标转换

CHANGETOOL（语句）

功　能

变更工具坐标系。

格　式

CHANGETOOL < 工具坐标系编号 >

说　明

将工具坐标系变更为 < 工具坐标系编号 > 所指定的坐标系。

该定义直至下一次执行 CHANGETOOL 语句前有效。

< 工具坐标系编号 > 在 0 ～ 63 是有效的。

工具 0 是机械界面坐标系。

相关项目

TOOL、 TOOLPOS

注意事项

如果执行 TAKEARM，则 TOOL0 被设定。

应用示例

```
DIM li1 As Integer
DIM lp1 As Position
Li1 = 1
Lp1 = (10, 10, 5, 0, 9, 0, 1)
TOOL 1, lp1                    ' 在工具坐标系编号 1 上，定义用位置型变量 lp1
                               ' 所表示的工具坐标系。
CHANGETOOL li1                 ' 变更为用 li1 所表示的工具坐标系编号的工具坐标系。
```

CHANGEWORK（语句）

功　能

变更用户坐标系。

格　式

CHANGEWORK < 用户坐标系编号 >

说　明

将用户坐标系变更为 < 用户坐标系编号 > 所指定的坐标系。

该定义直至下一次执行 CHANGEWORK 语句前有效。

< 用户坐标系编号 > 在 0 ～ 7 是有效的。

用户 0 是底座坐标系。

相关项目

WORK、 WORKPOS

注意事项

- 如果执行 TAKEARM，则 WORK0 被设定。（但是，子程序没有这种限制。）
- 利用 "WORK" 指令改变现在所选择之编号的工件坐标系值，机械手的工件坐标系也不会发生改变。请利用 "CHANGEWORK" 再次选择。

应用示例

```
DIM li1 As Integer
DIM lp1 As Position
Li1 = 1
Lp1 = (10, 10, 5, 0, 9, 0, 1)
WORK 1, lp1                  ' 在用户坐标系编号 1 上，定义用位置型变量 lp1
                             ' 所表示的用户坐标系。
CHANGEWORK li1               ' 变更为用 li1 所表示的用户坐标系编号的用户坐标系。
```

CURTOOL（系统变量）【Ver.1.4 以上版本】

功　　能

得出当前所设定的 TOOL 编号。

格　　式

CURTOOL

说　　明

当前所设定的 TOOL 编号被存放。

相关项目

TOOL、 CHANGETOOL、 TOOLPOS

应用示例

```
DIM li1 As Integer
li1 = CURTOOL
TOOL CURTOOL+I1, P1
CHANGETOOL CURTOOL+I1
P1 = TOOLPOS (CURTOOL)
```

CURWORK（系统变量）【Ver.1.4 以上版本】

功　　能

得出当前所设定的 WORK 编号。

格　　式

CURWORK

说　　明

当前所设定的 WORK 编号被存放。

相关项目

WORK、 CHANGEWORK、 WORKPOS

应用示例

```
DIM li1 As Integer
li1 = CURWORK
WORK CURWORK+I1, P1
CHANGEWORK CURWORK+I1
P1 = WORKPOS (CURWORK)
```

12.7　干扰检查

SETAREA（语句）

功　能

选择进行干扰检查的区域。

格　式

SETAREA < 干扰检查区域编号 >

说　明

< 干扰检查区域编号 > 选则预先定义的干扰检查区域。有效的区域编号是 0 ～ 31。通过 AREA 指令必须事先被定义。

执行 SETAREA 指令之后，到 RESETAREA 指令被执行中间，进行干扰检查。一旦干扰被检测出，到执行 RESETAREA 指令，再次 SETAREA 为止，不检测下一个干扰。

干扰检查，如果 SETAREA 指令被执行会用 0 ～ 31 的区域编号所表示，对于最大 32 个区域可以同时进行。

再者，SETAREA 指令不对用 AREA 指令所定义的指定 I / O 进行初始化。根据输出入控制语句直接设定 I / O 等时，需要用 RESETAREA 指令进行初始化。

相关项目

AREA、 RESETAREA、 AREAPOS、 AREASIZE

应用示例

```
DIM lp1 As Position
DIM lv1 As Vector
Lp1 = (10, 10, 5, 0, 9, 0, 1)
Lv1 = (50, 10, 50)
AREA 2, lp1, lv1, 104, 1        ' 在 2 号的区域，定义用 lp1 所表示的位置，
                                ' 用 lv1 所表示的区域定义 I / O 编号 104。
SETAREA 2                       ' 将 2 号的区域检查置为有效。
RESETAREA 2                     ' 将 2 号的区域检查进行初始化。
```

RESETAREA（语句）

功　能

初始化干扰检查。

格　式

RESETAREA < 初始化区域编号 >

说　明

在查出干扰时将设定的 I / O 重新进行设定，将干扰检查无效化。

根据输出入控制语句直接复位 I / O 时，即便发生再次干扰 I / O 也不被设定。为了进行再次干扰检查，执行 RESETAREA 指令进行初始化，再次执行 SETAREA 指令。

有效的区域编号是 0 ～ 31。

相关项目

AREA、 SETAREA、 AREAPOS、 AREASIZE

注意事项

因为是以多项任务动作，执行了 SETAREA 立即就有 RESET 命令的话，根据检查区域的时机有可能 RESET 这项先被执行。

应用示例

```
DIM lp1 As Position
DIM lv1 As Vector
Lp1 = (10, 10, 5, 0, 9, 0, 1)
Lv1 = (50, 10, 50)
AREA 2, lp1, lv1, 104, 1        ' 在 2 号的区域，定义用 lp1 所表示的位置，
                                ' 用 lv1 所表示的区域定义 I / O 编号 104。
SETAREA 2                       ' 将 2 号的区域检查置为有效。
RESETAREA 2                     ' 将 2 号的区域检查进行初始化。
```

SETPOSTUREAREA（语句）【Ver.3.0 以上版本】

功　　能

将高感度位置姿势检测功能设为有效。

格　　式

SETPOSTUREAREA

说　　明

将利用 POSTUREAREA 设定的高感度位置姿势检测功能设为有效。

执行 SETPOSTUREAREA 指令后，在 RESETPOSTUREAREA 指令执行之前，就会进行高感度位置姿势检测。

利用 SETPOSTUREAREA 指令无法将利用 POSTUREAREA 指令定义的 I/O 初始化。

相关项目

POSTUREAREA、 RESETPOSTUREAREA

注意事项

高感度位置姿势检测功能为可选功能。请购买功能许可证。

应用示例

```
POSTUREAREA, P1,V0,24,-1,0, TAR_POSTURE ( V2 , 1 ), TAR_EXJOINT ( ( 7 , 30 , 4 ), ( 8 , 90 , 1) )
SETPOSTUREAREA
```

RESETPOSTUREAREA（语句）【Ver.3.0 以上版本】

功　　能

将高感度位置姿势检测功能设为无效。

格　　式

RESETPOSTUREAREA

说　　明

将利用 SETPOSTUREAREA 生效的高感度位置姿势检测功能设为无效。

执行 SETPOSTUREAREA 指令后，在 RESETPOSTUREAREA 指令执行之前，就会进行高感度位置姿势检测。

利用 RESETPOSTUREAREA 指令，将利用 POSTUREAREA 指令定义的 I/O 置于 OFF。

相关项目

POSTUREAREA、 SETPOSTUREAREA

注意事项

高感度位置姿势检测功能为可选功能。请购买功能许可证。

应用示例

```
POSTUREAREA, P1,V0,24,-1,0, TAR_POSTURE ( V2 , 1 ), TAR_EXJOINT ( ( 7 , 30 , 4 ), ( 8 , 90 , 1) )
SETPOSTUREAREA
MOVE P,P1
RESETPOSTUREAREA
```

12.8 伺服内部数据

GetSrvData（系统变量）【Ver.1.5 以上版本】

功　能

获取机械手轴的伺服内部数据。（对应 Ver. 1.5 以上版本）

格　式

< 伺服内部数据 > = GetSrvData(< 数据编号 >)

说　明

获取用 < 数据编号 > 所指定的伺服内部数据。

< 伺服内部数据 > 用关节型的数据安装机械手轴的数据。通过数据编号可以获取以下的数据。

< 数据编号 >	< 伺服内部数据 >
1	电机速度当前值 [rpm]
2	电机角度偏差 [Pulse]
4	电机电流绝对值 ［额定比 %］
5	电机力矩指令值 （重力补偿量除外）［额定比 %］
8	各轴位置、角度指令值 ［mm 或 deg］
17	工具端速度 （工件坐标系、只有位置 3 成分） [mm / s]
18	工具端偏差 （工件坐标系，只有位置 3 成分） [mm]
19	工具端速度 （工具坐标系、只有位置 3 成分） [mm / s]
20	工具端偏差 （工具坐标系，只有位置 3 成分） [mm]

相关项目

GetJntData

注意事项

- 上述以外的数据编号为预约编号。即使指定到 30 也不会发生错误，但请不要指定。
- 如果在伺服单轴监视器执行过程中获取伺服数据，则有时会大幅度延长获取时间。与伺服单轴监视器功能并用时请予注意。
- 如果变更 < 数据编号 >，则有时会需要一定的变更时间。请将 < 数据编号 > 的切换设定为最小限。
- 请通过获取 (TAKEARM) 机械手控制权的任务执行。尚未获取机械手控制权的情况下，会发生 " 不能获取 21F7 臂信号 (semaphore)" 错误。

应用示例

```
defjnt vel
defsng absv,xvel,yvel,zvel
vel=GetSrvData(17)                          ' 获取工具端速度。
xvel=JOINT(1,vel)                           ' 抽出工件坐标 X 成分
yvel=JOINT(2,vel)                           ' 抽出工件坐标 Y 成分
zvel=JOINT(3,vel)                           ' 抽出工件坐标 Z 成分
absv = SQR(xvel*xvel+yvel*yvel+zvel*zvel)   ' 工具端合成速度计算
```

GetJntData（系统变量）

功　能

获取指定轴的伺服内部数据。（对应 Ver. 1.5 以上版本）

格　式

< 指定轴伺服内部数据 > = GetJntData(< 数据编号 >、 < 轴编号 >)

说　明

获取用 < 数据编号 > 和 < 轴编号 > 所指定的指定轴的伺服内部数据。

< 指定轴伺服内部数据 > 是 F 型 （单精度实数型）的数据，指定轴的数据被设定。通过数据编号可以获取以下的数据。

< 数据编号 >	< 伺服内部数据 >
1	电机速度当前值 [rpm]
2	电机角度偏差 [Pulse]
4	电机电流绝对值 ［额定比 %］
5	电机力矩指令值 （重力补偿量除外）［额定比 %］
8	各轴位置、角度指令值 ［mm 或 deg］

相关项目

GetSrvData

注意事项

- 上述以外的数据编号为预约编号。即使指定到 30 也不会发生错误，但请不要指定。
- 如果在伺服单轴监视器执行过程中获取伺服数据，则有时会大幅度延长获取时间。与伺服单轴监视器功能并用时请予注意。
- 如果变更 < 数据编号 >，则有时会需要一定的变更时间。请将 < 数据编号 > 的切换设定为最小限。
- 请通过获取　(TAKEARM)　控制权的任务执行。尚未获取控制权的情况下，会发生　" 不能获取信号 (semaphore)" 的错误。

应用示例

```
defsng vel
vel=GetJntData(1,7)                    ’获取 7 轴的电机速度。
```

12.9 电机电源

MOTOR {ON | OFF}（语句）【Ver.1.98 以上版本】

功　能

执行电机电源的 ON / OFF。

格　式

MOTOR {ON | OFF} [, <MOTOR 选件 >]

说　明

进行电机电源的 ON、 OFF。

如果用 <MOTOR 选件 > 指定 "RESULT", 在执行 MOTOR 指令时，即使在内部发生错误也不会停止程序（但是 MOTOR OFF 指令为安全起见，将发出错误讯息，程序将停止。）。届时的内部错误的错误代码，将储存到用 RESULT= （变量名）指定的变量中。（仅限于 Ver.2.6 以后的控制器）)

相关项目

INIT

注意事项

MOTOR OFF 命令与教导器上的电机 OFF 键进行完全相同的动作。停止机械手动作以及程序执行，处于连续停止的状态。

使用条件的 296: 将电机指令设定变更为 0 → 1 时，如果执行 MOTOR OFF 命令，则在程序执行过程中可以将电机 OFF，而程序不停止。但是，在机械手动作过程中不能执行 MOTOR OFF 命令。如果错误地在机械手动作过程中执行，则会发生错误停止程序。

MOTOR ON 指令，即使变更电机指令设定，动作也不会改变。

应用示例

```
PROGRAM  PRO1
------------
MOTOR  ON                       ' 电机电源 ON
------------
MOTOR  OFF                      ' 电机电源 OFF
------------
MOTOR ON, RESULT = I1           ' 将电机电源置于 ON。
                                'MOTOR ON 失败时，将错误代码储存到变量 I1 中，
                                ' 然后移到下一行。
------------
MOTOR OFF ,RESULT = I2          ' 将电机电源置于 OFF。
                                'MOTOR OFF 失败时，将错误代码储存到变量 I2 中。

                                ' 显示错误讯息，程序停止。
------------
END
```

12.10 CAL

EXECAL（语句）【Ver.1.98 以上版本】

功　能

CAL 执行

格　式

EXECAL

说　明

执行 CAL。

相关项目

INIT

注意事项

像 "E 系列 " 机械手那样、在不需要 CAL 的机械手上不会有任何反映。

应用示例

```
PROGRAM  PRO1
------------
EXECAL                          ' 执行 CAL
------------
END
```

12.11 特殊控制

将原来作为与伺服相关的 PAC 共用程序库 (Library) 使用的语句追加为指令（语句），实现了易操作性的提高。

ST_aspACLD（语句）【Ver.1.9 以上版本】

功　能

变更内部负荷条件值。以前端负荷质量 (g)、负荷重心位置 (mm)（注 1）指定所有负荷条件值。

格　式

ST_aspACLD < 前端负荷质量 >，< 前端负荷重心位置 X 坐标 >，< 前端负荷重心位置 Y 坐标 >，< 前端负荷重心位置 Z 坐标 >（注 1）

注 1: 4 轴机械手的 Ver.1.9 以上版本的情况下

ST_aspACLD < 前端负荷质量 >，< 前端负荷重心位置 X 坐标 >，< 前端负荷重心位置 Y 坐标 >，< 前端负荷惯量 >

说　明

前端负荷质量是机械手的前端轴所装配的负荷 （工具 + 工件）的质量。单位以 (g) 指定。

负荷重心位置在工具 0 坐标系上指定负荷的重心位置。单位是 (mm)。

负荷惯量是以负荷的重心位置为原点的，绕 Z 轴的惯量。单位是 $kgcm^2$。

用 6 轴机械手的示例进行说明就是，工具 0 坐标系的原点是 6 轴法兰中心， Y 成分为从法兰中心开始至 Φ5H7 (Φ6H7) 销钉方向（定向矢量方向）， Z 成分是通过法兰中心与法兰面垂直的方向（近似矢量方向）， X 成分是以定向矢量为 Y 轴、近似矢量为 Z 轴时的右手坐标系中的 X 轴方向 （常规矢量方向）。请参照编程手册 "4.7 ' 使用条件 ' 中的最佳可搬运质量设定功能 "。

前端负荷质量、负荷重心位置 X6、负荷重心位置 Y6、负荷重心位置 Z6（负荷惯量）4 个值中即使只变更 1 个值时， 4 个值也请全部记述。

负荷条件值的切换需要约 0.1 秒的执行时间。如果频繁地切换负荷条件，则会成为导致动作时间延迟的主要原因。此外，在通过 (Pass) 动作过程中切换负荷条件值时，会变为结束动作，所以请注意不要在障碍物附近的通过 (Pass) 动作过程中变更模式。

相关项目

相关项目编程手册 "4.7 ' 使用条件 ' 中的最佳可搬运质量设定功能 "

注意事项

- 前端负荷质量请用每台机械手上所设定的范围内的整数进行指定。如果指定其以外的数值，则会发生 "60d2 前端负荷设定值超过允许值 "。
- 前端负荷重心位置的输入请满足每台机械手所指定的范围。如果不满足范围，则会发生 "60d2 前端负荷设定值超过允许值 "。
- 内部负荷条件值对于外部负荷条件值请设定在以下范围。如果不满足范围，则会发生 "60d2 前端负荷设定值超过允许值 "。

  0.5 × 外部负荷条件值 ≦ 内部负荷条件值 ≦ 外部负荷条件值

应用示例

```
ST_aspACLD 8500,-50,100,80        ' 将内部前端负荷条件值设定为前端负荷质量 8500 (g)、
                                  ' 负荷重 ' 心位置 X 成分 -50 (mm)、 Y 成分 100
                                  '(mm)、 Z 成分 80 (mm)。
```

ST_aspChange（语句）【Ver.1.9 以上版本】

功　　能

选择最佳搬运质量设定模式的内部模式。

格　　式

ST_aspChange < 模式 >

说　　明

切换最佳可搬运质量设定模式。

< 设定值 >

0 → 无效
1 → 仅限 PTP 有效
2 → 仅限 CP 有效
3 → PTP、 CP 均有效

在模式切换时，需要约 0.1 秒的执行时间。如果频繁地切换模式，则会成为导致动作延迟的主要原因。此外，在通过 (Pass) 动作中切换模式时，会变为结束动作。请注意在障碍物附近的通过 (Pass) 动作中不要变更模式。

相关项目

相关项目编程手册 "4.7 ' 使用条件 ' 中的最佳可搬运质量设定功能 "

注意事项

请用 0 ～ 3 的整数指定 < 模式 >。如果指定其以外的数值，则会发生 "6003 超过了有效的数值范围 "。

应用示例

```
ST_aspChange 1                    ' 将最佳搬运质量设定模式的内部模式置为 1。
```

ST_SetGravity（语句）【Ver.1.9 以上版本】

功　能

修正各个关节的静负荷 （重力转矩），设定重力平衡。

格　式

ST_SetGravity

说　明

机械手的各个关节，由于重力承受向下的静荷载重量 （重力扭矩）。重力扭矩根据工具及工件的质量、重心位置、机械手的姿势而变化。设定电流限制，限制电机的扭矩时，如果限制扭矩在重力扭矩以下，会导致各关节在重力方向上动作。为此，此程序修正从限制扭矩到重力扭矩的部分，并且具有即使在设定电流限制时也不会在重力方向上动作的功能，起到维持重力平衡的作用。

相关项目

ST_SetCurLmt、 ST_ResetGravity、 ST_SetGrvOffset

注意事项

- 请通过获取 (TAKEARM) 机械手控制权的任务来执行。尚未获取机械手控制权的情况下，会发生 " 不能获取 21F7 臂信号 (semaphore)" 错误。
- 请正确设定前端负荷质量和负荷重心位置。在没被正确设定的状态下，如果将电流限制设定值进行减小（30 以下左右）设定，则有时会发生重力下降请予以注意。前端负荷质量和负荷重心位置设定，请参照 p. 4-15 "4.7 ' 使用条件 ' 中最佳可搬运质量设定功能 "。
- 不知道前端部负荷重量和负荷重心位置是否正确，即使正确进行了设定在电流限制设定时依然重力下降时，使用重力补偿修正功能 (ST_SetGrvOffset)，可以进行重力补偿值的修正。请参照重力补偿修正共用程序库 (Library)。
- 在教导器上将使用条件的重力补偿有效无效设定为 1 时，重力补偿为有效。用教导器设定为有效时，自启动控制器电源，执行校准之后开始，重力补偿变为有效。

应用示例

```
ST_SetGravity                ' 使重力补偿有效。
Delay 100                    ' 等待重力补偿的反映。
ST_SetCurLmt 2,30            ' 将 2 轴的电流限制值设定为 30%。
```

ST_ResetGravity（语句）【Ver.1.9 以上版本】

功　　能

将重力平衡设为无效。

格　　式

ST_ResetGravity

说　　明

将重力平衡设为无效。

相关项目

ST_ResetCurLmt、 ST_SetEralw

注意事项

- 请通过获取 (TAKEARM) 机械手控制权的任务来执行。尚未获取机械手控制权的情况下，会发生 " 不能获取 21F7 臂信号 (semaphore)" 错误。
- 如果在电流限制过程中执行命令，会发生 " 不能将 665b 重力补偿设为无效 " 错误。请在解除了电流限制之后再次执行。
- 在教导器上将使用条件的重力补偿有效无效设定即使设定为 0 时，重力补偿也为无效。但是，与 (2) 相同，电流限制过程中不能进行设定。

应用示例

ST_ResetGravity

ST_SetGrvOffset（语句）【Ver.1.9 以上版本】

功　　能

通过各个关节的重力转矩修正重力补偿值。

格　　式

ST_SetGrvOffset

说　　明

机械手的各个关节，由于重力承受向下的静荷载重量 （重力扭矩）。用重力补偿语句 (ST_SetGravity) 可以设定重力平衡，但前端负荷质量设定值和实际的负荷质量有偏离时，在重力平衡上会造成偏离。本功能在机械手停止的状态下推测重力扭矩并修正重力补偿值。

相关项目

ST_SetCurLmt、 ST_SetGravity、 ST_ResetGrvOffset

注意事项

- 请通过获取 (TAKEARM) 机械手控制权的任务来执行。尚未获取机械手控制权的情况下，会发生 " 不能获取 21F7 臂信号 (semaphore)" 错误。
- 请在电机电源为 ON 机械手停止状态下执行。在电机电源 OFF 状态下执行时，会发生 "6006 电机电源关闭 " 错误。在机械手动作中执行时，会发生 "600B 机械手正在 动作 " 错误。
- 执行重力补偿修正的姿势有较大的变化时，在修正上会发生偏离。此时，请再次实施修正。
- 使用条件的电流限制清零设定值为 1、 3、 5、 7 以外的情况下，接通电机电源时修正值被清零为 0。

应用示例

ST_SetGrvOffset

ST_ResetGrvOffset（语句）【Ver.1.9 以上版本】

功　　能

将重力补偿值的修正设为无效。

格　　式

ST_ResetGrvOffset

说　　明

将重力补偿值的修正设为无效。

相关项目

ST_SetGrvOffset

注意事项

- 请通过获取 (TAKEARM) 机械手控制权的任务来执行。尚未获取机械手控制权的情况下，会发生 " 不能获取 21F7 臂信号 (semaphore)" 错误。
- 请在机械手停止状态下执行。在机械手动作中执行时，会发生 "600B 机械手正在 动作 " 错误。即使电机电源 OFF 当中也可以执行。

应用示例

ST_ResetGrvOffset

ST_SetCurLmt（语句）【Ver.1.9 以上版本】

功　能

限制指定轴的电机电流值。

格　式

ST_SetCurLmt < 轴编号 >、 < 设定值 >

说　明

将 < 轴编号 > 所指定轴的电机电流值 （力矩）限制设定值。在插入操作、对接操作时，需要限制施加在工件上的力时使用。

< 设定值 > 中电机额定电流值为 100。设定超过各个轴的允许值时，被允许值限制。

请设定 1 以上的数值。设定 0 以下的数值时，会发生错误 6003 " 超出有效数值范围 "。

相关项目

ST_ResetCurLmt、 ST_SetGravity、 ST_SetGrvOffset、 ST_SetEralw

注意事项

- 正在进行电流限制设定时，因电机电流被限制，所以不能在最高速度、最高加速度下动作。请仅在必要的步骤使用电流限制。同时，使用电流限制时，请降低加速度。
- 使用电流限制即使限制推力，但在高速下工件碰撞时，也会因工件和夹治具和轴的惯性产生冲击力。电流限制请在工件接触之前设定，并降低速度。
- 电流限制请在机械手停止状态下设定。如果在通过 (Pass) 动作过程中设定，则会发生错误。
- 请通过获取 (TAKEARM) 机械手控制权的任务来执行。尚未获取机械手控制权的情况下，会发生 " 不能获取 21F7 臂信号 (semaphore)" 错误。
- 使用条件的电流限制清零设定值在 1、 3、 5、 7 以外的情况下，接通电机电源时电流限制被清零。
  电机电源接通之后，置电流限制为有效时，请将电流限制清零设定值的最下位比特设定为 1。

6 轴

- 设定电流限制时，请务必将重力补偿功能设为有效。在重力补偿功能无效的状态下设定电流限制时，会发生 " 不能设定 665a 电流限制 " 错误。重力补偿请参照 ST_SetGravity。
- 请正确设定前端负荷质量和负荷重心位置。在没被正确设定的状态下，如果将设定值进行减小 （30 以下左右）设定，则有时会发生重力下降请予以注意。前端负荷质量和负荷重心位置设定，请参照编程手册 "4.7 ' 使用条件 ' 中最佳可搬运质量设定功能 "。
- 电流限制清零设定值的最下位比特被设定为 1 时，接通电机电源之后，有时会发生重力下降。请在电机电源 OFF 状态下请复位电流限制 （执行 ST_ResetCurLmt），接通电机电源。
- 不知道前端部负荷重量和负荷重心位置是否正确，即使正确地进行设定依然重力下降时，请使用重力补偿修正功能 (ST_SetGrvOffset)，修正重力补偿值。

4 轴 HS-E

在没有气体平衡的 4 轴机械手的 Z 轴、T 轴上设定电流限制时，如果减小限制值，则会发生 Z 轴下降或 T 轴旋转。请在执行 Z 轴、 T 轴重力补偿值设定功能 (ST_SetZBalance) 之后，执行电流限制。

应用示例

6 轴

```
ST_SetGravity                ' 使重力补偿有效。
ST_SetGrvOffet               ' 修正重力补偿值。
ST_SetEralw 2, 20            ' 将 2 轴的偏差允许值设定为 20 度。
ST_SetCurLmt 2, 30           ' 将 2 轴的电流限制值设定为 30%。
```

4 轴

```
ST_SetEralw 2, 20            ' 将 2 轴的偏差允许值设定为 20 度。
ST_SetCurLmt 2, 30           ' 将 2 轴的电流限制值设定为 30%。
```

4 轴 HS-E

```
ST_SetZBalance               ' 设定 Z 轴重力补偿值。
ST_SetEralw 3, 100           ' 将 Z 轴的偏差允许值设定为 100mm。
ST_SetEralw 4, 30            ' 将 T 轴的偏差允许值设定为 30 度。
ST_SetCurLmt 3, 10           ' 将 Z 轴的电流限制值设定为 10%。
ST_SetCurLmt 4, 10           ' 将 T 轴的电流限制值设定为 10%。
```

ST_ResetCurLmt（语句）【Ver.1.9 以上版本】

功　能

解除指定轴的电机电流限制。

格　式

ST_ResetCurLmt < 轴编号 >

说　明

解除 < 轴编号 > 所指定轴的电机电流限制。解除之后，电流限制值和偏差允许值返回到默认值。

［Ver.1.4 以前版本］

如果将 0 指定为轴编号，则解除所有轴的电流限制。

［Ver.1.5 以上版本］

如果在轴编号上指定 0，则解除 ST_ResetCurLmt 执行任务的正在获取信号 (semaphore) 的轴的全部的电流限制。

相关项目

ST_SetCurLmt、 ST_ResetEralw

注意事项

- 在解除电流限制时，实施去除偏差处理。当由于外力有角度偏差时，根据设定的速度、加速度，去除偏差处理时间会有变化。要缩短去除偏差处理时间，请提高速度、加速度的设定。
- 即使在电机电源 OFF 状态也可以执行。在电机电源 OFF 状态下解除电流限制时，请在结束正在获取臂信号 (semaphore) 的任务之后，执行以下的程序。

```
PRO999
TAKEARM
ST_ResetCurLmt 0
END
```

［Ver 1.4 以前版本］

请通过获取 (TAKEARM) 机械手控制权的任务来执行。尚未获取机械手控制权的情况下，会发生 " 不能获取 21F7 臂信号 (semaphore)" 错误。

［Ver.1.5 以上版本］

请通过获取 (TAKEARM) 控制权的任务执行。指定尚未获取控制权的轴时，会发生 " 不能获取 27D* * 轴信号 (semaphore)" 错误。

应用示例

```
ST_ResetCurLmt 0                ' 解除所有轴的电流限制
ST_ResetGrvOffset               ' 将重力补偿修正值清零
```

ST_SetEralw（语句）【Ver.1.9 以上版本】

功　能

变更指定轴的偏差允许值。

格　式

ST_SetEralw < 轴编号 >、 < 设定值 >

说　明

设定在 < 轴编号 > 上所指定的轴的偏差允许值。在限制电流时由于外力产生角度偏差时使用。

< 设定值 > 在旋转关节时，用臂关节角度 （度）指定；在直动关节时以臂关节位移 (mm) 指定。

所谓偏差允许值，是为了保证安全，当伺服偏差达到一定值以上时发生的错误 "611* J* 偏差过大 " 的允许范围。当使用电流限制进行装配时，有时会发生由于外力产生伺服偏差的错误。为了避免该现象，在暂时将偏差允许值放大时使用。此外，为了在电流限制调整中尽早查出重力掉落，在偏差允许值设定变小时也使用。

相关项目

ST_SetCurLmt、 ST_ResetEralw

注意事项

请通过获取 (TAKEARM) 机械手控制权的任务来执行。尚未获取机械手控制权的情况下，会发生 "不能获取21F7臂信号 (semaphore)" 错误。

应用示例

```
ST_SetEralw 2, 20                    ' 将 2 轴的偏差允许值设定为 20 度。
```

ST_ResetEralw（语句）【Ver.1.9 以上版本】

功　能

将指定轴的偏差允许值返回到默认值。

格　式

ST_ResetEralw < 轴编号 >

说　明

将 < 轴编号 > 所指定轴的偏差允许值返回到默认值。

［Ver.1.4 以前版本］

如果将 0 指定为轴编号，则将所有轴的偏差允许值返回到默认值。

［Ver.1.5 以上版本］

如果在轴编号上指定 0，则将 ST_ResetEralw 执行任务的正在获取信号 (semaphore) 的轴的全部偏差允许值返回默认值。

相关项目

ST_ResetCurLmt、 ST_SetEralw

注意事项

在执行电流限制解除命令 (ST_ResetCurLmt) 时，偏差允许值也成为默认值。

［Ver.1.4 以前版本］

请通过获取 (TAKEARM) 机械手控制权的任务来执行。尚未获取机械手控制权的情况下，会发生 " 不能获取 21F7 臂信号 (semaphore)" 错误。

［Ver.1.5 以上版本］

请通过获取 (TAKEARM) 控制权的任务执行。指定尚未获取控制权的轴时，会发生 " 不能获取 27D* * 轴信号 (semaphore)" 错误。

应用示例

```
ST_ResetEralw 0                    ' 将所有轴的偏差允许值返回到默认值。
```

ST_OnSrvLock（语句）【Ver.1.9 以上版本】

功　　能

将指定轴置为伺服锁定状态。（4 轴机械手专用）

格　　式

ST_OnSrvLock < 指定轴 >

说　　明

提供与原来语言的 ON SVLOCK 命令同等的功能。

所谓伺服锁定是指机械手的臂被控制，其位置被锁定的状态。

相关项目

ST_OffSrvLock

注意事项

- 伺服锁定请在机械手停止状态下设定。如果在通过 (Pass) 动作过程中设定，则会发生错误。
- 请通过获取 (TAKEARM) 机械手控制权的任务来执行。尚未获取机械手控制权的情况下，会发生 " 不能获取 21F7 臂信号 (semaphore)" 错误。

应用示例

```
ST_OnSrvLock 1          '1 轴的伺服锁定
ST_OnSrvLock 0          ' 所有轴的伺服锁定
```

ST_OffSrvLock（语句）【Ver.1.9 以上版本】

功　能

解除指定轴的伺服锁定。（4 轴机械手专用）

格　式

ST_OffSrvLock < 指定轴 >

说　明

提供与原来语言的 OFF SVLOCK 命令同等的功能。

所谓伺服锁定是指机械手的臂被控制，其位置被锁定的状态。如果解除伺服锁定则机械手的臂的位置不能被定位，一旦施加外力则位置发生偏移。

相关项目

ST_OnSrvLock

注意事项

- 伺服锁定处于解除状态的轴，不能执行动作指令。
- 请在机械手停止状态下设定伺服锁定解除。如果在通过 (Pass) 动作过程中设定，则会发生错误。
- 请通过获取 (TAKEARM) 机械手控制权的任务来执行。尚未获取机械手控制权的情况下，会发生 " 不能获取 21F7 臂信号 (semaphore)" 错误。
- 使用条件的 "25: 电流限制清零 " 设定值的第 2 比特为 "0"（初始值）时，在电机电源接通时，伺服锁定解除被清零（伺服锁定）。电机电源接通之后紧接着将伺服锁定解除设为有效时，请将 "+2" 设定为电流限制清零设定值。

  注 : 使用条件 "25：电流限制清零 " 设定示例

	PWM	SVLock	Curlmt	
	*	*	*	
只在 SrvLock 为有效时	0	1	0	＝ 2
全都有效时	1	1	1	＝ 7

应用示例

```
ST_OnSrvLock 1                ’1 轴的伺服锁定
```

ST_SetCompControl（语句）【Ver.1.9 以上版本】

功　能

将力限制功能设为有效。（6 轴专用命令）

格　式

ST_SetCompControl

说　明

将力限制功能设为有效。用 ST_SetFrcLimit, ST_SetCompRate, ST_SetFrcCoord 等将设定的力限制条件设为有效。

相关项目

ST_SetFrcLimit、 ST_SetCompRate、 ST_SetFrcCoord、 ST_ResetCompControl
ST_SetCompFControl

注意事项

- 在重力补偿无效、电流限制有效的情况下，执行本语句时，则会发生错误 "60f5 不能执行力限制 "。请将电流限制置为无效，重力补偿置为有效后再次执行。有关电流限制无效化，请参照 ST_ResetCurLmt ； 重力补偿有效化请参照 ST_SetGravity。
- 如在电机电源 OFF 状态下执行，力限制为无效。此外，在力限制过程中将电机电源 OFF 时，力限制也为无效。
- 请通过获取 (TAKEARM) 机械手控制权的任务来执行。尚未获取机械手控制权的情况下，会发生 " 不能获取 21F7 臂信号 (semaphore)" 错误。
- 请在机械手停止状态下执行。在通过 (Pass) 动作过程中执行本语句时，则变为末端动作。此外，在机械手动作过程中执行本语句，发生 "600b 机械手正在动作 " 错误时，请用 Delay 命令等等待机械手停止之后再执行。
- 在力限制过程中机械手因外力等作用动作时，有时会发生 "611* 偏差过大 " 错误。当发生错误时，请变更偏差允许值。偏差允许值变更请参照 ST_SetEralw。
- 请不要在机械手接触状态等对机械手施加力的状态下执行本语句。在施加力的状态下将力限制功能设为有效时，请使用 ST_SetCompFControl。
- 执行 ST_SetCompControl 之后，有时由于重力补偿的修正值产生误差，机械手的姿势会发生在机械手重力方向上动作等很大变化。在力限制过程中姿势变化较大时，在中途通过 ST_ResetCompControl 将力限制设为无效之后，请再次通过 ST_SetCompControl 将力限制设为有效。

应用示例

```
ST_SetFrcCoord 1                          ' 设定力限制坐标系。
ST_SetFrcLimit 100,0,100,100,100,100      ' 设定力限制比例。
ST_SetCompControl                         ' 将力限制功能设为有效。
ST_SetEralw 1,90                          ' 设定偏差允许值。
```

ST_SetCompFControl（语句）【Ver.1.9 以上版本】

功　能

将力限制功能设为有效。（6 轴专用命令）

格　式

ST_SetCompFControl

说　明

与 ST_SetCompControl 一样将力限制功能设为有效。但是，通过 ST_SetCompControl 执行的重力补偿修正处理不被执行。

相关项目

ST_SetCompControl

注意事项

- 在重力补偿无效、电流限制有效的情况下，执行本语句时，则会发生错误 "60f5 不能执行力限制 "。请将电流限制置为无效，重力补偿置为有效后再次执行。
- 如在电机电源 OFF 状态下执行，力限制为无效。此外，在力限制过程中将电机电源 OFF 时，力限制也为无效。
- 请通过获取 (TAKEARM) 机械手控制权的任务来执行。尚未获取机械手控制权的情况下，会发生 " 不能获取 21F7 臂信号 (semaphore)" 错误。
- 请在机械手停止状态下执行。在通过 (Pass) 动作过程中执行本语句时，则变为末端动作。此外，在机械手动作过程中执行本语句，发生 "600b 机械手正在动作 " 错误时，请用 Delay 命令等等待机械手停止之后再执行。
- 请正确设定前端负荷设定。当前端负荷设定值与实际的前端负荷不同时，有时会向重力方向掉落。此外，如果通过 ST_SetGrvOffset 执行重力补偿修正，则能够防止重力降低。

应用示例

```
ST_SetGrvOffset                          ' 计算重力补偿修正值。
ST_SetFrcCoord 1                         ' 设定力限制坐标系。
ST_SetFrcLimit 100,0,100,100,100,100     ' 设定力限制比例。
STSetCompFControl                        ' 将力限制有效化。
```

ST_ResetCompControl（语句）【Ver.1.9 以上版本】

功　能

将力限制功能设为无效。（6 轴专用命令）

格　式

ST_ResetCompControl

说　明

将力限制功能设为无效。

相关项目

ST_SetCompControl

注意事项

- 请通过获取 (TAKEARM) 机械手控制权的任务来执行。尚未获取机械手控制权的情况下，会发生 " 不能获取 21F7 臂信号 (semaphore)" 错误。
- 请在机械手停止状态下执行。在通过 (Pass) 动作过程中执行本语句时，则变为末端动作。此外，在机械手动作过程中执行本语句，机械手紧急停止或发生错误 "612* 过电流 " 时，请将 < 设定值 > 设定为 1，或用 Delay 命令等等待机械手停止之后再执行。
- 在本语句执行中瞬时停止，进行步骤回退或程序清零操作时，会发生错误 "60f9 力限制无效操作异常 "。
- 在力限制有效时，通过 ST_SetEralw 变更了偏差允许值时，偏差允许值返回初始值。但是，执行 ST_ResetCompControl 发生错误时，有时会不返回初始值，所以力限制无效之后，请通过 ST_ResetEralw 将偏差允许值返回到初始值。
- 有时会发生错误 "608* J* 指令速度限界超程 "。在此种情况下，在执行 ST_ResetCompControl 前请通过 ST_aspChange 将最佳可搬运质量”模式变更为2 或3，执行之后将最佳可搬运质量设定”模式返回到原来值。

应用示例

```
ST_ResetCompControl          ’将力限制功能无效化。
ST_ResetEralw(0)             ’将偏差允许值初始化。
```

ST_SetFrcCoord（语句）【Ver.1.9 以上版本】

功　能

选择力限制设定坐标系。（6 轴专用命令）

格　式

ST_SetFrcCoord< 设定值 >

说　明

通过 ST_SetFrcLimit, ST_SetCompRate 选择设定的力限制值的坐标系。

设定值为 0 时，选择机械手的底座坐标；设定值为 1 时，选择工具坐标；设定值为 2 时，选择工件坐标。

相关项目

ST_SetFrcLimit、 ST_SetCompRate、 ST_SetFrcCoord、 ST_ResetCompControl

注意事项

- 请通过获取 (TAKEARM) 机械手控制权的任务来执行。尚未获取机械手控制权的情况下，会发生 " 不能获取 21F7 臂信号 (semaphore)" 错误。
- 在力限制过程中，不能执行。如果在力限制过程中执行本语句，会发生 "60fa 力限制过程中 " 错误。
- 当 < 设定值 > 为 1，选择了工具坐标时，工具坐标成为力限制有效设定 （执行 ST_SetCompControl）时的工具坐标。在力限制过程中，即使通过 changetool 命令变更工具坐标，力限制坐标也不会变化。
- 将 < 设定值 > 设为 2，选择工件坐标时，工件坐标成为力限制有效设定 （执行 ST_SetCompControl）时的工件坐标。在力限制过程中，即使通过 changework 命令变更工件坐标，力限制坐标也不会变化。
- 控制器电源启动之后，设定值立即被初始化，成为 0 （底座坐标）。

应用示例

```
ST_SetFrcCoord 1                    ' 将力限制坐标系设定为工具坐标。
Changetool 2                        ' 将工具坐标设定为工具 2。
ST_SetFrcLimit 100,0,100,100,100,100
                                    ' 设定力限制比例。
ST_SetCompControl                   ' 将工具 2 坐标系的 Y 方向的力限制比例设定为 0%，
                                    ' 将力功能设为有效。
```

ST_SetFrcLimit（语句）【Ver.1.9 以上版本】

功　能

设定力限制比例。（6 轴专用命令）

格　式

ST_SetFrcLimit <X 方向限制比例 >,<Y 方向限制比例 >,<Z 方向限制比例 >,<X 旋转限制比例 >,<Y 旋转限制比例 >,<Z 旋转限制比例 >

说　明

通过 ST_SetFrcCoord 对所设定的坐标系的 X 轴方向、Y 轴方向、Z 轴方向、X 轴旋转、Y 轴旋转、Z 轴旋转的力限制比例进行设定。

设定值在 0 ～ 100 范围进行设定。到小数点后 2 位有效。

相关项目

ST_ResetFrcLimit、 ST_SetFrcCoord、 ST_SetCompControl

注意事项

- 请通过获取 (TAKEARM) 机械手控制权的任务来执行。尚未获取机械手控制权的情况下，会发生 " 不能获取 21F7 臂信号 (semaphore)" 错误。
- 在力限制过程中，不能执行。如果在力限制过程中执行本语句，会发生 "60fa 力限制过程中 " 错误。
- 控制器电源启动之后，设定值立即被初始化，X 轴方向、Y 轴方向、Z 轴方向、X 轴转动、Y 轴转动、Z 轴转动均为 100。

应用示例

```
ST_SetFrcCoord 1                    ' 将力限制坐标系设定为工具坐标。
ST_SetFrcLimit 100,0,100,100,100,100
                                    ' 设定力限制比例。
ST_SetCompControl                   ' 将工具坐标系的 Y 方向的力限制比例设定为 0%，
                                    ' 将力功能设为有效。
```

ST_ResetFrcLimit（语句）【Ver.1.9 以上版本】

功　　能

将力限制比例初始化。（6 轴专用命令）

格　　式

ST_ResetFrcLimit

说　　明

将力限制比例初始化。 X 轴方向、 Y 轴方向、 Z 轴方向、 X 轴转动、 Y 轴转动、 Z 轴转动均为 100%。

相关项目

ST_SetFrcLimit

注意事项

- 请通过获取 (TAKEARM) 机械手控制权的任务来执行。尚未获取机械手控制权的情况下，会发生 " 不能获取 21F7 臂信号 (semaphore)" 错误。
- 在力限制过程中，不能执行。如果在力限制过程中执行本语句，会发生 "60fa 力限制过程中 " 错误。

应用示例

```
ST_ResetCompControl             ’将力限制设为无效。
ST_ResetFrcLimit                ’将力限制比例初始化。
```

ST_SetCompRate（语句）【Ver.1.9 以上版本】

功　能

设定力限制时的柔度的比例。（6 轴专用命令）

格　式

ST_SetCompRate <X 方向柔度 >,<Y 方向柔度 >,<Z 方向柔度 >,<X 旋转柔度 >,<Y 旋转柔度 >,<Z 旋转柔度 >

说　明

通过 ST_SetFrcCoord 对所设定的坐标系的 X 轴方向、Y 轴方向、Z 轴方向、X 轴旋转、Y 轴旋转、Z 轴旋转的柔度比例进行设定。

设定值在 0 ～ 100 范围内设定。 0 时为最柔软。到小数点后 2 位有效。

相关项目

ST_ResetCompRate、 ST_SetFrcCoord、 ST_SetCompControl

注意事项

- 请通过获取 (TAKEARM) 机械手控制权的任务来执行。尚未获取机械手控制权的情况下，会发生 " 不能获取 21F7 臂信号 (semaphore)" 错误。
- 在力限制过程中，不能执行。如果在力限制过程中执行本语句，会发生 "60fa 力限制过程中 " 错误。
- 控制器电源启动之后，设定值立即被初始化，X 轴方向、Y 轴方向、Z 轴方向、X 轴转动、Y 轴转动、Z 轴转动均为 100。

应用示例

```
ST_SetFrcCoord1                    ’将力限制坐标系设定为工具坐标。
ST_SetCompRate 100,0,100,100,100,100
                                   ’设定柔度比例。
ST_SetCompControl                  ’将工具坐标系的 Y 方向的柔度比例设定为 0%，
                                   ’将力功能设为有效。
```

ST_ResetCompRate（语句）【Ver.1.9 以上版本】

功　　能

将力限制时的柔度的比例初始化。（6 轴专用命令）

格　　式

ST_ResetCompRate

说　　明

将 X 轴方向、 Y 轴方向、 Z 轴方向、 X 轴转动、 Y 轴转动、 Z 轴转动的柔度比例初始化，全部设为 100。

相关项目

ST_SetCompRate

注意事项

- 请通过获取 (TAKEARM) 机械手控制权的任务来执行。尚未获取机械手控制权的情况下，会发生 " 不能获取 21F7 臂信号 (semaphore)" 错误。
- 在力限制过程中，不能执行。如果在力限制过程中执行本语句，会发生 "60fa 力限制过程中 " 错误。

应用示例

```
ST_ResetCompControl          ’将力功能设为无效。
ST_ResetCompRate             ’将柔度比例初始化。
```

ST_SetFrcAssist（语句）【Ver.1.9 以上版本】

功　能

设定力限制时的补偿力。（力限制特殊功能语句）（6 轴专用命令）

格　式

SetFrcAssist <X 方向补偿力 >,<Y 方向补偿力 >,<Z 方向补偿力 >,<X 转动补偿力矩 >,<Y 转动补偿力矩 >,<Z 转动补偿力矩 >

说　明

对 SetFrcCoord 所设定的坐标系的 X 轴方向、Y 轴方向、Z 轴方向、X 轴转动、Y 轴转动、Z 轴转动的补偿力、力矩进行设定。力限制最大值的 10% 为设定最大值。

力设定值的单位是 [N]。力矩设定值的单位是 [Nm]。到小数点后 1 位有效。

相关项目

ST_ResetFrcAssist、 ST_SetFrcCoord、 ST_SetCompControl

注意事项

- 请通过获取 (TAKEARM) 机械手控制权的任务来执行。尚未获取机械手控制权的情况下，会发生 " 不能获取 21F7 臂信号 (semaphore)" 错误。
- 有时机械手会沿施加补偿力、力矩的方向运转。当发生运转时，请降低设定值。
- 控制器电源启动之后，设定值立即被初始化，X 轴方向、Y 轴方向、Z 轴方向、X 轴转动、Y 轴转动、Z 轴转动均为 0。

应用示例

```
ST_SetFrcCoord 1                    ' 将力限制坐标系设定为工具坐标。
ST_SetFrcAssist -30,0,0,0,0,0
                                    ' 在 -X 方向设定 30 [N] 的补偿力。
ST_SetFrcLimit 0,100,100,100,100,100
                                    ' 设定力限制比例。
CALL SetCompControl                 ' 将力功能设为有效。 X 方向被 0% 力限制，
                                    ' 在 -X 方向施加 30 [N] 的力。
```

ST_ResetFrcAssist（语句）【Ver.1.9 以上版本】

功　　能

将力限制时的补偿力初始化。（力限制特殊功能）
（6 轴专用命令）

格　　式

ST_ResetFrcAssist

说　　明

通过 ST_SetFrcCoord 将所设定的坐标系的 X 轴方向、Y 轴方向、Z 轴方向、X 轴旋转、Y 轴旋转、Z 轴旋转的补偿力、力矩设为 0。

相关项目

ST_SetFrcAssist

注意事项

请通过获取 (TAKEARM) 机械手控制权的任务来执行。尚未获取机械手控制权的情况下，会发生 "不能获取21F7臂信号 (semaphore)" 错误。

应用示例

```
ST_ResetCompControl          ' 将力限制设为无效。
ST_ResetFrcAssist            ' 将补偿力、力矩设定为 0。
```

ST_SetCompJLimit（语句）【Ver.1.9 以上版本】

功　能

设定力限制时的电流限制值。（力限制特殊功能）（6 轴专用命令）

格　式

ST_SetCompJLimit<J1 电流限制值 >,<J2 电流限制值 >,<J3 电流限制值 >,<J4 电流限制值 >,<J5 电流限制值 >,<J6 电流限制值 >

说　明

设定力限制时的电流限制值。电机额定电流值为 100。通过 ST_SetFrcLimit 将全方位设定为 0 时，电机电流被限制在设定值以下。

设定值在 0 ～ 100 范围内设定。

相关项目

ST_ResetCompJLimit、 ST_SetCompControl

注意事项

- 请通过获取 (TAKEARM) 机械手控制权的任务来执行。尚未获取机械手控制权的情况下，会发生 " 不能获取 21F7 臂信号 (semaphore)" 错误。
- 控制器电源启动之后，设定值立即被初始化，电流限制值全部变为 0。
- 在 ST_SetCompJLimit 上设定比较大的值时，有时机械手会产生振动而发生错误。在此情况下，请通过 ST_SetCompRate, ST_SetDumpRate 调整力限制的柔度。

应用示例

```
ST_SetFrcCoord 1                          ' 将力限制坐标系设定为工具坐标。
ST_SetCompJLimit 30,0,0,0,0,0             ' 将 J1 轴的电流限制值设定为 30%。
ST_SetFrcLimit 100,0,100,100,100,100      ' 设定力限制比例。
ST_SetCompControl                         ' 将力功能设为有效。
```

ST_ResetCompJLimit（语句）【Ver.1.9 以上版本】

功　　能

将力限制时的电流限制值初始化。（力限制特殊功能）
（6 轴专用命令）

格　　式

ST_ResetCompJLimit

说　　明

将力限制时的电流限制值进行初始化，所有轴设定为 0。

相关项目

ST_SetCompJLimit、 ST_SetCompControl

注意事项

请通过获取 (TAKEARM) 机械手控制权的任务来执行。尚未获取机械手控制权的情况下，会发生 "不能获取21F7臂信号 (semaphore)" 错误。

应用示例

```
ST_ResetCompControl                ' 将力功能设为无效。
ST_ResetCompJLimit                 ' 将电流限制值初始化。
```

ST_SetCompVMode（语句）【Ver.1.9 以上版本】

功　能

设定力限制时的速度控制模式。（力限制特殊功能）
（6 轴专用命令）

格　式

ST_SetCompVMode

说　明

将执行 ST_SetCompControl 时的力限制速度控制模式设为有效。

相关项目

ST_ResetCompVMode

注意事项

请通过获取 (TAKEARM) 机械手控制权的任务来执行。尚未获取机械手控制权的情况下，会发生 "不能获取21F7臂信号 (semaphore)" 错误。

应用示例

```
ST_SetCompVMode                 ’将力限制速度控制模式设为有效。
ST_SetCompControl               ’将力限制设为有效。
```

ST_ResetCompVMode（语句）【Ver.1.9 以上版本】

功　　能

将力限制时的速度控制模式设为无效。（力限制特殊功能）
（6 轴专用命令）

格　　式

ST_ResetCompVMode

说　　明

将执行 ST_SetCompControl 时的力限制速度控制模式设为无效。

相关项目

ST_SetCompVMode

注意事项

请通过获取 (TAKEARM) 机械手控制权的任务来执行。尚未获取机械手控制权的情况下，会发生 "不能获取21F7臂信号 (semaphore)" 错误。

应用示例

```
ST_ResetCompVMode          ' 将力限制速度控制模式设为无效。
ST_SetCompControl          ' 将力限制设为有效。
```

ST_SetCompEralw（语句）【Ver.1.9 以上版本】

功　　能

设定力限制时的工具端的位置、姿势偏差允许值。（6 轴专用命令）

格　　式

ST_SetCompEralw <X 方向偏差允许值 >,<Y 方向偏差允许值 >,<Z 方向偏差允许值 >,<X 旋转偏差允许值 >,<Y 旋转偏差允许值 >,<Z 旋转偏差允许值 >

说　　明

设定力限制时的工具端的位置、姿势偏差允许值。X、Y、Z 方向的偏差允许值的单位是 (mm)，X、Y、Z 转动的偏差允许值的单位是 （度）。到小数点后 1 位有效。

相关项目

ST_ResetCompEralw、 ST_SetFrcCoord

注意事项

- 请通过获取 (TAKEARM) 机械手控制权的任务来执行。尚未获取机械手控制权的情况下，会发生 " 不能获取 21F7 臂信号 (semaphore)" 错误。
- 控制器电源起动之后的偏差允许值的初始值是 X、Y、Z 方向为 100 (mm) X、Y、Z 转动为 30（度）。X、Y、 Z 周围是最大值为 175 （度）。进行超过最大值的设定时，会发生 "6003 超过有效数值范围 " 错误。
- 超过工具的位置、姿势偏差所设定的允许值时，会发生 "60f8 力限制时位置偏差过大异常 " 错误。
- 偏差设定坐标系成为通过 ST_SetFrcCoord 被设定的坐标系。例如，设定为 ST_SetFrcCoord (1) 时，SetCompEralw 的设定坐标系为工具坐标系。

应用示例

```
ST_SetFrcCoord 2                          ' 将力限制坐标系设为工件坐标。
ST_SetFrcLimit 100,0,100,100,100,100      ' 将工件坐标的 Y 方向的力限制比例设为 0%
ST_SetCompEralw 10,150,10,5,5,5           ' 将工件坐标 X、 Z 方向的偏差允许值设为 10 (mm)，
                                          'Y 方向的偏差允许值设为 150 (mm)，
                                          'X、 Y、 Z 转动的偏差允许值设为 5 （度）。
```

ST_ResetCompEralw（语句）【Ver.1.9 以上版本】

功　　能

将力限制时的工具端的位置、姿势偏差允许值初始化。（6 轴专用命令）

格　　式

ST_ResetCompEralw

说　　明

将力限制时的工具端的位置、姿势偏差允许值初始化。X、Y、Z 方向的偏差允许值的初始值是 100 (mm)，X、Y、 Z 转动的偏差允许值的初始值是 30 （度）。

相关项目

ST_SetCompEralw

注意事项

请通过获取 (TAKEARM) 机械手控制权的任务来执行。尚未获取机械手控制权的情况下，会发生 "不能获取21F7臂信号 (semaphore)" 错误。

应用示例

ST_ResetCompEralw

ST_SetDampRate（语句）【Ver.1.9 以上版本】

功　能

设定力限制时的粘性比例。（6 轴专用命令）

格　式

ST_SetDampRate <X 方向粘性比例 >,<Y 方向粘性比例 >,<Z 方向粘性比例 >,<X 旋转粘性比例 >,<Y 旋转粘性比例 >,<Z 旋转粘性比例 >

说　明

通过 ST_SetFrcCoord 对所设定的坐标系的 X 轴方向、Y 轴方向、Z 轴方向、X 轴旋转、Y 轴旋转、Z 轴旋转的粘性比例进行设定。

设定值在 0 ～ 100 范围内设定。 0 时的粘性最小。到小数点后 2 位有效。

相关项目

ST_ResetDampRate、 ST_SetFrcCoord、 ST_SetCompRate

注意事项

- 请通过获取 (TAKEARM) 机械手控制权的任务来执行。尚未获取机械手控制权的情况下，会发生 " 不能获取 21F7 臂信号 (semaphore)" 错误。
- 在力限制过程中，不能执行。如果在力限制过程中执行本语句，会发生 "60fa 力限制过程中 " 错误。
- 控制器电源启动之后，设定值立即被初始化，X 轴方向、Y 轴方向、Z 轴方向、X 轴转动、Y 轴转动、Z 轴转动均为 100。
- 请不要设定比通过 ST_SetCompRate 设定的柔度比例更小的值。当设定了比柔度比例更小的值时，有时机械手会因振动而停止。
- ST_SetDampRate 执行之后，如果设定 ST_SetCompRate，则粘性比例为 ST_SetCompRate 的设定值。

应用示例

```
ST_SetFrcCoord 1                          ' 将力限制坐标系设定为工具坐标。
ST_SetCompRate 100,0,100,100,100,100
                                          ' 设定柔度比例。
ST_SetDampRate 100,20,100,100,100,100
                                          ' 设定粘性比例。
ST_SetCompControl                         ' 将工具坐标系的 Y 方向的柔度比例设定为 0%，
                                          ' 粘性比例设定为 20%，并将力功能置为有效。
```

ST_ResetDampRate（语句）【Ver.1.9 以上版本】

功　　能

将力限制时的粘性比例初始化。（6 轴专用命令）

格　　式

ST_ResetDampRate

说　　明

将 X 轴方向、Y 轴方向、Z 轴方向、X 轴转动、Y 轴转动、Z 轴转动的粘性比例进行初始化，全部设定为 100。

相关项目

ST_SetDampRate

注意事项

- 请通过获取 (TAKEARM) 机械手控制权的任务来执行。尚未获取机械手控制权的情况下，会发生 " 不能获取 21F7 臂信号 (semaphore)" 错误。
- 在力限制过程中，不能执行。如果在力限制过程中执行本语句，会发生 "60fa 力限制过程中 " 错误。

应用示例

ST_ResetDampRate　　’将粘性比例初始化。

ST_SetZBalance（语句）【Ver.1.9 以上版本】

功　能

设定 Z 轴、 T 轴的重力补偿值。（4 轴专用）

格　式

ST_SetZBalance

说　明

没有气体平衡的 4 轴机械手的 Z 轴、 T 轴，会由于重力的作用而受到向下的静负荷 （重力转矩）。设定电流限制时，如果将限制值设定在重力转矩以下，则 Z 轴掉落、T 轴旋转。（4 轴专用）本功能在机械手处于停止状态下，会推定重力转矩、设定重力补偿值。通过使用本功能，可以防止在减小设定电流限制值时的 Z 轴掉落、 T 轴旋转。

相关项目

ST_ResetZBalance、 ST_SetCurLmt

注意事项

- 请通过获取 (TAKEARM) 机械手控制权的任务来执行。尚未获取机械手控制权的情况下，会发生 " 不能获取 21F7 臂信号 (semaphore)" 错误。
- 请在电机电源为 ON 机械手停止状态下执行。在电机电源 OFF 状态下执行时，会发生 "6006 电机电源关闭 " 错误。在机械手动作中执行时，会发生 "600B 机械手正在 动作 " 错误。
- 执行重力补偿值设定之后，工件重量发生变化时，设定值会产生偏离。在此种情况下请再次执行本命令。

应用示例

ST_SetZBalance

ST_ResetZBalance（语句）【Ver.1.9 以上版本】

功　　能

将重力补偿值的修正设为无效。（4 轴专用）

格　　式

ST_ResetZBalance

说　　明

将 4 轴机械手的 Z 轴、 T 轴的重力补偿值的设定设为无效。

相关项目

ST_SetZBalance

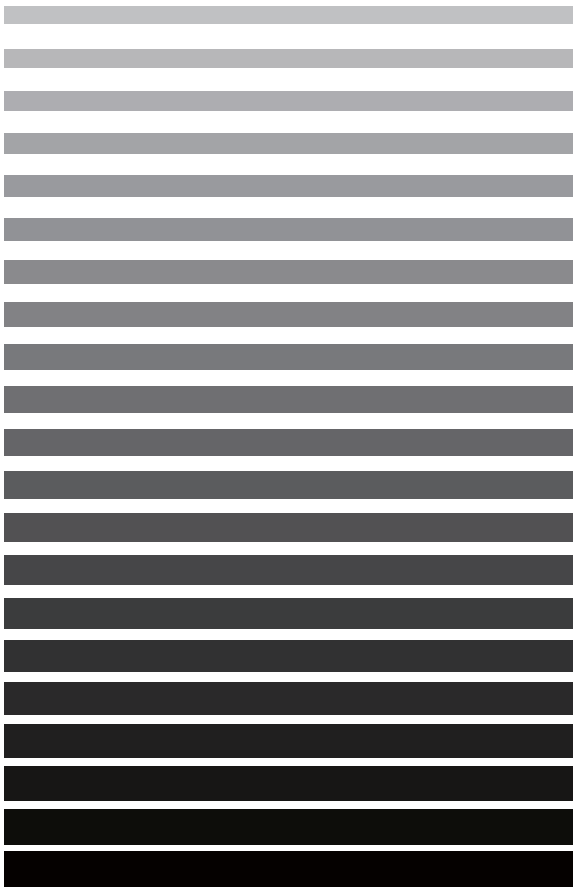
注意事项

- 请通过获取 (TAKEARM) 机械手控制权的任务来执行。尚未获取机械手控制权的情况下，会发生 " 不能获取 21F7 臂信号 (semaphore)" 错误。
- 请在机械手停止状态下执行。在机械手动作中执行时，会发生 "600B 机械手正在 动作 " 错误。即使电机电源 OFF 当中也可以执行。

应用示例

ST_ResetZBalance

第 13 章
输入输出控制语句



在本章中，介绍各种 I / O 的输入输出控制指令。

第 13 章　输入输出控制语句

13.1　I / O 端口

IN（语句）【依据 SLIM】

功　能

从通过 I / O 变量表示的 I / O 端口中读取数据。

格　式

IN < 算术变量名 > = <I / O 变量 >

说　明

将用 <I / O 变量 > 表示的端口的数据赋值用 < 算术变量名 > 指定的变量。

<I / O 变量 > 是用 DEFIO 语句定义的变量或是 I / O 型变量。

相关项目

OUT、 DEFIO

注意事项

当在 SINGLE 中定义的 I/O 型变量的内容，无法用单精度实数表示时，会显示错误 777F （实数转换失败）。

应用示例

```
DEFINT Li1, Li2
DEFIO samp1 = INTEGER, 220      ' 将 samp1 作为从端口 220 开始的 INTEGER 型 I / O 变量进行定义。

IN Li1 = samp1                  ' 将 samp1 的数据赋值到 Li1。
IN Li2 = IO[240]                ' 将端口 240 的数据赋值到 Li2。
OUT samp1 = Li1                 ' 将 Li1 的数据从以 samp1 定义的端口输出。

OUT IO[240] = Li2               ' 将 Li2 的数据从端口 240 输出。
```

OUT （语句）【依据 SLIM】

功　能

向通过 I / O 变量表示的 I / O 端口输出数据。

格　式

OUT <I / O 变量 > = < 输出数据 >

说　明

将 < 输出数据 > 的值在以 <I / O 变量 > 表示的端口地址输出。

<I / O 变量 > 是用 DEFIO 语句定义的变量或是 I / O 型变量。

相关项目

IN、 DEFIO

应用示例

```
DEFINT Li1, Li2
DEFIO samp1 = INTEGER, 220      ' 将 samp1 作为从端口 220 开始的 INTEGER 型 I / O 变量进行定义。

IN Li1 = samp1                  ' 将 samp1 的数据赋值到 Li1。
IN Li2 = IO[240]                ' 将端口 240 的数据赋值到 Li2。
OUT samp1 = Li1                 ' 将 Li1 的数据从以 samp1 定义的端口输出。

OUT IO[240] = Li2               ' 将 Li2 的数据从端口 240 输出。
```

IOBLOCK ON/OFF（语句）【依据 SLIM】

功　能

在执行动作指令时，同步执行 I / O 指令、运算指令等非动作指令。

格　式

IOBLOCK {ON|OFF}

说　明

IOBLOCK ON 和 IOBLOCK OFF 被成对使用，在被其覆盖的范围内，机械手在执行动作命令的过程中，将会并行执行其动作命令所持续的多个 I / O 命令和运算命令等非动作命令。

如果出现机械手动作命令，中断并列执行，等待当前正在执行的动作命令结束（通过 (Pass) 动作时开始通过 (Pass)），然后执行下个动作命令。下一个动作命令之后，接着是非动作命令的情况下，这些也与动作命令并行执行。在 IOBLOCK 内的动作命令执行过程中，如果执行 IOBLOCK OFF，则等待动作命令的执行之后，进行下一个步骤。

相关项目

注意事项

- 以下的情况下，不进行并行处理。
  在任何情况下，直至动作命令结束（通过 (Pass) 动作指定时，通过 (Pass) 动作开始），下一个步骤等待执行。
  - 在动作命令上追加动作选择时。
  - 执行 CHANGEWORK， CHANGETOOL， SPEED， JSPEED， ACCEL， JACCEL 时。
  - 执行原有语言共用程序库 (Library) "aspChange"、 "aspACLD"、机械臂动作共用程序库 (Library) "mvSetPulseWidth"、"mvSetTimeOut"、"mvReverseFlip"、"mvResetPulseWidth"、"mvResetTimeOut" 时。
- 在机械手动作过程中让步骤停止、瞬时停止，然后再开始运行时，有时不能进行与非动作指令并行的动作。
- 在 IOBLOCK 内连续 2 次执行动作命令时，若在最初的动作执行过程中让步骤停止，则下个动作结束之后步骤停止。在执行最初的动作时，进行瞬时停止并启动下个步骤时，也是在下个动作结束之后才步骤停止。

  例 :

  ```
  IOBLOCK ON
  MOVE P， J0      ← 即使在向 J0 的动作过程中让步骤停止，下一个动作 MOVE P， J1 也会被动作。
                     然后，步骤停止。
  MOVE P， J1
  SET IO[1]
  IOBLOCK OFF
  ```

- IOBLOCK 的有效范围只在定义的程序中有效，而在辅助程序中则无效。
  有以下程序时，在 SUB1 中用 PRO1 定义的 IOBLOCK 无效。IOBLOCK 的有效范围在每个程序中是独立的。
  例 :

  ```
  PROGRAM PRO1
  TAKEARM
  DEFPOS lp1， lp2
  IOBLOCK ON
  CALL SUB1
  MOVE P， lp1， lp2
  SET IO[1]
  ```

```
SET IO[2]
IOBLOCK OFF
    :
    :
```

- 在教导检查模式下， IOBLOCK 无效。

应用示例

```
TAKEARM                          ' 获得机械手控制权
IOBLOCK ON                       'I / O 命令与下一个的动作命令并行执行。
MOVE P (902.7, 0, 415.3, 180, 50, 180, 1)
                                 ' 向 (902.7, 0, 415.3, 180, 50, 180, 1)
                                 ' 的坐标 （PTP 控制）移动。
SET IO[240]                      ' 将 BIT 型端口 240 置于 ON。
SET IO[241], 40                  ' 将 BIT 型端口 241 在 40ms 之间置于 ON。
SET IO[SOL1]                     ' 将 I / O 变量 SOL1 所表示的端口置于 ON。
SET IO[104 TO 110]               ' 将 BIT 端口 104 ～ 110 置于 ON。
IF IO[242] THEN
  RESET IO[240]                  ' 将 BIT 型端口 240 置于 OFF。
  RESET IO[SOL1]                 ' 将 I / O 变量 SOL1 所表示的端口置于 OFF。
  RESET IO[104 TO 110]           ' 将 BIT 端口 104 ～ 110 置于 OFF。
ENDIF
IOBLOCK OFF
```

SET （语句）【依据 SLIM】

功　能

将 I / O 端口置为 ON。

格　式

SET <I / O 变量 > ［，＜输出时间 >］

说　明

将在 <I / O 变量 > 中指定的端口置于 ON。

如果指定 < 输出时间 >，则脉冲输出。（输出时间的单位是 ms。）

如果指定 < 输出时间 >，则在其时间没有经过之前，不向下一个命令进行。此外，输出时间的设定值是最低输出时间，实际的输出时间会受任务的优先次序影响而变化。

相关项目

RESET、 DEFIO

注意事项

- 在进行输出时间指定时，动作过程中其他的程序的存在、多功能教导器操作状况及周围机器的通信等原因，有时会延长输出时间。
- 在进行输出时间指定时，因为控制器的基准时间单位是 16.7ms，有时指定时间会偏离 ±16.7ms。
- 在使用时间指定 SET 时，请注意以下 2 点。
  - 如果根据时间指定 SET，端口在 ON 过程中由其他任务 RESET 相同的端口，从该时刻开始，该端口处于 OFF 状态。（时间指定 SET 变为无效。）
  - 如果根据时间指定 SET，端口在 ON 过程中由其他任务 RESET 相同的端口连续进行 SET，则经过指定时间之后，指定端口置于 OFF。（时间指定 SET 为有效。）
- 在使用输出时间时，如果在执行命令时进行瞬时停止后重新启动，请注意在瞬间停止时，也经过输出时间。
- 如果指定通用输出、手动输出、内部 I/O 端口以外的编号，即发生错误 21FB （系统区域写入错误）。尤其是对指定端口编号时使用算式的情况要加以注意。
- 无法用于在 SINGLE 中定义的 I/O 型变量。如执行，会显示错误 777F （实数转换失败）。

应用示例

```
TAKEARM                                  ’获得机械手控制权
IOBLOCK ON                               ’I / O 命令与下一个的动作命令并行执行。
MOVE P (902.7, 0, 415.3, 180, 50, 180, 1)
                                         ’向 (902.7, 0, 415.3, 180, 50, 180, 1)
                                         ’的坐标 （PTP 控制）移动。
SET IO[240]                              ’将 BIT 型端口 240 置于 ON。
SET IO[241], 40                          ’将 BIT 型端口 241 在 40ms 之间置于 ON。
SET IO[SOL1]                             ’将 I / O 变量 SOL1 所表示的端口置于 ON。
SET IO[104 TO 110]                       ’将 BIT 端口 104 ～ 110 置于 ON。
IF IO[242] THEN
   RESET IO[240]                         ’将 BIT 型端口 240 置于 OFF。
   RESET IO[SOL1]                        ’将 I / O 变量 SOL1 所表示的端口置于 OFF。
   RESET IO[104 TO 110]                  ’将 BIT 端口 104 ～ 110 置于 OFF。
ENDIF
IOBLOCK OFF

TAKEARM                                  ’获得机械手控制权
SET IO[I1 * 5]                           ’将 I1 的值 *5 的数值的端口置于 ON。
SET IO[27-24]                            ’将 27-24=3 的端口置于 ON。
```

RESET（语句）

功　能

将 I / O 端口置为 OFF。

格　式

RESET <I / O 变量 >

说　明

将在 <I / O 变量 > 中指定的端口置于 OFF。

相关项目

SET、 DEFIO

应用示例

```
TAKEARM                          ' 获得机械手控制权
IOBLOCK ON                       'I / O 命令与下一个的动作命令并行执行。
MOVE P (902.7, 0, 415.3, 180, 50, 180, 1)
                                 ' 向 (902.7, 0, 415.3, 180, 50, 180, 1)
                                 ' 的坐标 （PTP 控制）移动。
SET IO[240]                      ' 将 BIT 型端口 240 置于 ON。
SET IO[241], 40                  ' 将 BIT 型端口 241 在 40ms 之间置于 ON。
SET IO[SOL1]                     ' 将 I / O 变量 SOL1 所表示的端口置于 ON。
SET IO[104 TO 110]               ' 将 BIT 端口 104 ～ 110 置于 ON。
IF IO[242] THEN
  RESET IO[240]                  ' 将 BIT 型端口 240 置于 OFF。
  RESET IO[SOL1]                 ' 将 I / O 变量 SOL1 所表示的端口置于 OFF。
  RESET IO[104 TO 110]           ' 将 BIT 端口 104 ～ 110 置于 OFF。
ENDIF
IOBLOCK OFF
```

DETECT ON/OFF（语句）【Ver.3.2 以上版本】

功　能

检测夹治具输入的上升以及下降，将此时机械手的位置存放至变量。

格　式

- DETECT ON 的格式
  DETECT ON< 夹治具输入端口编号 >,<ROB>,< 存放位置前面变量 >,< 最大存放数 >,< 检测次数存放位置变量 >[,< 检测方向 >]
- DETECT OFF 的格式
  DETECT OFF< 检测结果存放位置变量 >[,< 夹治具输入端口编号 >]

说　明

执行 DETECT ON 之后，在执行 DETECT OFF 之前的时间段内，检测利用 < 夹治具输入端口编号 > 指定的夹治具输入的上升 （OFF 至 ON）、以及下降 （ON 至 OFF），并保存至利用 < 存放位置前面变量 > 指定的变量中。

DETECT 为扩展功能。请用密码 “1312” 进行功能扩展之后再使用。关于功能扩展，请参照操作指南的 " 扩展功能的追加 "。

DETECT ON

在 < 夹治具输入端口编号 > 处，指定检测的夹治具输入端口编号 （48~55）。

请在 <ROB> 处，指定 "ROB"。

在 < 存放位置前面变量 > 处，指定保存机械手位置的变量。当进行复数次检测时，在每次的变量编号上加 1，并将此编号保存至变量。

可指定P型以及J型的全球变量。如仅记述变量编号，则变为P型全球变量。例如，当记述1时，则指定的是P1。

在 < 最大存放数 > 处，指定检测时位置的最大存放数。

在 < 检测次数存放位置变量 > 处，指定存放检测次数的变量。可指定为 I 型全球变量。

在 < 检测方向 > 处，指定通过上升还是下降进行检测。

- 0：检测上升
- 1：检测下降
- 2：检测上升和下降两方面

省略时，指定为 0 （上升）。

DETECT OFF

在 < 检测次数存放位置变量 > 处，指定存放检测结果的变量。可指定为 I 型全球变量。

- 0：正常结束
- -1：检测次数与存放位置的数量不一致
- -2：检测次数比 “ 最大存放数 ” 多
- -3：上述 -1 与 -2 两方面

在 < 夹治具输入端口编号 > 处，指定结束检测的夹治具输入端口编号 （48~55）。当省略时， DETECT ON 时指定的夹治具输入全部结束。但是，对其他程序中利用 DETECT ON 指定的夹治具输入的检测，不会结束。

即使不执行 DETECT OFF，在执行 END 的时候，对其程序内利用 DETECT ON 指定的夹治具输入的检测也将全部结束。

当在同一程序内执行复数次 DETECT ON 时，只要指定相同的夹治具输入端口编号、相同的检测方向，最后执行的 DETECT ON 的设置就会生效。

例

```
DETECT ON, 50, ROB, P10, 30, I5, 0
DETECT ON, 50, ROB, P20, 10, I1, 0                    ' 此行的设置生效
```

即使指定相同的夹治具输入端口编号，如果检测方向不同，也会全部变为其他设定。

例

```
DETECT ON, 50, ROB, P10, 10, I1, 0                    ' 从 P10 处输入上升时的位置
DETECT ON, 50, ROB, P20, 10, I2, 1                    ' 从 P20 处输入下降时的位置
DETECT ON, 50, ROB, P30, 10, I3, 2                    ' 从 P30 处输入上升、下降时的位置
```

当在不同的程序中指定相同的夹治具输入端口编号、相同的检测方向时，将会显示错误 “77FB I/O 编号重复 ”。

注意事项

夹治具输入端口的数字过滤器时间 （数字信号处理）通常为 3msec，但一旦使用 DETECT 指令，即会切换为 50μ sec。一旦使用继电器接点输入等会产生蜂鸣的机器，有时将无法正确发挥功能，因此请多注意。

应用示例

```
PROGRAM PRO1
  TAKEARM
  MOVE P, P1
  DETECT ON, 48, ROB, 0, 5, I10, 0                    ' 将夹治具输入 48 上升时的位置存放至 P0
                                                      ' 最多存放 5 个。
                                                      ' 将实际的检测次数存放至 I10。
  DETECT ON, 49, ROB, 10, 5, I11, 1                   ' 将夹治具输入 49 下降时的位置
                                                      ' 存放至 P10，最多存放 5 个。
                                                      ' 将实际的检测次数存放至 I11。
  MOVE L, P2
  DETECT OFF, I0                                      ' 结束对夹治具输入 48 与 49 的检测
                                                      ' 将检测结果存放至 I0。
  IF I0 <> 0 THEN                                     ' 当检测结果非 “ 正常结束 ” 时，
    STOP                                              ' 程序停止。
  ELSE                                                ' 当检测结果 “ 正常结束 ” 时，
    MOVE L, P3                                        ' 移动至 P3。
  ENDIF
END
```

13.2 RS232C 及 Ethernet 端口

INPUT（语句）

功　能

得出来自 RS232C 及 Ethernet 端口的数据。

格　式

INPUT［#＜线路编号＞，］＜变量名＞［， WTIME＝＜超时时间＞［， RVAL＝＜存放变量＞］］

[,＜变量名＞［， WTIME＝＜超时时间＞［， RVAL＝＜存放变量＞］]…]

说　明

将 RS232C 和 Ethernet 端口接收的数据存放在用＜变量名＞指定的变量内。

在＜线路编号＞中，指定使用的 RS232C 和 Ethernet 端口的线路编号。若省略＜线路编号＞，则为默认值的 ch2。ch1 不能指定为多功能教导器用。（参照 "2.4.1 线路编号 "）若以同数的变量收到复数的信息，则在各变量间添加逗号 (，)。

将至间隔符 （CR 或 CR+LF）之前的全部内容存放在指定的变量内。

间隔符不能读入变量内。

波特率以系统参数指定。

＜超时时间＞［RC7_Ver2.2 以上版本］

每个变量均可指定超时时间。没有输入数据时，在经过指定时间之后，结束对于当前的变量的输入等待，移动至下一个控制。单位为 ms，但是，实际的延迟时间以 1 / 60s 单位增减。

＜存放变量＞［RC7_Ver2.2 以上版本］

＜超时时间＞的设定成对使用。若收到输入数据正常脱开时， TRUE (1) 进入，若超时脱开时，则 FALSE (0) 进入指定的变量。

注意

- 在接收数据之前，为了清空接收数据的输入缓冲区残留的数据，执行 FLUSH 指令。
- 如果输入的数据超过赋值的变量形式所能表示的最大值，其结果将成为该形式的最大值。

相关项目

FLUSH、 PRINT、 WRITE

应用示例

例 1:

```
DIM li1 As Integer
defint li
DEFSTR ls1, ls2, ls3, ls4
INPUT #1, ls1                          ' 将来自 ch2 的数据写入 ls1。
INPUT #1, li1, ls2, ls3                ' 将来自 ch2 的数据写入 li1、 ls2、 ls3。
INPUT ls4                              ' 将来自 ch2 的数据写入 ls4。
INPUT ls4 ,WTIME=100,RVAL=li           ' 在没有输入时， 100msec 的时间之后进
                                       ' 行下一个指令。表示 li 中超时
                                       ' 所遗漏的 0 输入。
```

注意

以下应用示例 2 和 3 的字符串信息以逗号 （，）断开，因可接收以与其断开的个数相同的要素数的变量，所以不需要多余的字符串演算。

例 2: 从外部传送 "505.425, -125.325, 400.238" 时 （6 轴机械手的示例）

```
DIM lv1 AS VECTOR
INPUT #1,lv1
LETP P11=lv1                           ' 将接收的 x， y， z 赋值 P11
LETR P11=RVEC(P10)                     ' 将基准位置 P10 的值赋值姿势数据
LETF P11=FIG(P10)                      ' 将基准位置 P10 的值赋值形态数据
MOVE P,@E P11,S=100
```

例 3: 从外部传送 "505.425, -125.325, 400.238, 180, 0, 180, 5" 时 （6 轴机械手的示例）

```
INPUT #1,P11
MOVE P,@E P11,S=100
```

LINEINPUT（语句）

功　能

从 RS232C 及 Ethernet 端口中读取数据直到间隔符，并对字符串型变量赋值。

格　式

LINEINPUT［#＜线路编号＞，］＜字符串型变量名＞［，WTIME＝＜超时时间＞［，RVAL＝＜存放变量＞］］

说　明

从 RS232C 和 Ethernet 端口接收的数据中将至间隔符 （CR 或 CR+LF）之前的全部文字存放在用＜字符串型变量名＞指定的变量内。间隔符不能读入变量内。

在＜线路编号＞中，指定使用的 RS232C 和 Ethernet 端口的线路编号。若省略＜线路编号＞，则为默认值的 ch2。ch1 不能指定为多功能教导器用。（参照 "2.4.1 线路编号 "）

在＜字符串变量名＞内指定字符串型变量的名称。

＜超时时间＞［RC7_Ver2.2 以上版本］

在指定时间内没有输入间隔符时，结束 lineinput 语句的执行，控制转移到下一个指令。单位为 ms，但是，实际的延迟时间以 1 / 60s 单位增减。

＜存放变量＞［RC7_Ver2.2 以上版本］

＜超时时间＞的设定成对使用。若收到输入数据正常脱开时， TRUE (1) 进入，若超时脱开时，则 FALSE (0) 进入指定的变量。

应用示例

```
DEFSTR ls1,ls2,ls3,ls4
DEFINT li
LINEINPUT #0, ls1                    ' 从端口 1 接收字符串，赋值到 ls1。
LINEINPUT #1, ls2                    ' 从端口 2 接收字符串，赋值到 ls2。
LINEINPUT ls3                        ' 从端口 2 接收字符串，赋值到 ls3。
LINEINPUT #1,ls4,WTIME=100,RVAL=li
                                     ' 在没有输入到间隔符时，处理移动至
                                     '100msec 时间之后的指令。
                                     ' 表示 li 中超时所遗漏的 0 输入。
```

PRINT（语句）

功　能

从 RS232C 及 Ethernet 端口输出数据。

格　式

PRINT［#< 线路编号 >，］< 讯息 >［< 分隔符 >< 讯息 >…］［< 分隔符 >］

说　明

从 RS232C 和 Ethernet 端口向外部输出 < 讯息 > 的值。

在 < 线路编号 > 中，指定使用的 RS232C 和 Ethernet 端口的线路编号。若省略 < 线路编号 >，则为默认值的 ch2。ch1 不能指定为多功能教导器用。

（参照 "2.4.1 线路编号 "）

< 分隔符 > 内可使用逗号 (，) 和分号 (；)。

为逗号 (，) 时，< 讯息 > 与 < 讯息 > 之间，加入空格字符。

<讯息> 之后指定逗号 (，)，若其后没有 <讯息> 时，最后的 <讯息> 之后送出间隔符（CR 或 CR+LF），完成输出。

为分号 (；) 时，与 < 讯息 > 之间不能加入空格字符。< 讯息 > 之后指定分号 (；)，若其后没有 < 讯息 > 时，最后的 < 讯息 > 之后送出间隔符 （CR 或 CR+LF），完成输出。

输出姿势 (Pose) 型数据时，用空格字符将各个要素分开输出。

相关项目

WRITE

应用示例

```
DEFINT li1, li2
DEFSTR ls1, ls2
PRINT #1, li1, ls1; ls2        ' 从 ch2 输出 li1、 ls1、 ls2 的值。（li1 和 ls1
                               ' 之间加入空格， ls1 和 ls2 之间不空格。
                               )
PRINT li2                      ' 从 ch2 输出 li2 的值。
```

WRITE（语句）

功　能

从 RS232C 及 Ethernet 端口输出数据。

格　式

WRITE ［# < 线路编号 >，］ < 讯息 > ［， < 讯息 >…］

说　明

从 RS232C 和 Ethernet 端口向外部输出 < 讯息 > 的值。

在 < 线路编号 > 中，指定使用的 RS232C 和 Ethernet 端口的线路编号。若省略 < 线路编号 >，则为默认值的 ch2。ch1 不能指定为多功能教导器用。（参照 "2.4.1 线路编号 "）

如果开始写复数的 < 讯息 >，以逗号 (，) 断开。

与 PRINT 语句有如下差异。

- 字符串的数据以双引号 (") 括起输出。
- 在输出信息结束时，附加间隔符 （CR 或 CR+LF）。
- 断开 < 讯息 > 的逗号 (，) 直接输出。
- 输出姿势 (Pose) 型数据时，用逗号 (，) 将各个要素分开输出。

相关项目

PRINT

应用示例

```
DEFINT li1, li2
DEFSTR ls1, ls2
WRITE #1, li1, ls1, ls2          ' 从 ch2 输出 li1、 ls1、 ls2 的值。
WRITE li2                        ' 从 ch2 输出 li2 的值。
```

FLUSH（语句）

功　能

清空输入缓冲区。

格　式

FLUSH ［#< 线路编号 >］

说　明

清空 RS232C 和 Ethernet 端口的输入缓冲区。

在 < 线路编号 > 中，指定清空输入缓冲区的 RS232C 和 Ethernet 端口的线路编号。若省略 < 线路编号 >，则为默认值的 ch2。 ch1 不能指定为多功能教导器用。（参照 "2.4.1 线路编号 "）

在接收数据之前，为了清空接收数据的输入缓冲区残留的数据，执行 FLUSH 指令。

相关项目

INPUT

应用示例

```
FLUSH #1                    ' 清空 ch2 的线路的输入缓冲区。
FLUSH                       ' 清空 ch2 的线路的输入缓冲区。
```

13.3 串行二进制通信 （RS-232C 以及 Ethernet 端口）

PRINTB【Ver.1.5 以上版本】

功　能

向 RS-232C 及 Ethernet 端口输出 1 字节 (Byte) 数据。

格　式

PRINTB # < 端口编号 >、 < 输出数据存放 I 变量 >

< 端口编号 >

输出端口编号
（1: 控制器 RS-232C 端口、 -1: μVision 板 RS-232C 端口
4 ～ 7: Ethernet 服务器端口，8 ～ 15: Ethernet 用户端口）

< 输出数据存放 I 变量 >

存放输出数据的 I 型变量

说　明

将以 < 输出数据存放 I 变量 > 指定的 I 型变量的下位字节 (Byte) 输出到指定端口。

应用示例

```
'!TITLE " ＜标题＞ "
PROGRAM sample
        .
        .
        .
        PRINTB #1,I10                    ' 将存放在 I10 的数据下位字节 (Byte) 输出到 RS-232C 端口
        .
        .
        .
END
```

INPUTB【Ver.1.5 以上版本】

功　能

从 RS-232C 及 Ethernet 端口输入 1 字节 (Byte) 数据。

格　式

INPUTB［#＜端口编号＞，］＜输入数据存放 I 变量＞［，WTIME＝＜超时时间＞［，RVAL＝＜存放变量＞］］

＜端口编号＞

输入端口编号
（1: 控制器 RS-232C 端口、 -1: μVision 板 RS-232C 端口
4 ～ 7: Ethernet 服务器端口，8 ～ 15: Ethernet 用户端口）
省略时 1: 控制器 RS-232C 端口

＜输入数据存放 I 变量＞

输入数据所存放的 I 型变量的编号

＜超时时间＞［RC7_Ver2.2 以上版本］

没有输入数据时，在经过指定时间之后，结束 inputb 语句的执行，控制转移到下一个指令。单位为 ms，但是，实际的延迟时间以 1 / 60s 单位增减。

＜存放变量＞［RC7_Ver2.2 以上版本］

＜超时时间＞的设定成对使用。若收到输入数据正常脱开时， TRUE (1) 进入，若超时脱开时，则 FALSE (0) 进入指定的变量。

说　明

在＜输入数据存放 I 变量＞所指定的 I 型变量上，存放从端口输入的数据。

注意

不设定超时时间，而用该指令进行数据输入时，如果端口中不存在数据，则要无限期等待。在对数据是否存在进行确认时，请使用 com_state 指令。

应用示例

```
'!TITLE " ＜标题＞ "
PROGRAM sample
        defint li
        .
        .
        .
        INPUTB #1,I10                       ' 将从 RS-232C 端口输入的数据存放在 I10 上
        INPUTB #1,I10,WTIME=100,RVAL=li     ' 在没有输入时， 100msec 的时间之后进
                                            ' 行下一个指令。表示 li 中超时
                                            ' 所遗漏的 0 输入。
        .
        .
        .
END
```

LPRINTB【Ver.1.5 以上版本】

功　能

从 RS-232C 及 Ethernet 端口输出多字节 (Byte) 数据。

格　式

LPRINTB # < 端口编号 >、 < 存放排列前头的要素 >、 < 输出字节 (Byte) 数 >

< 端口编号 >

输出端口编号
（1: 控制器 RS-232C 端口、 -1: μVision 板 RS-232C 端口
4 ～ 7: Ethernet 服务器端口，8 ～ 15: Ethernet 用户端口）

< 存放排列开头因素 >

输出数据存放的排列的开头要素

< 输出字节 (Byte) 数 >

输出的数据的字节 (Byte) 数

说　明

将从输出数据存放的排列的指定要素只指定的输出字节 (Byte) 数量的数据输出到指定端口。

应用示例

```
'!TITLE " ＜标题＞ "
PROGRAM sample
        .
        .
        .
        LPRINTB #1,I64, 30          ' 连续从 I64 到 I93 的数据的下位字节 (Byte)，
                                    ' 从 RS-232C 端口输出
        .
        .
        .
END
```

LINPUTB【Ver.1.5 以上版本】

功　能

从 RS-232C 及 Ethernet 端口输入多字节 (Byte) 数据。

格　式

LINPUTB［#＜端口编号＞,］＜存放排列开头要素＞,＜输入字节 (Byte) 数＞［, WTIME＝＜超时时间＞［, RVAL＝＜存放变量＞］］

＜端口编号＞

输入端口编号
（1: 控制器 RS-232C 端口、 -1: μVision 板 RS-232C 端口
4 ～ 7: Ethernet 服务器端口，8 ～ 15: Ethernet 用户端口）
省略时 1: 控制器 RS-232C 端口

＜存放排列开头因素＞

输入数据存放的排列的开头要素

＜输入字节 (Byte) 数＞

输入的数据的字节 (Byte) 数

＜超时时间＞［RC7_Ver2.2 以上版本］

没有输入字节 (Byte) 数的输入数据时，在经过指定时间之后，结束 linputb 语句的执行，控制转移到下一个指令。单位为 ms，但是，实际的延迟时间以 1 / 60s 单位增减。

＜存放变量＞［RC7_Ver2.2 以上版本］

＜超时时间＞的设定成对使用。若收到输入数据正常脱开时， TRUE (1) 进入，若超时脱开时，则 FALSE (0) 进入指定的变量。

说　明

从排列的指定因素，从只有指定输入字节 (Byte) 数指定端口输入数据。

注意

不设定超时时间而用该指令进行数据输入时，如果端口中不存在＜输入字节 (Byte) 数＞部分的数据，则要无限期等待。在对数据是否存在进行确认时，请使用 com_state 指令。

应用示例

```
'!TITLE " ＜标题＞ "
PROGRAM sample
        defint li
        .
        .
        LINPUTB #1, I64, 30                     ' 从 I64 到 I93 连接数据，从 RS-232C
                                                ' 端口输入
LINPUTB #1,I64, 30,WTIME=100,RVAL=li            ' 没有＜输入字节 (Byte) 数＞部分的输入时，
                                                ' 处理移动至 100msec 时间之后的指令。表示 li 中
                                                ' 超时所遗漏的 0 输入。
        .
        .
END
```

com_encom【Ver.1.5 以上版本】

功　能

为使用 RS-232C 端口或 Ethernet 用户端口进行二进制通信做准备。

RS-232C 端口的情况下，为了进行二进制通信而占有端口。

使用 Ethernet 用户端口时，打开用户端口，在端口与服务器之间建立连接。

格　式

com_ encom # < 端口编号 >

< 端口编号 >

输出端口编号
1: 控制器 RS-232C 端口、 -1: μVision 板 RS-232C 端口
8 ～ 15: Ethernet 用户端口

说　明

在进行二进制通信时，用 WINCAPSIII 等进行 PC ←→控制器间通信时，请勿使通信数据相互混杂。

二进制通信结束后，需要用 com_discom 指令释放通信端口。

注意

- 实行该指令以后，即使用 WINCAPSIII 等发送数据，也可作为二进制通信数据处理。 RS-232C 端口并非关闭，对此请予注意。
- 在 Ethernet 连接状态下，即使实行该指令之后，与 WINCAPSIII 间的数据通信也不受影响。

注意事项

如果以很短的间隔进行 com_discom、com_encom，可能会因 com_encom 而发生用户打开失败错误。请像上述用例那样，使用确认未建立线路的程序。

请勿在 For loop 的过程中使用 com_discom、 com_encom。

应用示例

< 使用 Ethernet 端口时 >

```
PROGRAM sample
DEFINT Flg

com_state #8,Flg                    ' 获取端口的状态
IF Flg=-1 THEN                      ' 未建立线路的话
com_encom #8                        ' 端口打开
ENDIF

END
```

com_discom【Ver.1.5 以上版本】

功　能

为了结束二进制通信释放 RS-232C 端口。（释放 COM 端口）

使用 Ethernet 外端口时，关闭外端口，切断连接。

格　式

com_discom # < 端口编号 >

说　明

为了其他的用途，将进行二进制通信所占有的通信端口释放。

注意

为了防止数据的混合，如果执行该指令，则要将 RS-232C 端口进行一次清空。

注意

如果在使用 Ethernet 时进行该处理，则未发送数据不等待发送结束即将其切断，所以在接受一方会持续等待接受。

注意事项

如果以很短的间隔进行 com_discom、 com_encom，可能会因 com_encom 而发生用户打开失败错误。请像 com_encom 的用例那样，使确认未建立线路的程序变为可使用状态。

请勿在 For loop 的过程中使用 com_discom、 com_encom。

应用示例

```
'!TITLE " ＜标题＞ "
PROGRAM sample
        .
        .
        .
        com_discom #1                    ' 从端口的二进制通信的释放
        .
        .
        .
END
```

com_state【Ver.1.5 以上版本】

功　能

RS-232C 及获取 Ethernet 端口的状态。

格　式

com_state # < 端口编号 >，< 状态存放 I 型变量 >

说　明

RS-232C 及获取 Ethernet 通信缓冲区的残余数据数。但是，通信端口错误发生时返回 -1。
Ethernet 使用时，若不能确立该端口的连接时，也返回 -1。

应用示例

```
'!TITLE " ＜标题＞ "
PROGRAM sample
        .
        .
        .
        com_state #1,I280                    ' 获取在 I280 中通信缓冲区内积存的数据数
        .
        .
        .
END
```

13.4 多功能教导器

PRINTMSG（语句）

功　能

将讯息附上字幕和图标，在多功能教导器的彩色液晶中显示。

格　式

PRINTMSG < 讯息字符串 >， < 图标类型 >， < 字幕字符串 >

说　明

将自变量 (argument) 的各自指定的讯息附上字幕和图标，在多功能教导器的彩色液晶中显示。

图标类型	图标
0	
1	
2	
3	
4	

可以表示的 < 讯息字符串 > 最大为 60 字符， < 字幕字符串 > 最大是 40 字符。

相关项目

PRINTDBG、 PRINTLBL

应用示例

```
PRINTMSG "Hello World !", 1, " 讯息 "
                                    ' 追加字幕和图标显示 "Hello World !"。
```

PRINTDBG（语句）

功　能

向调试视窗输出数据。

格　式

PRINTDBG < 讯息 > ［< 分隔符 >< 讯息 >…］［< 分隔符 >］

说　明

在教导器的程序调试视窗上输出 < 讯息 > 的值。
< 分隔符 > 内可使用逗号 (，) 和分号 (；)。
在逗号 (，) 时，逗号和后面的 < 讯息 > 之间，加入空格字符。
为分号 (；) 时，与 < 讯息 > 之间不能加入空格字符。
输出姿势 (Pose) 型数据时，用空格字符将各个要素分开显示。
在 < 讯息 > 的最后没有 < 分隔符 > 时，进行换行。

相关项目

PRINTMSG、 PRINTLBL

应用示例

```
PRINTDBG "DEBUG"                    ' 向调试视窗输出 "DEBUG"。
```

BUZZER（语句）

功　能

鸣响蜂鸣器。

格　式

BUZZER < 鸣响时间 >

说　明

在 < 鸣响时间 > 上指定的时间使多功能教导器的蜂鸣器鸣响。
< 鸣响时间 > 的单位是 msec。

应用示例

```
BUZZER 3000                         ' 蜂鸣器鸣响 3 秒钟。
```

PRINTWARNING（语句）【RC7_Ver.2.2 以上版本】

功　能

在教导器警告讯息显示部分显示讯息。

格　式

PRINTWARNING < 表示讯息 > ［，< 讯息颜色 >］

说　明

在教导器警告讯息显示部分显示讯息。

表示讯息的最大长度用半角至 55 个字符。其以上部分可舍去。

讯息颜色

0: 黑色 1: 青色 2: 绿色 3: 红色 其他 : 黑色 （省略时也为黑色。）

仅限表示讯息，不记录错误 LOG。

仅限教导器的功能。

应用示例

PRINTWARNING "Hello World!",1

PRINTLBL（语句）

功　能

向用户定义按钮设定标签 （字幕）。

格　式

PRINTLBL < 面板编号 >，< 按钮编号 >，< 字幕字符串 >

说　明

在多功能教导器的各操作面板的用户定义按钮设定标签 （字幕）。

< 面板编号 > 中到 0 ～ 6 为止的编号可指定。

< 按钮编号 > 中到 0 ～ 11 为止的编号可指定。

以 6 个半角字符以内的字符串指定 < 字幕字符串 >。

相关项目

PRINTDBG、 PRINTMSG

应用示例

```
PRINTLBL 3, 1, "E_STOP"          ' 将第 3 个面板的第 1 个按钮的标签以 "E_ STOP" 设定。
```

set_button【Ver.1.5 以上版本】

功　能

进行按钮参数的设定。

格　式

set_button < 按钮编号 >、 < 参数类型 >、 < 变更参数值 >

< 按钮编号 >

操作盘上所配置的按钮在总体中的编号

< 参数类型 >

色彩、位置等按钮的属性 （参照下表）

< 变更参数值 >

按钮分别具有的变更参数值 （参照下表）

< 参数类型 >	意思	< 变更参数值 >（注 1）
1	左上 X 坐标	0 ～ 640、点单位
2	左上 Y 坐标	0 ～ 350、点单位
3	右下 X 坐标	0 ～ 640、点单位
4	右下 Y 坐标	0 ～ 350、点单位
5	按钮类型	0: 什么也没有 1: 标签 2: 线 17: 2D 按钮 （变量变更） 18: 3D 按钮 （变量变更） 19: 3D 按钮 （变量变更） 20: 圆 （变量变更） 33: 2DLED （指示灯） 34: 圆 LED （指示灯） 35: 3D 按钮 (IO 变更) 36: 预约 （请勿使用。） 37: 预约 （请勿使用。） 38: 箱
6	按钮状态	0: 字符中心 1: 字符左首 2: 字符右首

< 参数类型 >	意思	< 变更参数值 > （注 1）
7	背景颜色	0: 黑 1: 蓝 2: 绿 3: 青绿色 4: 红 5: 桃红色 6: 棕 7: 浅灰色 8: 灰色 9: 浅蓝色 10: 浅绿色 11: 浅青绿色 12: 浅红色 13: 浅桃红色 14: 黄 15: 白
8	文字颜色	0: 黑 1: 蓝 2: 绿 3: 青绿色 4: 红 5: 桃红色 6: 棕 7: 浅灰色 8: 灰色 9: 浅蓝色 10: 浅绿色 11: 浅青绿色 13: 浅桃红色 14: 黄 15: 白
9	可以使用的状态	0: 不使用 1: 使用
10	可视状态	0: 不可见 1: 可见
11	变量种类 （注 2）	1: 整数 2: 实数 3: 倍精度实数 4: 字符串 （最大 32 字符）
12	变量编号	通过变更变量按钮进行变更的变量编号
13	I / O 编号	通过 IO 变更按钮变更的变量编号 (128~511)
14	显示页面	编号按钮所显示的页面编号

注 1: 如果将整数以外的值指定为变更参数值，则会发生错误 " 数据标题异常 "。请用整数值输入。

注 2: 用数值输入按钮使用实数、倍精度实数时，因为实数需要 13 个字符、倍精度实数需要 22 个字符的区域，所以按钮区域的设定要与此相一致。

说　明

将 < 参数类型 > 的值变为 < 变更参数值 >、变更 < 按钮编号 > 上所指定的编号的按钮规格。

不同按钮类型的示例

表示每个按钮类型的示例。

①标签

②线

③ 2D 按钮 （变更变量）

④ 3D 按钮 （变更变量）

⑤ 3D 按钮 （变更变量）

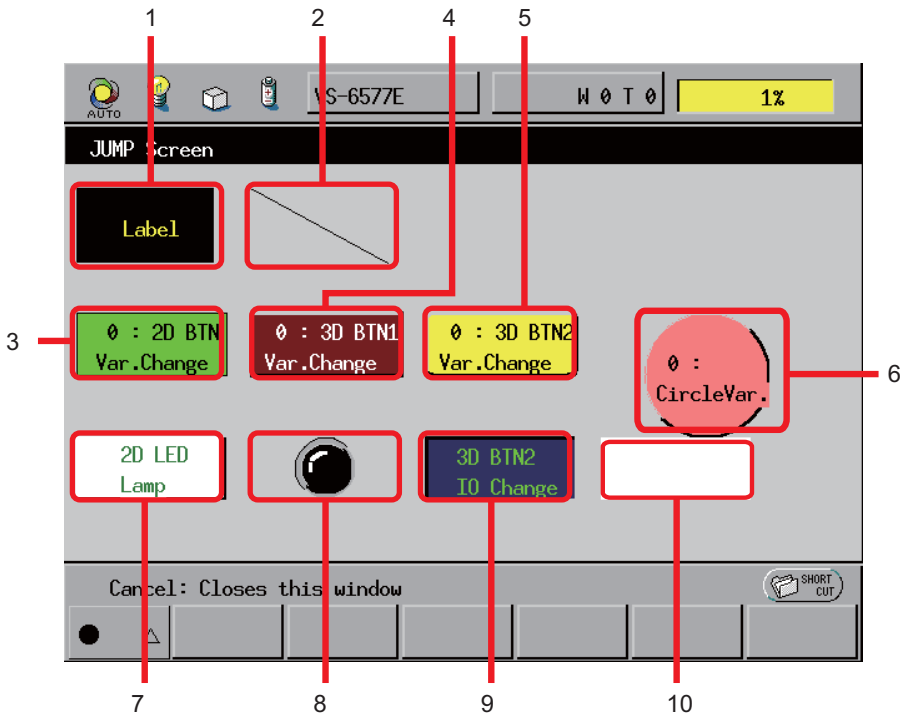
⑥圆

⑦ 2 DLED

⑧圆 LED

⑨ 3D 按钮 （IO 变更）

⑩箱



应用示例

```
'!TITLE " <标题> "
PROGRAM sample1
        .
        .
        .
        set_button (btn_no),(1),(minx)
        set_button (btn_no),(2),(miny)
        set_button (btn_no),(3),(maxx)
        set_button (btn_no),(4),(maxy)
        .
        .
        .
END
```

set_page【Ver.1.5 以上版本】

功　能

进行页面参数的设定。

格　式

set_page < 页编号 >、 < 参数类型 >、 < 变更参数值 >

< 页编号 >

操作盘上所配置的页在总体中的编号

< 参数类型 >

色彩、位置等页的属性 （参照下表）

< 变更参数值 >

页分别具有的变更参数值 （参照下表）

< 参数类型 >	意思	< 变更参数值 >
1	页的种类	0 固定
2	按钮状态	未使用
3	背景颜色	0: 黑 1: 蓝 2: 绿 3: 青绿色 4: 红 5: 桃红色 6: 棕 7: 浅灰色 8: 灰色 9: 浅蓝色 10: 浅绿色 11: 浅青绿色 12: 浅红色 13: 浅桃红色 14: 黄 15: 白

< 参数类型 >	意思	< 变更参数值 >
4	文字颜色 （Ver.1.5 未使用）	0: 黑 1: 蓝 2: 绿 3: 青绿色 4: 红 5: 桃红色 6: 棕 7: 浅灰色 8: 灰色 9: 浅蓝色 10: 浅绿色 11: 浅青绿色 12: 浅红色 13: 浅桃红色 14: 黄 15: 白
5	可以使用的状态	0: 不使用 1: 使用
6	可视状态	0: 不可见 1: 可见

说　明

将 < 参数类型 > 的值变为 < 变更参数值 >、变更 < 页编号 > 上所指定的编号的页的规格。

应用示例

```
'!TITLE " ＜标题＞ "
PROGRAM sample2
        .
        .
        .
        set_page panel_no,P_BGCOLOR,GLAY          ' 页的背景颜色设定
        set_page panel_no,P_USESTATE,ON           ' 页使用的设定
        set_page panel_no,P_VISSTATE,ON           ' 页可视的设定
        .
        .
        .
END
```

change_bCap 【Ver.1.5 以上版本】

功　能

进行按钮的字幕设定。

格　式

change_bCap < 按钮编号 >、 < 指定字幕 >

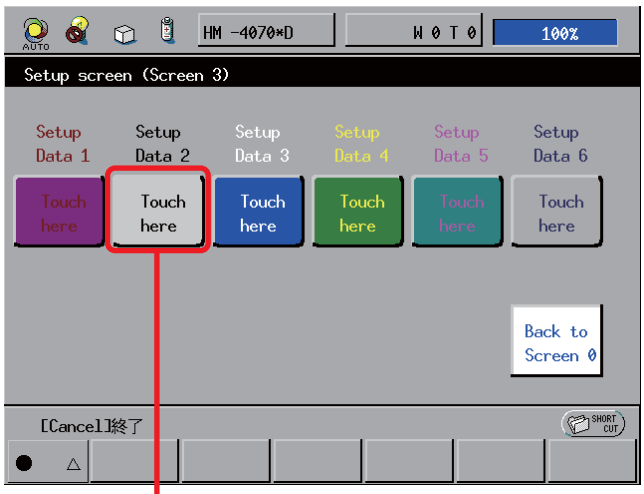
< 按钮编号 >

操作盘上所配置的按钮在总体中的编号

< 指定字幕 >

按钮中心所显示的字符串

（图示示例）



按钮字幕示例

说　明

将在 < 按钮编号 > 上指定的编号的按钮的中心所显示的字符串进行 < 指定字幕 > 变更。

应用示例

```
'!TITLE " <标题> "
PROGRAM sample3
        .
        .
        .
        bcap4 = " 工件切断 "
        btn_no = 3
        .
        .
        .
        change_bCap btn_no,bcap4
        .
        .
        .
END
```

change_pCap【Ver.1.5 以上版本】

功　能

进行页面的字幕设定。

格　式

change_pCap < 页编号 >、 < 指定字幕 >

< 页编号 >

　　操作盘上所配置的页在总体中的编号

< 指定字幕 >

　　页的标题栏表示的字符串

（表示示例）



说　明

变更用页编号指定的编号的页的标题栏表示的字符串。

应用示例

```
'!TITLE " <标题> "
PROGRAM sample4
        .
        .
        .
        pcap4 = " 工件切断 "
        page_no = 3
        .
        .
        .
        change_pCap page_no,pcap4
        .
        .
        .
END
```

disp_page【Ver.1.5 以上版本】

功　能

进行页面的显示。

格　式

disp_page < 页编号 >

< 页编号 >

操作盘上所配置的页在总体中的编号

说　明

将页编号上所指定的编号的页面显示在操作盘画面上。

应用示例

```
'!TITLE " ＜标题＞ "
PROGRAM sample3
        .
        .
        .
        'disp_page panel_no                    ' 指定画面的表示
        .
        .
        .
END
```

第 14 章
多项任务控制语句

多项任务控制是 PAC 的特征之一。在本章中，介绍多项任务控制指令。

第 14 章　多项任务控制语句

14.1　任务控制

RUN（语句）

功　能

并列启动其他程序。

格　式

RUN < 程序名 >[(< 自变量 (argument)>[,< 自变量 (argument)>…])][,<RUN 选择 >]

说　明

从当前正在执行的程序中并行启动在 < 程序名 > 上所指定的程序。不能将自程序进行 RUN。

< 自变量 >

在 < 自变量 (argument)> 中，只能使用传值。指定传址时，自动进行传值，但不能使用独立排列。

<RUN 选件 >

可以对 <RUN 选件 > 指定下列 3 个选件。

PRIORITY （或者 P）

指定程序的优先度。如果省略，则作为默认值 128 进行处理。

数值越小，则优先度越高。设定范围在 102 ～ 255 之间。

注意

特权任务的优先顺序不能变更。

CYCLE （或者 C）

指定切换周期 （反复启动程序时的每个周期的间隔）。单位是 msec。设定范围在 1 ～ 2147483647 之间。

带有自变量 (argument) 的程序，添加 CYCLE 选择不能启动。

RESULT （Ver.2.6 以后的控制器）

执行 RUN 指令时，即使发生内部错误也不会停止程序。内部错误的错误代码将储存到用 RESULT=（变量名）指定的变量中。

相关项目

CALL、 GOSUB

注意事项

- 通过 SUSPEND 命令让动作停止过程中的任务再次 RUN 时，请动作停止之后进行 RUN。如果在动作停止结束之前进行 RUN 使其再动作，则会造成指令速度限制超程等错误。
- 在使用 CYCLE 选择时，如果在命令执行过程中进行瞬间停止而再进行启动，则瞬时停止过程中也要进行循环同步经过，对此请予注意。
- 在教导检查模式下， RUN 命令不能执行。

应用示例

```
DEFINT Li1 = 1, Li2 = 2, Li3 = 3
RUN samp1, C = 1000                    ' 将 samp1 进行并行 (C = 1000) 启动。
RUN samp2 (Li1)                        ' 在 samp2 上添加 Li1 的自变量 (argument) 并行启动。
RUN samp3 (Li1, Li2), PRIORITY = 129
                                       ' 在 samp3 上添加 Li1、 Li2 的自变量 (argument) 并行 (P = 129) 启动。

RUN samp4 (Li1, Li2), PRIORITY = 150
                                       ' 在 samp4 上添加 Li1、 Li2 的自变量 (argument) 并行 (P = 150) 启动。

RUN samp5 (Li1, Li2, Li3), PRIORITY = 120
                                       ' 在 samp5 上添加 Li1、 Li2、 Li3 的自变量 (argument) 并行 (P = 120) 启
                                       动。

RUN samp6, RESULT = I1                 ' 启动 samp6。
                                       'RUN 失败时，将错误代码储存到变量 I1 中，然后移到下一行。
```

KILL（语句）

功　　能

强制结束任务。

格　　式

KILL < 程序名 >

说　　明

强制结束 < 程序名 > 所指定的任务 （程序）的执行。

对自程序不能进行 KILL。如果对自程序进行 KILL，则会发生错误。为了强制结束自程序，请使用 STOP 命令。

相关项目

SUSPEND、 STOP

注意事项

强制结束正在获取臂信号 (semaphore) 的任务，通过其他任务获取臂信号 (semaphore) 时，有时会发生 " 臂信号 (semaphore) 获取失败 " 的错误。此时，在其他任务获取臂信号 (semaphore) 之前，请勿加入计时器等。

应用示例

```
RUN samp1                    ' 并行启动 samp1。
    .
    .
    .
KILL samp1                   ' 结束 samp1。
```

SUSPEND（语句）

功　　能

暂停任务。

格　　式

SUSPEND < 程序名 >

说　　明

暂停所指定的任务。

对自程序不能进行 SUSPEND。为了瞬间停止自程序，请使用 HOLD 命令。

相关项目

KILL、 HOLD

注意事项

通过 SUSPEND 命令让动作停止过程中的任务再次 RUN 时，请动作停止之后进行 RUN。如果在动作停止结束之前进行 RUN 使其再次动作，则会发生指令速度限制超程等错误。

应用示例

```
SUSPEND samp1                    ' 将 samp1 的任务的执行暂停。
```

DEFEND（语句）

功　能

保护任务。

格　式

DEFEND {ON|OFF}

说　明

通常，程序任务在每隔一定的时间内，会将执行权让给比自己优先次序更高或者同等的程序任务。

不把执行权让给其他的程序任务，而需要一次处理时，要使用 DEFEND 指令。DEFEND ON 执行之后，执行权被确保直至 DEFEND OFF 被执行为止。但是，如果执行动作命令和 DELAY、WAIT、SET IO 的时间指定的各个指令，则执行权转移到其他的程序任务。

注意

- 用 DEFEND ON 设定保护的区间要尽可能短。DEFEND ON 执行之后，如果成为无限循环状态，则执行权不转移到其他的程序任务。
- 即使用 DEFEND ON 进行保护，若在以下情况时，保护状态也能自动地被释放。
  - 在执行了 END 指令时（调出的程序最后的 END 指令除外）
  - 执行 KILL 指令时
  - 通过多功能教导器或者 I / O，机械手控制器被初始化时

相关项目

应用示例

```
DEFEND ON                    ' 保护自任务。
SET IO[100]                  ' 因为被保护，必须连续被执行以下的 3 个命令。

SET IO[102]
SET IO[104]
DEFEND OFF                   ' 释放自任务的保护。
```

STATUS（函数）

功　能

得出程序的状态。

格　式

STATUS(< 程序名 >)

说　明

< 程序名 > 所指定的程序的状态以整数被存放。

值	状态	
1	running	正在执行
2	stopping	停止中
3	suspend	暂停中
4	delay	延迟中
5	pending	待机中
6	step stopped	步骤停止中

相关项目

DELAY、 HALT、 HOLD、 STOP

注意事项

- 不能获取自身的 STATUS。
- 在显示程序正在执行的状态下，除了 1:running 之外，4:delay、5:pending 也适合，根据执行指令的处理状况变化。当判断程序是否正在执行时，请参考应用示例中的程序。
- 要想获取利用 CALL 指令打开之程序的状态，请指定所打开之程序的程序名。例如有

```
PROGRUM PRO1
      :
   CALL SUB1
      :
```

这样一个程序，在调出程序 “SUBI” 的过程中，要想获取程序 "SUBI" 的状态，请指定 "PRO1"。如果指定 "SUB1"，将无法获得正确的状态。

```
I0 = STATUS(PRO1)
```

应用示例

```
DIM li1 As Integer
li1 = STATUS(samp1)                    ’将 samp1 的程序状态以整数赋值 li1。

’====== 程序状态的区别处理 ======

DEFINT iX
iX = STATUS(PRO1)
SELECT CASE iX
  CASE 2                               ’停止中
                                       ’停止中的处理
  CASE 3,6                             ’暂时停止中，步骤停止中
                                       ’暂时停止中，步骤停止中的处理
  CASE ELSE                            ’正在执行、延迟中、待机中
                                       ’正在执行的处理
END SELECT
```

SUSPENDALL（语句）【Ver.1.98 以上版本】

功　　能

停止特权任务以外的所有的程序。

格　　式

SUSPENDALL

说　　明

让特权任务以外的所有任务停止，成为从正在停止的步骤可以让其动作的 " 连续停止 " 状态，将机械手运行过程中的输出置为 OFF。

相关项目

SUSPEND、 KILLALL、 ROBOTSTOP、 CONTINUERUN

注意事项

执行 SUSPENDALL 命令程序停止之后，通过 RUN 命令可以执行程序的继续。

此外通过 CONTINUERUN 命令也可以连续启动。

让运行停止过程中的任务再次 RUN 时，请在动作停止之后再进行 RUN。如果在动作结束之前进行 RUN 使其再动作，则会发生指令速度限制超程等错误。

SUSPENDALL 命令执行之后 （与机械手停止输入等、 TP、从外部的停止相关输入全部相同）大约 0.5 秒钟程序不能启动。

应用示例

```
PROGRAM TSR1
------------
SUSPENDALL                    ’所有任务瞬时停止  进入连续停止状态
------------
CONTINUERUN                   ’连续启动
END
```

KILLALL（语句）【Ver.1.98 以上版本】

功　　能

强制结束特权任务以外的所有的任务。（相当于程序复位）

格　　式

KILLALL

说　　明

强制结束机械手当前正在执行的特权任务以外的所有任务，将机械手运行过程中的输出置为 OFF。

相关项目

KILL、 SUSPENDALL、 ROBOTSTOP

注意事项

不能继续让被停止的任务连续动作。 KILLALL 命令执行之后 （与机械手停止输入等、 TP、从外部的停止相关输入全部相同）大约 0.5 秒钟内程序不能启动。

应用示例

```
PROGRAM TSR1
------------
KILLALL                          ’所有任务停止 程序复位状态
------------
END
```

CONTINUERUN（语句）【Ver.1.98 以上版本】

功　　能

进行连续启动。

格　　式

CONTINUERUN

说　　明

对连续停止状态的任务进行继续运行，将所有的任务从被停止步骤的中途进行再启动。

相关项目

SUSPENDALL、 ROBOTSTOP

注意事项

在允许连续开始状态时，只可以从特权任务执行。

应用示例

```
PROGRAM TSR1
------------
SUSPENDALL                    ’所有任务瞬时停止 进入连续停止状态
------------
CONTINUERUN                   ’连续启动
------------
END
```

ROBOTSTOP（语句）【Ver.1.98 以上版本】

功　　能

进行机械手停止。

格　　式

ROBOTSTOP

说　　明

让特权任务以外的所有任务停止，成为从正在停止的步骤可以让其动作的 " 连续停止 " 状态，将机械手运行过程中的输出置为 OFF。此外将电机电源置为 OFF。

补充说明

备注与 "SUSPENDALL" 指令的不同点是电机电源被 OFF，由此引起的外部输出发生变化。 ROBOTSTOP 命令执行之后 （与机械手停止输入等、 TP、从外部的停止相关输入全部相同）大约 0.5 秒钟内程序不能启动。

此外，该指令是多功能教导器的设定，通过［F1 程序］－［F6 辅助功能］－［F7 连续设定］的操作所显示画面的参数 0: 连续 （0: 无效 1: 连续）是无效时，程序不连续停止而处于停止状态。

相关项目

SUSPENDALL、 CONTINUERUN

应用示例

```
PROGRAM TSR1
------------
ROBOTSTOP                          ’机械手停止
------------
END
```

TAKEARMSTATE （函数）【RC7_Ver.2.2 以上版本】

功　能

返回臂组控制权的获取状态。
0: 尚未获取、1: 已经获取

格　式

TAKEARMSTATE (< 臂组 No>)

说　明

返回当前的指定臂组 No 控制权的获取状态。
< 臂组 No>
指定想获取状态的臂组 No。

应用示例

```
PROGRAM PRO1
------------
I1 = TAKEARMSTATE (1)              ’获取臂组 1 的获取状态
------------
END
```

LOCKSTATE （函数）【RC7_Ver.2.2 以上版本】

功　能

获取锁定的状态。
0: OFF、　1: ON

格　式

LOCKSTATE

说　明

获取当前的锁定的状态。

应用示例

```
PROGRAM PRO1
------------
I1 = LOCKSTATE                     ’获取机器锁定的状态
------------
END
```

DEADMANSTATE （函数）【RC7_Ver.2.2 以上版本】

功　能

双重安全开关的状态。

0: OFF、　1: ON

格　式

DEADMANSTATE

说　明

获取当前的双重安全开关的状态。

应用示例

```
PROGRAM PRO1
------------
I1 = DEADMANSTATE                    ’获取双重安全开关的状态
------------
END
```

SEMIDSTATE（函数）【RC7_Ver.2.2 以上版本】

功　　能

返回所指定的信号 (semaphore) ID 是有效还是无效。

格　　式

SEMIDSTATE(< 信号 (semaphore) ID>)

说　　明

返回所指定的信号 (semaphore) ID 的 " 有效 (1) ／无效 (0) "。

CREATESEM 执行时信号 (semaphore) ID 为有效， DELETESEM 执行时信号 (semaphore) ID 为无效。

相关项目

CREATESEM、 DELETESEM

应用示例

TSR1 创建 PRO1、 PRO2 上所使用的信号 (semaphore)。

使用 SEMIDSTATE 只在信号 (semaphore) ID 非有效时创建信号 (semaphore)。

```
'!TITLE " 信号 (semaphore) 创建 "
PROGRAM TSR1
  FOLDER DEFINT SEMID                    'SEMID 只在非有效 (1) 时创建信号 (semaphore)
  IF SEMIDSTATE (SEMID) <> 1
    SEMID = CREATESEM (0)
  ENDIF
END

'!TITLE " 用信号 (semaphore) 控制向 I0 进行写入 "
PROGRAM PRO1
  EXTERN DEFINT SEMID
  TAKESEM SEMID
    I0=1
  GIVESEM SEMID
END

'!TITLE " 用信号 (semaphore) 控制向 I0 进行写入 "
PROGRAM PRO2
  EXTERN DEFINT SEMID
  TAKESEM SEMID
    I0=1
  GIVESEM SEMID
END
```

14.2 信号 (semaphore)

信号 (semaphore) 是在让多个任务同步进行 （同步控制）、或在同时不要运行 （排他控制）时，用于任务之间的通信 （信号 (semaphore) 的联络）。

为了利用信号 (semaphore)，首先要用 CREATESEM 指令生成信号 (semaphore)，获取信号 (semaphore) ID。基于信号 (semaphore) ID，可以从多个的信号 (semaphore) 中指定特定的信号 (semaphore)。

进行同步或排他控制时 , 等待其他任务指令一侧的任务执行 TAKESEM 指令 , 发出指令一侧的任务等待执行 GIVESEM 指令。发出指令一方的任务在准备好后，执行 GIVESEM 指令，允许对等待信号 (semaphore) 的任务进行处理执行。

一个 GIVESEM 指令只对一个等待信号 (semaphore) 任务有效。

多个的任务在等待相同信号 (semaphore) ID 的信号 (semaphore) 时 , 从哪个任务开始执行是可以从先到达顺序或优先顺序的 2 个排队 （等待执行）方式选择的。

CREATESEM （函数）

功　能

生成信号 (semaphore)。

格　式

CREATESEM(< 算式 >)

说　明

生成信号 (semaphore)、获取信号 (semaphore) ID。

如果没有信号 (semaphore) ID，则其他的信号 (semaphore) 相关的指令就不能使用，所以要在信号 (semaphore) 使用之前必须执行该 CREATESEM 指令。

用自变量 (argument) 指定任务的排队（等待执行）方式。决定多个任务在等待相同信号 (semaphore) 时的执行顺序。

排队方式有以下 2 种。

自变量	
0	先着顺序
0 以外	任务的优先顺序

先到达顺序是执行 TAKESEM 指令的顺序。

任务的优先顺序是被 RUN 指令的选择 (PRIORITY) 指定的。

如果执行 CREATESEM，则在该时刻就是信号 (semaphore) 存在的状态，所以即使不执行 GIVESEM，也可以执行 TAKESEM。

信号 (semaphore) 可以生成 32 个。

相关项目

DELETESEM、 FLUSHSEM、 GIVESEM、 TAKESEM

注意事项

CREATESEM 命令使用时的注意 :

- 由于错误的使用方法而发生的现象
  根据 CREATESEM 命令已经被生成的具有信号 (semaphore) ID 的程序在动作中、暂停待机时，
  - 根据 SEMCREATE 改写信号 (semaphore) ID 存放变量、
    或者
  - 如果从多功能教导器上故意对其变量进行改写，则具有信号 (semaphore) ID 的程序不能取得信号 (semaphore) 而一直处于待机中。
- 例
  启动 PRO1、在 PRO2 动作中通过 STOP 键进行瞬时停止。那之后如果再启动 PRO1，由于被 i1 存放的信号 (semaphore) ID 被变更，所以 PRO3 会无限期地持续等待信号 (semaphore)。

```
PROGRAM PRO1
  i1=CREATESEM(0)
  RUN PRO2
  RUN PRO3
END

PROGRAM PRO2
  TAKESEM i1
  .
  .
  .
  GIVESEM i1
  .
  .
  .
END

PROGRAM PRO3
  TAKESEM i1
  .
  .
  .
  GIVESEM i1
END
```

- 发生时的处置方法
  为了从该状态中跳出，请停止处于无限期等待的程序 （示例为 PR03）之后，启动调出原来的程序 （示例为 PR01）。
- 为了正确使用请遵守的事项
  请注意不要改写处于等待状态的信号 (semaphore) 的 ID。
  具体请遵守以下 2 点。
  - 使用同一变量编号的信号 (semaphore) 只要不被明确删除，就只能生成一次。
  - 管理信号 (semaphore) ID 的变量不用于其他用途。

应用示例

```
DEFINT Li1, Li2, Li3 = 1
Li1 = CREATESEM(Li3)        ' 用 Li3 所指定的排队方式生成信号 (semaphore)，
                            ' 在 Li1 上得出信号 (semaphore) ID。
Li2 = CREATESEM(Li3)        ' 用 Li3 所指定的排队方式生成信号 (semaphore)，
                            ' 在 Li2 上得出信号 (semaphore) ID。
TAKESEM Li1                 ' 获取在 Li1 上被指定的信号 (semaphore)。
TALESEM Li2, 100            ' 获取在 Li2 上被指定的信号 (semaphore)。但是， 100ms 超时。

RUN samp1
GIVESEM Li1                 ' 从等待具有在 Li1 上被指定的信号 (semaphore)
                            ' 的一个任务的状态下进行释放。
FLUSHSEM Li2                ' 从等待具有在 Li2 上被指定的信号 (semaphore)
                            ' 的所有任务的状态下进行释放。
DELETESEM Li1               ' 删除在 Li1 上被指定的信号 (semaphore) ID 的信号 (semaphore)。

DELETESEM Li2               ' 删除在 Li2 上被指定的信号 (semaphore) ID 的信号 (semaphore)。
```

DELETESEM（语句）

功　能

删除信号 (semaphore)。

格　式

DELETESEM< 信号 (semaphore) ID>

说　明

删除在 < 信号 (semaphore) ID> 上所指定的具有信号 (semaphore) ID 的信号 (semaphore)。

相关项目

CREATESEM、 FLUSHSEM、 GIVESEM、 TAKESEM

应用示例

```
DEFINT Li1, Li2, Li3 = 1
Li1 = CREATESEM(Li3)        ' 用 Li3 所指定的排队方式生成信号 (semaphore)，
                            ' 在 Li1 上得出信号 (semaphore) ID。
Li2 = CREATESEM(Li3)        ' 用 Li3 所指定的排队方式生成信号 (semaphore)，
                            ' 在 Li2 上得出信号 (semaphore) ID。
TAKESEM Li1                 ' 获取在 Li1 上被指定的信号 (semaphore)。
TALESEM Li2, 100            ' 获取在 Li2 上被指定的信号 (semaphore)。但是， 100ms 超时。

RUN samp1
GIVESEM Li1                 ' 从等待具有在 Li1 上被指定的信号 (semaphore)
                            ' 的一个任务的状态下进行释放。
FLUSHSEM Li2                ' 从等待具有在 Li2 上被指定的信号 (semaphore)
                            ' 的所有任务的状态下进行释放。
DELETESEM Li1               ' 删除在 Li1 上被指定的信号 (semaphore) ID 的信号 (semaphore)。

DELETESEM Li2               ' 删除在 Li2 上被指定的信号 (semaphore) ID 的信号 (semaphore)。
```

FLUSHSEM（语句）

功　能

释放信号 (semaphore) 等待任务。

格　式

FLUSHSEM< 信号 (semaphore) ID>

说　明

对于在 < 信号 (semaphore) ID> 上所指定的等待信号 (semaphore) 的所有的任务，允许重新开始处理。

相关项目

CREATESEM、 DELETESEM、 GIVESEM、 TAKESEM

应用示例

```
DEFINT Li1, Li2, Li3 = 1
Li1 = CREATESEM(Li3)        ' 用 Li3 所指定的排队方式生成信号 (semaphore)，
                            ' 在 Li1 上得出信号 (semaphore) ID。
Li2 = CREATESEM(Li3)        ' 用 Li3 所指定的排队方式生成信号 (semaphore)，
                            ' 在 Li2 上得出信号 (semaphore) ID。
TAKESEM Li1                 ' 获取在 Li1 上被指定的信号 (semaphore)。
TALESEM Li2, 100            ' 获取在 Li2 上被指定的信号 (semaphore)。但是， 100ms 超时。

RUN samp1
GIVESEM Li1                 ' 从等待具有在 Li1 上被指定的信号 (semaphore)
                            ' 的一个任务的状态下进行释放。
FLUSHSEM Li2                ' 从等待具有在 Li2 上被指定的信号 (semaphore)
                            ' 的所有任务的状态下进行释放。
DELETESEM Li1               ' 删除在 Li1 上被指定的信号 (semaphore) ID 的信号 (semaphore)。

DELETESEM Li2               ' 删除在 Li2 上被指定的信号 (semaphore) ID 的信号 (semaphore)。
```

GIVESEM（语句）

功　能

释放信号 (semaphore) 等待任务。

格　式

GIVESEM< 信号 (semaphore) ID>

说　明

释放在 < 信号 (semaphore) ID> 上所指定的信号 (semaphore)。

释放时，如果有一个在 < 信号 (semaphore) ID> 上所指定的等待信号 (semaphore) 的任务，则允许重新开始处理。存在多个信号 (semaphore) 等待任务的情况下，执行顺序在信号 (semaphore) 生成时被 CREATESEM 指令所指定的排队方式决定。

相关项目

CREATESEM、 DELETESEM、 FLUSHSEM、 TAKESEM

应用示例

```
DEFINT Li1, Li2, Li3 = 1
Li1 = CREATESEM(Li3)        ' 用 Li3 所指定的排队方式生成信号 (semaphore)，
                            ' 在 Li1 上得出信号 (semaphore) ID。
Li2 = CREATESEM(Li3)        ' 用 Li3 所指定的排队方式生成信号 (semaphore)，
                            ' 在 Li2 上得出信号 (semaphore) ID。
TAKESEM Li1                 ' 获取在 Li1 上被指定的信号 (semaphore)。
TALESEM Li2, 100            ' 获取在 Li2 上被指定的信号 (semaphore)。但是， 100ms 超时。

RUN samp1
GIVESEM Li1                 ' 从等待具有在 Li1 上被指定的信号 (semaphore)
                            ' 的一个任务的状态下进行释放。
FLUSHSEM Li2                ' 从等待具有在 Li2 上被指定的信号 (semaphore)
                            ' 的所有任务的状态下进行释放。
DELETESEM Li1               ' 删除在 Li1 上被指定的信号 (semaphore) ID 的信号 (semaphore)。

DELETESEM Li2               ' 删除在 Li2 上被指定的信号 (semaphore) ID 的信号 (semaphore)。
```

TAKESEM（语句）

功　　能

获取具有指定信号 (semaphore) ID 的信号 (semaphore)。

格　　式

TAKESEM < 信号 (semaphore) ID>[,< 超时时间 >]

说　　明

获取在 < 信号 (semaphore) ID> 上所指定的信号 (semaphore)。

在获取时如果其他任务正在获取信号 (semaphore)，则等待其信号 (semaphore) 被释放。

在选择的 < 超时时间 > 上，可以以 ms 为单位对等待时间的限度进行指定。在指定的时间内不能获取信号 (semaphore) 时，会发生错误。

相关项目

CREATESEM、 DELETESEM、 FLUSHSEM、 GIVESEM

注意事项

在使用超时时间时，如果在命令执行过程中进行瞬时停止而再启动，则请注意在暂停过程中也经过等待时间。

应用示例

```
DEFINT Li1, Li2, Li3 = 1
Li1 = CREATESEM(Li3)        ' 用 Li3 所指定的排队方式生成信号 (semaphore)，
                            ' 在 Li1 上得出信号 (semaphore) ID。
Li2 = CREATESEM(Li3)        ' 用 Li3 所指定的排队方式生成信号 (semaphore)，
                            ' 在 Li2 上得出信号 (semaphore) ID。
TAKESEM Li1                 ' 获取在 Li1 上被指定的信号 (semaphore)。
TALESEM Li2, 100            ' 获取在 Li2 上被指定的信号 (semaphore)。但是， 100ms 超时。

RUN samp1
GIVESEM Li1                 ' 从等待具有在 Li1 上被指定的信号 (semaphore)
                            ' 的一个任务的状态下进行释放。
FLUSHSEM Li2                ' 从等待具有在 Li2 上被指定的信号 (semaphore)
                            ' 的所有任务的状态下进行释放。
DELETESEM Li1               ' 删除在 Li1 上被指定的信号 (semaphore) ID 的信号 (semaphore)。

DELETESEM Li2               ' 删除在 Li2 上被指定的信号 (semaphore) ID 的信号 (semaphore)。
```

14.3 臂信号 (semaphore)

TAKEARM（语句）

功　能

获取臂组。当省略了臂组 NO 时，获取臂组 0 （仅限机械手）的控制权。
获取时，将内部速度、加速度、减速度设定为 100。
当获取的臂组包含机械手轴时，将工具坐标、工件坐标返回到原点。

格　式

TAKEARM[< 臂组 NO>][<KEEP= 初始化设定值 >]

说　明

已经具有控制权的任务或从其任务中调出的子程序中执行了 TAKEARM 指令时，继续照此进行处理。

< 臂组 NO> ［V1.5 以上版本］

获取臂组。当省略了臂组 NO 时，
获取臂组 0 （仅限机械手）的控制权。

<KEEP = 初始化设定值 > ［V1.4 以上版本］

初始化设定值 0: 将工具坐标、工件坐标返回原点，将内部速度、加速度、
减速度设定为 100。

初始化设定值

1: 将工具坐标、工件坐标、内部速度、加速度、
减速度设定为当前所设定的值。

<KEEP = 初始化设定值 > 省略时设定为 KEEP = 0 （将工具坐标、工件坐标返回到原点，
内部速度、加速度、减速度为 100)。

(1) 臂组 0 （仅限机械手）时的说明

没有获取臂组的任务，要想执行下表的机械手动作命令时会发生错误。执行这些动作命令的程序，请必须用 TAKEARM 指令获取控制权。

需要控制权的机械手动作命令

种类	指令
定义语句	HOME, TOOL, WORK
机械手控制语句	APPROACH, DEPART, DRAW, DRIVE, DRIVEA, GOHOME, MOVE, ROTATEH, ROTATE, SPEED, JSPEED, ACCEL, JACCEL, DECEL, JDECEL, CHANGETOOL, CHANGEWORK, LETENV, 最佳可搬运质量设定共用程序库 (Library)、臂动作共用程序库 (Library)

注意

- 在以下情况时，自动释放机械手控制权。
  - 一 执行 END 指令时
    （调出的程序最后的 END 指令除外）
  - 一 执行 KILL 指令时
  - 一 执行 HALT 指令时
  - 一 执行 STOP 指令时
  - 一 通过多功能教导器或者 I / O，机械手控制器被初始化时
- 如果执行 TAKEARM 指令，则下一个处理自动被执行。
  - 一 将 TOOL 定义初始化为 TOOL0，将工件坐标设定为机械手的底座坐标
  - 一 将内部速度、内部加速度、内部减速度设定为 100
- 程序暂停中是不进行释放的。

(2) 附加轴相关的说明

没有获取臂组的任务，要想执行下表的动作命令时会发生错误。执行这些动作命令时，请必须用 TAKEARM 指令获取控制权。

需要臂组的指令

指令	需要的臂组
HOME、 TOOL、 WORK、 APPROACH、 DEPART、 DRAW、 GOHOME、 MOVE、 ROTATEH、 ROTATE、 CHANGETOOL、 CHANGEWORK、 DRIVE、 DRIVEA、 SPEED、 JSPEED、 ACCEL、 JACCEL、 DECEL、 JDECEL、 INTERRUPT、 LETENV、最佳可搬运质量设定共用程序库 (Library)、臂动作共用程序 (Library)	需要包括机械手的臂组
DRIVE、DRIVEA、SPEED、JSPEED、ACCEL、JACCEL、DECEL、JDECEL、 INTERRUPT、 LETENV、 POSCLR	需要臂组 （也可以包括机械手的轴）

注意

- DRIVE、 DRIVEA 需要包括让其动作的轴在内的臂组。
  例　DRIVE(7、 10)
  为了运行 7 轴，需要取得包括 7 轴在内的臂组。
- 使用 MOVE 命令的 EX (EXA) 选择时，需要获取包括用机械手轴和 EX (EXA) 运行的附加轴在内的臂组。
  例 MOVE P,P0 EX((7,10))
  这种情况下，需要获取包括机械手轴和 7 轴在内的臂组。
- 速度设定的指令，只对当前获取的臂组的轴进行变更。
- LETENV 命令，需要获取包括变更参数所影响的轴在内的臂组。

相关项目

GIVEARM、 TAKEVIS、 GIVEVIS

注意事项

- 不能在一个程序中获取两个组。
  但是相同的臂组时可以再次获取。

```
例 :   TAKEARM 0
       MOVE P, P0
       TAKEARM 0          ' 即使获取组 0 之后，组 0 也能再次获取。
       TAKEARM 1          ' 获取组 0 之后，因正在获取组 1，
                          ' 错误
```

- 将 TAKEARM 的 KEEP 选择设为 1 获取组时，速度不被初始化，而是以以前的臂组的速度动作。
  例 : 如下的臂组被设定，有以下的 3 个程序时。

	J1	J2	J3	J4	J5	J6	J7	J8
Group 0	○	○	○	○	×	×	×	×
Group 1	×	×	×	×	×	×	○	○
Group 2	○	○	○	○	×	×	○	○
Group 3	×	×	×	×	×	×	×	×
Group 4	×	×	×	×	×	×	×	×

```
PROGRAM PRO1          PROGRAM PRO2          PROGRAM PRO3
TAKEARM 1             TAKEARM 2             TAKEARM 1 keep=1
SPEED 10              SPEED 20              DRIVE(7,10)
DRIVE(7,10)           DRIVE(7,10)           END
END                   END
```

首先让 PRO1 动作。 7 轴以速度 10 运行。
然后让 PRO2 动作。 7 轴以速度 20 运行。
最后让 PRO3 动作时，因为 KEEP 初始化设定值是 1，所以 7 轴以以前的臂组 1 的速度 10 运行。

应用示例

例 1:

```
PROGRAM PRO1                  '
  TAKEARM                     ' 在进行机械手动作程序的开头，执行 TAKEARM。
                              '
  MOVE P, P1
  MOVE P, P2
  GIVEARM 1                   ' 在结束时用 GIVEARM 释放机械手控制权。因为之后
                              ' 的 END 自动释放，所以不需要执行。

END
```

例 2：

```
PROGRAM PRO2
  MOVE P, P3                  '×: 如果不执行 TAKEARM 而执行 MOVE 命令，
                              ' 会发生错误。
END
```

例 3：

```
PROGRAM PRO3                    ’
  TAKEARM                       ’
  MOVE P, P4                    ’
  CALL SUB1                     ’
END                             ’
PROGRAM SUB1                    ’
  MOVE P, P5                    ’ 因为在 PRO3 上已经执行 TAKEARM，所以不发生错误。
END                             ’
```

例 4:

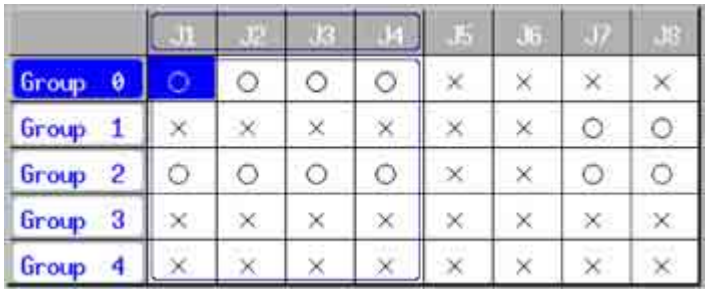
```
PROGRAM PRO4                    ’
  TAKEARM                       ’
  SPEED 50                      ’
  CHANGEWORK 3                  ’
  CHANGETOOL 1                  ’
  MOVE P, P5                    ’
  CALL PRO5                     ’
END                             ’
PROGRAM PRO5                    ’
  TAKEARM                       ’ 因为作为 PRO4 的子程序被调出，所以即使和 PRO4
                                ’ 重复进行 TAKEARM，也不会发生错误。
  MOVE P, P6                    ’ 但是，工具坐标、工件坐标成为 0，内部速度成为 100。
                                ’
END                             ’
```

例 5:

```
PROGRAM PRO6                    ’
  TAKEARM                       ’
  RUN PRO7                      ’
  MOVE P, P7                    ’
END                             ’
PROGRAM PRO7                    ’
  TAKEARM                       ’×: 如果要想用作为和已经具有机械手控制权的 PRO6
                                ’ 不同的其他的任务被启动的 PRO7 获取机械手执行权，
                                ’ 就会发生错误。
  MOVE P, P7                    ’
END                             ’
```

例 6: 有附加轴的示例

以下的臂组被设定时的程序示例。（4 轴机械手的示例）



	J1	J2	J3	J4	J5	J6	J7	J8
Group 0	○	○	○	○	×	×	×	×
Group 1	×	×	×	×	×	×	○	○
Group 2	○	○	○	○	×	×	○	○
Group 3	×	×	×	×	×	×	×	×
Group 4	×	×	×	×	×	×	×	×

```
PROGRAM PRO1
TAKEARM 1                    ' 获取臂组 1。（包括 7 轴的臂组）
DRIVEA(7,100)                ' 将 7 轴运行至 100 度的位置。
END

PROGRAM PRO2
TAKEARM 2                    ' 获取臂组 2。（包括机械手轴、7 轴在内的臂
                             ' 组）
MOVE P,P0 EX((7,10))         ' 让机械手轴和附加轴同时动作。
DRIVEA(7,100)                ' 将 7 轴运行至 100 度的位置。
END
```

GIVEARM（语句）

功　能

释放当前正在获取的臂组。

臂组 0（仅限机械手）时，开放机械手控制权。

格　式

GIVEARM

说　明

释放当前正在获取的臂组。由此，其他的任务可以获取臂组的控制权。

没有获取臂组的任务要执行 GIVEARM 指令时，则发生错误。

注意

在以下情况时，机械手控制权被自动释放。因此，有省略 GIVEARM 指令也没关系的情况。

- 在执行了 END 指令时 （调出的程序最后的 END 指令除外）
- 执行 KILL 指令时
- 通过多功能教导器或者 I / O，机械手控制器被初始化时

相关项目

TAKEARM、 TAKEVIS、 GIVEVIS

注意事项

GIVEARM 执行时，等待当前正在获取的臂组的轴完全停止。因此，即使在 GIVEARM 之前进行路径动作指定路径动作也不会进行。

应用示例

例 1:

```
TAKEARM                 ' 在进行机械手动作程序的开头，执行 TAKEARM。
MOVE P, lp1
MOVE P, lp2
GIVEARM                 ' 在结束 1 时用 GIVEARM 释放机械手控制权。
                        ' 因为之后的 END 自动释放，所以不需要执行。
```

例 2（有附加轴的示例）：

```
PROGRAM PRO1
TAKEARM 1               ' 获取臂组 1。
GIVEARM                 ' 释放当前正在获取的臂组 1。
END

PROGRAM PRO2
TAKEARM 2               ' 获取臂组 2。
END                     ' 程序停止时，释放当前正在获取的臂组 2。
```

TAKEVIS（语句）

功　能

获取视觉处理权。

格　式

TAKEVIS

说　明

获取以选择被内置在机械手控制器中的 μVISION 板上的视觉处理权。μVISION 板没有被内置的状态下，如果执行 TAKEVIS 指令则会发生错误。

其他的任务获取视觉处理权，在没能获取处理权时则会发生错误。

没有获取视觉处理权的任务要想执行视觉命令时，则会发生错误。在执行视觉命令的程序中，请在视觉命令之前，用 TAKEVIS 指令获取视觉处理权。

在具有视觉处理权的任务、或者被该任务调出的辅助程序中执行了 TAKEVIS 指令时，继续在该状态下进行处理。

注意

在以下情况时，视觉处理权被自动释放。

- 在执行了 END 指令时 （调出的程序最后的 END 指令除外）
- 执行 KILL 指令时
- 通过多功能教导器或者 I / O，机械手控制器被初始化时

相关项目

TAKEARM、 GIVEARM、 GIVEVIS

应用示例

```
DEFINT Li1, Li2
TAKEVIS                                  ’获取视觉处理权。
VISSCREEN 1, 0, 1
VISCLS
FOR Li1 = 0 TO 255
  Li2 = VISREFTABLE(1, Li1)              ’获取表 1 号的数据。
  VISLOC 10, 10                          ’显示位置的设定
  VISDEFCHAR 1, 1, 2                     ’显示字符的设定
  VISPRINT " 数据 " ; Li1 ; "=" ; Li2    ’显示
NEXT Li1
GIVEVIS                                  ’视觉处理权的释放
```

GIVEVIS（语句）

功　能

释放视觉处理权。

格　式

GIVEVIS

说　明

释放以选择被内置在机械手控制器中的 μVISION 板上的视觉处理权。由此，其他的任务可以获取视觉处理权。 μVISION 板没有被内置的状态下，如果执行 GIVEVIS 指令则会发生错误。

尽管其他的任务已经获取视觉处理权，但要想执行 GIVEVIS 指令时，仍会发生错误。

注意

在以下情况时，机械手控制权被自动释放。因此，有省略 GIVEVIS 指令也没关系的情况。

- 在执行了 END 指令时 （调出的程序最后的 END 指令除外）
- 执行 KILL 指令时
- 通过多功能教导器或者 I / O，机械手控制器被初始化时

相关项目

TAKEARM、 GIVEARM、 TAKEVIS

应用示例

```
DEFINT Li1, Li2
TAKEVIS                                  ' 获取视觉处理权。
VISSCREEN 1, 0, 1
VISCLS
FOR Li1 = 0 TO 255
  Li2 = VISREFTABLE(1, Li1)              ' 获取表 1 号的数据。
  VISLOC 10, 10                          ' 显示位置的设定
  VISDEFCHAR 1, 1, 2                     ' 显示字符的设定
  VISPRINT " 数据 " ; Li1 ; "=" ; Li2    ' 显示
NEXT Li1
GIVEVIS                                  ' 视觉处理权的释放
```

14.4 特权任务

INIT【Ver.1.7 以上版本】

功　能

根据特权任务参数的状态，进行电机 ON、 CAL、速度的设定。

格　式

INIT [， <INIT 选件 >]

说　明

- 特权任务使用 / 尚未使用为 ［尚未使用］时什么也不执行。
- 特权任务使用 / 尚未使用为 ［使用］时，执行以下的处理。

  INIT 命令执行模式为 ［（电机 ON + CAL） : 非执行］时
  - 如果 INIT 启动速度是 10，则将控制器的外部速度设为 10。
  - 如果 INIT 启动速度是 100，则将控制器的外部速度设为 100。

  INIT 命令执行模式为 ［（电机 ON + CAL） : 执行］时
  - 如果 INIT 启动速度是 10，则将外部速度设为 10、电机置为 ON 以及执行 CAL。
  - 如果 INIT 启动速度是 100，则将外部速度设为 100、电机置为 ON
- 如果在 <INIT 选件 > 记述 "RESULT"，则在执行 INIT 指令时，即使发生内部错误，也不会停止特权任务。将 RESULT=（变量名）中发生的错误的错误代码储存到指定的变量中。（仅限于 Ver.2.6 以后的控制器））

注意事项

- 请勿让在无限循环中只执行 INIT 命令的程序和机械手动作程序同时执行。
- INIT 执行过程中，请注意有时正在动作的程序的显示处于待机过程中。
- 请勿从多个特权任务同时执行 INIT 命令。

应用示例

```
'!TITLE " 初始化 "
PROGRAM TSR1
  INIT, RESULT = I1          ' 进行电机 ON、 CAL、速度的设定。
                             'INIT 失败时，将错误代码储存到变量 I1 中，然后移到下一行。

END
```

SETOCCUPATIONTIME（语句）【RC7_Ver.2.0 以上版本】

功　能

变更特权任务的占有时间。

格　式

SETOCCUPATIONTIME < 占用时间 >

说　明

- 特权任务使用 / 尚未使用为［尚未使用］时什么也不执行。
- 特权任务使用 / 尚未使用为［使用］时，执行以下的处理。
  在 < 占用时间 > 里，可以设定 0, 2, 4, 6, 8 的任何一个。各自的设定值意味是，
  0: 0ms/8ms
  2: 2ms/8ms
  4: 4ms/8ms
  6: 6ms/8ms
  8: 8ms/8ms
  在特权任务动作过程中也可进行设定，并立即被反映。

注意事项

设定为 0m / 8ms 时，特权任务使用 PAC 程序的空闲时间进行动作。在没有空闲时间时特权任务不动作。

设定为 8ms / 8ms 时，在特权任务动作过程中，PAC 程序不动作。

为了使 PAC 程序和特权任务同时运行，在特权任务中需要加入 DELAY 或者信号 (semaphore) 等待等的处理。

应用示例

有以下 3 个程序时，通常想优先执行 PRO1 和 PRO2，但只有在机械手动作时，在需要优先进行特权任务的动作而多少牺牲 PRO2 的执行等时使用。

```
PROGRAM TSR1                    ' 特权任务在任何动作范围的检查程序
  -------------
  -------------
END
PROGRAM PRO1
  TAKEARM
  -------------
  SETOCCUPATIONTIME 4           ' 将特权任务的占用时间设为 4msec / 8msec。
  MOVE P,P0                     ' 机械手动作
  SETOCCUPATIONTIME 2           ' 将特权任务返回到默认的 2msec / 8msec
  -------------
END
PROGRAM PRO2
  DO WHILE 1
  -------------
  LOOP
END
```

INITWAITERR（语句）【RC7_Ver.2.2 以上版本】

功　能

初始化错误检测。（特权任务专用命令）

格　式

INITWAITERR

说　明

使用错误检测命令 (WAITERROR) 进行错误检测时，将错误检测初始化。

如果执行该命令，则将此前检测的错误进行清空。

在使用错误检测命令时如果不用该命令对错误检测进行初始化，则有不能正常进行错误检测的情况。

相关项目

WAITERROR、 CURERRSTATUS

应用示例

```
PROGRAM TSR1
-------------
INITWAITERR                    ’错误检测的初始化
-------------
  .
  .
  .
WHILE 1                        ’反复开始
  -------------
  I1 = WAITERROR               ’错误检测
  I2 = CURERRSTATUS            ’获取当前的错误发生状态
  -------------
WEND                           ’反复结束
-------------
END
```

WAITERROR（函数）【RC7_Ver.2.2 以上版本】

功　能

检测错误发生。（特权任务专用命令）

格　式

WAITERROR

说　明

返回检测到的错误的错误编码。

如果执行该命令，则控制器将一直等待发生错误为止，在发生错误时，返回至发生的错误的错码。

按照发生的顺序检测错误，错误检测开始时，如果不使用错误检测初始化命令 (INITWAITERR) 进行错误检测，则有可能出现不能正常检测错误的情况。

在连续发生错误时，能够检测的错误是按照从最后发生的错误到 32 个之前的顺序进行检测的。

相关项目

INITWAITERR、 CURERRSTATUS

应用示例

```
PROGRAM TSR1
-------------
INITWAITERR                    ’错误检测的初始化
-------------
  .
  .
  .
WHILE 1                        ’反复开始
  -------------
  I1 = WAITERROR               ’错误检测
  I2 = CURERRSTATUS            ’获取当前的错误发生状态
  -------------
WEND                           ’反复结束
-------------
END
```

CURERRSTATUS（函数）【RC7_Ver.2.2 以上版本】

功　能

返回错误发生状态。（特权任务专用命令）

格　式

CURERRSTATUS

说　明

返回当前的错误发生状态、" 错误发生中 (1) ／没有错误 (0) "。
该命令是在进行错误检测时判断当前是否发生错误时使用的。

相关项目

INITWAITERR、 WAITERROR

应用示例

```
PROGRAM TSR1
-------------
INITWAITERR                    ’错误检测的初始化
-------------
  .
  .
  .
WHILE 1                        ’反复开始
  -------------
  I1 = WAITERROR               ’错误检测
  I2 = CURERRSTATUS            ’获取当前的错误发生状态
  -------------
WEND                           ’反复结束
-------------
END
```

第 15 章
函数

在本章中，就 PAC 上所准备的各种函数进行说明。

第 15 章　函数

15.1　算术函数

ABS（函数）【依据 SLIM】

功　能

得出算式的值的绝对值。

格　式

ABS (< 算式 >)

说　明

得出在 < 算式 > 上所指定的值的绝对值。

< 算式 > 中包含倍精度实数时，得出的数值为倍精度，但在其他情况下则为单精度。

补充说明

所谓数值的绝对值，是将该数值的符号去除之后的值。例如，ABS (-1) 和 ABS (1) 都要返回 1。

相关项目

应用示例

```
DEFSNG lf1, lf2, lf3, lf4
lf1 = ABS(-2)                ' 将 -2 的绝对值赋值 f1。
lf2 = ABS(lf3 / lf4)         ' 将 (lf3 / lf4) 的绝对值赋值 lf2。
```

EXP（函数）【依据 SLIM】

功　能

得出将自然对数作为基数的指数函数。

格　式

EXP (< 算式 >)

说　明

< 算式 > 中包含倍精度实数时，得出的数值为倍精度，但在其他情况下则为单精度。

补充说明

- 自变量 (argument) number 的值如果超过 709.782712893，则会发生溢出错误。常量 e 的值是约 2.718282。
- Exp 函数是 Log 函数的反函数，有时也被称为反对数。

相关项目

LOG、 LOG10

应用示例

```
DEFSNG lf1, lf2, lf3, lf4
lf1 = EXP(2)                    ' 自然对数将底的平方赋值 lf1。
lf2 = EXP(lf3 / lf4)            ' 自然对数将底的 (lf3 / lf4) 次幂赋值 lf2。
```

INT（函数）【依据 SLIM】

功　　能

得出不超过被指定值的最大的整数值。

格　　式

INT (< 算式 >)

说　　明

返回不超过 < 算式 > 值的最大的整数。

如果溢出的情况下，会成为与 < 算式 > 同符号的整数的最大值。

相关项目

应用示例

```
DIM li1 As Integer
li1 = INT(123.456)          ' 舍去小数点以下的 . 赋值整数变量 li1。
                            '123.456->123
```

LOG（函数）【依据 SLIM】

功　能

得出自然对数。

格　式

LOG (< 算式 >)

说　明

得出在 < 算式 > 上所指定的值的自然对数。

在 < 数值 > 上指定大于 0 的值。

< 算式 > 中包含倍精度实数时，得出的数值为倍精度，但在其他情况下则为单精度。

补充说明

所谓自然对数，是以常量 e 为底的对数。常量 e 的值是约 2.718282。任意数值 x 的 n 为底的对数，可以按如下所示用 n 的自然对数除 x 的自然对数而得出。

Lognx=Logex / Logen

相关项目

EXP、 LOG10

应用示例

```
DEFSNG lf1, lf2, lf3, lf4
lf1 = LOG(2)                    ' 将 2 的自然对数赋值 lf1。
lf2 = LOG(lf3 / lf4)            ' 将 (lf3 / lf4) 的自然对数赋值 lf2。
```

LOG10（函数）【依据 SLIM】

功　能

得出常用对数。

格　式

LOG10 (< 算式 >)

说　明

得出在 < 算式 > 上所指定的值的常用对数。常用对数的底是 10。

在 < 数值 > 上指定大于 0 的值。

< 算式 > 中包含倍精度实数时，得出的数值为倍精度，但在其他情况下则为单精度。

相关项目

EXP、 LOG

应用示例

```
DEFSNG lf1, lf2, lf3, lf4
lf1 = LOG10(2)                  ' 将 2 的常用对数赋值 lf1。
lf2 = LOG10(lf3 / lf4)          ' 将 (lf3 / lf4) 的常用对数赋值 lf2。
```

POW （函数）【依据 SLIM】

功　　能

求取幂乘。

格　　式

POW (< 底 >,< 指数 >)

说　　明

得出用 < 底 > 的 < 指数 > 所指定的值的幂乘。

< 底 >、 < 指数 > 中包含倍精度实数时，得出的数值为倍精度，但在其他情况下则为 单精度。

相关项目

EXP

注意事项

在幂乘的计算中第 1 自变量 (argument) 为负值时，会发生域错误。

在 < 底 >、 < 指数 > 中包含倍精度实数时，精度只能被保证到 15 位。

在 < 底 >、 < 指数 > 中不包含倍精度实数时，精度只能被保证到 7 位。

应用示例

```
DEFSNG lf1, lf2
DEFINT li1, li2
li1 = POW(5, 2)                    ' 将 5 的平方赋值 li1。
lf1 = POW(lf2, li2)                ' 将 lf2 的 li2 次幂赋值 lf1。
```

MAX（函数）【依据 SLIM】

功　能

抽出最大值。

格　式

MAX (< 算式 >,< 算式 >[,< 算式 >…])

说　明

从任意个数的 < 算式 > 中抽出最大值。

算式的个数最大是 32 个。

补充说明

如果使用 Min 函数以及 Max 函数，则可以求出所指定的合计方法或者组的域的最小值和最大值。例如，在调查运费的最高额和最低额时使用该函数。没有指定合计方法时，图表全部成为目标。

相关项目

MIN

应用示例

```
DEFSNG lf1, lf2, lf3, lf4, lf5, lf6
DEFINT li1, li2, li3, li4, li5, li6
li1 = MAX(1, 2)                    ' 将 (1, 2) 内的最大值赋值 li1。
li2 = MAX(li3, li4, li5, li6)      ' 将 (li3, li4, li5, li6)
                                   ' 内的最大值赋值 li2。
lf1 = MAX(lf2, lf3, lf4, lf5, lf6) ' 将 (lf2, lf3, lf4, lf5, lf6)
                                   ' 内的最大值赋值 lf1。
```

MIN（函数）【依据 SLIM】

功　能

抽出最小值。

格　式

MIN (< 算式 >,< 算式 >[,< 算式 >…])

说　明

从任意个数的 < 算式 > 中抽出最小值。

算式的个数最大是 32 个。

补充说明

如果使用 Min 函数以及 Max 函数，则可以求出所指定的合计方法或者组的域的最小值和最大值。例如，在调查运费的最高额和最低额时使用该函数。没有指定合计方法时，图表全部成为目标。

相关项目

MAX

应用示例

```
DEFSNG lf1, lf2, lf3, lf4, lf5, lf6
DEFINT li1, li2, li3, li4, li5, li6
li1 = MIN(1, 2)                    ' 将 (1, 2) 内的最小值赋值 li1。
li2 = MIN(li3, li4, li5, li6)      ' 将 (li3, li4, li5, li6)
                                   ' 内的最小值赋值 li2。
lf1 = MIN(lf2, lf3, lf4, lf5, lf6) ' 将 (lf2, lf3, lf4, lf5, lf6)
                                   ' 内的最小值赋值 lf1。
```

RND（函数）【依据 SLIM】

功　能

发生大于 0 小于 1 的随机数。

格　式

RND (< 算式 >)

说　明

根据 < 算式 > 的值，按下表所示进行处理。

< 算式 > 取整数值。加入实数值时，进行舍去转换为整数。

算式的值	处理
算式 < 0	将 < 算式 > 设为随机数的种子值。如果种子值是相同的，则 RND 函数的返回值总是同一值。
算式 = 0	得出在此之前所得出的随机数。如果机械手控制器的电源接通之后，一次也没有得出随机数时，则返回 0。
算式 > 0	得出随机数系列的下一个随机数。

补充说明

Rnd 函数返回大于 0、小于 1 的范围的值。根据自变量 (argument) 的值，决定 Rnd 函数返回的随机数。只要初始种子值不变，一系列的 Rnd 函数返回随机数序列是相同的。这是因为，连续的各个 Rnd 函数将随机数系列中的之前的随机数作为种子值，分别生成下一个的随机数。

注意

随机数的种子值相同时，其后所得到的随机数系列为同一个系列。需要每次得出不同的系列时，作为种子值，请使用 TIMER 函数的结果值等。

相关项目

应用示例

```
DIM array(10) As Single
array(0) = RND(-TIMER)          ' 用 TIMER 函数赋予随机数的种子值。
FOR I1 = 1 TO 9
  array(I1) = RND(1)            ' 得出随机数。
NEXT I1
```

SGN（函数）

功　能

调查符号。

格　式

SGN (< 算式 >)

说　明

调查 < 算式 > 的符号，返回以下的数值。

正时	1
0 时	0
负时	-1

相关项目

应用示例

```
DEFSNG lf1, lf2
lf2 = SGN(lf1)                    ' 返回实数型变量 lf1 的符号。
```

SQR（函数）【依据 SLIM】

功　能

得出平方根。

格　式

SQR(< 算式 >)

说　明

得出 < 算式 > 的数值的平方根 (square)。

< 算式 > 中包含倍精度实数时，得出的数值为倍精度，但在其他情况下则为单精度。

< 算式 > 必须是 0 以上的值。

相关项目

应用示例

```
DEFSNG lf1, lf2, lf3, lf4, lf5, lf6, lf7
lf1 = SQR(2)                    ' 将 2 的平方根赋值 lf1。
lf2 = SQR(lf4)                  ' 将 lf4 的平方根赋值 lf2。
lf3 = SQR(lf5 + lf6) * lf7      ' 将 (lf5 + lf6) 平方根乘以 lf7 之后的数值赋值 lf3。
```

15.2 三角函数

ACOS（函数）【依据 SLIM】

功　能

得出反余弦 (arc cosine)。

格　式

ACOS(< 算式 >)

说　明

得出 < 算式 > 的数值的反余弦值。

所得出的值为度，范围从 0 到 180。

< 算式 > 中包含倍精度实数时，得出的数值为倍精度，但在其他情况下则为单精度。

相关项目

SIN、 TAN、 ASIN、 ATN、 ATN2

应用示例

```
DEFSNG lf1, lf2, lf3, lf4, lf5, lf6, lf7
lf1 = ACOS(0.5)                ' 将 0.5 的反余弦值赋值 lf1。
lf2 = ACOS(lf4 / 2)            ' 将 (lf4 / 2) 的反余弦值赋值 lf2。
lf3 = ACOS(lf5 / lf6) * lf7    ' 将 (lf5 / lf6) 的反余弦值乘以
                               'lf7 之后的数值赋值 lf3。
```

ASIN（函数）【依据 SLIM】

功　能

得出反正弦 (arc sine)。

格　式

ASIN(< 算式 >)

说　明

得出 < 算式 > 的值的反正弦值。

所得出的值为度，范围从 -90 到 90。

< 算式 > 中包含倍精度实数时，得出的数值为倍精度，但在其他情况下则为单精度。

相关项目

SIN、 COS、 TAN、 ACOS、 ATN、 ATN2

应用示例

```
DEFSNG lf1, lf2, lf3, lf4, lf5, lf6, lf7
lf1 = ASIN(0.5)                 ' 将 0.5 的反正弦值赋值 lf1。
lf2 = ASIN(lf4 / 2)             ' 将 (lf4 / 2) 的反正弦值赋值 lf2。
lf3 = ASIN(lf5 / lf6) * lf7     ' 将 (lf5 / lf6) 的反正弦值乘以 lf7 之后的数值
                                ' 赋值 lf3。
```

ATN（函数）【依据 SLIM】

功　能

得出反正切 (arc tangent)。

格　式

ATN(< 算式 >)

说　明

得出 < 算式 > 的值的反正切值。

所得出的值为度，范围从 -90 到 90。

< 算式 > 中包含倍精度实数时，得出的数值为倍精度，但在其他情况下则为单精度。

补充说明

- Atn 函数，是将直角三角形的 2 条边的比作为自变量 (argument) (number) 接受，返回所对应的角度。这里的 2 条边是指夹持直角的 2 个边。2 条边的比，是指将所求角的相反一侧的边（对边）的长度用另一侧的边（底边、即相邻所求角一侧的边）的长度相除的数值。结果值为 -π / 2 ～ π / 2 的范围内的值（单位为弧度）。为了把角度的单位从度转换成弧度，要在度上乘以 π / 180。为了从弧度转换为度，要在弧度上乘以 180 / π。
- Atn 函数是 Tan 函数的反三角函数。Tan 函数，将角度作为自变量 (argument) 接受，返回夹持包含其角度在内的直角三角形的直角的 2 条边的比。请留意 Atn 函数与正切的倒数余切（ 1 / 正切）的不同之处。

相关项目

SIN、 COS、 TAN、 ASIN、 ACOS、 ATN2

应用示例

```
DEFSNG lf1, lf2, lf3, lf4, lf5, lf6, lf7
lf1 = ATN(0.5)              ' 将 0.5 的反正切值赋值 lf1。
lf2 = ATN(lf4 / 2)          ' 将 (lf4 / 2) 的反正切值赋值 lf2。
lf3 = ATN(lf5 / lf6) * lf7  ' 在 (lf5 / lf6) 的反正切值上乘以 lf7 的值赋值 lf3。
```

ATN2（函数）【依据 SLIM】

功　能

得出以算式 1 除以算式 2 之后的反正切 (arc tangent)。

格　式

ATN2 (< 算式 1>,< 算式 2>)

说　明

得出用 < 算式 2> 除以 < 算式 1> 的数值的反正切值。

所得出的值为度，范围从 -180 到 180。

< 算式 > 中包含倍精度实数时，得出的数值为倍精度，但在其他情况下则为单精度。

ATN2 的数值范围如下。

	第一象限	第二象限	第三象限	第四象限
下限值	0	90	-180	-90
上限值	90	180	90	- 0

相关项目

SIN、 COS、 TAN、 ASIN、 ACOS、 ATN

应用示例

```
DEFSNG lf1, lf2, lf3, lf4, lf5, lf6, lf7, lf8
lf1 = ATN2(1, 2)              ' 将 (1, 2) 的反正切值赋值 lf1。
lf2 = ATN2(lf4, lf5)          ' 将 (lf4, lf5) 的反正切值赋值 lf2。
lf3 = ATN2(lf6, lf7) * lf8    ' 在 (lf6, lf7) 的反正切值上乘以 lf8 之后的数值赋值 lf3。
```

COS（函数）【依据 SLIM】

功　能

得出余弦 (cosine)。

格　式

COS(< 算式 >)

说　明

得出 < 算式 > 的值的余弦值。

< 算式 > 的单位用度指定。

< 算式 > 中包含倍精度实数时，得出的数值为倍精度，但在其他情况下则为单精度。

以弧度 (RAD) 输入自变量 (argument) 时，在常量之后加上 RAD。

补充说明

Cos 函数，是将角度作为自变量 (argument) 接受，返回包含其角度在内的直角三角形的 2 条边的比。这里所说的所谓 2 条边，是指夹持自变量 (argument) number 所指定角的 2 个边。2 条边的比，是指短一方的边（底边）的长度用另一方的边 （斜边）的长度相除的值。结果值为 -1 ～ 1 范围内的值。为了把角度的单位从度转换成弧度，要在度上乘以 π / 180。为了从弧度转换为度，要在弧度上乘以 180 / π。

相关项目

SIN、 TAN、 ASIN、 ACOS、 ATN、 ATN2

应用示例

```
DEFSNG lf1, lf2, lf3, lf4, lf5
lf1 = COS(0.78525RAD)        ' 将 0.78525 （弧度）的余弦值赋值 lf1。
lf2 = COS(lf4)               ' 将 lf4 余弦值赋值 lf2。
lf3 = COS(45) * lf5          ' 将在 45 度的余弦值上乘以 lf5 之后的数值赋值 lf3。
```

SIN （函数）【依据 SLIM】

功　能

得出正弦 (sine)。

格　式

SIN (< 算式 >)

说　明

得出 < 算式 > 的值的正弦值。

< 算式 > 的单位用度指定。

< 算式 > 中包含倍精度实数时，得出的数值为倍精度，但在其他情况下则为单精度。

以弧度 (RAD) 输入自变量 (argument) 时，在常量之后加上 RAD。

补充说明

Sin 函数，是将角度作为自变量 (argument) 接受，返回包含其角度在内的直角三角形的 2 条边的比。在此所说的 2 条边是指所指定的角的相反一侧的边 （对边）和斜边。2 条边的比是，对边的长度用斜边的长度相除的值。结果值为 -1 ～ 1 范围内的值。为了把角度的单位从度转换成弧度，要在度上乘以 π / 180。为了从弧度转换为度，要在弧度上乘以 180 / π。

相关项目

COS、 TAN、 ASIN、 ACOS、 ATN、 ATN2

应用示例

```
DEFSNG lf1, lf2, lf3, lf4, lf5
lf1 = SIN(0.78525RAD)          ' 将 0.78525 （弧度）的正弦值赋值 lf1。
lf2 = SIN(lf4)                 ' 将 lf4 的正弦值赋值 lf2。
lf3 = SIN(45) * lf5            ' 将在 45 度的正弦值上乘以 lf5 之后的数值赋值 lf3。
```

TAN（函数）【依据 SLIM】

功　能

得出正切 (tangent)。

格　式

TAN (< 算式 >)

说　明

得出 < 算式 > 的值的正切值。

< 算式 > 的单位用度指定。

< 算式 > 中包含倍精度实数时，得出的数值为倍精度，但在其他情况下则为单精度。

以弧度 (RAD) 输入自变量 (argument) 时，在常量之后加上 RAD。

补充说明

Tan 函数，是将角度作为自变量 (argument) 接受，返回包含其角度在内的直角三角形的 2 条边的比。这里的 2 条边是指夹持直角的 2 个边。2 条边的比，是指指定的角的相反一侧的边（对边）的长度用另一侧的边（底边、即与指定的角相邻一侧的边）的长度相除的值。为了把角度的单位从度转换成弧度，要在度上乘以 π / 180。为了从弧度转换为度，要在弧度上乘以 180 / π。

相关项目

SIN、 COS、 ASIN、 ACOS、 ATN、 ATN2

应用示例

```
DEFSNG lf1, lf2, lf3, lf4, lf5
lf1 = TAN(0.78525RAD)          ' 将 0.78525 （弧度）的正切值赋值 lf1 。
lf2 = TAN(lf4)                 ' 将 lf4 的正切值赋值 lf2。
lf3 = TAN(45) * lf5            ' 将在 45 度的正切值上乘以 lf5 之后的数值赋值 lf3。
```

15.3 角度转换

DEGRAD（函数）【依据 SLIM】

功　能

将单位转换为弧度。

格　式

DEGRAD (< 算式 >)

说　明

将 < 算式 > 所指定的值从度转换为弧度。

相关项目

RADDEG

应用示例

```
DEFSNG lf1, lf2, lf3
lf1 = DEGRAD(90)                    ' 将 90 转换为弧度之后的值赋值 lf1。
lf2 = SIN(DEGRAD(lf3 + 20))         ' 将 (lf3 + 20) 转换为弧度之后的值的正弦值赋值 lf2。
```

RAD（函数）

功　能

将通过弧度赋予的数值转换为角度。

格　式

< 数值 > RAD

说　明

将通过弧度能赋予的 < 数值 > 转换为角度进行运算。

在 < 数值 > 处指定常量。

相关项目

RADDEG

注意事项

在 < 数值 > 处不能使用变量或保留字 （P1 等）。

当使用变量或保留字时，请使用 RADDEG。

应用示例

```
DIM lf1 As SINGLE
lf1 = 2RAD                          ’将 2 弧度转换为角度赋值 lf1。
```

RADDEG（函数）【依据 SLIM】

功　　能

将单位转换为度。

格　　式

RADDEG (< 算式 >)

说　　明

将 < 算式 > 所指定的值从弧度转换为度。

相关项目

RAD、 DEGRAD

应用示例

```
DEFSNG lf1, lf2
lf1 = RADDEG(1.5705)          ' 将 1.5705 转换为度之后的值赋值 lf1。
lf2 = RADDEG( PI / 2)         ' 将 (π / 2) 转换为度之后的值的正弦值赋值 lf2。
```

15.4 速度转换

MPS（函数）

功　能

将指定的速度值 （mm/sec）转换为对 CP 动作时的最大内部速度的比率 （%）。

格　式

MPS (< 算式 >)

说　明

通过与指定速度的指令 "SPEED" 配合使用，可利用速度值 （mm/sec）指定速度。
指定的速度值的单位是 mm/sec。

相关项目

SPEED

注意事项

由于这是转换内部速度的指令，所以实际动作会受指定外部速度的影响。
例：指定为 SPEED MPS （50），当外部速度为 50% 时，实际速度变为 25mm/sec。

应用示例

```
SPEED MPS(50)                    ' 将内部移动速度定为 50mm/sec。
```

RPM （函数）【Ver.2.8 以上版本】

功　　能

将指定轴编号的旋转数 （rpm）转换为对 PTP 动作时的最大内部速度的比率 （%）。

格　　式

RPM （< 轴编号 >、< 算式 >）

说　　明

将指定轴编号的旋转数 （rpm）转换为对 PTP 动作时的最大内部速度的比率 （%）。
单位为 rpm。

相关项目

SPEED

注意事项

这是一个转换内部速度的指令，因此，其实际动作会受到指定外部速度的影响。
例：指定 SPEED RPM （7,50），当外部速度为 50% 时， 7 轴的旋转速度为 25rpm。
如果是附加轴，一旦指定了旋转轴，即指所指定的旋转数为输出端的旋转数。
如果指定的是直动轴，则变为输入端的旋转数 （电机的旋转数）。

应用示例

```
SPEED RPM(7, 50)                    ' 将 7 轴的内部速度变为 50rpm。
```

15.5 时间函数

SEC（函数）

功　能

将以秒为单位赋予的数值变换为毫秒。

格　式

< 数值 > SEC

说　明

将以秒能赋予的 < 数值 > 变换为毫秒进行运算。用于毫秒设定的命令等。

相关项目

应用示例

```
DELAY(10SEC)                    ' 等待 10 秒钟经过。
```

15.6　矢量

AVEC（函数）

功　能

抽出近似矢量。

格　式

AVEC(< 齐次变换型 >)

说　明

从齐次变换型坐标抽出近似矢量。

相关项目

OVEC、 PVEC

应用示例

```
DEFTRN lt1, lt2, lt3
DEFVEC lv1, lv2
lv1 = AVEC(lt1)                 ' 将 lt1 的近似矢量赋值 lv1。
lv2 = AVEC(lt2 * lt3)           ' 将 (lt2 * lt3) 的值的近似矢量赋值 lv2。
```

OVEC（函数）

功　能

抽出定向矢量。

格　式

OVEC (< 齐次变换型 >)

说　明

从齐次变换型坐标抽出定向矢量。

相关项目

AVEC、 PVEC

应用示例

```
DEFTRN lt1, lt2, lt3
DEFVEC lv1, lv2
lv1 = OVEC(lt1)              ' 将 lt1 的定向矢量赋值 lv1。
lv2 = OVEC(lt2 * lt3)        ' 将 (lt2 * lt3) 的值的定向矢量赋值 lv2。
```

PVEC（函数）

功　能

抽出位置矢量。

格　式

PVEC ({< 齐次变换型 >|< 位置型 >})

说　明

从齐次变换型坐标或者位置型坐标中，抽出位置矢量。

相关项目

AVEC、 OVEC、 RVEC

应用示例

```
DIM lt1 As Trans
DIM lv1 As Vector
DIM lp1 As Position
lv1 = PVEC(lt1)                    ' 将 lt1 的位置矢量赋值 lv1。
```

6 轴

```
lv1 = PVEC(lp1 + (100, 200, 0, 0, 0, 0) )
                                   ' 将 lp1 + (100, 200, 0, 0, 0, 0) 的位置矢量赋值 lv1。
```

4 轴

```
lv1 = PVEC(lp1 + (100, 200, 0, 0) )
                                   ' 将 lp1 + (100, 200, 0, 0) 的位置矢量赋值 lv1。
```

MAGNITUDE（函数）

功　能

得出矢量的大小。

格　式

MAGNITUDE (< 矢量型 >)

说　明

得出矢量型坐标的值的矢量的大小。

相关项目

DIST

应用示例

```
DEFSNG lf1, lf2
DIM lv1 As Vector
lf1 = MAGNITUDE((10, 10, 10))        ' 将 (10, 10, 10) 的矢量的大小赋值 lf1。
lf2 = MAGNITUDE(lv1)                 ' 将 lv1 的矢量的大小赋值 lf2。
```

15.7 姿势数据型转换

J2P（函数）

功　能

从关节型转换为位置型。

格　式

J2P （< 关节型 >[,<J2P 选件 >]）

说　明

将在 < 关节型 > 所指定的关节型数据转换为位置型的数据。所得出的数值，反映了当时所设定的工具、工件坐标系。

<J2P 选件 > 为可省略的 Result 存放变量 （Ver.3.0 以上版本）。指定 I 型变量。一旦指定 <J2P 选件 >，在执行 J2P 指令时，内部即使发生错误，也不会停止程序，并将内部错误的错误编码存放至指定变量。

相关项目

J2T、 P2J、 P2T、 T2J、 T2P

应用示例

```
DEFPOS lp1, lp2
DIM lj1 As Joint
lp1 = J2P(lj1)                    ' 将 lj1 转换为位置型的数据赋值 lp1。

lp1 = J2P(lj1, li1)               'J2P 指令失败时，将错误编码存放至变量 li1，
                                  ' 转移到下一行的步骤。

6 轴
lp2 = J2P((0, 0, 90, 0, 0, 0))    ' 将 (0, 0, 90, 0, 0, 0) 转换为位置型的数据赋值 lp2。

4 轴
lp2 = J2P((0, 0, 300, 0))         ' 将 (0, 0, 300, 0) 转换为位置型的数据赋值 lp2。
```

J2T （函数）

功　　能

从关节型转换为齐次变换型。

格　　式

J2T （< 关节型 >[,<J2T 选件 >]）

说　　明

将在 < 关节型 > 所指定的关节型数据转换为齐次变换型的数据。所得出的数值，反映了当时所设定的工具、工件坐标系。

<J2T 选件 > 为可省略的 Result 存放变量 （Ver.3.0 以上版本）。指定 I 型变量。一旦指定 <J2T 选件 >，在执行 J2T 指令时，内部即使发生错误，也不会停止程序，并将内部错误的错误编码存放至指定变量。

相关项目

J2P、 P2J、 P2T、 T2J、 T2P

应用示例

```
DEFTRN lt1, lt2
DIM lj1 As Joint
lt1 = J2T(lj1)                  ' 将 lj1 转换为齐次变换型的数据赋值 lt1。

lt1 = J2T(lj1, li1)             'J2T 指令失败时，将错误编码存放至变量 li1，
                                ' 转移到下一行的步骤。

6 轴
lt2 = J2T((0, 0, 90, 0, 0, 0))  ' 将 (0, 0, 90, 0, 0, 0) 转换为齐次变换型的数据赋值 lt2。

4 轴
lt2 = J2T((0, 0, 300, 0))       ' 将 (0, 0, 300, 0) 转换为齐次变换型的数据赋值 lt2。
```

P2J（函数）

功　能

从位置型转换为关节型。

格　式

P2J （< 位置型 >[,<P2J 选件 >]）

说　明

将在 < 位置型 > 上所指定的数据转换为关节型。

在位置型数据的 FIG 的值为 -1 （未定）时，用当前的形态进行转换。所得出的数值，反映了当时所设定的工具、工件坐标系。

<P2J 选件 > 为可省略的 Result 存放变量 （Ver.3.0 以上版本）。指定 I 型变量。一旦指定 <P2J 选件 >，在执行 P2J 指令时，内部即使发生错误，也不会停止程序，并将内部错误的错误编码存放至指定变量。

相关项目

J2P、 J2T、 P2T、 T2J、 T2P、 CURFIG

应用示例

```
DEFJNT lj1, lj2
DIM lp1 As position
lj1 = P2J(lp1)                    ' 将 lp1 以当前的形态转换为关节型的数据赋值
                                  'lj1。
lj1 = P2J(lp1, li1)               'P2J 指令失败时，将错误编码存放至变量 li1，
                                  ' 转移到下一行的步骤。

6 轴
lj2 = P2J((0, 0, 90, 0, 0, 180))  ' 将 (0, 0, 90, 0, 0, 180) 以当前的形态转换为关节型的
                                  ' 数据赋值 lj2。

4 轴
lj2 = P2J((100, 100, 300, 45))    ' 将 (100, 100, 300, 45) 以当前的形态转换为关节型的
                                  ' 数据赋值 lj2。
```

P2T（函数）

功　能

从位置型转换为齐次变换型。

格　式

P2T （< 位置型 >[,<P2T 选件 >]）

说　明

将在 < 位置型 > 上所指定的数据转换为齐次变换型。所得出的数值，反映了当时所设定的工具、工件坐标系。<P2T 选件 > 为可省略的 Result 存放变量 （Ver.3.0 以上版本）。指定 I 型变量。一旦指定 <P2T 选件 >，在执行 P2T 指令时，内部即使发生错误，也不会停止程序，并将内部错误的错误编码存放至指定变量。

相关项目

J2P、 J2T、 P2J、 T2J、 T2P

应用示例

```
DEFTRN lt1, lt2
DIM lp1 As position
lt1 = P2T(lp1)                    ' 将 lp1 转换为齐次变换型的数据赋值 lt1。

lt1 = P2T(lp1,li1)                'P2T 指令失败时，将错误编码存放至变量 li1，
                                  ' 转移到下一行的步骤。

6 轴
lt2 = P2T((350, 0, 450, 0, 0, 180))   ' 将 (350, 0, 450, 0, 0, 180) 转换为齐次变换型的数据
                                      ' 赋值 lt2。

4 轴
lt2 = P2T((100, 100, 300, 45))    ' 将 (100, 100, 300, 45) 转换为齐次变换型的数据赋值
                                  'lt2。
```

T2J（函数）

功　能

从关节型转换为齐次变换型。

格　式

T2J （< 齐次变换型 >[,<P2T 选件 >]）

说　明

将在 < 齐次变换型 > 上所指定的数据转换为关节型。

齐次变换型数据的 FIG 的值为 -1 （未定）时，以当前的形态进行转换。

所得出的数值，反映了当时所设定的工具、工件坐标系。

<T2J 选件 > 为可省略的 Result 存放变量 （Ver.3.0 以上版本）。指定 I 型变量。一旦指定 <T2J 选件 >，在执行 T2J 指令时，内部即使发生错误，也不会停止程序，并将内部错误的错误编码存放至指定变量。

相关项目

J2P、 J2T、 P2J、 P2T、 T2P、 CURFIG

应用示例

```
DEFJNT lj1, lj2, lj3
DEFTRN lt1, lt2
DIM li1 As Integer
lj1 = T2J(lt1)                    ' 将 lt1 以当前的形态转换为关节型的数据赋值
                                  'lj1。
lj2 = T2J(lt2)                    ' 将 lt2 以 0 的形态转换为关节型的数据赋值 lj2。

lj3 = T2J((350, 0, 450, 0, 1, 0, 0, 0, -1))
                                  ' 将齐次变换型数据 (350, 0, 450, 0, 1, 0, 0, 0, -1)
                                  ' 转换为关节型的数据赋值 lj3。
lj1 = T2J(lt1,li1)                'T2J 指令失败时，将错误编码存放至变量 li1，
                                  ' 转移到下一行的步骤。
```

T2P（函数）

功　能

从齐次变换型转换为位置型。

格　式

T2P （< 齐次变换型 >[,<T2P 选件 >]）

说　明

将在 < 齐次变换型 > 上所指定的数据转换为位置型。所得出的数值，反映了当时所设定的工具、工件坐标系。

<T2P 选件 > 为可省略的 Result 存放变量 （Ver.3.0 以上版本）。指定 I 型变量。一旦指定 <T2P 选件 >，在执行 T2P 指令时，内部即使发生错误，也不会停止程序，并将内部错误的错误编码存放至指定变量。

相关项目

J2P、 J2T、 P2J、 P2T、 T2J

应用示例

```
DEFPOS lp1, lp2
DIM lt1 As Trans
lp1 = T2P(lt1)                          ' 将 lt1 转换为位置型的数据赋值 lp1。

lp2 = T2P(350, 0, 450, 0, 1, 0, 0, 0, -1))
                                        ' 将齐次变换型数据 (350, 0, 450, 0, 1, 0, 0, 0, -1) 转换
                                        ' 为位置型的数据赋值 lp2。
lp1 = T2P(lt1,li1)                      'T2P 指令失败时，将错误编码存放至变量 li1，
                                        ' 转移到下一行的步骤。
```

TINV（函数）

功　能

计算齐次变换型的逆矩阵。

格　式

TINV （< 齐次变换型 >[,<TINV 选件 >]）

说　明

将在 < 齐次变换型 > 上所指定的数据逆转换。

<TINV 选件 > 为可省略的 Result 存放变量（Ver.3.0 以上版本）。指定 I 型变量。一旦指定 <TINV 选件 >，在执行 TINV 指令时，内部即使发生错误，也不会停止程序，并将内部错误的错误编码存放至指定变量。

相关项目

应用示例

```
DEFTRN lt1, lt2, Lt3
lt1 = TINV((350, 0, 450, 0, 1, 0, 0, 0, -1))
                                    ' 将 (350, 0, 450, 0, 1, 0, 0, 0, -1) 的逆矩阵赋值 lt1。

lt2 = TINV(lt3)                     ' 将 lt3 的逆矩阵赋值 lt2。
lt2 = TINV(lt1,li1)                 'TINV 指令失败时，将错误编码存放至变量 li1，
                                    ' 转移到下一行的步骤。
```

NORMTRN（函数）【Ver.1.8 以上版本】

功　能

进行齐次变换型的正规化计算。

格　式

NORMTRN(< 齐次变换型 >[,<NORMTRN 选件 >])

说　明

将在 < 齐次变换型 > 上所指定的数据正规化。

正规化 … 以近似矢量为基准将定向矢量正交化，将近似、定向矢量单位矢量化。

<NORMTRN 选件 > 为可省略的 Result 存放变量 （Ver.3.0 以上版本）。指定 I 型变量。一旦指定 <NORMTRN 选件 >，在执行 NORMTRN 指令时，内部即使发生错误，也不会停止程序，并将内部错误的错误编码存放至指定变量。

相关项目

应用示例

```
DEFTRN lt1, lt2
lt1 = NORMTRN ((350, 0, 450, 0, 1, 0, 0, 0, -1))

lt1 = NORMTRN (lt2)
lt2 = NORMTRN(lt1,li1)          'NORMTRN 指令失败时，将错误编码存放至变量 li1，
                                ' 转移到下一行的步骤。
```

15.8　距离抽出

DIST（函数）【依据 SLIM】

功　能

返回 2 点间的距离。

格　式

DIST (< 位置型 1>,< 位置型 2>)

说　明

得出 < 位置型 1> 和 < 位置型 2> 的距离。

相关项目

应用示例

```
DEFSNG lf1, lf2
DEFPOS lp1, lp2, lp3

6 轴
lf1 = DIST((350, 0, 450, 0, 0, 180), lp1)
                                    ' 将 (350, 0, 450, 0, 0, 180) 和 lp1 之间的距离赋值 lf1。

4 轴
lf1 = DIST((350, 0, 300, 45), lp1)
                                    ' 将 (350, 0, 300, 45) 和 lp1 之间的距离赋值 lf1。

lf2 = DIST(lp2, lp3)                ' 将 lp2 和 lp3 之间的距离赋值 lf2。
```

15.9 形态分量

FIG（函数）

功　能

抽出形态。

格　式

FIG ({< 位置型 >|< 齐次变换型 >})

说　明

从位置型、齐次变换型抽出形态。

相关项目

CURFIG、 LETF、机械手形态 （附录 2）

应用示例

```
DIM li1 As Integer
DIM lp1 As position
li1 = FIG(lp1)                          ' 将 lp1 的形态赋值 li1。

6 轴
li1 = FIG(lp1 + (0, 0, 100, 0, 0, 0) )  ' 将 lp1 + (0, 0, 100, 0, 0, 0) 的形态赋值 li1。

4 轴
li1 = FIG(lp1 + (0, 0, 100, 0) )        ' 将 lp1 + (0, 0, 100, 0) 的形态赋值 li1。
```

15.10 角度分量

JOINT（函数）

功　能

从关节型坐标抽出角度。

格　式

JOINT (< 轴编号 >,< 关节型 >)

说　明

将从 < 关节型 > 上所指定的关节型坐标抽出 < 轴编号 > 所指定的一个轴的关节角度。

相关项目

LETJ

应用示例

```
DEFJNT lj1, lj2
DEFSNG lf1, lf2
DIM li1 As Integer
lf1 = JOINT(1, lj1)                    ' 将 lj1 的 1 轴的关节角度赋值 lf1。
lf2 = JOINT(li1, lj2)                  ' 将 lj2 的 li1 的值的轴的关节角度赋值 lf2。
```

6 轴

```
lf1 = JOINT(l, lj1 + (10, 10, 0, 0, 0, 0) )
                                       ' 将 lj1 + (10, 10, 0, 0, 0, 0) 的 1 轴的关节角度赋值 lf1。
```

4 轴

```
lf1 = JOINT(l, lj1 + (10, 10, 0, 0) )
                                       ' 将 lj1 + (10, 10, 0, 0) 的 1 轴的关节角度赋值 lf1。
```

15.11 轴分量

POSX（函数）【依据 SLIM】

功　能

抽出 X 分量。

格　式

POSX ({< 位置型 >|< 矢量型 >})

说　明

从位置型或者矢量型的坐标，抽出 X 分量。

相关项目

POSY、 POSZ

应用示例

```
DIM lf1 As Single
DIM lp1 As Position
lf1 = POSX(lp1)                          ' 将 lp1 的 X 分量赋值 lf1。

6 轴
lf1 = POSX(lp1 + (100, 100, 100, 0, 0, 0) )
                                         ' 将 lp1 + (100, 100, 100, 0, 0, 0) 的 X 分量赋值 lf1。

4 轴
lf2 = POSX(lp1 + (100, 100, 100, 0) )
                                         ' 将 lp1 + (100, 100, 100, 0) 的 X 分量赋值 lf1。
```

POSY（函数）【依据 SLIM】

功　能

抽出 Y 分量。

格　式

POSY ({< 位置型 >|< 矢量型 >})

说　明

从位置型或者矢量型的坐标，抽出 Y 分量。

相关项目

POSX、 POSZ

应用示例

```
DIM lf1 As Single
DIM lp1 As Position
lf1 = POSY(lp1)                          ' 将 lp1 的 Y 分量赋值 lf1。
```

6 轴

```
lf1 = POSY(lp1 + (100, 100, 100, 0, 0, 0) )
                                         ' 将 lp1 + (100, 100, 100, 0, 0, 0) 的 Y 分量赋值 lf1。
```

4 轴

```
lf1 = POSY(lp1 + (100, 100, 100, 0) )
                                         ' 将 lp1 + (100, 100, 100, 0) 的 Y 分量赋值 lf1。
```

POSZ（函数）【依据 SLIM】

功　能

抽出 Z 分量。

格　式

POSZ ({< 位置型 >|< 矢量型 >})

说　明

从位置型或者矢量型的坐标，抽出 Z 分量。

相关项目

POSX、 POSY

应用示例

```
DIM lf1 As Single
DIM lp1 As Position
lf1 = POSZ(lp1)                          ' 将 lp1 的 Z 分量赋值 lf1。

6 轴
lf1 = POSZ(lp1 + (100, 100, 100, 0, 0, 0) )
                                         ' 将 lp1 + (100, 100, 100, 0, 0, 0) 的 Z 分量赋值 lf1。

4 轴
lf1 = POSZ(lp1 + (100, 100, 100, 0) )
                                         ' 将 lp1 + (100, 100, 100, 0) 的 Z 分量赋值 lf1。
```

15.12 旋转分量

POSRX（函数）

功　能

抽出 X 轴旋转分量。

格　式

POSRX(< 位置型 >)

说　明

从 < 位置型 > 所指定的位置型的坐标，抽出 X 轴旋转分量。

相关项目

POSRY、 POSRZ

应用示例

```
DIM lf1 As Single
DIM lp1 As Position
lf1 = POSRX(lp1)                    ' 将 lp1 的 X 轴旋转分量赋值 lf1。
```

POSRY（函数）

功　　能

抽出 Y 轴旋转分量。

格　　式

POSRY(< 位置型 >)

说　　明

从 < 位置型 > 所指定的位置型的坐标，抽出 Y 轴旋转分量。

相关项目

POSRX、 POSRZ

应用示例

```
DIM lf1 As Single
DIM lp1 As Position
lf1 = POSRY(lp1)                    ' 将 lp1 的 Y 轴旋转分量赋值 lf1。
```

POSRZ（函数）

功　　能

抽出 Z 轴旋转分量。

格　　式

POSRZ(< 位置型 >)

说　　明

从 < 位置型 > 所指定的位置型的坐标，抽出 Z 轴旋转分量。

相关项目

POSRX、 POSRY

应用示例

```
DIM lf1 As Single
DIM lp1 As Position
lf1 = POSRZ(lp1)                    ' 将 lp1 的 Z 轴旋转分量赋值 lf1。
```

POST（函数）

功　能

抽出 T 轴旋转分量。

格　式

POST(< 位置型 >)

说　明

从 < 位置型 > 所指定的位置型的坐标，抽出 T 轴旋转分量。

相关项目

应用示例

```
DIM lf1 As Single
DIM lp1 As Position
lf1 = POST(lp1)                    ' 将 lp1 的 T 轴旋转分量赋值 lf1。
```

15.13 姿势分量

RVEC（函数）

功　能

抽出姿势。

格　式

RVEC(< 位置型 >)

说　明

从位置型的坐标抽出姿势。

相关项目

PVEC

应用示例

```
DIM lp1 As position
DIM lv1 As Vector
lv1 = RVEC(lp1)                    ' 将 lp1 的姿势赋值 lv1。
```

15.14 位置函数

AREAPOS（函数）

功　能

将进行干扰检查的区域的中心位置和长方体的方向以位置型进行返回。

格　式

AREAPOS (< 区域编号 >)

说　明

指定被 AREA 指令定义的 < 区域编号 >，将当前的设定值 （中心位置）以位置型进行返回。

< 区域编号 > 区域编号 (0 ～ 31)

相关项目

AREA、 SETAREA、 RESETAREA、 AREASIZE

应用示例

```
DIM lp1 As Position
lp1 = AREAPOS(1)                ' 将进行干扰检查的区域的中心位置和立方体的方向赋值
                                'lp1。
```

AREASIZE（函数）

功　能

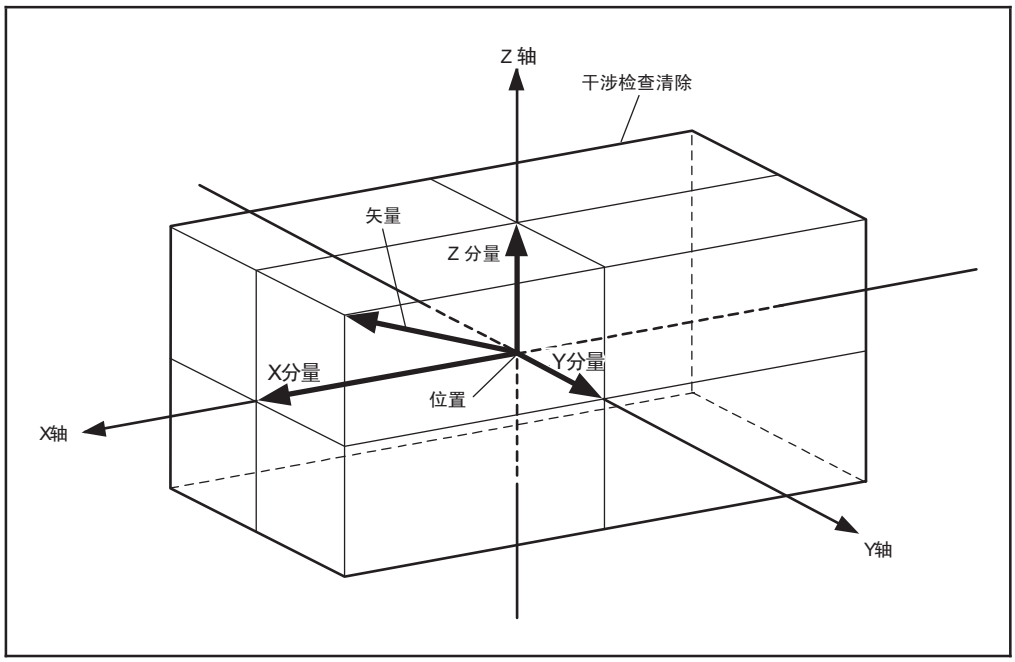
将定义进行干扰检查的区域的长方体的大小（各个边的长度）以矢量型进行返回。

格　式

AREASIZE(< 区域编号 >)

说　明

通过指定以 AREA 指令定义的 < 区域编号 >，将定义该区域的长方体的大小（各个边的长度）按矢量型返回。
长方体的各个边的长度是矢量的 X、 Y、 Z 各分量的长度的倍数。



相关项目

AREA、 SETAREA、 RESETAREA、 AREAPOS

应用示例

```
DIM lv1 As Vector
lv1 = AREASIZE(1)                    ' 将定义进行干扰检查区域的长方体的大小赋值 lv1。
```

TOOLPOS（函数）

功　能

将工具坐标系以位置型进行返回。

格　式

TOOLPOS (< 工具坐标系编号 >)

说　明

通过指定被 TOOL 指令定义的 < 工具坐标系编号 >，将工具坐标系的设定值以位置型 进行返回。在此种情况下，-1 （不定）进入位置型数据的 FIG 分量中。

< 工具坐标系编号 > 工具坐标系编号 (1 ～ 63)

相关项目

TOOL、 CHANGETOOL

应用示例

```
DIM lp1 As Position
lp1 = TOOLPOS(1)                    ’将工具坐标系赋值 lp1。
```

WORKPOS（函数）

功　能

将用户坐标系以位置型进行返回。

格　式

WORKPOS (< 用户坐标系编号 >)

说　明

通过指定被 WORK 指令定义的 < 用户坐标系编号 >，将该工件坐标系的设定值以位置型进行返回。在此种情况下，-1 （不定）进入位置型数据的 FIG 分量中。

< 用户坐标系编号 > 用户坐标系编号 (1 ～ 7)

相关项目

WORK、 CHANGEWORK

应用示例

```
DIM lp1 As Position
lp1 = WORKPOS(1)                    ’将用户坐标系赋值 lp1。
```

WORKATTRIBUTE（函数）【Ver.3.2 以上版本】

功　能

获取指定工件坐标系的属性值。

格　式

WORKATTRIBUTE （< 工件坐标系编号 >）

说　明

获取利用<工件坐标系编号>指定的工件坐标的属性值（0：标准，1：固定工具（外部TCP））。（Ver.3.2以上版本）

关于外部 TCP 功能，请参照操作指南的 " 外部 TCP 功能 （以工件坐标原点为中心进行旋转动作） "。

相关项目

WORK、 CHANGEWORK

应用示例

```
I55=WORKATTRIBUTE(5)                ’获取工件坐标 5 的属性值至 I55。
```

15.15 字符串函数

ASC（函数）

功　能

变换为字符编码。

格　式

ASC(< 字符串 >)

说　明

将 < 字符串 > 的最初字符变换为字母编码。

注意

该函数只返回字符串的最初的 1 个字节 (Byte) 数的值。因此，最初的字符即使是汉字，Shift-JIS 编码的头 1 个字节 (Byte) 数也被返回。

相关项目

CHR$, 字符编码表 （附录 1）

应用示例

```
DIM li1 As Integer
li1 = ASC( "ABCDEFGH" )                ' 将 A 的字母编码 65 (&H41) 赋值 li1。
```

BIN$（函数）【依据 SLIM】

功　能

将算式的值转换为 2 进制数的字符串。

格　式

BIN$(< 算式 >)

说　明

将在 < 算式 > 上所指定的值转换为 2 进制数 , 并将该值转换为字符串。

相关项目

CHR$、 HEX$、 STR$、 VAL

应用示例

```
DIM li1 As Integer
DEFSTR ls1, ls2
ls1 = BIN$(20)                  ' 将 20 变换为 2 进制数赋值 ls1。
ls2 = BIN$(li1)                 ' 将 li1 变换为 2 进制数赋值 ls2。
```

CHR$（函数）【依据 SLIM】

功　能

将 ASCII 编码转换为字符。

格　式

CHR$(< 算式 >)

说　明

得出具有在 < 算式 > 上所指定的值的字母编码的字符。

相关项目

BIN$、 HEX$、 STR$、 VAL、字符编码表 （附录 1）

应用示例

```
DEFSTR ls1, ls2
ls1 = CHR$(49)                  ' 将具有 49 的字母编码的字符赋值 ls1。
ls2 = CHR$(&H4E)                ' 将具有 &H4E 的字母编码的字符赋值 ls2。
```

SPRINTF$（函数）

功　能

将表达式变换为所指定的格式化，并作为字符串返回。

格　式

SPRINTF$ (< 格式化 >,< 表达式 >)

说　明

在 < 格式化 > 中，包含原样被输出的字符串和为了变换 < 表达式 > 的内容（作为字符串的一部分）而输出的变换指定字符串。

变换指定字符串开头是 %，以下连接字符 （串）。

修饰变换指定字符串含义的 0 个以上的标志 (Flag) （顺序不同）。

- 写出字符串的域的最小宽度（选择）。被变换的值的字符数小于域的最小宽度时，为了满足域的最小宽度，左侧 （指定以下的左对齐标志 (Flag) 时为右侧）为空白 （默认）可以填入。域的宽度值是星号 (*)（以后叙述）而且是整数值。
- 精度 （选择）。根据精度被指定的是通过 d、 i、 o、 x 以及 X 变换指定记号变换所表示的最小位数、通过 e、E 以及 f 变换指定记号变换所表示的小数点以下的位数、通过 g 以及 G 变换指定记号变换所表示的最大有效位数、以及通过 s 变换指定记号变换从字符串所能写入的最大字符数。精度的值是句号 (.) 和与其相接的星号 (*)（以后 叙述）而且是整数值 （选择）。如果只指定句号，则精度被设定为 0。将精度和其他的变换指定记号一起指定时的动作是尚未定义的。
- 变换指定记号
  如前所述，在域宽度和精度上可以指定星号 (*)。在此种情况下，通过整数型的自变量 (argument) 宽度或者精度的值被指定。指定域宽度或者精度、或者这两个值的自变量 (argument) ，必须在被变换的自变量 (argument) （只有存在时）的前面 （将域宽度置于精度前面）进行指定。如果在域的宽度上指定负的值，则在－变换指定记号的后面指定正的域宽度是相同的。如果在精度上指定负的值，则与省略了精度是相同的。
  标志 (Flag) 的意思如下。

  -　：在域内将变换结果左对齐 （不指定该标志 (Flag) 时为右对齐）。

  +　：在带有符号的变换中，总是在数字的开头加上 + 或者 - 的符号 （不指定该标志 (Flag) 时，只在负值时加上 - 的符号）。

  空格字符　：在带有符号的变换中开头字符不是符号时，或者根据带有符号变换不产生任何字符时，要在结果的开头加上空格字符。如果同时指定空格字符和 + 标志 (Flag)，则空格字符被忽视。

  #　：根据 " 置换形式 " 进行变换。在 O 变换指定记号进行的变换中，精度被 变更，使结果的开头位为 0。在 x （或者 X）变换指定记号进行的变换中，在 0 以外的结果的开头被加上 "0x"（或者 0X）。在 e、E、f、g 或者 G 变换指定记号进行的变换中，必须在结果处（可以没有小数位的数字）被加上小数点字符。（通常，在这些变换中，只有在有小数位的数字时 才被加上小数点字符。）在 g 以及 G 变换指定记号进行的变换中，末尾的 0 不被删除。其他的变换上的动作是尚未定义的。

  0　：在 d、i、o、x、X、e、E、f、g 以及 G 转换指定记号进行的变换中，要在开头（符号或者底数的后面）通过填入 0 而确保最小域的宽度。不能通过空格字符进行填入。同时指定 0 标志 (Flag) 和－标志 (Flag) 时，0 标志 (Flag) 被忽视。根据 d、i、o、x 以及 X 的转换指定记号，指示精度的转换时， 0 标志 (Flag) 被忽视。对于与其他转换指定记号一起使用时的动作不进行规定。

变换指定记号的意思如下。

d、 i	：	将整数型自变量 (argument) 的值以 " [-]dddd" 的形式变换为带有符号的 10 进制数的字符串。根据被指定的精度，决定所表示的最小位数。变换后的值不满最小位数时，开头可以用 0 填入。默认的精度是 1。如果用精度 0 转换 0 的值，则不输出字符。
o、 x、 X	：	将整数型的自变量 (argument) 以 "dddd" 的形式变换为没有符号的 8 进制数 (o)、或者 16 进制数 （ x 或者 X）的字符串。在 x 变换指定记号中使用小写字符 (abcdef)、在 X 变换指定记号中使用大写字符 (ABCDEF)。根据被指定的精度，决定所表示的最小位数。变换后的值不满最小位数时，开头可以用 0 填入。默认的精度是 1。如果用精度 0 转换 0 的值，则不输出字符。
f	：	将实数型的自变量 (argument) 以 " [-]ddd.ddd" 的形式转换为 10 进制数表示的字符串。根据被指定的精度，决定小数点以下的位数。如果精度没被指定，则小数点以下的位数为 6 位。精度是 0，# 标志 (Flag) 没有被指定时，小数点以下的字符不被表示。在小数点以下的字符被表示时，最低也要被表示的是其前面的 1 位。值被取整为适当的位数。
e、 E	：	将实数型的自变量 (argument) 变换为 "[-]d.ddde + / - dd" 的形式的字符串。在小数点字符的前面被显示 1 位 （自变量 (argument) 如果不是 0 则是 0 以外）。小数点以下的位数，由精度 (precision) 所决定。如果精度不被指定，则小数点以下为 6 位。精度是 0，# 标志 (Flag) 没有被指定时，小数点以下的字符不被输出。值被取整为适当的位数。如果使用 E 转换指定记号，则指数不是用 e 而是用 E 显示。指数一般在 2 位以上。值如果是 0，则指数部分也为 0。
g、 G	：	将实数型的自变量 (argument) 变换为 f 或者 e（G 时为 E）的形式。根据精度决定有效位数。精度为 0 时，有效位数为 1。被使用的形式由被转换的值决定。 e（或者 E）的形式变换只是在变换结果的指数部分比 -4 小或者与精度相等或者比精度大时使用。小数点以下的末尾的 0 被删除。小数点字符只是在有小数点以下的数字时被显示。
c	：	将整数型的自变量 (argument) 变换为 1 字节 (Byte) 的 ASCII 字符，输出转换后的字符。
S	：	系统预约
%	：	输出 % 的字符。自变量 (argument) 不被变换。变换的完全指定是 %%。

关于变换指定不正确时的动作，没有进行规定。

在变换结果大于域宽度时，即使转换结果被压缩，域也不会按照变换结果进行扩展。

相关项目

HEX$、 STR$

应用示例

```
S1 = SPRINTF$("% d",123)          ' "123" 被赋值 S1。
S2 = SPRINTF$("%-6.3f", 1.23)     ' "1.230" 被赋值 S2。
S3 = SPRINTF$("%e",123.456#)      '"1.234560e + 002" 被赋值 S3。
S4 = SPRINTF$("%g",12345678#)     '"1.23457e + 007" 被赋值 S4。
S5 = SPRINTF$("%#x",128)          ' "0x80" 被赋值 S5。
```

HEX$（函数）【依据 SLIM】

功　能

以字符串形式得出从 10 进制数变换为 16 进制数后的值。

格　式

HEX$(< 算式 >)

说　明

将在 < 算式 > 上所指定的值从 10 进制数变换为 16 进制数，并将其值变换为字符串。

补充说明

自变量 (argument) 不是整数时，在赋值变换之前要取整为不超过其值的整数。

相关项目

BIN$、 CHR$、 STR$、 VAL

应用示例

```
DIM li1 As Integer
DEFSTR ls1, ls2
ls1 = HEX$(20)                  ' 将 20 变换为 16 进制数赋值 ls1。
ls2 = HEX$(li1)                 ' 将 li1 变换为 16 进制数赋值 ls2。
```

LEFT$（函数）【依据 SLIM】

功　能

抽出左部分字符串。

格　式

LEFT$ (< 字符串 >,< 位数 >)

说　明

从 < 字符串 > 的左侧抽出 < 位数 > 部分的字符串。

字符串中含有 NULL 值时，返回 NULL 值。

在 < 位数 > 上指定字符串的长度以上的值时，返回所有字符串。

注意

< 位数 > 作为字节 (Byte) 数处理。因此，汉字编码请按 2 字节 (Byte) 计数。

相关项目

LEN、 MID$、 RIGHT$

应用示例

```
DIM li1 As Integer
DEFSTR ls1, ls2
ls1 = LEFT$( "abcdefg", 3 )          ' 从字符串 "abcdefg" 的左侧将 3 字符的字符串 "abc" ' 赋值 ls1。

ls2 = LEFT$( ls1, li1 )              ' 从 ls1 的左侧将 li1 的字符数部分的值的字符串赋值 ls2。
```

LEN（函数）【依据 SLIM】

功　能

以字节 (Byte) 数得出字符串的长度。

格　式

LEN(< 字符串 >)

说　明

得出用 < 字符串 > 所指定的字符串的长度的总计字节 (Byte) 数。

相关项目

MID$、 LEFT$、 RIGHT$

应用示例

```
DEFINT li1, li2
DIM ls1 As String
li1 = LEN( "abcdefg" )          ' 将字符串 "abcdefg" 的字符数 7 赋值 li1。
li2 = LEN( ls1 )                ' 将 ls1 的字节 (Byte) 数赋值 li2。
```

MID$（函数）【依据 SLIM】

功　能

从字符串中取出指定的字符数部分的字符串。

格　式

MID$ (< 字符串 >,< 起始位置 >[,< 位数 >])

说　明

从 < 字符串 > 的 < 起始位置 > 的字符抽出 < 位数 > 部分的字符串。

字符串中含有 NULL 值时，返回 NULL 值。

在 < 位数 > 上指定字符串的长度以上的值时，返回所有字符串。

省略 < 位数 > 时，或者只有在少于字符串内指定的位数的字符串时，从起始位置后的 所有的字符被返回。

在 < 起始位置 > 上指定字符串的长度以上的值时，返回所有字符串。

注意

< 位数 > 作为字节 (Byte) 数处理。因此，汉字编码请按 2 字节 (Byte) 计数。

相关项目

LEN、 LEFT$、 RIGHT$

应用示例

```
DEFSTR ls1, ls2
ls1 = MID$( "abcdefg", 2, 3 )          ' 将字符串 "bcd" 赋值 ls1。
ls2 = MID$( ls1, 2, 2 )                ' 将 ls1 的第 2 号到 2 位赋值 ls2。
```

ORD （函数）【依据 SLIM】

功　能

变换为字符编码。

格　式

ORD(< 字符串 >)

说　明

将 < 字符串 > 的最初字符变换为字母编码。

注意

该函数只返回字符串的最初的 1 个字节 (Byte) 数的值。因此，最初的字符即使是汉字，Shift-JIS 编码的头 1 个字节 (Byte) 数也被返回。此外，字符串为 " " 时 , 返回 0。

相关项目

CHR$, 字符编码表 （附录 1）

应用示例

```
DIM li1 As Integer
li1 = ORD( "ABCDEFGH" )              ' 将 A 的字母编码 65 (&H41) 赋值 li1。
```

RIGHT$（函数）【依据 SLIM】

功　能

抽出字符串的右部分。

格　式

RIGHT$(< 字符串 >,< 位数 >)

说　明

从 < 字符串 > 的右侧抽出 < 位数 > 部分的字符串。

字符串中含有 NULL 值时，返回 NULL 值。

在 < 位数 > 上指定字符串的长度以上的值时，返回所有字符串。

注意

< 位数 > 作为字节 (Byte) 数处理。因此，汉字编码请按 2 字节 (Byte) 计数。

相关项目

LEN、 MID$、 LEFT$

应用示例

```
DIM li1 As Integer
DEFSTR ls1, ls2
ls1 = RIGHT$( "abcdefg", 3 )          ' 从字符串 "abcdefg" 的右侧将 3 字符的字符串 "efg" ' 赋值 ls1。

ls2 = RIGHT$( ls1, li1 )              ' 从 ls1 的右侧将 li1 的值的字符数部分的字符串赋值 ls2。
```

STRPOS（函数）【依据 SLIM】

功　　能

得出字符串的位置。

格　　式

STRPOS (< 字符串 1>,< 字符串 2>)

说　　明

得出 < 字符串 1> 内的 < 字符串 2> 的位置。

顺序从左数 1、 2…。

找不到时，返回 0。

相关项目

应用示例

```
DIM li1 As Integer
li1 = STRPOS("abcdefg", "bc")                    ' 将 "abcdefg" 内的 "bc" 的位置 (2) 赋值 li1。
```

STR$（函数）【依据 SLIM】

功　能

将数值变换为字符串。

格　式

STR$(< 算式 >)

说　明

将 < 算式 > 上所指定的数值变换为字符串。

补充说明

如果将数值变换为字符串，在结果值的开头为了显示符号所使用的空格经常被确保。数值为正时，在 Str 函数的结果值的开头被插入空格。该空格的意思为正记号。 Str 函数只将句号 (.) 作为有效的小数点记号进行识别。

相关项目

BIN$、 CHR$、 HEX$、 VAL

应用示例

```
DIM li1 As Integer
DEFSTR ls1, ls2
ls1 = STR$(20)                  ' 将 20 变换为字符串赋值 ls1。
ls2 = STR$(li1)                 ' 将 li1 变换为字符串赋值 ls2。
```

VAL（函数）【依据 SLIM】

功　能

从字符串变换为数值。

格　式

VAL (< 字符串 >)

说　明

将 < 字符串 > 上所指定的字符串变换为数值。

< 字符串 > 的最初的字符如果不是 +、 -、 &、或者数值，则 VAL 的值为 0。

补充说明

如果在字符串中发现数字以外的字符，则 Val 函数中止读入。日元记号 (￥) 和逗号 (,) 等，通常被视为数值的一部分的记号和字符，Val 函数也不作为数值进行解释。但是，Val 函数识别表示基数的前缀 &H （16 进制数）、 &B （2 进制数）。自变量 (argument) 的字符串中所包含的空格、双引号、换行被忽视。

相关项目

BIN$、 CHR$、 HEX$、 STR$

应用示例

```
DEFINT li1, li2, li3
li1 = VAL("&B100")          ' 将 "&B100" 变换为数值 （10 进制数为 4）赋值 li1。
li2 = VAL("&H20")           ' 将 "&H20" 变换为数值 （10 进制数为 32）赋值 li2。
li3 = VAL("-30")            ' 将 "-30" 变换为数值 （10 进制数为 -30）赋值 li3。
```

第 16 章
常量

在本章中，就 PAC 所准备的常量进行说明。

第 16 章　常量

16.1　内置常量

OFF（内置常量）

功　能

赋予 OFF (0) 值。

格　式

OFF

说　明

对于式赋予 OFF (0) 的值。

相关项目

ON

应用示例

```
IF I0 = TRUE THEN                ' 赋予布尔值的真 (1) 的值。
  I1 = ON                        ' 将 ON (1) 的值赋值整数型变量。
ELSEIF I0 = FALSE THEN           ' 赋予布尔值的假 (0) 的值。
  I1 = OFF                       ' 将 OFF (0) 的值赋值整数型变量。
ELSE
  D1 = PI                        ' 将 π 赋值实数型变量。
ENDIF
```

ON（内置常量）

功　能

赋予 ON (1) 值。

格　式

ON

说　明

对于式赋予 ON (1) 的值。

相关项目

OFF

应用示例

```
IF I0 = TRUE THEN                ' 赋予布尔值的真 (1) 的值。
  I1 = ON                        ' 将 ON (1) 的值赋值整数型变量。
ELSEIF I0 = FALSE THEN           ' 赋予布尔值的假 (0) 的值。
  I1 = OFF                       ' 将 OFF (0) 的值赋值整数型变量。
ELSE
  D1 = PI                        ' 将 π 赋值实数型变量。
ENDIF
```

PI（内置常量）

功　能

赋予 π 值。

格　式

PI

说　明

用双精度型返回 π 的值。

相关项目

应用示例

```
IF I0 = TRUE THEN                ' 赋予布尔值的真 (1) 的值。
  I1 = ON                        ' 将 ON (1) 的值赋值整数型变量。
ELSEIF I0 = FALSE THEN           ' 赋予布尔值的假 (0) 的值。
  I1 = OFF                       ' 将 OFF (0) 的值赋值整数型变量。
ELSE
  D1 = PI                        ' 将 π 赋值实数型变量。
ENDIF
```

FALSE（内置常量）

功　能

赋予布尔值的假 (0) 的值。

格　式

FALSE

说　明

对于式赋予布尔值的假 (0) 的值。

相关项目

TRUE

应用示例

```
IF I0 = TRUE THEN                ' 赋予布尔值的真 (1) 的值。
  I1 = ON                        ' 将 ON (1) 的值赋值整数型变量。
ELSEIF I0 = FALSE THEN           ' 赋予布尔值的假 (0) 的值。
  I1 = OFF                       ' 将 OFF (0) 的值赋值整数型变量。
ELSE
  D1 = PI                        ' 将 π 赋值实数型变量。
ENDIF
```

TRUE（内置常量）

功　能

赋予布尔值的真 (1) 的值。

格　式

TRUE

说　明

对于式赋予布尔值的真 (1) 的值。

相关项目

FALSE

应用示例

```
IF I0 = TRUE THEN                ' 赋予布尔值的真 (1) 的值。
  I1 = ON                        ' 将 ON (1) 的值赋值整数型变量。
ELSEIF I0 = FALSE THEN           ' 赋予布尔值的假 (0) 的值。
  I1 = OFF                       ' 将 OFF (0) 的值赋值整数型变量。
ELSE
  D1 = PI                        ' 将 π 赋值实数型变量。
ENDIF
```

第 17 章
时间 / 日期控制

在本章中，就确认日期、时刻和经过时间的指令及按时刻配置控制的指令进行说明。

第 17 章　时间 / 日期控制

17.1　时间 / 日期

DATE$（系统变量）【依据 SLIM】

功　能

得出当前的日期。

格　式

DATE$

说　明

当前的日期以 "yyyy / mm / dd"（年 / 月 / 日）的形式被存放。

相关项目

TIME$

应用示例

```
DIM ls1 As String
ls1 = DATE$                          ' 将当前的日期赋值 ls1。
```

TIME$（系统变量）【依据 SLIM】

功　能

得出当前的时间。

格　式

TIME$

说　明

当前的时间以 "hh : mm : ss"（时 : 分 : 秒）的形式存放。

时刻表示位 24 小时方式。

相关项目

DATE$

应用示例

```
DIM ls1 As String
ls1 = TIME$                          ' 将当前的时间赋值 ls1。
```

TIMER（系统变量）【依据 SLIM】

功　能

得出经过时间。

格　式

TIMER

说　明

以投入控制器的电源时刻为基准 (0)，获取以毫秒计时所经过的时间。

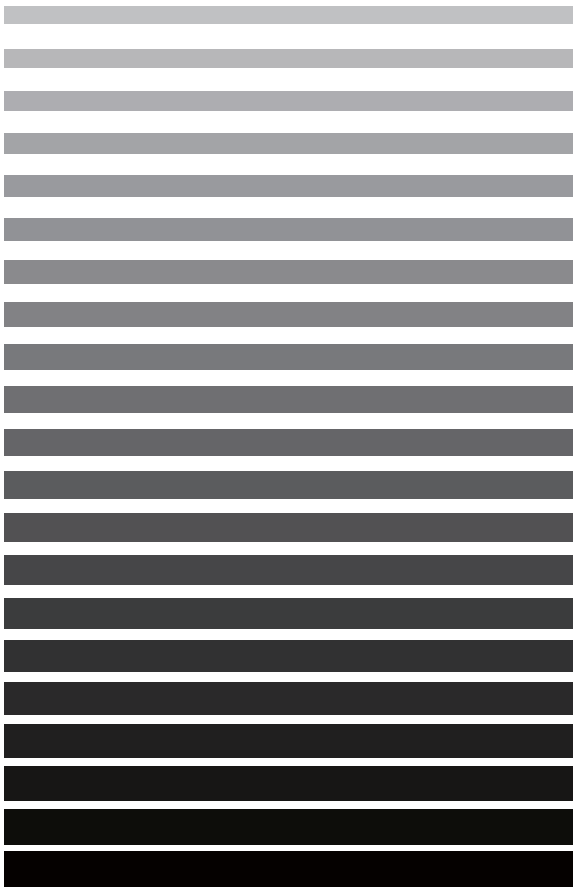
注意

每当时间经过 1431655750 毫秒时，都重新归零。

应用示例

```
DEFINT li1, li2, li3
li1 = TIMER                 ' 将从基准时间开始所经过的时间赋值 li1。
li2 = TIMER + li3           ' 将 li3 满足从基准时间开始的经过时间的值赋值 li2。
```

第 18 章
错误控制



在发生错误时，可以调查错误的原因、编写恢复程序。在本章中，就应对错误的指令进行介绍。

第 18 章　错误控制

18.1　错误信息

ERRMSG$（函数）

功　能

赋予错误讯息。

格　式

ERRMSG$(< 算式 >)

说　明

在用 < 算式 > 所指定的错误编号上赋予相对应的错误讯息。

应用示例

```
DIM ls1 As String
ls1 = ERRMSG$(&H600C)                    ’将错误编号 &H600C 错误讯息赋值 ls1。
```

SETERR（语句）【Ver.1.98 以上版本】

功　能

将用户定义错误设定到 Ⅰ型变量区域。

格　式

SETERR <. 错误编码 >

说　明

用错误存放功能定义用户定义错误的 Ⅰ型变量区域，在该区域进行设定，并将指针加上。

相关项目

CLRERR、 GETERRLVL、 GETERR

注意事项

需要设定错误存放功能。此外，用错误存放功能定义的 Ⅰ型变量区域，会因命令、TP 操作而变更，请予以注意。

应用示例

```
PROGRAM  PRO1
------------
SETERR  100                              ’在环形缓冲器上写入错误编码 "100"。
------------
END
```

GETERR（函数）【Ver.1.98 以上版本】

功　　能

从通过错误存放功能定义的缓冲中获取错误编码。

格　　式

GETERR(< 算式 >)

说　　明

从通过错误存放功能定义的缓冲中获取错误编码。在获取最新的错误时，在自变量 (argument) 上设定 "0"。

相关项目

CLRERR、 GETERRLVL、 SETERR

注意事项

需要设定错误存放功能。此外，用错误存放功能定义的 I 型变量区域，会因命令、TP 操作而变更，请予以注意。
此外，在没有设定错误存放功能时，该指令从通常的错误 Log 上清除。

应用示例

```
PROGRAM  PRO1
------------
I1 = GETERR(0)                    ’将最新的错误从环形缓冲区中清除。
------------
END
```

CLRERR（语句）【Ver.1.98 以上版本】

功　能

清空错误。

格　式

CLRERR

说　明

清空当前发生的错误。

相关项目

ERR、 SETERR、 GETERR

注意事项

只从特权任务可以执行。

应用示例

```
PROGRAM  TSR1
------------
I1 = SYSSTATE                          ' 获取系统状态
IF  (I1  AND  &H0082)  THEN            ' 如果发生异常或者发生警告
CLRERR                                 ' 错误清空
END IF
------------
END
```

GETERRLVL（函数）【Ver.1.98 以上版本】

功　　能

赋予错误编码的等级。

格　　式

GETERRLVL(< 算式 >)

说　　明

在用 < 算式 > 指定的错误编码上赋予对应的错误级别。

相关项目

CLRERR、 SETERR、 GETERR

注意事项

在没有相应的错误时，求出结果为 "-1"。

应用示例

```
PROGRAM  PRO1
------------
I1 = GETERRLVL(&H6001)              ’得到错误编码 "6001" 的错误级别。
------------
END
```

第 19 章
系统信息

在本章中，就获取机械手的系统信息的指令进行说明。

第 19 章　系统信息

19.1　系统

GETENV（函数）

功　能

获取系统的环境设定值。

格　式

GETENV （< 表编号 >，< 要素编号 >）

说　明

获取用 <表编号 > 和 <要素编号> 所指定的系统的环境设定值。环境设定值的项目，请参照附录3 "环境设定值 "。

相关项目

VER$, 环境设定值 （附录 3）

应用示例

```
DIM li1 As Integer
li1 = GETENV(9, 1)                    ' 将 9, 1 号的环境设定值的内容赋值 li1。
```

LETENV（语句）

功　能

设定系统的环境设定值。

格　式

LETENV < 表编号 >、 < 要素编号 >、 < 设定值 >

说　明

将用 < 表编号 > 和 < 要素编号 > 所指定的系统的环境设定值在 < 设定值 > 上进行设定。

环境设定值的项目，请参照附录 3 " 环境设定值 "。

如果指定不可能设定的要素编号和设定值，则在执行时会发生错误。

补充说明

鉴定设定是否可行的更简单的方法：在 WINCAPS Ⅲ的 [参数编辑] 画面中，属性一览没有 * 记号的项目就是可以设定的要素。但是，请将密码的级别置于 <programmer>，并进行登录。

注意

一般变更环境设定值时，需要重新启动控制器。

相关项目

VER$、 GETENV

应用示例

```
LETENV 6, 22, 5600                    ' 在 6, 22 号的环境设定值上设定 5600。
```

VER$ （函数）

功　能

获取各个模块的版本。

格　式

VER$ （< 算式 >）

说　明

与 <算式> 的值对应的模块的版本用字符串存放。有关与 <算式> 的值对应的模块，请参照附录 7的版本对应表。

相关项目

GETENV, 版本对应表 （附录 7）

应用示例

```
DEFSTR ls1, ls2, ls3
DIM li1 As Integer
ls1 = VER$(1)                    ’ 将主板的版本赋值 ls1。
ls2 = VER$(2)                    ’ 将 DIO 板的版本赋值 ls2。
ls3 = VER$(li1)                  ’ 将与 li1 的值对应的版本信息赋值 ls3。
```

该示例中，在 ls1、 ls2 内作为字符串以如下所示形式存放。（日期是版本完成的日期。）

ls1	1.000	07/30/1998
ls2	V1.00	8/03/1998

GETLANGUAGE （函数）【RC7_Ver.2.2 以上版本】

功　能

获取当前的语言设定。

格　式

GETLANGUAGE

说　明

获取当前的语言设定。

0: 日语 1: 英语 2: 德文 3: 韩文 4: 中文来表示。

应用示例

```
defint li
li=GETLANGUAGE
```

19.2 Log

STARTLOG（语句）

功　能

开始伺服控制 LOG 的记录。

格　式

STARTLOG

说　明

为了开始伺服控制 LOG 的记录，使用 STARTLOG 指令。

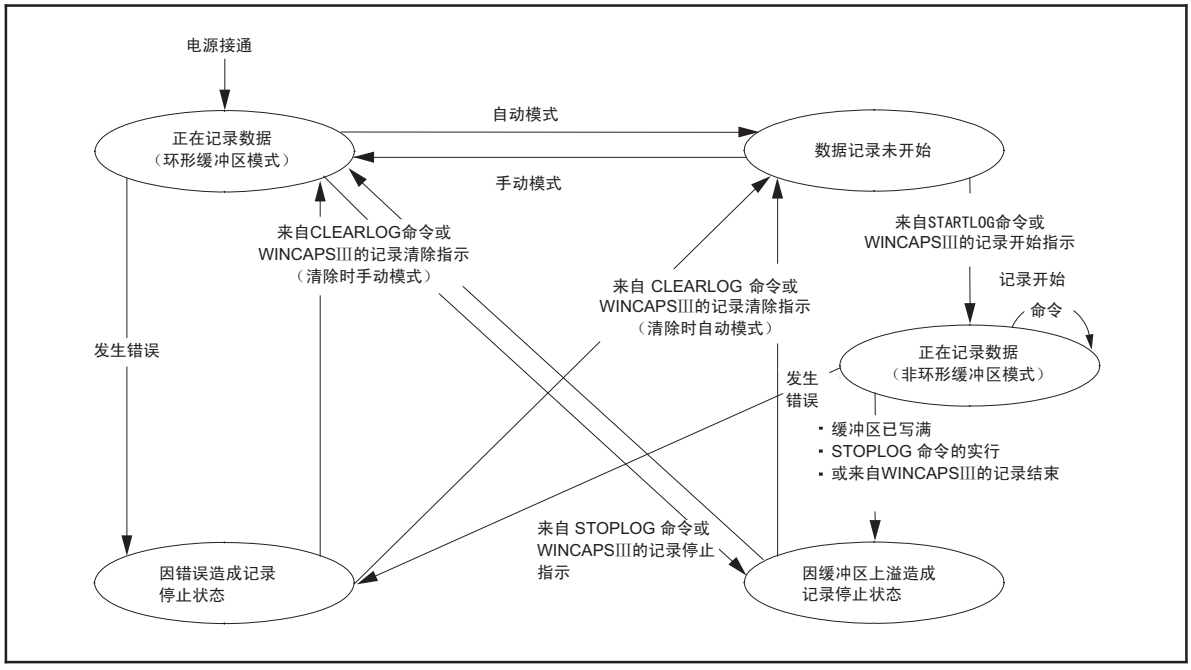
为了确实记录伺服控制 LOG，在执行 STARTLOG 指令之前，请预先执行 CLEARLOG 指令，或用 WINCAPS Ⅲ进行伺服 LOG 清空操作。已经开始伺服控制 LOG 的记录时，即使执行 STARTLOG 指令，也无法记录。

在手动模式和教导检查模式下，自动开始记录伺服控制 LOG。

在自动模式下，程序中有记录开始指示，或利用 WINCAPS Ⅲ开始记录的操作，开始记录伺服控制 LOG。

在发生错误和缓冲充满 （ 10 秒钟）时会停止伺服控制 LOG 的记录。在手动模式下，可以保持到发出记录停止的指示为止的最新 10 秒钟的 LOG。

LOG 动作的状态迁移图



相关项目

CLEARLOG、 STOPLOG

应用示例

STARTLOG ’开始伺服控制 LOG 的记录。

CLEARLOG（语句）

功　　能

对伺服控制 LOG 的记录进行初始化。

格　　式

CLEARLOG

说　　明

在程序中，为了指示伺服控制 LOG 的初始化，使用 CLEARLOG 指令。

缓冲区已写满，或由于发生错误而停止，在重新开始记录之前，需要清空 LOG 的内容（初始化）。请参照 P19-4 "LOG 动作的状态迁移图 "。

为了确实记录伺服控制 LOG，在实行 STARTLOG 指令之前，请实行 CLEARLOG 指令。在已经开始记录的状态，即使实行 STARTLOG 指令，也不能记录。

相关项目

STARTLOG、 STOPLOG

应用示例

```
CLEARLOG                    ' 对伺服控制 LOG 的记录进行初始化。
```

STOPLOG（语句）

功　　能

停止伺服控制 LOG 的记录。

格　　式

STOPLOG

说　　明

在程序中，为了指示伺服控制 LOG 的停止，使用 STOPLOG 指令。

在自动模式下，当缓冲区已满时，自动停止 LOG 记录，通常不需在程序内实行 STOPLOG 指令。

相关项目

STARTLOG、 CLEARLOG

应用示例

```
STOPLOG                              ' 开始伺服控制 LOG 的记录。
```

STARTTRACELOG（语句）【Ver.2.7 以上版本】

功　　能

开始记录跟踪 LOG。

格　　式

STARTTRACELOG

说　　明

指定开始记录程序内的跟踪 LOG。

跟踪 LOG 利用 WINCAPS Ⅲ接收 LOG 数据，并阅览。

同时记录跟踪 LOG（多功能）与单跟踪 LOG。（Ver.2.7 以上版本）

相关项目

STOPTRACELOG、 CLEARTRACELOG

应用示例

```
CLEARTRACELOG                ' 清空跟踪 LOG 信息。
STARTTRACELOG                ' 开始记录跟踪 LOG。
....
动作
....
STOPTRACELOG                 ' 结束对跟踪 LOG 的记录。
```

STOPTRACELOG（语句）【Ver.2.7 以上版本】

功　能

结束对跟踪 LOG 的记录。

格　式

STOPTRACELOG

说　明

指定结束程序内跟踪 LOG 的记录。

跟踪 LOG 利用 WINCAPS Ⅲ接收 LOG 数据，并阅览。

同时停止记录跟踪 LOG （多功能）与单跟踪 LOG。（Ver.2.7 以上版本）

相关项目

STARTTRACELOG、 CLEARTRACELOG

应用示例

```
CLEARTRACELOG                ' 清空跟踪 LOG 数据。
STARTTRACELOG                ' 开始记录跟踪 LOG。
....
动作
....
STOPTRACELOG                 ' 结束对跟踪 LOG 的记录。
```

CLEARTRACELOG（语句）【Ver.2.7 以上版本】

功　能

清空跟踪 LOG 数据。

格　式

CLEARTRACELOG

说　明

清空程序内的跟踪 LOG 数据。

跟踪 LOG 利用 WINCAPS Ⅲ接收 LOG 数据，并阅览。

同时清除跟踪 LOG （多功能）与单跟踪 LOG 的数据。（Ver.2.7 以上版本）

相关项目

STARTTRACELOG、 STOPTRACELOG

应用示例

```
CLEARTRACELOG                ' 清空跟踪 LOG 数据。
STARTTRACELOG                ' 开始记录跟踪 LOG。
....
动作
....
STOPTRACELOG                 ' 结束对跟踪 LOG 的记录。
```

STARTVARLOG（语句）【Ver.2.7 以上版本】

功　能

开始记录变量 LOG。

格　式

STARTVARLOG

说　明

指定开始在程序内记录变量 LOG。

变量 LOG 利用 WINCAPS Ⅲ接收 LOG 数据，并阅览。

记录 LOG 的变量只能是在监视器上登录，并在 WINCAPS Ⅲ的 "LOG" 栏中打勾的变量。（Ver.2.7 以上版本）

相关项目

STOPVARLOG、 CLEARVARLOG

应用示例

```
CLEARVARLOG                    ’清空变量 LOG 数据。
STARTVARLOG                    ’开始记录变量 LOG。
....
动作
....
STOPVARLOG                     ’结束对变量 LOG 的记录。
```

STOPVARLOG（语句）【Ver.2.7 以上版本】

功　能

结束对变量 LOG 的记录。

格　式

STOPVARLOG

说　明

指定结束在程序内对变量 LOG 的记录。

变量 LOG 利用 WINCAPS Ⅲ接收 LOG 数据，并阅览。

记录 LOG 的变量只能是在监视器上登录，并在 WINCAPS Ⅲ的 "LOG" 栏中打勾的变量。（Ver.2.7 以上版本）

相关项目

STARTVARLOG、 CLEARVARLOG

应用示例

```
CLEARVARLOG                    ' 清空变量 LOG 数据。
STARTVARLOG                    ' 开始记录变量 LOG。
....
动作
....
STOPVARLOG                     ' 结束对变量 LOG 的记录。
```

CLEARVARLOG（语句）【Ver.2.7 以上版本】

功　能

清空变量 LOG 数据。

格　式

CLEARVARLOG

说　明

清除程序内的变量 LOG 数据。

变量 LOG 利用 WINCAPS Ⅲ接收 LOG 数据，并阅览。

记录 LOG 的变量只能是在监视器上登录，并在 WINCAPS Ⅲ的 "LOG" 栏中打勾的变量。（Ver.2.7 以上版本）

相关项目

STARTVARLOG、 STOPVARLOG

应用示例

```
CLEARVARLOG                    ’清空变量 LOG 数据。
STARTVARLOG                    ’开始记录变量 LOG。
....
动作
....
STOPVARLOG                     ’结束对变量 LOG 的记录。
```

19.3　动作模式

CHGEXTMODE（语句）【Ver.1.98 以上版本】

功　　能

进行外部自动模式切换。

格　　式

CHGEXTMODE

说　　明

从内部自动模式进行向外部自动模式的切换。

相关项目

INIT、 CHGINTMODE

注意事项

不能进行从手动、教导检查模式的切换。仅限来自特权任务的实行时有效，而且可在用户任务停止状态下实行。

应用示例

```
PROGRAM  TSR1
------------
CHGEXTMODE                    ’外部自动模式切换
------------
CHGINTMODE                    ’内部自动模式切换
------------
END
```

CHGINTMODE（语句）【Ver.1.98 以上版本】

功　　能

进行内部自动模式切换。

格　　式

CHGINTMODE

说　　明

从外部自动模式进行向内部自动模式的切换。

相关项目

INIT、 CHGEXTMODE

注意事项

不能进行从手动、教导检查模式的切换。仅限来自特权任务的实行时有效，而且可在用户任务停止状态下实行。

应用示例

```
PROGRAM  TSR1
------------
CHGEXTMODE                    ’外部自动模式切换
------------
CHGINTMODE                    ’内部自动模式切换
------------
END
```

CUROPTMODE （语句）【Ver.1.98 以上版本】

功　能

获取动作模式。

格　式

CUROPTMODE

说　明

获取当前动作模式（手动、教导检查、内部自动、外部自动）的信息。

1: 手动
2: 教导
3: 内部自动
4: 外部自动

应用示例

```
PROGRAM  PRO1
------------
I1 = CUROPTMODE
------------
END
```

SYSSTATE（语句）【Ver.1.98 以上版本】

功　能

获取控制器的状态。

格　式

SYSSTATE

说　明

获取控制器状态。根据 I / O 配置的设定，有效的数据发生变化。下述表示获取可能的数据。

Bit	RC5 控制器的情况	RC7 控制器的情况 （使用 Mini I / O 时）
0	机械手正在运转	←
1	机械手异常	←
2	伺服 ON 状态	←
3	机械手初始化完成 （选择 I / O 标准模式时） / 机械手电源已投入 （选择 I / O 互换模式时）	机械手初始化完成
4	自动模式	←
5	外部模式	←
6	电池耗尽警告	←
7	机械手警告	预约
8	连续开始许可	←
9	SS 模式	←
10	机械手停止	←
11	自动运转允许	←
12 ～ 15	预约	←
16	程序开始复位 （选择 I / O 互换模式时）	预约
17	CAL 完成 （选择 I / O 互换模式时）	预约
18	正在教导 （选择 I / O 互换模式时）	预约
19	1 个循环完成 （选择 I / O 互换模式时）	预约
20 ～ 23	预约	预约
24	指令处理完成 （选择 I / O 标准模式时）	指令处理完成
25 ～ 31	预约	←

注意

有关 RC7 控制器 （使用 Mini I / O 时）的注意事项

为了获取 “ 运行准备 ” 的状态，请将 Bit2 的 " 伺服 ON 状态 " 和 Bit5 的 " 外部模式 " 的逻辑积 (AND) 作为其状态使用。

应用示例

```
PROGRAM  TSR1
------------
I1 = SYSSTATE                          ' 获取系统状态
IF  (I1  AND  &h0082)  THEN            ' 如果发生异常或者发生警告
CLRERR                                 ' 错误清空
END  IF
------------
END
```

第 20 章
预处理器

若使用 WINCAPS Ⅲ编写程序，则可利用本章说明的预处理器指令。
WINCAPS Ⅲ在编译之前，按照预处理器指令的定义，自动置换指定的字符串和文件之后，进行编译。
若使用预处理器指令，则可利用共用程序库 (Library) 的程序或记述易于阅读程序。

第 20 章　预处理器

20.1　符号常量、宏定义

#define（预处理器语句）

功　能

使用指定字符串替换程序中所指定的常量或宏名。

格　式

#define < 符号常量 > < 字符串 >

或

#define < 宏名 （自变量 (argument)） > < 包括自变量 (argument) 的字符串 >

说　明

使用指定字符串替换程序中的 < 符号常量 > 或 < 宏名 >。为宏名时，包括自变量 (argument) 也可置换。

程序中的 < 符号常量 > 或 < 宏名 > 不能置换以双引号 " " 括起的字符串。

#define 语句必须写在 1 行内。

在 < 符号常量 > 与 < 字符串 > 之间，必须插入 1 个以上的空格。

与括起宏名和自变量 (argument) ( ) 之间，不能插入空格。

< 符号常量 > 和 < 宏名 > 可重新定义 , 但是必须先用 #undef 进行无效化，然后进行定义才能有效。

< 符号常量 > 和 < 宏名 > 必须在 64 个字符以内。

1 个程序最多可使用 2048 个宏名。同时，宏函数的自变量 (argument) 数量没有限制。

相关项目

#undef

应用示例

```
#DEFINE NAME "Denso Corporation"
                                    ' 为 NAME 的符号常量赋予 "Denso Corporation"。

#DEFINE mAREA(radius) PI * POW(radius, 2)
                                    ' 将 mAREA (radius) 作为宏函数定义。
S1 = NAME                           ' 将 "Denso Corporation" 赋值入 S1。
D1 = mAREA(10)                      ' 将 PI*POW (10,2) 的计算值赋值 D1。
```

#undef（预处理器语句）

功　能

将通过 #define 定义的符号常量或宏的定义置为无效。

格　式

#undef {< 符号常量 > | < 宏名 >}

说　明

将通过 #define 定义的符号常量或宏的定义以该 #undef 以后的程序置为无效。带有自变量 (argument) 的宏指令，只指定宏名称。

相关项目

#define

应用示例

```
#UNDEF NAME                    ' 将用 #DEFINE 语句进行记号定义或者宏指令定义 NAME
                               ' 设为无效。
```

#error （预处理器语句）

功　　能

如果执行 #error 指令，则强制性置为编译错误。

格　　式

#error ［< 讯息 >］

说　　明

用于为了测试程序，故意导致错误等时使用。

编译使用 #error 指令的行时，产生错误。

作为错误讯息表示以 < 讯息 > 定义的内容。

相关项目

应用示例

#error "Error Occur"　　’编译时显示 "Error Occur" 讯息。

20.2 文件读取

#include（预处理器语句）

功　能

读入预处理程序。

格　式

#include " ［路径］文件名 "

#include < ［路径］文件名 >

说　明

在保存 #include 语句的位置，读取预处理器程序文件。如果省略文件的路径，则 " " 时，以当前目录、系统目录的顺序搜索文件。在 < > 时，只从系统目录搜索文件。以全路径指定路径时，只搜索该目录。

在用 #include 语句指定的文件中，可以进一步包含 #include 语句，能够最多进行 8 次嵌套。

可以指定的文件的扩展符有 H 和 PAC。

注意事项

使用文件夹功能 ［ RC7_Ver.2.2 以上版本］时的注意事项

注意

表示文件夹之间的分隔的 '\' 的显示，由显示系用 ' ＼ ' 显示。

例） " 教导器显示 ' ＼ '"、 " 小型教导器显示 '\' "

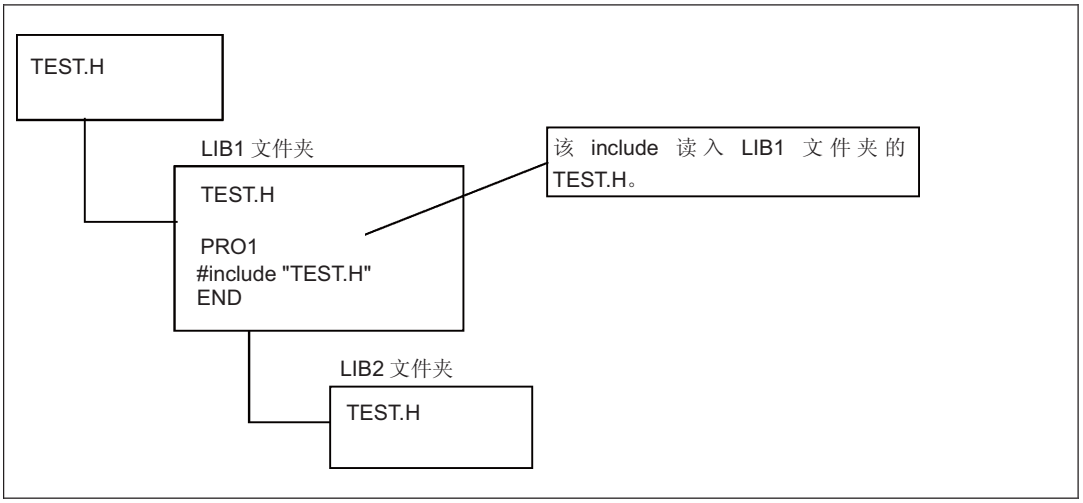
仅是显示上的不同，内部以相同内容处理。

(1) #include < 文件名 >

搜索系统文件夹。

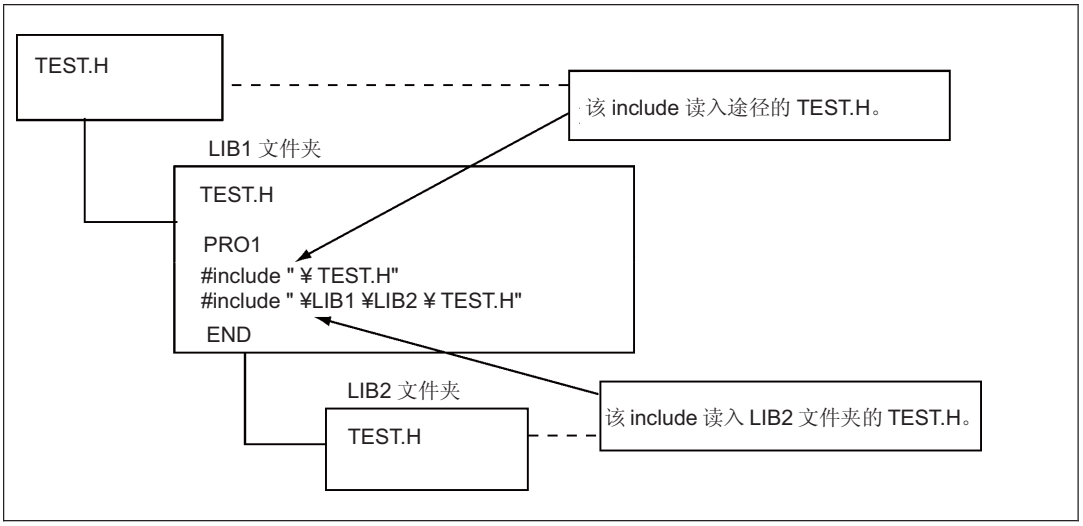
(2) #include " 文件名 "

在 include 语句所记述的 PAC 所存在的文件夹内中搜索。



(3) #include"￥（文件夹路径）￥文件名 "

从路径的文件夹路径开始的文件夹内中搜索。



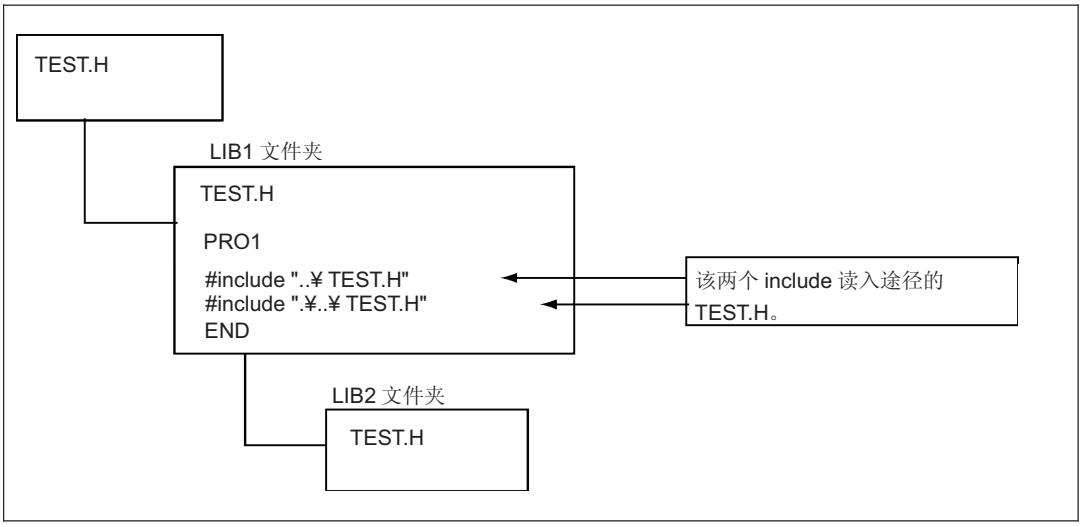
(4) #include "..\ 文件名 " 或者 #include ".\ 文件名 "

..￥: 搜索 1 个以上的文件夹内。

.￥: 搜索 include 语句所记述的 PAC 的文件夹内。

例)#include ".￥..￥TEST.H"

搜索 1 个以上的 TEST.H。



应用示例

```
#include "samp1.h"                    ' 将 samp1.h 文件在这一行展开。
```

20.3 最佳化

#pragma optimize（预处理器语句）

功　能

指定按程序单位执行的最优化处理。

格　式

#pragma optimize （"< 选择清单 >"， {on / off}）

说　明

指定按程序单位执行的最优化处理。在 PROGRAM 定义语句或者有效的语句行的前面记述。

在 < 选择清单 > 中，指定以下表中的任意个数的参数。

参数	最佳化的类型
A	排列检查编码的删除。
C	循环时间计算编码的删除。

补充说明

如果有效利用最佳化选择，则能将非运行命令的执行速度提高 10% ～ 20%。

注意事项

如果利用指令记述程序，将比项目参数的设定值更优先。

应用示例

```
#pragma optimize( "ac", ON )          ' 将排列检查编码以及循环时间计算编码的删除
                                      ' 设为有效。
```

第 21 章

视觉控制（机械手控制器：选择）

在本章中，对在μVision卡以及视觉装置μVision-21上可以使用的所有指令进行归纳。

第 21 章　视觉控制 （机械手控制器：选择）

21.1　图像输出入

CAMIN （语句）

功　能

将来自照相机的影像存放在图像储存器 （处理画面）。

格　式

CAMIN < 照相机编号 > ［， < 存放存储器编号 > ［， < 表编号 >］］

说　明

< 照相机编号 >

指定照相机编号。（1 或者 2）

< 存放存储器编号 >

指定存放存储器 （处理画面）的编号。 (0 ～ 3) 省略时，指定 0。

< 表编号 >

指定存放时的一览表编号。 (0 ～ 15) 省略时，指定 0。

注意

- 照相机未被连接或者因故障而不能正确输入时会发生错误。
- 存放时表编号不是 0 时，因为要进行表变更，所以画面上会出现乱像，但并不是故障。
- 执行后，表编号自动返回到 0 号。
- 本命令在机械手控制器中需要 μVision 卡 （选件）。

相关项目

CAMMODE、 CAMLEVEL、 VISDEFTABLE

应用示例

```
CAMMODE 1,0,0          ' 将照相机 1 号的影像在表 0 号 （与照相机的影像辉度相同）
                       ' 上进行转换，并存放在存放存储器 0 号上。
DELAY 2000             ' 停止 2 秒钟
CAMIN 1                ' 能够取得与 CAMIN 1, 0, 0 相同的结果。
DELAY 2000             ' 停止 2 秒钟
I1 = 1                 '
I2 = 0                 '
I3 = 3                 '
CAMIN I1,I2,I3         ' 将照相机 1 号的影像在表 3 号 （翻转）上进行转换，
                       ' 并存放在存放存储器 0 号上。
VISPLNOUT 0            ' 将存放存储器 0 号的图像输出到 （静止图像）监视器。
```

CAMMODE （语句）

功　能

设定存放照相机影像时的功能。

格　式

CAMMODE < 照相机编号 >， < 功能 >， < 存放方法 >

说　明

< 照相机编号 >

指定照相机编号。（1 或者 2）

< 功能 >

指定照相机的功能。（0 或者 1）

0: 正常 （照相机的设定为通常时）

1: 复位功能 （将照相机的动作复位后，保存照相机的影像。）

0: 帧输入

< 存放方法 >

指定存放方法。（0 或者 1）

将照相机的影像存放 1 帧。垂直方向的分解能达到最大。

1: 半帧输入

在使用半帧快门照相机的快门功能时进行设定。

没有半帧之间的延迟 （ 1 / 60 秒），可以输入没有摇晃的影像。

但是，垂直方向的分解能为 1 / 2。

注意

- 不设定本命令时，将遵循初始设定值。
- 本命令不变更初始设定。在电源再启动时，设定被丢失。
- 确认照相机的连接情况。在判断照相机不正常时，状态 VISSTATUS (0) 的值为 -1，正常时返回 0。
- 本命令在机械手控制器中需要 µVision 卡 （选件）。

相关项目

CAMIN、 VISSTATUS

应用示例

```
CAMMODE 1,0,0                  ' 将照相机 1 号的功能设定为正常，将输入方法设为帧。

I1 = VISSTATUS(0)              ' 正常时 I1 = 0
IF I1 = 0 THEN
  CAMIN 1                      ' 将照相机 1 号的影像在表 0 号 （与照相机的影像辉度相
                               ' 同） ' 上进行转换并存放在存放存储器 0 号上。
  VISPLNOUT 0                  ' 将存放存储器 0 号的图像输出到 （静止图像）监视器。

  VISLOC 10,10                 ' 设定显示位置
  VISPRINT " 输入正常 "        ' 在画面上显示字符。
ELSE
  VISLOC 10,10                 ' 设定显示位置
  VISPRINT " 照相机异常 "      ' 在画面上显示字符。
END IF
```

CAMLEVEL（语句）

功　能

设定照相机影像的输入等级。

格　式

CAMLEVEL < 照相机编号 >， < 下限等级 >， < 上限等级 >

说　明

< 照相机编号 >

指定照相机编号。（1 或者 2）

< 下限等级 >

设定照相机影像的输入等级的下限等级。（0 ～ 93）

< 上限等级 >

设定照相机影像的输入等级的上限等级。（7 ～ 100）

注意

- 本命令将最大输入范围的下限指定为 0%，上限作为 100% 来指定输入范围的设定。
- 将设定的下限、上限等级的范围设为 256 级调节。
- 设定需要是 0 ≦下限等级 < 上限等级 ≦ 100 而且上限－下限 ≦ 7。
- 暗淡的影像和明亮的影像，在部分提高辉度的分解能时，通过调整可以获得更详细的影像信息。
- 不设定本命令时，将遵循初始设定值。
- 本命令不变更初始设定。在电源再启动时，设定被丢失。
- 本命令在机械手控制器中需要 μVision 卡 （选件）。
- 本命令在执行以下的照相机输入 (CAMIN) 时有效。

相关项目

CAMIN、 VISCAMOUT

应用示例

```
CAMLEVEL 1,0,100          ' 将照相机 1 号的输入等级范围设为最大。
VISCAMOUT 1               ' 将来自照相机的影像 （动画影像）显示在监视器上。
CAMIN 1                   ' 将照相机 1 号的影像在表 0 号 （与照相机的影像辉度相
                          ' 同）' 上进行转换并存放在存放存储器 0 号上。
DELAY 2000                ' 停止 2 秒钟
CAMLEVEL 1,30,80          ' 将照相机 1 号的输入等级下限设定为 30%，
                          ' 上限设定为 80%。
VISCAMOUT 1               ' 将来自照相机的影像 （动画影像）显示在监视器上。
CAMIN 1
```

VISCAMOUT（语句）

功　能

将来自照相机的影像显示在监视器上。

格　式

VISCAMOUT < 照相机编号 > ［，< 表编号 >］

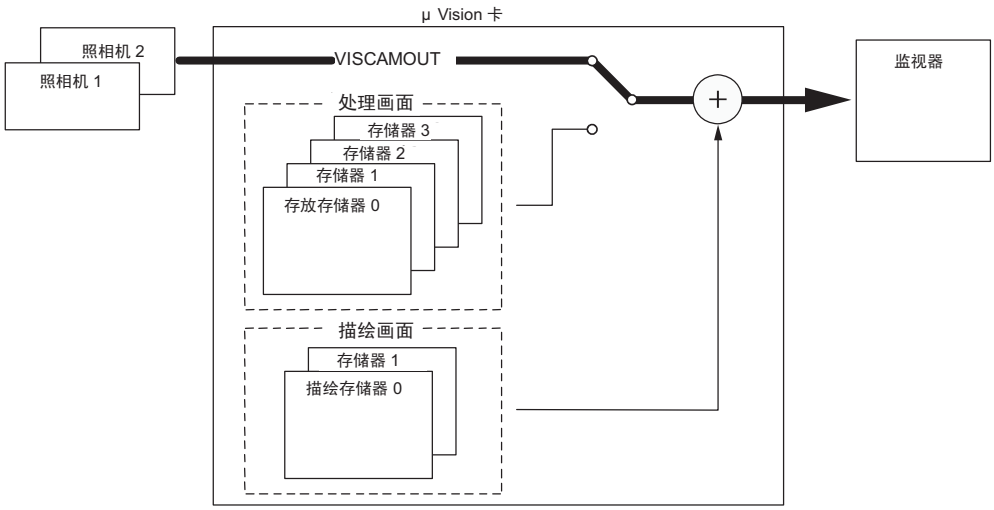
说　明

< 照相机编号 >

指定照相机编号。（1 或者 2）

< 表编号 >

指定显示时的一览表编号。 (0 ～ 15) 省略时，指定 1。



注意

本命令在机械手控制器中需要 μVision 卡 （选件）。

相关项目

VISDEFTABLE、 VISPLNOUT、 VISOVERLAY、 CAMMODE

应用示例

```
VISCAMOUT 1,1          ' 将来自照相机 1 号的影像 （动画影像）在表 1 号
                       ' （0 ～ 175 约压缩 70% 辉度）上进行转换并显示在监视器上。
VISCAMOUT 1            ' 可以取得与 VISCAMOUT 1, 1 相同的结果。
```

VISPLNOUT （语句）

功　能

将存放存储器的图像显示在监视器上。

格　式

VISPLNOUT < 存放存储器编号 > ［，< 表编号 >］

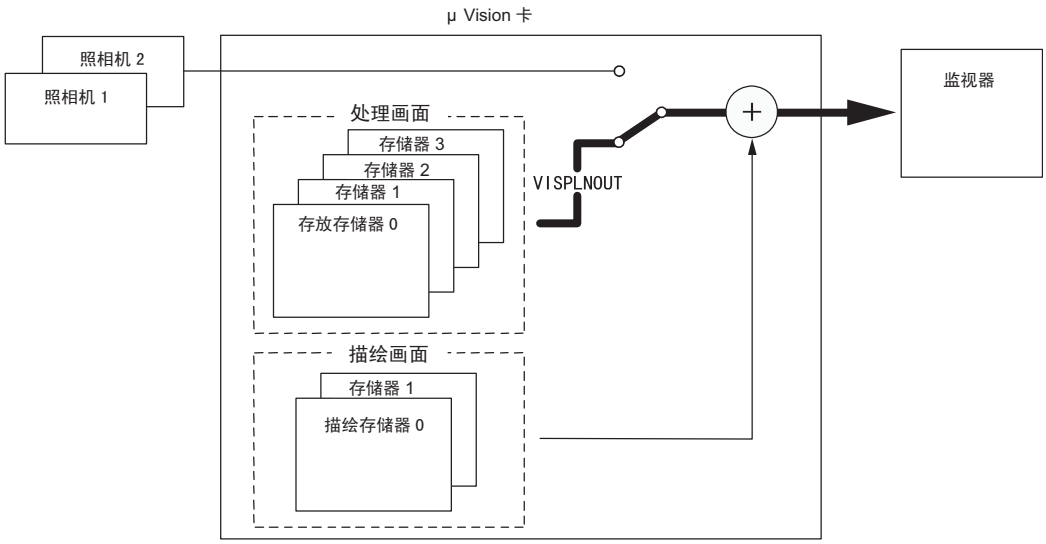
说　明

< 存放存储器编号 >

指定存放存储器编号 （处理画面编号）。（0 ～ 3）

< 表编号 >

指定显示时的一览表编号。 (0 ～ 15) 省略时，指定 1。



相关项目

VISDEFTABLE、 VISCAMOUT、 VISOVERLAY

应用示例

```
VISPLNOUT 0,1          ' 将存放存储器 0 号的图像 （停止图像）在表 1 号
                       ' （0 ～ 175 约压缩 70% 辉度）上进行转换并显示在监视器上。
VISPLNOUT 0            ' 可以取得与 VISPLNOUT 0,1 相同的结果。
```

VISOVERLAY（语句）

功　能

将描绘画面的信息显示在监视器上。

格　式

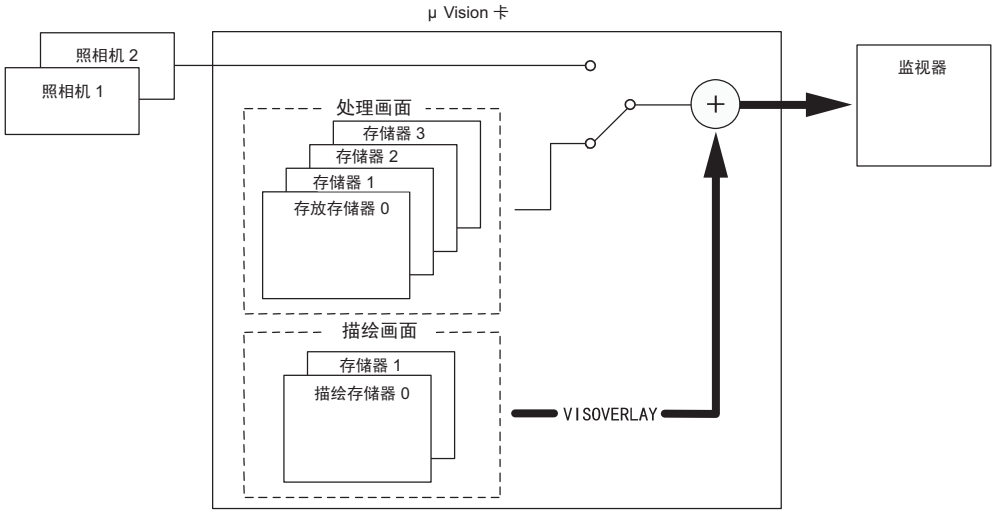
VISOVERLAY < 编号 >

说　明

< 编号 >

设定描绘画面的显示。（0 ～ 3）

0: 不显示描绘画面。
1: 显示描绘画面 0 号。
2: 显示描绘画面 1 号。
3: 同时显示 0、 1 两个描绘画面。



相关项目

VISCAMOUT、 VISPLNOUT

应用示例

```
VISOVERLAY 3                    ' 描绘目标画面的设定
VISSCREEN 1,0                   '
VISCLS 0                        '
VISLOC 10,10                    ' 设定显示位置
VISPRINT " 在描绘画面 0 上描绘 "  ' 在画面上描绘字符。
VISSCREEN 1,1                   ' 描绘目标画面的设定
VISLOC 10,11                    ' 设定显示位置
VISPRINT " 在描绘画面 1 上描绘 "  ' 在画面上描绘字符。
VISOVERLAY 0                    ' 中止描绘画面的显示。
DELAY 5000                      ' 停止 5 秒钟
VISOVERLAY 1                    ' 将描绘画面 0 号显示在监视器上。
DELAY 5000                      ' 停止 5 秒钟
VISOVERLAY 2                    ' 将描绘画面 1 号显示在监视器上。
DELAY 5000                      ' 停止 5 秒钟
VISOVERLAY 3                    ' 将描绘画面 0,1 号显示在监视器上。
```

VISDEFTABLE （语句）

功　能

设定照相机影像的输入、图像输出时的一览表数据。

格　式

VISDEFTABLE< 表编号 >， < 输入值 >， < 输出值 >

说　明

< 表编号 >

指定一览表编号。（5 ～ 15）

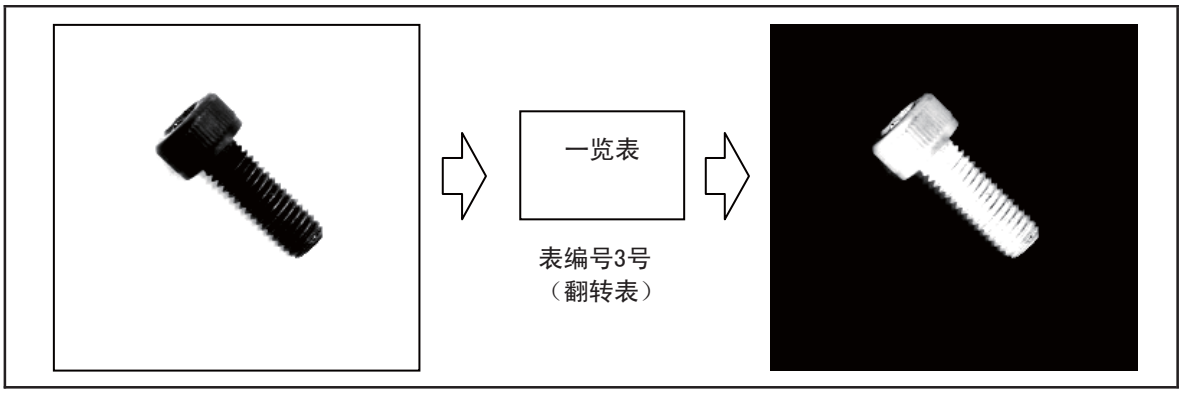
< 输入值 >

指定表的输入值。（0 ～ 255）

< 输出值 >

指定表的输出值。（0 ～ 255）

使用一览表进行转换的示例



注意

- 不设定本命令时，将遵循初始设定值。
- 本命令需要 2 ～ 3 （秒）的执行时间。
- 表编号 0 ～ 4 是被预先设定为认为有用的表。本命令不能进行变更。
  表 0 号 : 正常 (0 ～ 255)
  表 1 号 : 压缩 70% 辉度 (0 ～ 175)
  表 2 号 : γ 修正
  表 3 号 : 翻转
  表 4 号 : 压缩 70% 辉度翻转
- 本命令在机械手控制器中需要 μVision 卡 （选件）。
- 执行本命令时，请勿将电源 OFF。电源重新 ON 时，系统判断前回的命令没有正常结束，将把与视觉有关的信息进行初始化。

相关项目

CAMIN、 VISCAMOUT、 VISPLNOUT、 VISREFTABLE

应用示例

```
VISSCREEN 1,0,1                ’
VISCLS 0                       ’
VISCAMOUT 1,1                  ’将来自照相机 1 号的影像 （动画影像）在表 1 号
                               ’（0 ～ 175 约压缩 75% 辉度）上进行转换并显示在监视器上。
DELAY 5000                     ’停止 5 秒钟
VISDEFCHAR 4,4,2               ’
FOR I1 = 0 TO 255              ’
  VISLOC 10,10                 ’
  VISPRINT I1                  ’
  VISDEFTABLE 5,I1,255-I1      ’表 5 号的设定
NEXT I1                        ’
VISCAMOUT 1,5                  ’将来自照相机 1 号的影像 （动画影像）翻转并显示在监视器上。
```

VISREFTABLE （函数）

功　　能

参照一览表的数据。

格　　式

VISREFTABLE （< 表编号 >，< 输入值 >）

说　　明

< 表编号 >

指定一览表编号。（0 ～ 15）

< 输入值 >

指定表的输入值。（0 ～ 255）

相关项目

VISDEFTABLE

应用示例

```
VISSCREEN 1,0,1                 '
VISCLS 0                        '
FOR I1 = 0 TO 255               '
  I2 = VISREFTABLE(1,I1)        ' 获取表 1 号的数据。
  VISLOC 10,10                  ' 显示位置的设定
  VISDEFCHAR 1,1,2              ' 显示字符的设定
  VISPRINT " 数据 ";I1;"=";I2   ' 显示
NEXT I1                         '
```

21.2 视窗设定

WINDMAKE（语句）

功　能

指定图像处理的范围。

格　式

WINDMAKE < 视窗形状 >，< 视窗编号 >，< 参数 1>，< 参数 2>，…

直线视窗（指定 2 点）：

WINDMAKE P，< 视窗编号 >，< 始点 X 坐标 >，< 始点 Y 坐标 >，< 终点 X 坐标 >，< 终点 Y 坐标 >

直线视窗（指定长度、角度）：

WINDMAKE L，< 视窗编号 >，< 长度 > [，< 角度 >]

圆视窗：

WINDMAKE C，< 视窗编号 >，< 半径 >

椭圆视窗：

WINDMAKE E，< 视窗编号 >，< 宽度 >，< 高度 >

扇形视窗：

WINDMAKE S，< 视窗编号 >，< 外径 >，< 内径 >，< 起始角度 >，< 结束角度 >，< 分节锥角 >[，< 模式 >]

矩形视窗：

WINDMAKE R，< 视窗编号 >，< 宽度 >，< 高度 > [，< 角度 > [，< 模式 >]]

说　明

< 视窗形状 >

指定视窗的形状。

P: 直线视窗 Point to Point（指定 2 点）
L: 直线视窗 Line（指定长度、角度）
C: 圆视窗 Circle
E: 椭圆视窗 Ellipse
S: 扇形视窗 Section
R: 矩形视窗 Rectangle

< 视窗编号 >

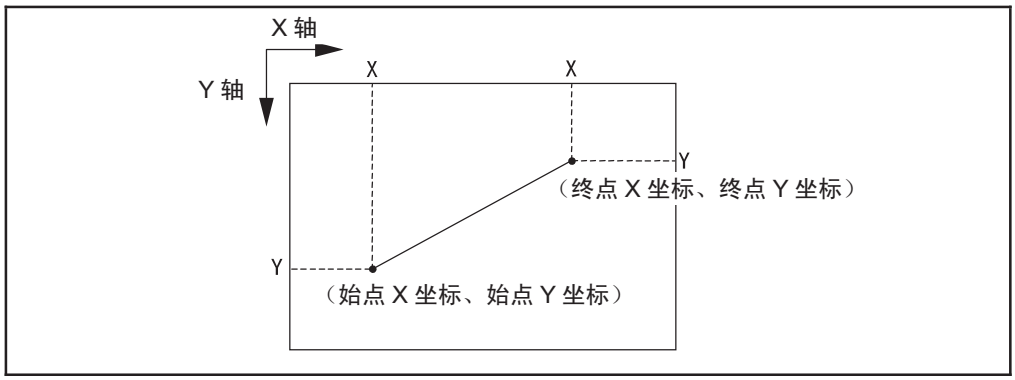
指定视窗的编号。（0 ～ 511）

＜参数＞

指定设定各个视窗形状时的值。

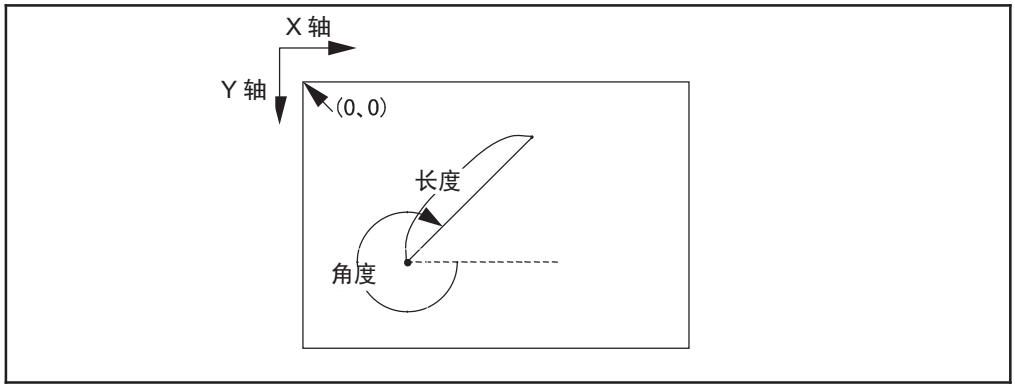
直线视窗 （指定 2 点）：

< 始点 X 坐标 > 指定直线的始点 X 坐标。（0 ～ 511）
< 始点 Y 坐标 > 指定直线的始点 Y 坐标。（0 ～ 479）
< 终点 X 坐标 > 指定直线的终点 X 坐标。（0 ～ 511）
< 终点 Y 坐标 > 指定直线的终点 Y 坐标。（0 ～ 479）



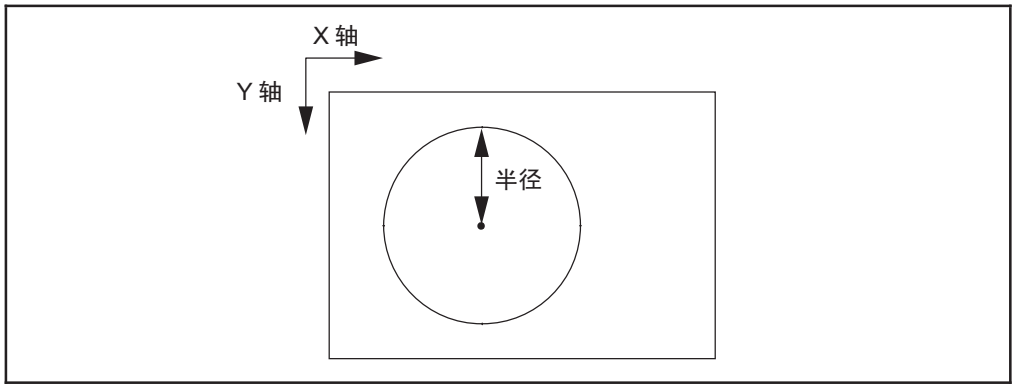
直线视窗 （指定长度、角度）：

< 长度 > 指定直线的长度。（1 ～ 512）
< 角度 > 指定直线的角度。省略时 0° 被指定。



圆视窗：

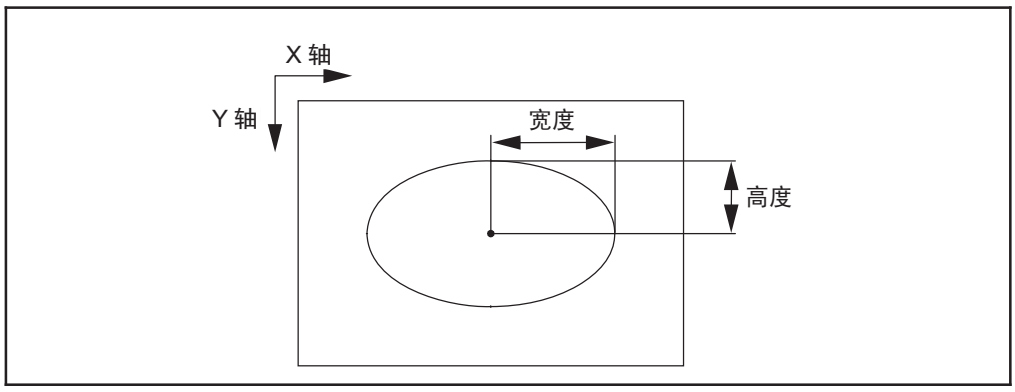
< 半径 > 指定圆的半径。（1 ～ 240）



椭圆视窗 :

< 宽度 > 指定椭圆的宽度。（ 1 ～ 256）

< 高度 > 指定椭圆的高度。（ 1 ～ 240）



扇形视窗 :

< 外径 > 指定扇形的外径。（外形 > 内径）(1 ～ 9999)

< 内径 > 指定扇形的内径。（外形 > 内径）(1 ～ 9999)

< 起始角度 > 指定扇形的起始角度。 （-720 ～ 720）

< 结束角度 > 指定扇形的结束角度。 （-720 ～ 720）

< 分节锥角 > 指定处理时的分解能。（0.1 ～ 360）

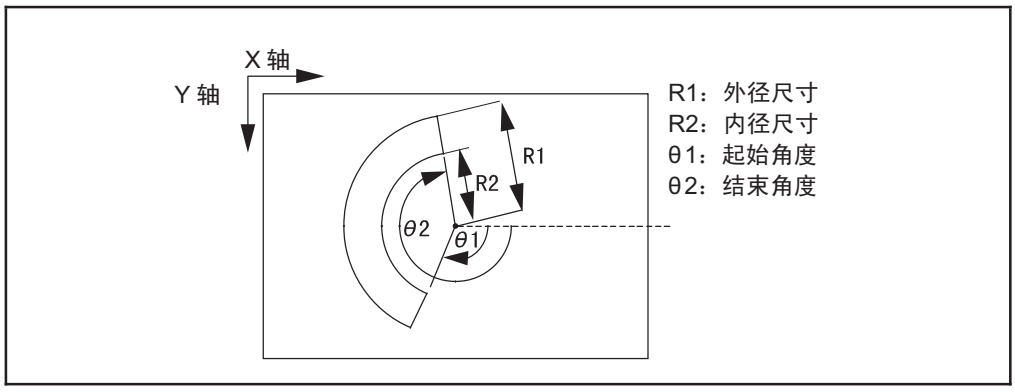
< 模式 > 指定描绘时的模式。（0 ～ 2）

0: 将 VISPROJ、 VISEDGE 的扫描方向指定为子午线方向。

1: 将 VISPROJ、 VISEDGE 的扫描方向指定为圆周方向。

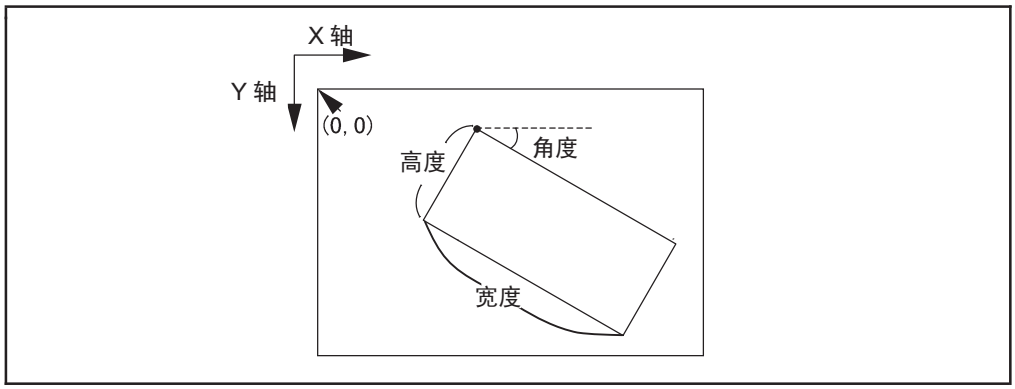
2: 不指定 VISPROJ、 VISEDGE 的扫描方向。

省略时，指定 2。



矩形视窗 :

< 宽度 > 指定矩形的宽度。（1 ～ 512）
< 高度 > 指定矩形的高度。（1 ～ 480）
< 角度 > 指定描绘的直线的角度。省略时 0° 被指定。（-720 ～ 720）
< 模式 > 指定描绘时的模式。（0 ～ 2）
0: 将 VISPROJ、 VISEDGE 的扫描方向指定为宽度方向。
1: 将 VISPROJ、 VISEDGE 的扫描方向指定为高度方向。
2: 不指定 VISPROJ、 VISEDGE 的扫描方向。
省略时，指定 0。



注意

- 本命令不变更初始设定。电源重新启动时，初始设定值复活。
- 需要永久变更时，请根据 WINCAPS Ⅲ进行编辑。
- 本命令在机械手控制器中需要 μVision 卡 （选件）。

相关项目

VISHIST、VISBINA、VISFILTER、VISMASK、VISMEASURE、VISPROJ、VISEDGE、VISREADQR、BLOB、SHMODEL、 SHCORNER、 SHCIRCLE

应用示例

```
VISSCREEN 1,0,1                          '
VISCLS 0                                 '
VISCAMOUT 1                              '
CAMIN 1                                  '
VISPLNOUT 0                              '
WINDMAKE P,1,50,100,100,150              ' 创建直线视窗 （指定 2 点）。
WINDMAKE L,2,100,45                      ' 创建直线视窗 （指定长度、角度）。
WINDMAKE C,3,50                          ' 创建圆视窗。
WINDMAKE E,4,50,100                      ' 创建椭圆视窗。
WINDMAKE S,5,100,80,90,300,1,2           ' 创建扇形视窗。
WINDMAKE R,6,100,50,45,2                 ' 创建矩形视窗。
VISMEASURE 1,100,100,1,1,128             '
WINDDISP 1                               '
VISMEASURE 2,150,150,1,1,128             '
WINDDISP 2                               '
VISMEASURE 3,200,200,1,1,128             '
WINDDISP 3                               '
VISMEASURE 4,250,250,1,1,128             '
WINDDISP 4                               '
VISMEASURE 5,300,300,1,1,128             '
WINDDISP 5                               '
VISMEASURE 6,350,350,1,1,128             '
WINDDISP 6                               '
```

WINDCLR（语句）

功　能

消除设定的视窗信息。

格　式

WINDCLR < 视窗编号 >

说　明

< 视窗编号 >

指定视窗的编号。（0 ～ 511）

注意

- 本命令不变更初始设定。电源重新启动时，初始设定值复活。
- 需要永久删除时，请从 WINCAPS Ⅲ中消除。
- 本命令在机械手控制器中需要 μVision 卡 （选件）。

相关项目

WINDMAKE

应用示例

```
VISSCREEN 1,0,1                    ’
VISCLS 0                           ’
VISCAMOUT 1                        ’
WINDMAKE R,1,50,100,0,2            ’
FOR I1 = 0 TO 7                    ’
  I2 = WINDREF(1,I1)               ’ 获取视窗 1 号的数据。
  VISLOC 0,I1                      ’ 显示位置的设定
  VISPRINT " 数据 ";I1;"=";I2      ’ 显示
NEXT I1                            ’
WINDCLR 1                          ’
VISLOC 0,9                         ’
VISPRINT " 视窗信息消除后 "        ’
FOR I1 = 0 TO 7                    ’
  I2 = WINDREF(1,I1)               ’ 获取视窗 1 号的数据。
  VISLOC 0,10+I1                   ’ 显示位置的设定
  VISPRINT " 数据 ";I1;"=";I2      ’ 显示
NEXT I1                            ’
```

WINDCOPY（语句）

功　能

复制视窗的数据。

格　式

WINDCOPY < 复制原件视窗编号 >， < 复制位置视窗编号 >

说　明

< 复制原件视窗编号 >

指定进行复制的视窗的编号。（0 ～ 511）

< 复制位置视窗编号 >

指定复制位置视窗编号。（0 ～ 511）

注意

- 本命令不变更初始设定。电源重新启动时，复制的设定被丢失。
- 本命令在机械手控制器中需要 μVision 卡 （选件）。

相关项目

WINDMAKE

应用示例

```
VISSCREEN 1,0,1                          ’
VISCLS 0                                 ’
VISCAMOUT 1                              ’
WINDCLR 2                                ’
WINDMAKE R,1,50,100,0,2                  ’
VISLOC 0,0                               ’
VISPRINT " 复制原件视窗信息 "             ’
                                         ’
FOR I1 = 0 TO 7                          ’
  I2 = WINDREF(1,I1)                     ’ 获取视窗 1 号的数据。
  VISLOC 0,1+I1                          ’ 显示位置的设定
  VISPRINT " 数据 ";I1;"=";I2            ’ 显示
NEXT I1                                  ’
VISLOC 0,9                               ’
VISPRINT " 视窗信息复制之前 "             ’
                                         ’
FOR I1 = 0 TO 7                          ’
  I2 = WINDREF(2,I1)                     ’ 获取视窗 1 号的数据。
  VISLOC 0,10+I1                         ’ 显示位置的设定
  VISPRINT " 数据 ";I1;"=";I2            ’ 显示
NEXT I1                                  ’
VISLOC 0,18                              ’
VISPRINT " 视窗信息复制之后 "             ’
                                         ’
FOR I1 = 0 TO 7                          ’
  I2 = WINDREF(2,I1)                     ’ 获取视窗 1 号的数据。
  VISLOC 0,19+I1                         ’ 显示位置的设定
  VISPRINT " 数据 ";I1;"=";I2            ’ 显示
NEXT I1                                  ’
```

WINDREF（函数）

功　能

取得视窗的信息。

格　式

WINDREF （< 视窗编号 >，< 项目 >）

说　明

< 视窗编号 >

指定视窗的编号。（0 ～ 511）

< 项目 >

指定想要取得信息的编号。（0 ～ 9）

项目编号 0: 有无视窗的设定　　返回值 有＝ 0 无 =-1

项目编号 1: 视窗形状　　返回值 （参照下表）

项目编号 2: 视窗原点的 X 坐标

项目编号 3: 视窗原点的 Y 坐标

项目编号 4 ～ 9: 视窗的各个设定数据　　返回值 各个设定数据 （参照下表）

	项目						
视窗形状	1	4	5	6	7	8	9
直线 （指示 2 点）	0	始点 X 坐标	始点 Y 坐标	终点 X 坐标	终点 Y 坐标	-1	-1
直线 （指定长度、角度）	1	长度	角度	-1	-1	-1	-1
圆	2	半径	-1	-1	-1	-1	-1
椭圆	3	宽度	高度	-1	-1	-1	-1
扇形	4	外径	内径	起始角度	结束角度	节面角	模式
矩形	5	宽度	高度	角度	模式	-1	-1

注意

- 没有设定数据时，返回值为 -1。
- 所取得的数据是当前的设定数据，不是初始设定数据。
- 本命令在机械手控制器中需要 µVision 卡 （选件）。

相关项目

WINDMAKE

应用示例

```
VISSCREEN 1,0,1                    '
VISCLS 0                           '
VISCAMOUT 1                        '
WINDMAKE R,1,50,100,0,2            '
FOR I1 = 0 TO 7                    '
  I2 = WINDREF(1,I1)               ' 获取视窗 1 号的数据。
  VISLOC 0,I1                      ' 显示位置的设定
  VISPRINT " 数据 ";I1;"=";I2      ' 显示
NEXT I1
```

WINDDISP （语句）

功　能

描绘所设定的视窗。

格　式

WINDDISP < 视窗编号 >

说　明

< 视窗编号 >

指定视窗的编号。（0 ～ 511）

注意

- 在视窗信息里有视窗的处理原点位置，但该数据由各个图像处理命令所决定。因此如果在执行处理命令后执行本命令，则被显示在正确的位置。
- 新设定视窗后，视窗原点被设定为 X = 0、 Y = 0。
- 描绘的画面会成为被 VISSCREEN 设定的画面。
- 本命令在机械手控制器中需要 μVision 卡 （选件）。

相关项目

WINDMAKE、 VISSCREEN

应用示例

```
VISOVERLAY 1                ' 将描绘专用画面 0 号显示在监视器上。
VISSCREEN 1,0,1             ' 在描绘画面 0 号上进行即时描绘。
WINDMAKE R,1,200,200,0,2    ' 矩形的视窗创建
WINDDISP 1                  ' 视窗的显示 （原点 X = 0、 Y = 0）
VISHIST 1,100,100           ' 执行直方柱状体处理
WINDDISP 1                  ' 视窗的显示 （原点 X =100、 Y =100）
```

21.3 描绘

VISSCREEN（语句）

功　能

指定描绘的画面。

格　式

VISSCREEN < 描绘目标 >，< 画面编号 > [，< 描绘模式 >]

说　明

< 描绘目标 >

指定存放存储器、覆盖存储 （画面）中的一个。（0 或者 1）

0: 存放存储器 （处理画面）

1: 覆盖存储 （描绘专用画面）

< 画面编号 >

各个画面上的画面编号

存放存储器 （处理画面）: 0 ～ 3 的 4 个画面

覆盖存储 （描绘专用画面）: 0、 1 的 2 个画面

< 描绘模式 >

指定描绘时的显示方法。（0 或者 1）

0: 即时显示 OFF

1: 即时显示 ON

省略时，指定 1。

注意

- 电源接通之后的初始设定被设定为 VISSCREEN 1, 0, 1。
- 即时显示为 ON 时，执行描绘命令时，描绘结果被监视显示，为 OFF 时，通过执行 VISPLNOUT 和 VISOVERLAY 中的任意一个对应描绘目标的命令，画面被更新。
- 在即使描绘为 ON 时，执行描绘命令时因为包含着大约 0.02 秒的更新时间，所以频繁进行描绘时，其累积增大。将即时描绘置于 OFF，通过归纳显示能够缩短处理时间。
- 本命令在机械手控制器中需要 μVision 卡 （选件）。

相关项目

VISBRIGHT、VISCLS、VISPUTP、VISLINE、VISPTP、VISRECT、VISCIRCLE、VISELLIPSE、VISSECT、VISCROSS、 VISLOC、 VISDEFCHAR、 VISPRINT

应用示例

```
VISSCREEN 1,0,1                ' 在描绘画面 0 号上进行即时描绘。
VISCLS 0                       '
FOR I1 = 100 TO 200 STEP 2     '
  VISRECT I1,I1,200,200        ' 描绘宽度 200 高度 200 的矩形。
NEXT I1                        '
VISCLS 0                       ' 画面的清空
VISSCREEN 1,0,0                ' 在描绘专用画面 0 号上进行描绘。
FOR I1 = 100 TO 200 STEP 2     '
  VISRECT I1,I1,200,200        ' 描绘宽度 200 高度 200 的矩形。
NEXT I1                        '
VISOVERLAY 1                   ' 将描绘专用画面 0 号显示 （更新）在监视器上。
```

VISBRIGHT（语句）

功　能

指定描绘时的辉度值。

格　式

VISBRIGHT < 辉度值 >

说　明

< 辉度值 >

指定描绘时的辉度值。（0 ～ 255）

注意

- 描绘目标为存放存储器（处理画面）时，用指定辉度进行描绘，但在覆盖存储（描绘专用存储器）时，则为以下的显示辉度。
  辉度值 128 ～ 255: 255（白色）
  辉度值 1 ～ 127: 1（黑色）
  辉度值 0: 直通（透明）
- 接通电源时，初始设定被设定为 VISBRIGHT 255。
- 本命令在机械手控制器中需要 μVision 卡（选件）。

相关项目

VISSCREEN、VISCLS、VISPUTP、VISLINE、VISPTP、VISRECT、VISCIRCLE、VISELLIPSE、VISSECT、VISCROSS、 VISLOC、 VISDEFCHAR、 VISPRINT

应用示例

```
VISPLNOUT 0                          ' 显示存放存储器 0 号（处理画面）。
VISSCREEN 0,0,1                      ' 在处理画面 0 号上进行即时描绘。
VISCLS 0                             '
FOR I1 = 10 TO 200 STEP 5            '
  VISBRIGHT I1                       '
  VISRECT 100+I1,100,200,200         ' 描绘宽度 200 高度 200 的矩形。
NEXT I1                              '
DELAY 500                            ' 停止 0.5 秒钟
VISCLS 0                             ' 画面的清空
VISSCREEN 1,0,1                      ' 在描绘专用画面 0 号上进行即时描绘。
FOR I1 = 10 TO 200 STEP 5            '
  VISBRIGHT I1                       '
  VISRECT 100+I1,100,200,200         ' 描绘宽度 200 高度 200 的矩形。
NEXT I1                              '
```

VISCLS （语句）

功　能

用指定辉度全面涂抹 （清空）模式所指定的画面。

格　式

VISCLS ［< 模式 > ［,< 辉度值 >］］

说　明

< 模式 >

(0 ～ 3) 省略时，指定 2。

0: 用指定辉度全部涂抹被 VISSCREEN 命令设定的画面。
1: 用指定辉度全部涂抹所有处理画面。
2: 用指定辉度全部涂抹所有描绘画面。
3: 用指定辉度全部涂抹所有的画面。

< 辉度值 >

指定全部涂抹 （清空）时的辉度值。 (0 ～ 255) 省略时，指定 0。

注意

- 描绘目标为存放存储器 （处理画面）时，用指定辉度进行清空，但在覆盖存储 （描绘专用存储器）时，则为以下的显示辉度。
  辉度值 128 ～ 255: 255 （白色）
  辉度值 1 ～ 127: 1 （黑色）
  辉度值 0: 直通 （透明）
- 本命令在机械手控制器中需要 µVision 卡 （选件）。

相关项目

VISSCREEN

应用示例

```
VISSCREEN 1,0,1                 ’
VISCLS 0                        ’
VISSCREEN 1,1,1                 ’
VISCLS 0                        ’
VISPLNOUT 0                     ’显示存放存储器 0 号 （处理画面）。
VISSCREEN 0,0,1                 ’在处理画面 0 号上进行即时描绘。
FOR I1 = 10 TO 200 STEP 5       ’
  VISCLS 0,I1                   ’用 I1 的指定辉度清空画面。
NEXT I1                         ’
DELAY 500                       ’停止 0.5 秒钟
VISCLS 0                        ’用辉度 0 清空画面。
VISRECT 200,200,200,200         ’描绘宽度 200 高度 200 的矩形。
DELAY 500                       ’停止 0.5 秒钟
VISCLS 0                        ’清空描绘画面的矩形。
VISCAMOUT 1                     ’
VISSCREEN 1,0,1                 ’在描绘专用画面 0 号上进行即时描绘。
VISRECT 100,100,200,200         ’描绘宽度 200 高度 200 的矩形。
DELAY 500                       ’停止 0.5 秒钟
VISCLS 0                        ’清空描绘画面的矩形。
VISCLS 0,127                    ’清空 （黑色）描绘画面。
DELAY 500                       ’停止 0.5 秒钟
VISCLS 0,255                    ’清空 （白色）描绘画面。
DELAY 500                       ’停止 0.5 秒钟
VISCLS 0                        ’直通 （透明）描绘画面。
```

VISPUTP （语句）

功　能

在画面上描绘点。

格　式

VISPUTP <X 坐标 >， <Y 坐标 >

说　明

<X 坐标 >

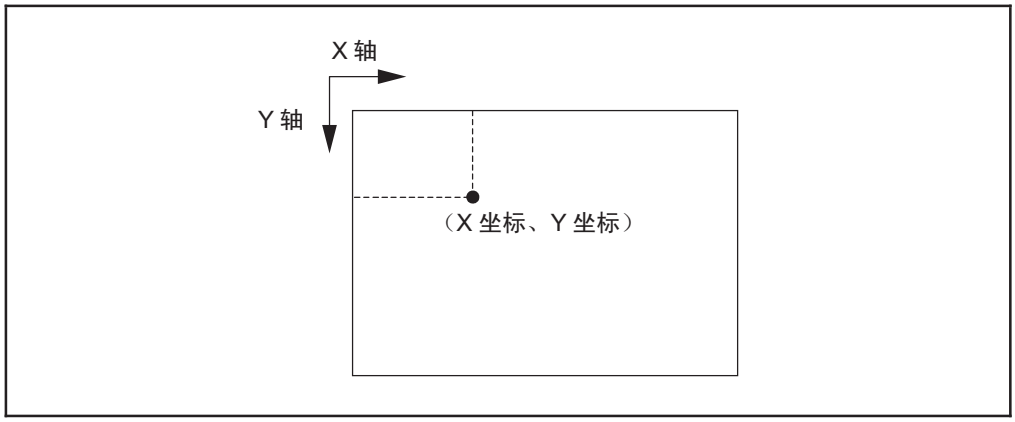
指定描绘点的 X 坐标。

<Y 坐标 >

指定描绘点的 Y 坐标。

注意

- X、 Y 坐标的数值没有进行检查。即使超出描绘的范围也不会发生错误。
- 目标画面成为被 VISSCREEN 设定的画面。
- 描绘的辉度值成为 VISBRIGHT 所指定的辉度。
- 本命令在机械手控制器中需要 μVision 卡 （选件）。



相关项目

VISSCREEN、 VISBRIGHT

应用示例

```
VISPLNOUT 0                    ' 显示存放存储器 0 号 （处理画面）。
VISSCREEN 0,0,1                ' 在处理画面 0 号上进行即时描绘。
VISCLS 0                       ' 画面的清空
VISBRIGHT 255                  ' 描绘的辉度值的设定
FOR I1 = 10 TO 200 STEP 5      '
  VISPUTP 100+I1,100           ' 在 (100 + I1, 100) 坐标上描绘点。
NEXT I1                        '
```

VISLINE（语句）

功　能

在画面上描绘直线。

格　式

VISLINE <X 坐标 >， <Y 坐标 >， < 长度 > [， < 角度 >]

说　明

<X 坐标 >

指定描绘直线的 X 坐标。

<Y 坐标 >

指定描绘直线的 Y 坐标。

< 长度 >

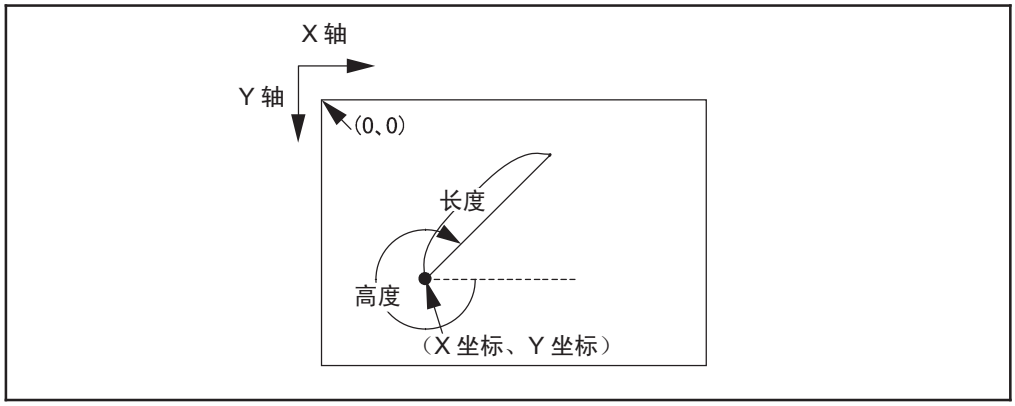
指定描绘的直线的长度。

< 角度 >

指定描绘的直线的角度。省略时 0° 被指定。

注意

- X、 Y 坐标、长度、角度的数值没有进行检查。即使超出描绘的范围也不会发生错误。
- 目标画面成为被 VISSCREEN 设定的画面。
- 描绘的辉度值成为 VISBRIGHT 所指定的辉度。
- 本命令在机械手控制器中需要 μVision 卡 （选件）。



相关项目

VISSCREEN、 VISBRIGHT

应用示例

```
VISPLNOUT 0                    ' 显示存放存储器 0 号 （处理画面）。
VISSCREEN 0,0,1                ' 在处理画面 0 号上进行即时描绘。
VISCLS 0                       ' 画面的清空
VISBRIGHT 255                  ' 描绘的辉度值的设定
FOR I1 = 0 TO 360 STEP 5
  VISLINE 256,256,100,I1       ' 描绘长度 100 的直线。
NEXT I1
```

VISPTP （ステートメント）

功　能

在画面上描绘连接 2 点间的直线。

格　式

VISPTP < 始点 X 坐标 >， < 始点 Y 坐标 >， < 终点 X 坐标 >， < 终点 Y 坐标 >

说　明

< 始点 X 坐标 >

指定描绘直线的始点 X 坐标。

< 始点 Y 坐标 >

指定描绘直线的始点 Y 坐标。

< 终点 X 坐标 >

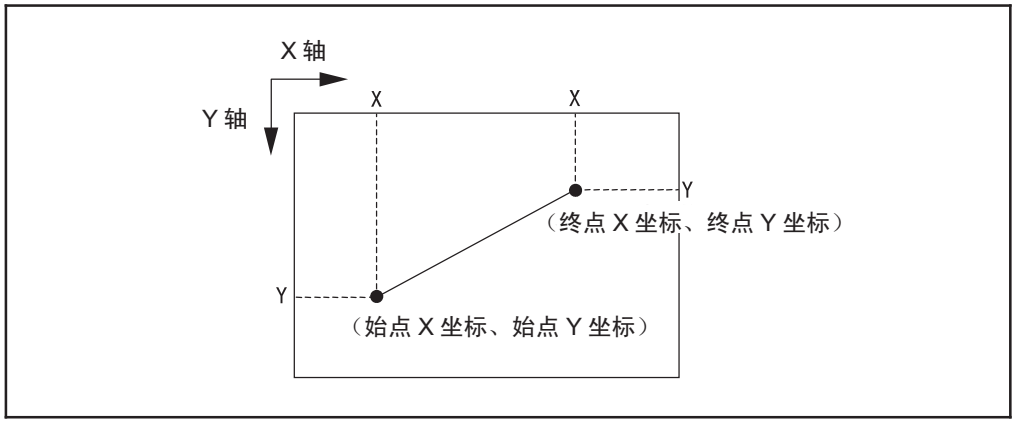
指定描绘直线的终点 X 坐标。

< 终点 Y 坐标 >

指定描绘直线的终点 Y 坐标。

注意

- X、 Y 坐标的数值没有进行检查。即使超出描绘的范围也不会发生错误。
- 目标画面成为被 VISSCREEN 设定的画面。
- 描绘的辉度值成为 VISBRIGHT 所指定的辉度。
- 本命令在机械手控制器中需要 μVision 卡 （选件）。



相关项目

VISSCREEN、 VISBRIGHT

应用示例

```
VISPLNOUT 0                    ' 显示存放存储器 0 号 （处理画面）。
VISSCREEN 0,0,1                ' 在处理画面 0 号上进行即时描绘。
VISCLS 0                       ' 画面的清空
VISBRIGHT 255                  ' 描绘的辉度值的设定
FOR I1 = 200 TO 300            '
  VISPTP 100,100,200,I1        ' 描绘连接 2 点间的直线。
NEXT I1                        '
```

VISRECT（语句）

功　能

在画面上描绘矩形。

格　式

VISRECT <X 坐标 >， <Y 坐标 >， < 宽度 >， < 高度 > [， < 模式 > [， < 角度 >]]

说　明

<X 坐标 >

指定描绘矩形的 X 坐标。

<Y 坐标 >

指定描绘矩形的 Y 坐标。

< 宽度 >

指定描绘的矩形的宽度。

< 高度 >

指定描绘的矩形的高度。

< 模式 >

指定描绘时的模式。

0: 只描绘矩形的外形线。

1: 描绘全部涂抹的矩形。

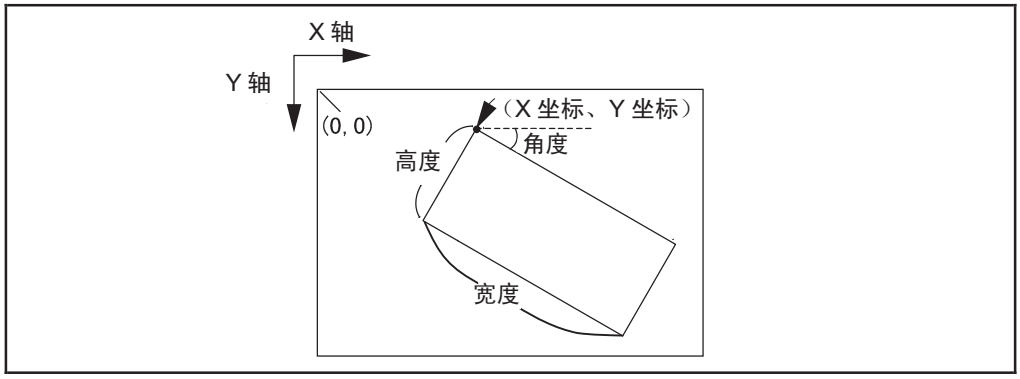
省略时，指定 0。

< 角度 >

指定描绘的直线的角度。省略时 0° 被指定。

注意

- X、 Y 坐标、长度、角度的数值没有进行检查。即使超出描绘的范围也不会发生错误。
- 目标画面成为被 VISSCREEN 设定的画面。
- 描绘的辉度值成为 VISBRIGHT 所指定的辉度。
- 本命令在机械手控制器中需要 µVision 卡 （选件）。



相关项目

VISSCREEN、 VISBRIGHT

应用示例

```
VISPLNOUT 0                              ' 显示存放存储器 0 号 （处理画面）。
VISSCREEN 0,0,1                          ' 在处理画面 0 号上进行即时描绘。
VISCLS 0                                 ' 画面的清空
VISBRIGHT 255                            ' 描绘的辉度值的设定
FOR I1 = 0 TO 360 STEP 5                 '
   VISRECT 256,256,100,100,0,I1          ' 描绘高度 100 宽度 100 的矩形。
NEXT I1                                  '
VISRECT 200,200,50,50,215                ' 描绘高度 50 宽度 50 的且全部涂抹的矩形。
```

VISCIRCLE（语句）

功　能

在画面上描绘圆。

格　式

格式 VISCIRCLE <X 坐标 >， <Y 坐标 >， < 半径 > [， < 模式 >]

说　明

<X 坐标 >

指定描绘圆的 X 坐标。

<Y 坐标 >

指定描绘圆的 Y 坐标。

< 半径 >

指定描绘的圆的半径。

< 模式 >

指定描绘时的模式。

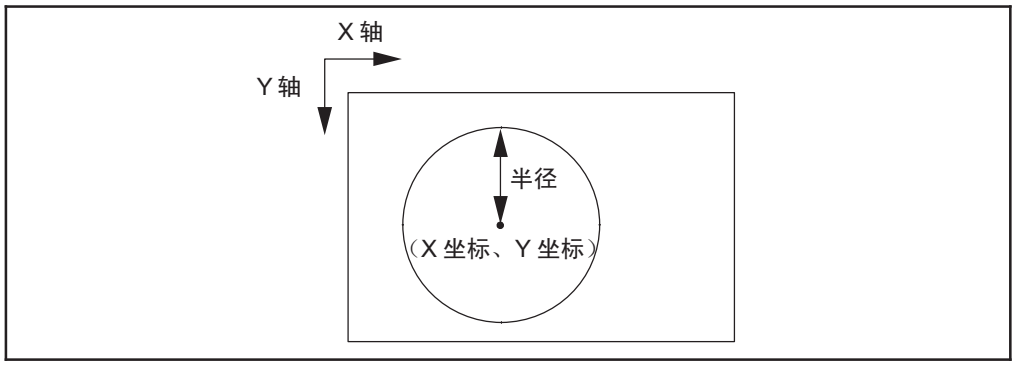
0: 只描绘圆的外形线。

1: 描绘全部涂抹的圆。

省略时，指定 0。

注意

- X、 Y 坐标的数值没有进行检查。即使超出描绘的范围也不会发生错误。
- 目标画面成为被 VISSCREEN 设定的画面。
- 描绘的辉度值成为 VISBRIGHT 所指定的辉度。
- 本命令在机械手控制器中需要 μVision 卡 （选件）。



相关项目

VISSCREEN、 VISBRIGHT

应用示例

```
VISPLNOUT 0                    ' 显示存放存储器 0 号 （处理画面）。
VISSCREEN 0,0,1                ' 在处理画面 0 号上进行即时描绘。
VISCLS 0                       ' 画面的清空
VISBRIGHT 255                  ' 描绘的辉度值的设定
VISCIRCLE 255,255,100          ' 描绘半径 100 的圆。
VISCIRCLE 200,200,50,1         ' 描绘半径 50 的全部涂抹的圆。
```

VISELLIPSE （语句）

功　能

在画面上描绘椭圆。

格　式

VISELLIPSE <X 坐标 >， <Y 坐标 >， < 宽度 >， < 高度 > [， < 模式 >]

说　明

<X 坐标 >

指定描绘椭圆的 X 坐标。

<Y 坐标 >

指定描绘椭圆的 Y 坐标。

< 宽度 >

指定描绘的椭圆的宽度。

< 高度 >

指定描绘的椭圆的高度。

< 模式 >

指定描绘时的模式。

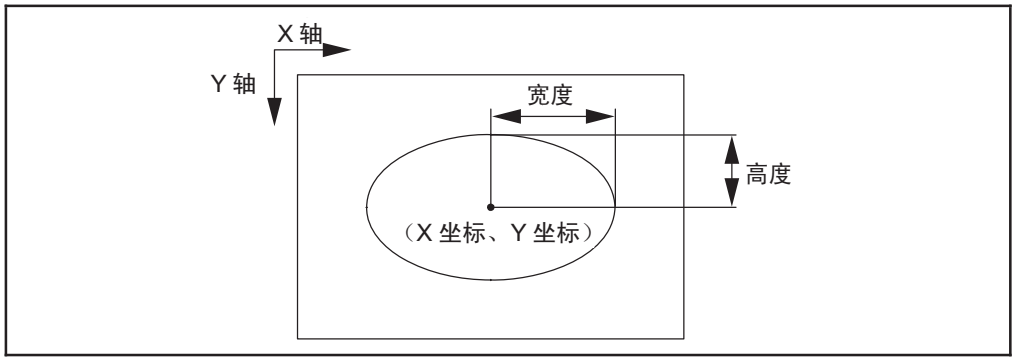
0: 只描绘椭圆的外形线。

1: 描绘全部涂抹的椭圆。

省略时，指定 0。

注意

- X、 Y 坐标、宽度、高度的数值没有进行检查。即使超出描绘的范围也不会发生错误。
- 目标画面成为被 VISSCREEN 设定的画面。
- 描绘的辉度值成为 VISBRIGHT 所指定的辉度。
- 本命令在机械手控制器中需要 μVision 卡 （选件）。



相关项目

VISSCREEN、 VISBRIGHT

应用示例

```
VISPLNOUT 0                    ' 显示存放存储器 0 号 （处理画面）。
VISSCREEN 0,0,1                ' 在处理画面 0 号上进行即时描绘。
VISCLS 0                       ' 画面的清空
VISBRIGHT 255                  ' 描绘的辉度值的设定
VISELLIPSE 255,255,100,50      ' 描绘宽度 100 高度 50 的椭圆。
VISELLIPSE 255,255,50,100,1    ' 描绘宽度 50 高度 100 的全部涂抹的椭圆。
```

VISSECT （语句）

功　能

在画面上描绘扇形。

格　式

VISSECT <X 坐标 >， <Y 坐标 >， < 外径 >， < 内径 >， < 起始角度 >，< 结束角度 >

说　明

<X 坐标 >

指定描绘扇形的 X 坐标。

<Y 坐标 >

指定描绘扇形的 Y 坐标。

< 外径 >

指定描绘的扇形的外径。

< 内径 >

指定描绘的扇形的内径。

< 起始角度 >

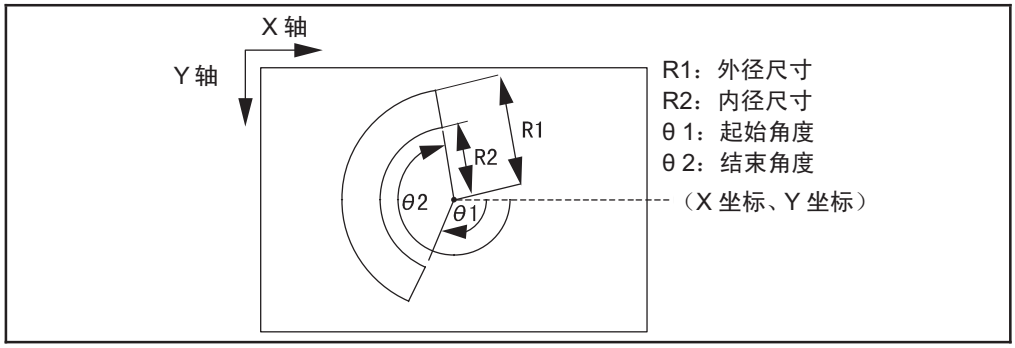
指定描绘的扇形的起始角度。

< 结束角度 >

指定描绘的扇形的结束角度。

注意

- 指定的数值没有进行检查。即使超出描绘的范围也不会发生错误。
- 目标画面成为被 VISSCREEN 设定的画面。
- 描绘的辉度值成为 VISBRIGHT 所指定的辉度。
- 本命令在机械手控制器中需要 μVision 卡 （选件）。



相关项目

VISSCREEN、 VISBRIGHT

应用示例

```
VISPLNOUT 0                          ' 显示存放存储器 0 号 （处理画面）。
VISSCREEN 0,0,1                      ' 在处理画面 0 号上进行即时描绘。
VISCLS 0                             ' 画面的清空
VISBRIGHT 255                        ' 描绘的辉度值的设定
VISSECT 255,255,100,50,100,260       ' 描绘扇形。
```

VISCROSS（语句）

功　能

在画面上描绘十字标记。

格　式

VISCROSS <X 坐标 >， <Y 坐标 > [， < 轴长 1> [， < 轴长 2> [， < 角度 >]]]

说　明

<X 坐标 >

指定描绘十字标记的 X 坐标。

<Y 坐标 >

指定描绘十字标记的 Y 坐标。

< 轴长 1>

指定描绘的十字标记的轴长。省略时，指定 10。

< 轴长 2>

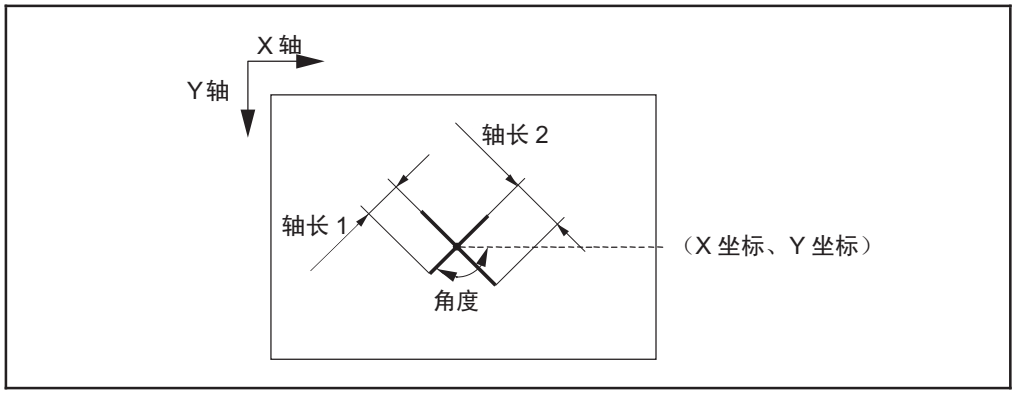
指定描绘的十字标记的轴长。省略时，指定 10。

< 角度 >

指定描绘的十字标记的角度。省略时，指定 0°。

注意

- 指定的数值没有进行检查。即使超出描绘的范围也不会发生错误。
- 目标画面成为被 VISSCREEN 设定的画面。
- 描绘的辉度值成为 VISBRIGHT 所指定的辉度。
- 本命令在机械手控制器中需要 μVision 卡 （选件）。



相关项目

VISSCREEN、 VISBRIGHT

应用示例

```
VISPLNOUT 0                    ' 显示存放存储器 0 号 （处理画面）。
VISSCREEN 0,0,1                ' 在处理画面 0 号上进行即时描绘。
VISCLS 0                       ' 画面的清空
VISBRIGHT 255                  ' 描绘的辉度值的设定
VISCROSS 255,255,10,20,45      ' 描绘十字标记。
```

VISLOC （语句）

功　能

指定字符的显示位置。

格　式

VISLOC <X 位置 >， <Y 位置 > ［， < 模式 >］

说　明

＜ X 位置＞

指定描绘字符的 X 位置。

<Y 位置 >

指定描绘字符的 Y 位置。

< 模式 >

指定 X、 Y 位置数据的坐标模式。省略时，指定 0。

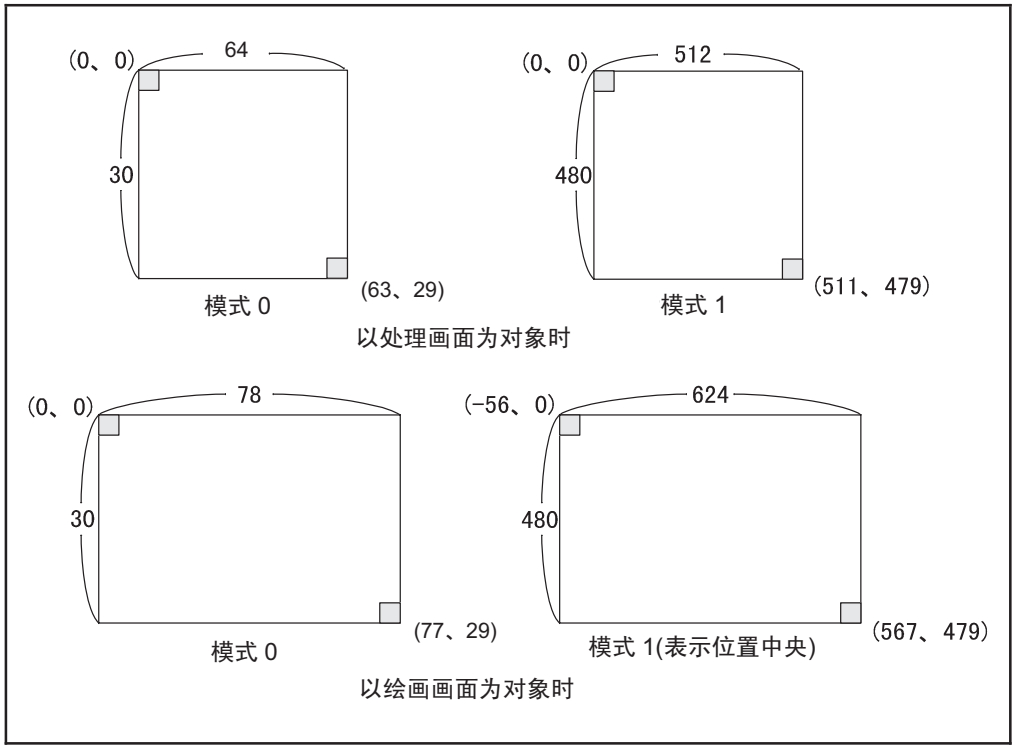
0: 用位、行指定。

1: 用 XY 坐标系指定。

指定的数值没有进行检查。即使超出描绘的范围也不会发生错误。

注意

- 接通电源时，初始设定设为 VISLOC 0， 0， 0。
- 将模式置于 0 时，将描绘字符之后的最终位置进行存储。
- 字符的显示到达右端时，自动换行。
- 本命令在机械手控制器中需要 μVision 卡 （选件）。



相关项目

VISPRINT

应用示例

```
VISPLNOUT 0                          ' 显示存放存储器 0 号 （处理画面）。
VISSCREEN 0,0,1                      ' 在处理画面 0 号上进行即时描绘。
VISCLS 0                             ' 画面的清空
VISBRIGHT 255                        ' 描绘的辉度值的设定
VISLOC 10,10                         ' 指定显示字符的位置。
VISPRINT " 试验模式 0"               '
VISLOC 10,10,1                       ' 指定显示字符的位置。
VISPRINT " 试验模式 1"               '
VISPRINT " 试验模式 1 的继续 "       '
VISPRINT "******** 换行 ********"
```

VISDEFCHAR（语句）

功　能

指定字符的大小和显示方法。

格　式

VISDEFCHAR < 横向大小 >， < 纵向大小 >， < 显示方法 >

说　明

< 横向大小 >

指定字符的横向大小。（1 ～ 4）

< 纵向大小 >

指定字符的纵向大小。（1 ～ 4）

< 显示方法 >

指定字符的显示方法。（0 ～ 3）

0: 白色字符
1: 黑色字符
2: 凸白字符
3: 凸黑字符

注意

- 接通电源时，初始设定指定 VISDEFCHAR 1, 1, 0。
- 字符大小以纵横 16 x 16 字节数为基准。（VISDEFCHAR 1、 1、 *）



VISDEFCHAR 1, 1, ·

VISDEFCHAR 2, 2, ·

- 白色、黑色字符以原来背景不变，只显示字符。
- 凸白字符是以背景为黑、将字符变白，凸黑字符是以背景为白、将字符变黑进行显示，所以即使在背景造成字符难于显现时，也容易看见字符。
- 本命令在机械手控制器中需要 µVision 卡 （选件）。

相关项目

VISPRINT

应用示例

```
VISPLNOUT 0                    ' 显示存放存储器 0 号 （处理画面）。
VISSCREEN 0,0,1                ' 在处理画面 0 号上进行即时描绘。
VISCLS 128                     ' 画面的清空
VISLOC 10,10                   ' 显示位置的设定
VISDEFCHAR 1,1,0               ' 指定字符大小、显示方法。
VISPRINT " 大小 1"             '
VISLOC 10,11                   ' 显示位置的设定
VISDEFCHAR 2,2,1               ' 指定字符大小、显示方法。
VISPRINT " 大小 2"             '
VISLOC 10,13                   ' 显示位置的设定
VISDEFCHAR 2,2,2               ' 指定字符大小、显示方法。
VISPRINT " 凸白字符 "          '
VISDEFCHAR 2,2,3               ' 指定字符大小、显示方法。
VISPRINT " 凸黑字符 "          '
```

VISPRINT （语句）

功　能

在画面上显示字符、数字。

格　式

VISPRINT < 讯息 > ［< 分隔符 > < 讯息 > …］

说　明

< 讯息 >

指定用 "" 指定的字符、变量、常量等。

< 分隔符 >

使用逗号 (，) 、分号 (；)。
逗号 : 将讯息之间用空格空开。
分号 : 不空开讯息之间。

在 < 讯息 > 的最后如果指定逗号或者分号则不进行换行，在该行中继续根据下一个 VISPRINT 的显示。

注意

在显示实数变量时，在画面上显示到小数点以下第四位。

相关项目

VISSCREEN、 VISBRIGHT、 VISLOC、 VISDEFCHAR

应用示例

```
VISPLNOUT 0                                  ’显示存放存储器 0 号 （处理画面）。
VISSCREEN 0,0,1                              ’在处理画面 0 号上进行即时描绘。
VISCLS 0                                     ’画面的清空
VISLOC 10,10                                 ’显示位置的设定
VISDEFCHAR 1,1,0                             ’指定字符大小、显示方法。
I1= 10
F1 = 0.999
VISPRINT " 整数变量 ";I1," 实数变量 F1";F1
```

21.4 图像处理

VISWORKPLN（语句）

功　能

指定处理目标的存放存储器 （处理画面）。

格　式

VISWORKPLN< 存放存储器编号 >

说　明

< 存放存储器编号 >

指定处理目标的存放存储器编号。 (0 ～ 3) 省略时，指定 0。

注意

- 接通电源时，初始设定被设定为 VISWORKPLN 0。
- 本命令在机械手控制器中需要 µVision 卡 （选件）。

相关项目

VISGETP、VISHIST、VISBINA、VISMEASURE、VISPROJ、VISEDGE、VISREADQR、BLOB、SHDEFMODEL、SHMODEL、SHCORNER、SHCIRCLE

应用示例

```
VISWORKPLN 0                    ' 在存放存储器 0 号上指定处理目标。
```

VISGETP（函数）

功　能

从存放存储器 （处理画面）中获取指定坐标的辉度。

格　式

VISGETP(<X 坐标 >， <Y 坐标 >)

说　明

<X 坐标 >

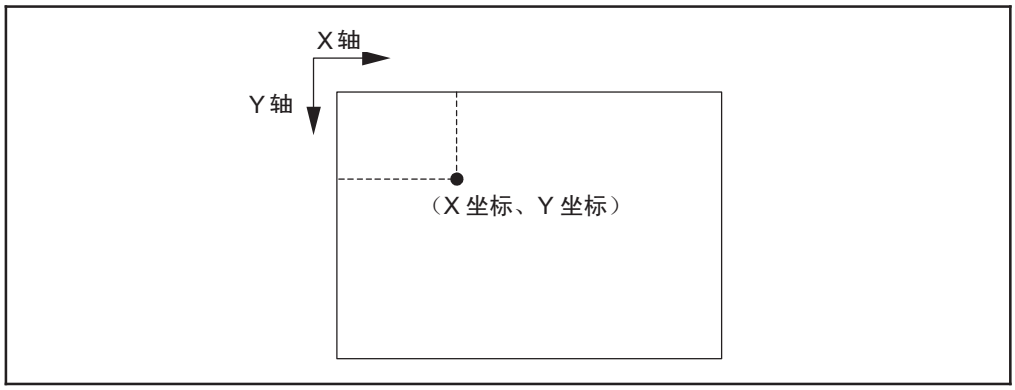
指定 X 坐标。（0 ～ 511）

<Y 坐标 >

指定 Y 坐标。（0 ～ 479）

注意

- 处理目标成为 VISWORKPLN 所指定的画面。
- 本命令在机械手控制器中需要 μVision 卡 （选件）。



相关项目

VISWORKPLN

应用示例

```
VISPLNOUT 0                    ' 显示存放存储器 0 号 （处理画面）。
FOR I1 = 10 TO 200 STEP 5      '
  I2 = VISGETP(100+I1,100)     ' 将 (100 + I1,100) 坐标的辉度值赋值 I2。
  VISLOC 10,10                 ' 显示位置的设定
  VISDEFCHAR 1,1,3             ' 指定显示字符大小、显示方法。
  VISPRINT " 辉度值 = "; I2    '
NEXT I1
```

VISHIST（语句）

功　能

得到画面的直方柱状体 （辉度分布）。

格　式

VISHIST < 视窗编号 >， <X 坐标 >， <Y 坐标 >

说　明

< 视窗编号 >

　指定视窗的编号。（0 ～ 511）

<X 坐标 >

　指定 X 坐标。（0 ～ 511）

<Y 坐标 >

　指定 Y 坐标。（0 ～ 479）

注意

- 在视窗上指定处理范围。
- 指定的视窗的位置超出画面时，执行结果发生错误。
- 所能指定的视窗的形状只是矩形角度为 0°。在其他的视窗情况下，则发生错误。
- 处理目标成为 VISWORKPLN 所指定的画面。
- 处理结果可以用 VISREFHIST 命令读出。
- 处理结果被记忆在存储器中。被保持到再次执行本命令。
- 本命令在机械手控制器中需要 μVision 卡 （选件）。

相关项目

WINDMAKE、 VISWORKPLN、 VISREFHIST

应用示例

```
WINDMAKE R,1,100,100,0,2        ' 将视窗 1 号设定在矩形视窗上。
CAMIN 1                         ' 在存放存储器上获取照相机影像。
VISWORKPLN 0                    ' 将处理的目标图像设定在存放存储器 0 号。
VISHIST 1,100,100               ' 以坐标 (100, 100) 为原点，在视窗 1 号执行直方图。

FOR I1 = 0 TO 255               '
  I2 = VISREFHIST(I1)           ' 将辉度值 I1 的分布数据赋值 I2。
  VISLOC 10,10                  ' 显示位置的设定
  VISDEFCHAR 1,1,3              ' 指定显示字符大小、显示方法。
  VISPRINT " 分布数据 =" ;  I2  '
NEXT I1
```

VISREFHIST（函数）

功　　能

读出直方柱状体结果。

格　　式

VISREFHIST （辉度值）

说　　明

< 辉度值 >

指定读出的直方柱状体的辉度值。（0 ～ 255）

相关项目

VISHIST

应用示例

```
WINDMAKE R,1,100,100,0,2          ' 将视窗 1 号设定在矩形视窗上。
CAMIN 1                           ' 在存放存储器上获取照相机影像。
VISWORKPLN 0                      ' 将处理的目标图像设定在存放存储器 0 号。
VISHIST 1,100,100                 ' 以坐标 (100, 100) 为原点，在视窗 1 号
                                  ' 执行直方图。
FOR I1 = 0 TO 255                 '
  I2 = VISREFHIST(I1)             ' 将辉度值 I1 的分布数据赋值 I2。
  VISLOC 10,10                    ' 显示位置的设定
  VISDEFCHAR 1,1,3                ' 指定显示字符大小、显示方法。
  VISPRINT " 分布数据 =" ;  I2    '
NEXT I1
```

VISLEVEL（函数）

功　能

根据直方柱状体结果求取 2 值化标准值。

格　式

VISLEVEL （< 模式 > [， < 基准面积 > [， < 处理目标 >]]）

说　明

< 模式 >

指定求取 2 值化基准值的方法。（0 ～ 2）

0: 模式法

1: 判别分析法

2: P 瓷砖法

关于内容请阅览 P5-4 "5.1.1.5 2 值化基准值 "。

< 基准面积 >

指定根据 P 瓷砖法求取 2 值化基准值时的面积值。 (1 ～ 245760) 省略时，指定 1。

< 处理目标 >

指定根据 P 瓷砖法求取 2 值化基准值时的目标 （白色、黑色）。（0 或者 1）省略时，指定 0。

0: 黑色 （从辉度值 0 开始累积）

1: 白色 （从辉度值 255 开始累积）

注意

- 在模式法、判别分析法中不使用 < 基准面积 > < 处理目标 >。
- 在使用本功能时，需要预先执行 VISHIST。
- 指定数据有误、或者处理结果异常时，返回值为 -1。
- 本命令在机械手控制器中需要 μVision 卡 （选件）。

相关项目

VISHIST

应用示例

```
VISCLS 0                                ’
WINDMAKE R,1,100,100,0,2                ’将视窗 1 号设定在矩形视窗上。
CAMIN 1                                 ’在存放存储器上获取照相机影像。
VISWORKPLN 0                            ’将处理的目标图像设定在存放存储器 0 号。
VISHIST 1,100,100                       ’以坐标 (100, 100) 为原点，在视窗 1 号执行直方图。

WINDDISP 1                              ’
I2 = VISLEVEL(0)                        ’用模式法求取 2 值化基准值。
VISLOC 10,10                            ’
IF I2 = -1 THEN VISPRINT " 错误 " ELSE VISPRINT "2 值化基准值 = " ； I2
I2 = VISLEVEL(1)                        ’根据判别分析法求取 2 值化基准值。
VISLOC 10,11                            ’
IF I2 = -1 THEN VISPRINT " 错误 " ELSE VISPRINT "2 值化基准值 = " ； I2
I2 = VISLEVEL(2,100,0)                  ’根据 P 瓷砖法求取 2 值化基准值。
VISLOC 10,12                            ’
IF I2 = -1 THEN VISPRINT " 错误 " ELSE VISPRINT "2 值化基准值 = " ； I2
```

VISBINA（语句）

功　能

将画面进行 2 值化处理。

格　式

VISBINA < 视窗编号 >， <X 坐标 >， <Y 坐标 >， <2 值下限 > ［， <2 值上限 >］

说　明

< 视窗编号 >

指定视窗的编号。（0 ～ 511）

<X 坐标 >

指定 X 坐标。（0 ～ 511）

<Y 坐标 >

指定 Y 坐标。（0 ～ 479）

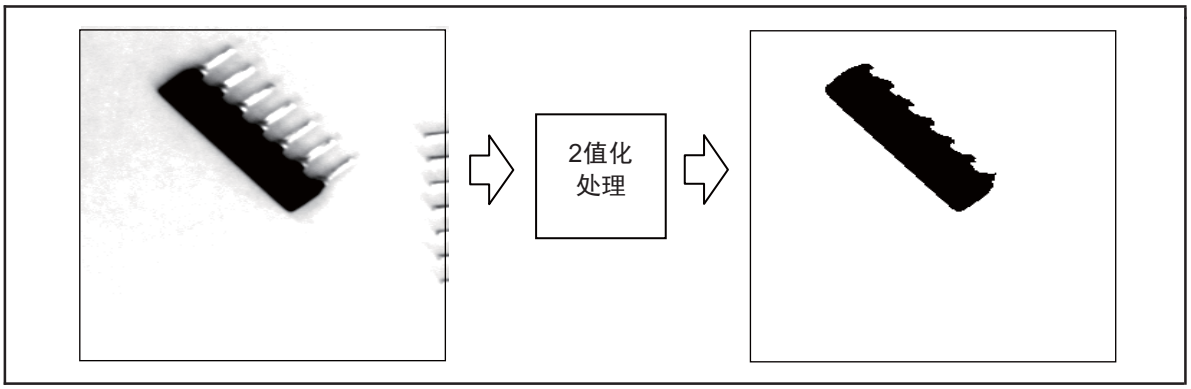
<2 值下限 >

指定 2 值化基准值时的下限等级。（0 ～ 254 下限 < 上限）

<2 值上限 >

指定 2 值化基准值时的上限等级。（1 ～ 255 下限 < 上限）
省略时，指定 255。

2 值化示例



注意

- 在视窗上指定处理范围。
- 指定的视窗的位置超出画面时，执行结果发生错误。
- 所能指定的视窗的形状只是矩形角度为 0°。在其他的视窗情况下，则发生错误。
- 处理目标成为 VISWORKPLN 所指定的画面。
- 本命令在机械手控制器中需要 μVision 卡 （选件）。

相关项目

WINDMAKE、 VISWORKPLN

应用示例

```
VISSCREEN 1,0,1                 ' 在描绘画面 0 号上进行即时描绘。
WINDMAKE R,1,100,100,0,2        ' 将视窗 1 号设定在矩形视窗上。
CAMIN 1                         ' 在存放存储器上获取照相机影像。
VISWORKPLN 0                    ' 将处理的目标图像设定在存放存储器 0 号。
VISPLNOUT 0                     ' 将存放存储器 0 号显示在监视器上。
VISBINA 1,100,100,128,255       ' 将视窗内进行 2 值化。
WINDDISP 1                      ' 描绘视窗。
```

VISBINAR（语句）

功　能

将画面进行 2 值化显示。

格　式

VISBINAR < 模式 > ［， <2 值下限 > ［， 2 值上限］］

说　明

< 模式 >

指定 2 值化基准值显示的模式。（0 或者 1）

0: 中止 2 值化显示，返回原来的显示。

1: 执行 2 值化显示。

<2 值下限 >

指定 2 值化基准值时的下限等级。（0 ～ 254 下限 < 上限）

<2 值上限 >

指定 2 值化基准值时的上限等级。（1 ～ 255 下限 < 上限）

省略时，指定 255。

注意

- 将当前显示在监视器上的全部画面进行 2 值化显示。
- 存放存储器 （处理画面）即使被选择显示，存储的内容也不会改写。
- 只要不将模式置为 0，会继续 2 值化显示。
- 本命令在机械手控制器中需要 μVision 卡 （选件）。

相关项目

VISBINA

应用示例

```
VISBINAR 1,128,255          ' 开始 2 值化显示。
VISCAMOUT 1                 ' 将照相机 1 号显示在监视器上。
DELAY 5000                  ' 停止 5 秒钟
VISPLNOUT 0                 ' 将存放存储器 0 号显示在监视器上。
DELAY 5000                  ' 停止 5 秒钟
VISBINAR 0                  ' 返回原来的显示。
```

VISFILTER（语句）

功　能

对图像进行过滤器处理。

格　式

VISFILTER < 视窗编号 >， <X 坐标 >， <Y 坐标 >， < 处理画面 >， < 存放画面 > ［， < 模式 >］

说　明

< 视窗编号 >

指定视窗的编号。（0 ～ 511）

<X 坐标 >

指定 X 坐标。（0 ～ 511）

<Y 坐标 >

指定 Y 坐标。（0 ～ 479）

< 处理画面 >

指定存放存储器编号。（0 ～ 3）

< 存放画面 >

指定存放过滤器处理结果的存放存储器编号。 (0 ～ 3)

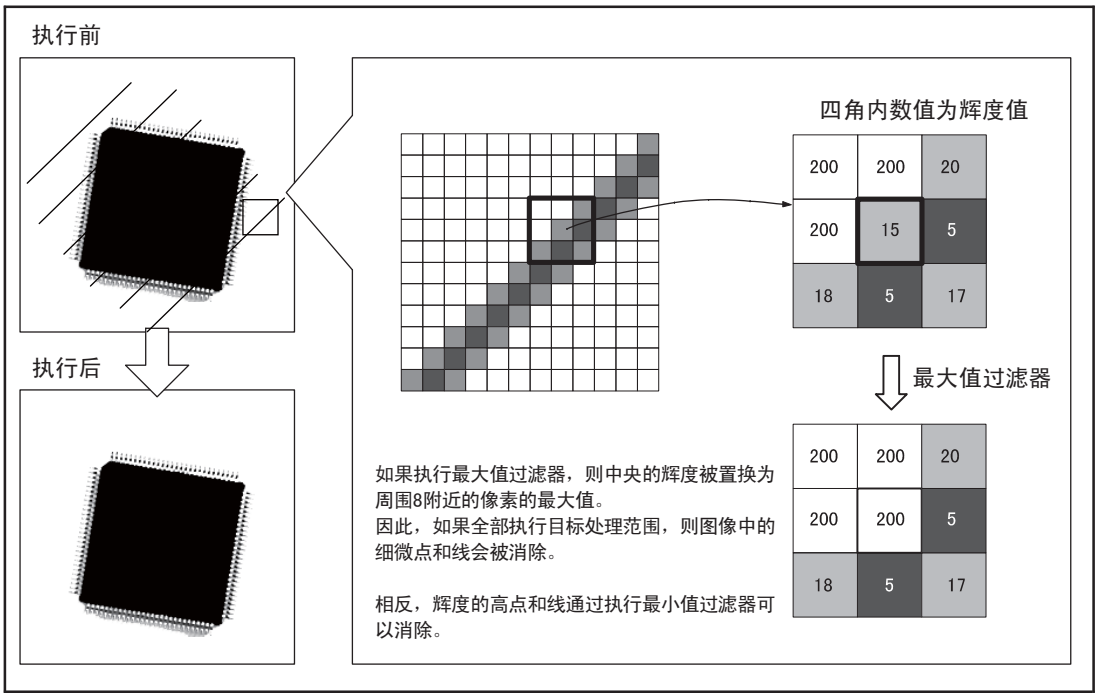
< 模式 >

指定过滤器处理 （ 3 × 3 空间过滤器）的类型。

0: 最小值过滤器

1: 最大值过滤器

省略时，指定 0。



注意

- 处理画面和存放画面的编号相同时，会发生错误。
- 在视窗上指定处理范围。
- 指定的视窗的位置超出画面时，执行结果发生错误。
- 所能指定的视窗的形状只是矩形角度为 0°。在其他的视窗情况下，则发生错误。
- 本命令在机械手控制器中需要 μVision 卡 （选件）。

相关项目

WINDMAKE

应用示例

```
VISSCREEN 1,0,1                 ' 在描绘画面 0 号上进行即时描绘。
WINDMAKE R,1,100,100,0,2        ' 将视窗 1 号设定在矩形视窗上。
CAMIN 1                         ' 在存放存储器上获取照相机影像。
VISPLNOUT 1                     ' 将存放存储器 1 号显示在监视器上。
VISFILTER 1,100,100,0,1,0       '
VISFILTER 1,100,100,1,0,0       '
VISFILTER 1,100,100,0,1,0       '
VISFILTER 1,100,100,1,0,0       '
VISFILTER 1,100,100,0,1,0       '
WINDDISP 1                      ' 描绘视窗。
```

VISMASK （语句）

功　能

进行图像间运算。

格　式

VISMASK＜视窗编号＞，<X 坐标＞，<Y 坐标＞，＜画面 1>，＜画面 2>，＜模式＞[，<2 值下限＞[，<2 值上限＞]]

说　明

＜视窗编号＞

指定视窗的编号。（0 ～ 511）

<X 坐标＞

指定 X 坐标。（0 ～ 511）

<Y 坐标＞

指定 Y 坐标。（0 ～ 479）

＜画面 1>

指定运算的存放存储器编号。（0 ～ 3）

＜画面 2>

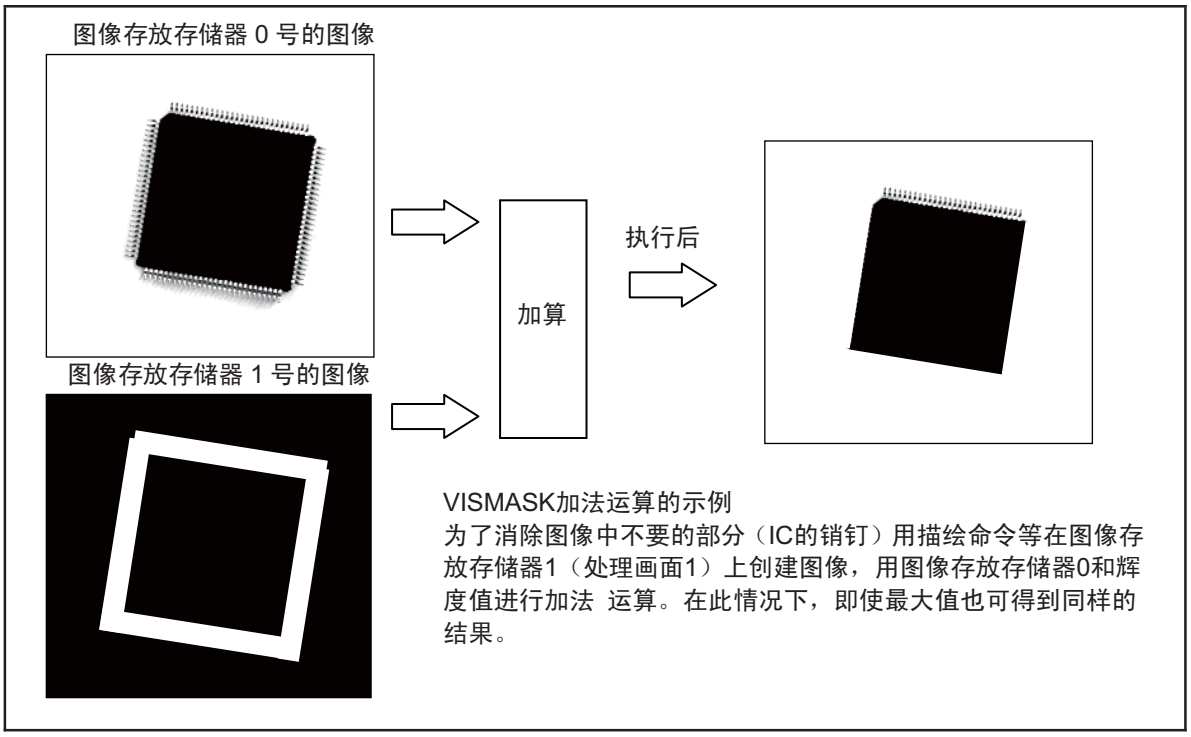
指定运算的存放存储器编号。成为处理结果的存放位置。（0 ～ 3）

＜模式＞

指定图像间运算的类型。（0 ～ 10）

0: 2 值 AND
1: 2 值 OR
2: 2 值 XOR
3: AND （将辉度值 2 进制数化，每个比特位上 AND）
4: OR （将辉度值 2 进制数化，每个比特位上 OR）
5: XOR （将辉度值 2 进制数化，每个比特位上 XOR）
6: 加法运算 （辉度值在 255 以上时，固定在 255）
7: 减法运算 （辉度值在 0 以下时，固定在 0）
8: 最大值 （选择辉度值大的一方）
9: 最小值 （选择辉度值小的一方）

10: 绝对值 （辉度值的差的绝对值）



VISMASK加法运算的示例
为了消除图像中不要的部分（IC的销钉）用描绘命令等在图像存放存储器1（处理画面1）上创建图像，用图像存放存储器0和辉度值进行加法 运算。在此情况下，即使最大值也可得到同样的结果。

注意

- 处理画面和存放画面的编号相同时，会发生错误。
- 在视窗上指定处理范围。
- 指定的视窗的位置超出画面时，执行结果发生错误。
- 所能指定的视窗的形状只是矩形角度为 0°。在其他的视窗情况下，则发生错误。
- 在 2 值化运算中，画面 1 也被 2 值化处理。
- 本命令在机械手控制器中需要 µVision 卡 （选件）。

相关项目

VISBINA

应用示例

```
VISSCREEN 0,0,1                ' 在存放存储器 0 号上进行即时描绘。
WINDMAKE R,1,512,480,0,2       ' 将视窗 1 号设定在矩形视窗上。
CAMIN 1                        ' 在存放存储器上获取照相机影像。
VISCOPY 0,1                    ' 将存放存储器 0 号复制在 1 号上。
VISBRIGHT 255                  ' 将描绘辉度设定为 255。
VISRECT 100,100,100,100,1      ' 在画面上描绘全部涂抹的矩形。
VISPLNOUT 1                    ' 将存放存储器 1 号显示在监视器上。
VISMASK 1,0,0,0,1,10           ' 运算 2 画面之间的差的绝对值。
WINDDISP 1                     ' 描绘视窗。
```

VISCOPY （语句）

功　能

复制画面。

格　式

VISCOPY < 复制原件画面 >， < 复制位置画面 >

说　明

< 复制原件画面 >

指定复制时的复制原件的存放存储器编号。（0 ～ 3）

< 复制位置画面 >

指定复制时的复制位置的存放存储器编号。（0 ～ 3）

注意

- 复制原件和复制位置的编号相同时会发生错误。
- 在画面上显示复制位置的编号时，遵照 VISSCREEN 的描绘模式的指令。
- 本命令在机械手控制器中需要 μVision 卡 （选件）。

相关项目

VISSCREEN

应用示例

```
VISSCREEN 1,0,1          ' 在描绘画面 0 号上进行即时描绘。
CAMIN 1                  ' 在存放存储器上获取照相机影像。
VISCOPY 0,1              ' 将存放存储器 0 号的图像复制在 1 号上。
VISPLNOUT 1              ' 将存放存储器 1 号显示在监视器上。
```

VISMEASURE（语句）

功　能

测量视窗内的特征 （面积、重心、主轴角）。

格　式

VISMEASURE < 视窗编号 >，<X 坐标 >，<Y 坐标 >，< 处理目标 >，< 模式 >，<2 值下限 > [，<2 值上限 >]

说　明

< 视窗编号 >

指定视窗的编号。（0 ～ 511）

<X 坐标 >

指定 X 坐标。 (0 ～ 511) （Ver. 1.6 以上版本 : -16384 ～ 16383）

<Y 坐标 >

指定 Y 坐标。 (0 ～ 479) （Ver. 1.6 以上版本 : -16384 ～ 16383）

注意

指定画面之外时，则发生 " 视窗异常 "。

< 处理目标 >

指定测量的目标。（0 或者 1）

0: 黑色 （辉度值 < 2 值化基准值下限， 2 值化基准值上限 < 辉度值）

1: 白色 （2 值化基准值下限 ≦辉度值 ≦ 2 值化基准值上限）

< 模式 >

指定测量的特征。

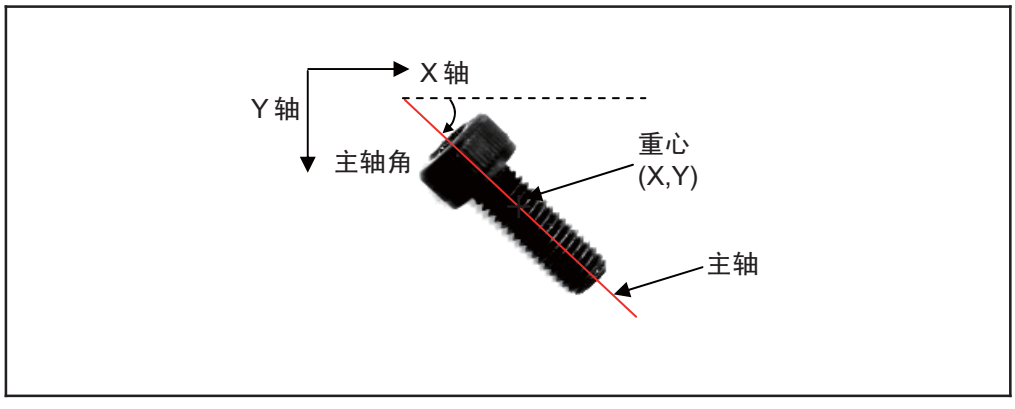
0: 累积辉度、面积

1: 累积辉度、面积、重心 （ 1 次力矩）

2: 累积辉度、面积、重心 （ 1 次力矩）、主轴角 （ 2 次力矩）

注意

- 测量的特征越多，则处理时间越长。
- 在执行本命令时，由于将处理画面 3 号作为工件区域使用，所以处理画面 3 号的数据不被保证。此外，处理画面 3 号 (VISWORKPLN 3) 不能处理。
- 本命令在机械手控制器中需要 µVision 卡 （选件）。



<2 值下限 >

指定 2 值化基准值时的下限等级。（0 ～ 254 下限 < 上限）

<2 值上限 >

指定 2 值化基准值时的上限等级。（1 ～ 255 下限 < 上限）

省略时，指定 255。

注意

- 在视窗上指定处理范围。（只有扇形视窗不能测量主轴角）

		模式		
	○ : 可以 ×: 不可以	0	1	2
直线 （2 点指示）	Windmake P	○	○	○
直线 （长度、角度）	Windmake L	○	○	○
圆	Windmake C	○	○	○
椭圆	Windmake E	○	○	○
扇形	Windmake S	○	○	×
四角	Windmake R	○	○	○

- 指定的视窗的位置超出画面时，执行结果发生错误。
- 以下是用处理结果获取函数可以得到的数据。

	VISSTATUS （n)
n	项目
0	执行结果 0 = 正常 -1 = 异常
1	unknown
2	执行时间

	VISGETNUM （a、 b）
b	a = 0
0	面积
1	重心 X 坐标值
2	重心 Y 坐标值
3	主轴角角度
4	累积辉度
5	1 次惯性矩 X
6	1 次力矩 Y
7	2 次惯性矩 X
8	2 次力矩 Y
9	2 次力矩 XY

- 本命令在机械手控制器中需要 µVision 卡 （选件）。

相关项目

WINDMAKE、 VISWORKPLN、 VISGETNUM、 VISSTATUS

应用示例

```
VISSCREEN 0,0,1                          ' 在存放存储器 0 号上进行即时描绘。
WINDMAKE R,1,512,480,0,2                 ' 将视窗 1 号设定在矩形视窗上。
CAMIN 1                                  ' 在存放存储器 0 号上指定照相机影像。
VISWORKPLN 0                             ' 将目标指定在存放存储器 0 号上。
VISPLNOUT 0                              '
VISMEASURE 1,0,0,1,2,128                 ' 求取累积辉度、面积、重心、主轴角。
IF VISSTATUS(0) = 0.0 THEN               '
  FOR I1 = 0 TO 9                        '
    VISLOC 0,I1                          '
    VISPRINT " 结果 ";I1;"=";VISGETNUM(0,I1)
  NEXT I1                                '
END IF                                   '
```

VISPROJ（语句）

功　能

测量视窗内的投影数据。

格　式

VISPROJ < 视窗编号 >， <X 坐标 >， <Y 坐标 >， < 处理目标 >， <2 值下限 > ［， <2 值上限 >］

说　明

< 视窗编号 >

指定视窗的编号。（0 ～ 511）

<X 坐标 >

指定 X 坐标。 (0 ～ 511) （Ver. 1.6 以上版本 : -16384 ～ 16383）

<Y 坐标 >

指定 Y 坐标。 (0 ～ 479) （Ver. 1.6 以上版本 : -16384 ～ 16383）

注意

指定画面之外时，则发生 " 视窗异常 "。

< 处理目标 >

指定测量的目标。（0 或者 1）

0: 黑色 （辉度值 < 2 值化基准值下限， 2 值化基准值上限 < 辉度值）

1: 白色 （2 值化基准值下限 ≦辉度值 ≦ 2 值化基准值上限）

<2 值下限 >

指定 2 值化基准值时的下限等级。（0 ～ 254 下限 < 上限）

<2 值上限 >

指定 2 值化基准值时的上限等级。（1 ～ 255 下限 < 上限）

省略时，指定 255。

注意

- 处理范围在视窗上指定。
  ○ : 可以 ×: 不可以

直线 （2 点指示）	Windmake P	○
直线 （长度、角度）	Windmake L	○
圆	Windmake C	×
椭圆	Windmake E	×
扇形	Windmake S	○
四角	Windmake R	○

- 指定的视窗的位置超出画面时，执行结果发生错误。

- 以下是用处理结果获取函数可以得到的数据。

	VISSTATUS (n)
n	项目
0	执行结果 0 = 正常 -1 = 异常
1	投影数据个数
2	执行时间

	VISGETNUM （a、 b）
b	a = 0 ～ 511
0	面积
1	unknown
2	unknown
3	unknown
4	辉度积分
5	unknown
6	unknown
7	unknown
8	unknown
9	unknown

- 在执行本命令时，由于将处理画面 3 号作为工件区域使用，所以处理画面 3 号的数据不被保证。此外，处理画面 3 号 (VISWORKPLN 3) 不能处理。
- 本命令在机械手控制器中需要 μVision 卡 （选件）。

相关项目

WINDMAKE、 VISWORKPLN、 VISGETNUM、 VISSTATUS

应用示例

```
VISSCREEN 0,0,1                 ’在存放存储器 0 号上进行即时描绘。
WINDMAKE R,1,100,20,0,1         ’将视窗 1 号设定在矩形视窗上。
CAMIN 1                         ’在存放存储器上获取照相机影像。
VISWORKPLN 0                    ’将目标指定在存放存储器 0 号上。
VISPROJ 1,100,100,1,128         ’测量投影数据。
IF VISSTATUS(0) = 0.0 THEN      ’
  FOR I1 = 0 TO 19              ’
    VISPRINT VISGETNUM(I1,0)    ’
  NEXT I1                       ’
END IF                          ’
```

VISEDGE（语句）

功　能

测量视窗内的边缘。

格　式

VISEDGE < 视窗编号 >， <X 坐标 >， <Y 坐标 >， < 步骤 >， < 处理目标 >
< 等级 > [， < 模式 > [， <2 值下限 > [， <2 值上限 >]]]

说　明

< 视窗编号 >

指定视窗的编号。（0 ～ 511）

<X 坐标 >

指定 X 坐标。 (0 ～ 511) （Ver. 1.6 以上版本 : -16384 ～ 16383）

<Y 坐标 >

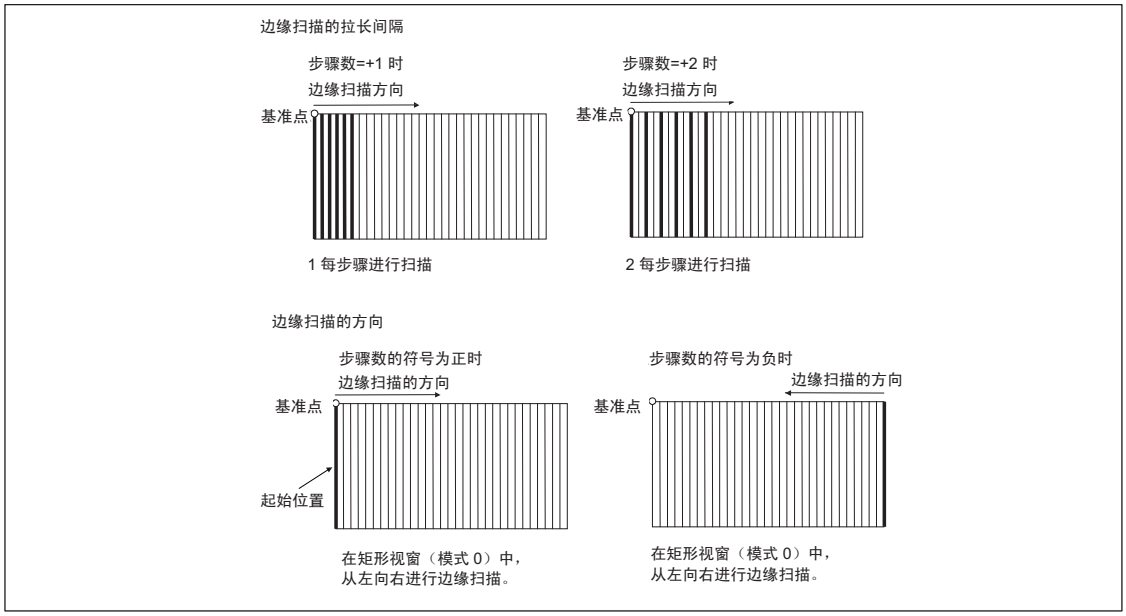
指定 Y 坐标。 (0 ～ 479) （Ver. 1.6 以上版本 : -16384 ～ 16383）

注意

指定画面之外时，则发生 " 视窗异常 "。

< 步骤 >

指定与边缘的扫描方向和拉长的间隔。（-511 ～ 511）



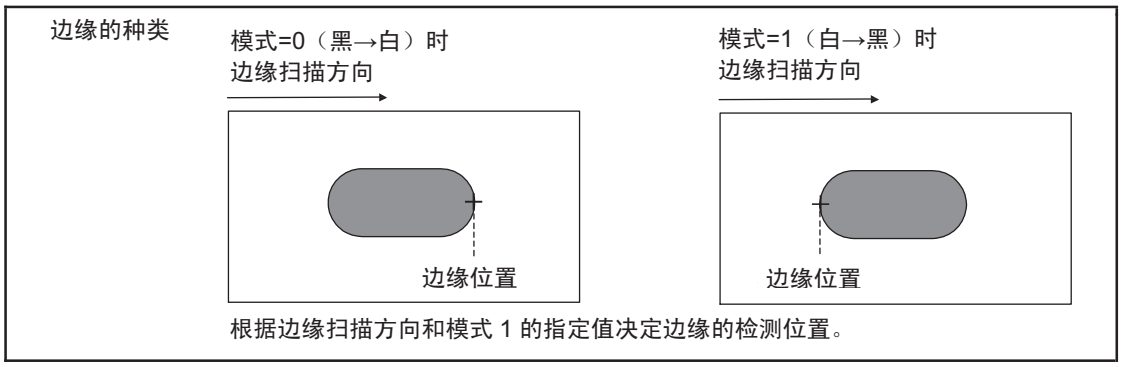
< 处理目标 >

指定测量的目标。（0 ～ 2）

0: 从黑色到白色的变化点 （边缘）

1: 从白色到黑色的变化点

2: 变化点 （0、 1 的两个边缘）



< 等级 >

指定检测边缘的等级。（0 ～ 512）

< 模式 >

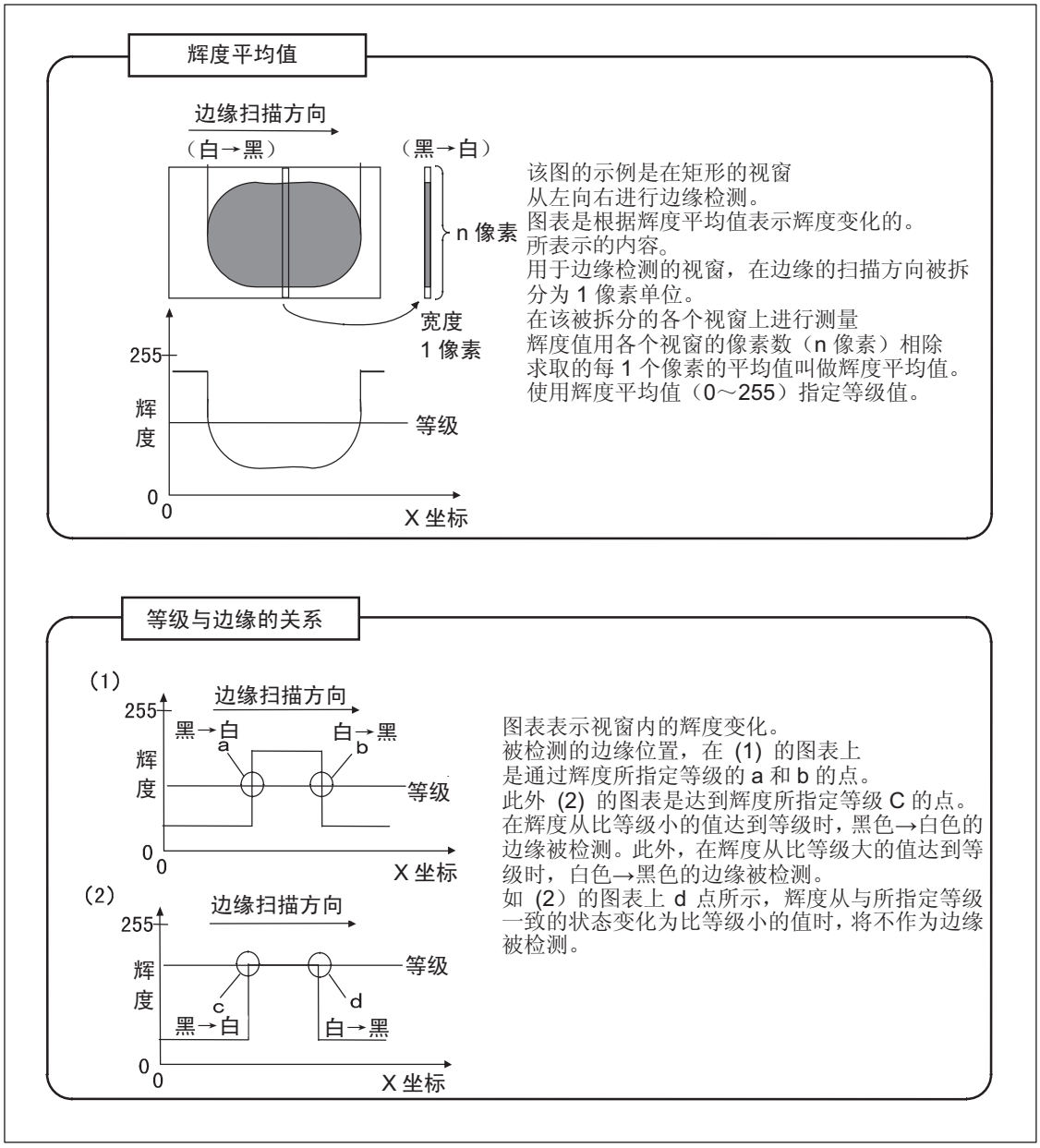
指定边缘的检测方法。省略时 0 被指定。

0: 平均辉度的绝对值

1: 平均辉度的差分值

2: 面积的绝对值

3: 面积的差分值



<2 值下限 >

指定 2 值化基准值时的下限等级。（0 ～ 254 下限 < 上限）
省略时 0 被指定。

<2 值上限 >

指定 2 值化基准值时的上限等级。（1 ～ 255 下限 < 上限）
省略时，指定 255。

注意

- 在视窗上指定处理范围。
  ○ : 可以 ×: 不可以

直线 （2 点指示）	Windmake P	○
直线 （长度、角度）	Windmake L	○
圆	Windmake C	×
椭圆	Windmake E	×
扇形	Windmake S	○
四角	Windmake R	○

- 指定的视窗的位置超出画面时，执行结果发生错误。
- 以下是用处理结果获取函数可以得到的数据。

	VISSTATUS (n)
n	项目
0	执行结果 0 = 正常 -1 = 异常
1	边缘检测数
2	执行时间

	VISGETNUM （a、 b）
b	a = 0 ～ 511
0	Unknown
1	X 坐标值
2	Y 坐标值
3	角度 （*）
4	Unknown
5	Unknown
6	Unknown
7	Unknown
8	Unknown
9	Unknown

* 只有用扇形 （模式 = 0）指定处理范围时，可以取得测量结果 （角度）。

- 处理画面成为用 VISWORKPLN 确定的画面。
- 本命令在机械手控制器中需要 μVision 卡 （选件）。
- 增大步骤数时，检测时间被缩短，检测分解能降低。
- 在扇形视窗上设定 360° 以上时，进行 360° 以上的部分的解析。

相关项目

WINDMAKE、 VISWORKPLN、 VISGETNUM、 VISSTATUS

应用示例

```
VISSCREEN 1,0,1                    ' 在描绘画面 0 号上进行即时描绘。
VISPLNOUT 0                        '
VISCLS 0                           '
WINDMAKE R,1,300,20,0,0            ' 将视窗 1 号设定在矩形视窗上。
CAMIN 2                            ' 在存放存储器上获取照相机影像。
VISWORKPLN 0                       ' 将目标指定在存放存储器 0 号上。
VISPLNOUT 0                        '
VISEDGE 1,100,100,1,0,128          ' 测量边缘。
WINDDISP 1                         '
I1 = VISSTATUS(0)                  '
IF I1 = 0 THEN                     '
  FOR I1 = 0 TO VISSTATUS(1)-1
      VISCROSS VISPOSX(I1),VISPOSY(I1)
  NEXT I1                          '
  I1 = VISSTATUS(1)                '
  IF I1 = 0 THEN                   '
    VISLOC 10,10                   '
    VISPRINT " 没有发现边缘。 "
  END IF                           '
ELSEIF I1 <> 0 THEN                '
  VISLOC 10,10                     '
  VISPRINT " 不能进行测量。 "      '
END IF                             '
```

21.5　编码识别

VISREADQR（语句）

功　能

读取 QR 编码。

格　式

VISREADQR < 视窗编号 >， <X 坐标 >， <Y 坐标 >， < 模式 >，［， <2 值下限 >　［， <2 值上限 >］］

说　明

< 视窗编号 >

指定视窗的编号。（0 ～ 511）

<X 坐标 >

指定 X 坐标。（0 ～ 511）

<Y 坐标 >

指定 Y 坐标。（0 ～ 479）

< 模式 >

指定 2 值化的方法。

0: 自动 2 值化 （用判别分析法将处理范围内进行 2 值化。）

1: 用指定值 2 值化基准值 （根据命令中的 2 值化标准值进行 2 值化。）

<2 值下限 >

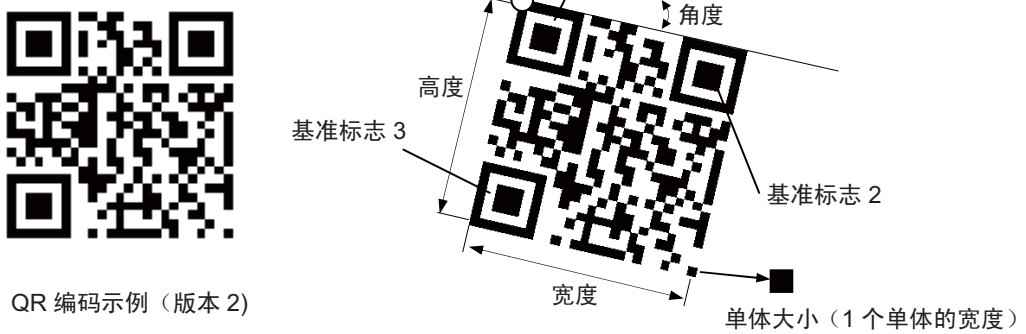
指定 2 值化基准值时的下限等级。（0 ～ 254 下限 < 上限）

省略时，指定 0。

<2 值上限 >

指定 2 值化基准值时的上限等级。（1 ～ 255 下限 < 上限）

省略时，指定 255。



注意

- 在视窗上指定处理范围。
- 所能指定的视窗的形状只是矩形角度为　0°。其他的视窗或者指定的视窗的位置超出画面时就会发生错误。
- 处理目标成为 VISWORKPLN 所指定的画面。
- 编码的内容被存放在板内的存储器上，通过 VISGETSTR 获取。
- 编码中有错误等不能读取时，VISSTATUS (0) 的结果为 -1。
- 以下是用处理结果获取函数可以得到的数据。

	VISSTATUS (n)
n	项目
0	执行结果 0 = 正常 -1 = 异常
1	字符数 （字节 (byte) 单位）
2	执行时间

	VISGETNUM （a、 b）			
b	a = 0	a = 1	a = 2	a = 3
0	字符数 （字节 (byte) 单位）	unknown	unknown	unknown
1	X 坐标值	基准标志 1X 坐标值	基准标志 2X 坐标值	基准标志 3X 坐标值
2	Y 坐标值	基准标志 1Y 坐标值	基准标志 2Y 坐标值	基准标志 3Y 坐标值
3	角度	unknown	unknown	unknown
4	宽度	unknown	unknown	unknown
5	高度	unknown	unknown	unknown
6	版本	unknown	unknown	unknown
7	单体大小	unknown	unknown	unknown
8	模式识别 ［V1.4 以上版本］	unknown	unknown	unknown
9	unknown	unknown	unknown	unknown

- 本命令在机械手控制器中需要 µVision 卡 （选件）。

相关项目

WINDMAKE、 VISWORKPLN、 VISGETNUM、 VISSTATUS、 VISGETSTR

应用示例

```
VISSCREEN 1,0,1                    ’
VISCLS                             ’
WINDMAKE R,1,512,480,0,2           ’将视窗 1 号设定在矩形视窗上。
CAMIN 1                            ’在存放存储器上获取照相机影像。
VISPLNOUT 0                        ’
VISREADQR 1,0,0,0                  ’QR 编码的读取
I1 = VISSTATUS(0)                  ’
VISPRINT I1,VISSTATUS(1)           ’
IF I1 = 0 THEN                     ’
  VISLOC 10,10                     ’
  VISPRINT VISGETSTR(1,VISSTATUS(1))
END IF                             ’
VISCAMOUT 1                        ’
```

21.6 标注

BLOB （语句）

功　能

执行标注。

格　式

BLOB < 视窗编号 >，<X 坐标 >，<Y 坐标 >，< 处理目标 >，<2 值下限 > [，<2 值上限 > [，< 连接 > [，< 面积下限 > [，< 分类 >]]]]

说　明

< 视窗编号 >

指定视窗的编号。（0 ～ 511）

<X 坐标 >

指定 X 坐标。（0 ～ 511）

<Y 坐标 >

指定 Y 坐标。（0 ～ 479）

< 处理目标 >

指定用标注所求取的目标。（0 或者 1）

0: 黑色 （辉度值 < 2 值化基准值下限， 2 值化基准值上限 < 辉度值）

1: 白色 （2 值化基准值下限 ≦辉度值 ≦ 2 值化基准值上限）

<2 值下限 >

指定 2 值化基准值时的下限等级。（0 ～ 254 下限 < 上限）

<2 值上限 >

指定 2 值化基准值时的上限等级。（1 ～ 255 下限 < 上限）

省略时，指定 255。

< 连接 >

指定连接的条件。（0 或者 1）

0: 连接 4 附近 （检查左右、上下的相邻像素的状态）

1: 连接 8 附近 （检查左右、上下、也包括斜向的相邻像素的状态）

< 面积下限 >

指定标注时所忽视的面积值的下限。（0 ～ 245760）

< 分类 >

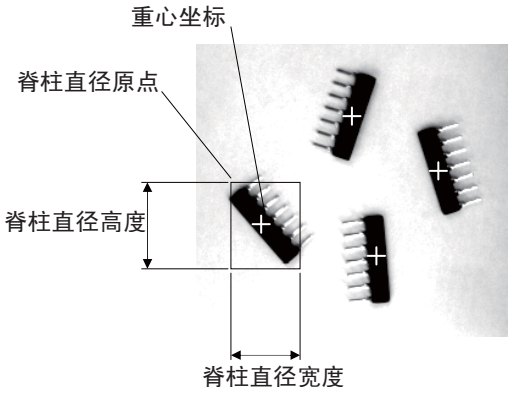
指定用标注所求取的编号的排列替换。（0 ～ 2）

0: 可以求取的顺序

1: 面积值降序

2: 面积值升序

标注的事例



通过执行标注，可以求取多个物体的数量、位置等。

在左边的示例中，标注数为4。

此外，用BLOBMEASURE更可以单独求取物体的倾斜（方向：主轴角）和周长。

注意

- 在视窗上指定处理范围。
- 指定的视窗的位置超出画面时，执行结果发生错误。
- 所能指定的视窗的形状只是矩形角度为 0°。在其他的视窗情况下，则发生错误。
- 处理目标成为 VISWORKPLN 所指定的画面。
- 以下是用处理结果获取函数可以得到的数据。

	VISSTATUS (n)
n	项目
0	执行结果 0 = 正常 -1 = 异常
1	标签数
2	执行时间

	VISGETNUM （a、 b）
b	a = 0 ～ max511
0	面积
1	重心 X 坐标值
2	重心 Y 坐标值
3	unknown
4	脊柱直径原点 Y 坐标
5	脊柱直径原点 X 坐标
6	脊柱直径宽度
7	脊柱直径高度
8	unknown
9	unknown

- 在指定时间内没有结束标注时，则会造成错误。
- 本命令在机械手控制器中需要 μVision 卡 （选件）。

相关项目

WINDMAKE、 VISWORKPLN、 VISGETNUM、 VISSTATUS

应用示例

```
VISSCREEN 1,0,1                    ' 在描绘画面 0 号上进行即时描绘。
VISCLS 0                           '
WINDMAKE R,1,512,480,0,2           ' 将视窗 1 号设定在矩形视窗上。
CAMIN 1                            ' 在存放存储器 0 号上指定照相机影像。
VISPLNOUT 0                        '
VISWORKPLN 0                       ' 将目标指定在存放存储器 0 号上。
BLOB 1,0,0,0,128                   ' 执行标注。
I1 = VISSTATUS(0)                  '
IF I1 = 0 THEN                     '
  I2 = VISSTATUS(1)                '
  VISDEFCHAR 1,1,2                 '
  VISLOC 10,10                     '
  VISPRINT I1,I2                   '
  IF I2 <> 0 THEN                  '
    FOR I1 = 0 TO I2 -1            '
      VISLOC 10,11                 '
      VISPRINT VISGETNUM(I1,1),VISGETNUM(I1,2)
      VISCROSS VISGETNUM(I1,1),VISGETNUM(I1,2)
    NEXT I1                        '
  END IF                           '
END IF                             '
VISCAMOUT 1                        '
```

BLOBMEASURE（语句）

功　能

进行目标标签编号的特征测量。

格　式

BLOBMEASURE < 标签编号 > , < 特征 >

说　明

< 标签编号 >

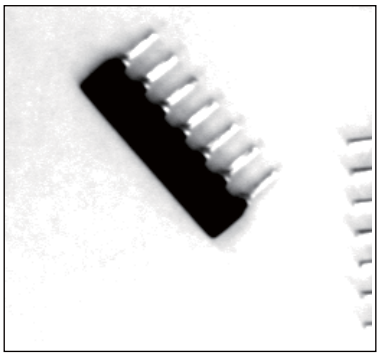
指定标注所求取的标签编号。 (0 ～ 511)

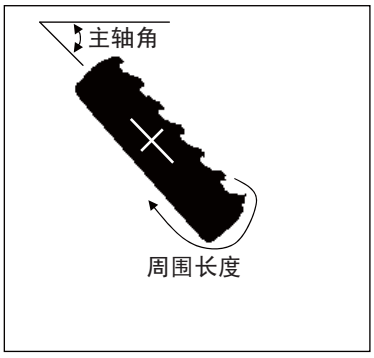
< 特征 >

指定想要求取的特征。（0 或者 1）

0: 主轴角度

1: 周围长度





2值化基准值不改变原图像而是在内部自动运行。
不需要预先进行2值化处理。

注意

- 在执行本函数之前，需要用 BLOB 预先进行标注。
- 求取周长时，需要保留执行标注时的原图像。
- 以下是用处理结果获取函数可以得到的数据。

	VISSTATUS (n)
n	项目
0	执行结果 0 = 正常 -1 = 异常
1	unknown
2	执行时间

	VISGETNUM （a、 b）
b	a = 0 ～ max511
0	面积
1	重心 X 坐标值
2	重心 Y 坐标值
3	主轴角角度
4	脊柱直径原点 X 坐标
5	脊柱直径原点 Y 坐标
6	脊柱直径宽度
7	脊柱直径高度
8	周围长度
9	unknown

- 本命令在机械手控制器中需要 μVision 卡 （选件）。

相关项目

BLOB

应用示例

```
VISSCREEN 1,0,1                          ' 在描绘画面 0 号上进行即时描绘。
WINDMAKE R,1,512,480,0,2                 ' 将视窗 1 号设定在矩形视窗上。
CAMIN 1                                  ' 在存放存储器上获取照相机影像。
BLOB 1,0,0,0,128                         ' 执行标注。
IF VISSTATUS(0)=0.0 THEN                 '
  IF VISSTATUS(1)<>0.0 THEN              '
    FOR I1 = 0 TO VISSTATUS(1)-1         '
      BLOBMEASURE I1,0                   ' 求取 I1 标签编号的主轴角度。
      VISCROSS VISGETNUM(I1,1),VISGETNUM(I1,2),10,20,VISGETNUM(I1,3)
    NEXT I1                              '
  END IF                                 '
END IF                                   '
```

BLOBLABEL（语句）

功　能

获取指定坐标的标签编号。

格　式

BLOBLABEL （<X 坐标 >， <Y 坐标 >）

说　明

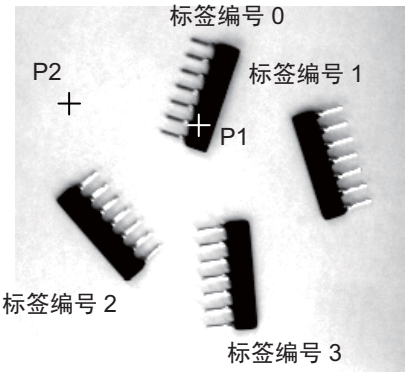
<X 坐标 >

指定 X 坐标。（0 ～ 511）

<Y 坐标 >

指定 Y 坐标。（0 ～ 479）

标签编号的获取示例



BLOBLABEL确认指定坐标是否被包含在各个标签的物体内。
标注数4、标签编号如左图所示对于被测量的图像，用BLOBLABEL指定P1的坐标时，返回值为标签编号的0。
指定P2时，由于没有相应的物体，所以返回值为-1。

注意

- 在执行本函数之前，需要用 BLOB 预先进行标注。
- 获取在指定的坐标上所存在的标签编号。
- 标签编号不存在时返回 -1。
- 本命令在机械手控制器中需要 μVision 卡 （选件）。

相关项目

BLOB

应用示例

```
VISSCREEN 1,0,1                          ’在描绘画面 0 号上进行即时描绘。
WINDMAKE R,1,512,480,0,2                 ’将视窗 1 号设定在矩形视窗上。
CAMIN 1                                  ’在存放存储器上获取照相机影像。
BLOB 1,0,0,0,128                         ’执行标注。
IF VISSTATUS(0)=0.0 THEN                 ’
  IF VISSTATUS(1)<>0.0 THEN              ’
    IF BLOBLABEL(100,100)<>-1 THEN
      VISLOC 100,100,1                   ’
      VISPRINT " 标签编号 =" ;  BLOBLABEL (100,100)
    END IF                               ’
  END IF                                 ’
END IF                                   ’
```

BLOBCOPY（语句）

功　能

复制目标标签编号。

格　式

BLOBCOPY < 标签编号 >， < 复制位置画面 >， <X 坐标 >， <Y 坐标 >

说　明

< 标签编号 >

指定标注所求取的标签编号。（0 ～ 511）

< 复制位置画面 >

指定复制位置的存放存储器编号。（0 ～ 3）

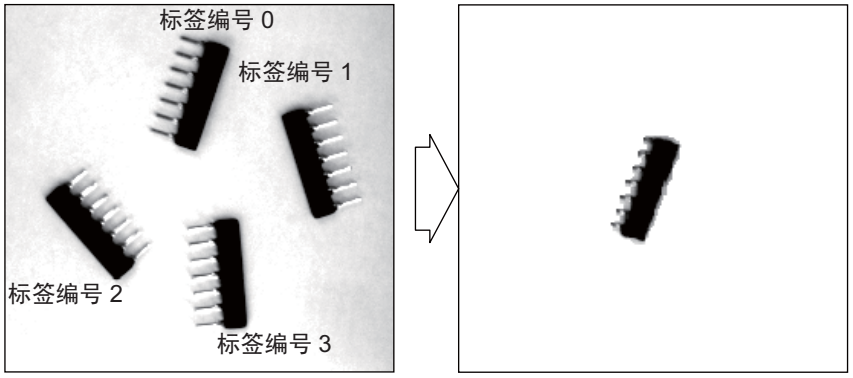
<X 坐标 >

指定复制位置的原点 X 坐标。（0 ～ 511）

<Y 坐标 >

指定复制位置的原点 Y 坐标。（0 ～ 479）

BLOBCOPY 的示例



复制指定的标签编号的物体图像。
左边的示例是将标签编号0复制在处理画面1上的结果。

注意

- 在执行本函数之前，需要用 BLOB 预先进行标注。
- 因为复制时是参照原画面 （标注目标画面），所以请不要变更画面。
- 本命令在机械手控制器中需要 μVision 卡 （选件）。

相关项目

BLOB

应用示例

```
VISSCREEN 0,1,1                 ' 在存放存储器 1 号上进行即时描绘。
VISCLS 128                      ' 清空画面。
VISSCREEN 1,0,1                 ' 在描绘画面 0 号上进行即时描绘。
VISCLS 0                        '
WINDMAKE R,1,512,480,0,2        ' 将视窗 1 号设定在矩形视窗上。
CAMIN 1                         ' 在存放存储器上获取照相机影像。
BLOB 1,0,0,0,128                ' 执行标注。
I1 = VISSTATUS(0)               '
IF I1 = 0 THEN                  '
  I1 = VISSTATUS(1)             '
  IF I1<>0 THEN                 '
    BLOBCOPY 0,1,100,100        ' 将标签 0 号复制在存放存储器 1 号上。
  END IF                        '
END IF                          '
VISPLNOUT 0                     '
DELAY 2000                      '
VISPLNOUT 1                     '
DELAY 2000                      '
VISCAMOUT 1                     '
```

21.7　检索功能

SHDEFMODEL（语句）

功　能

进行检索模型的登录。

格　式

SHDEFMODEL < 模型编号 >， <X 坐标 >， <Y 坐标 >， < 宽度 >， < 高度 >，< 偏移 X>，
< 偏移 Y> ［， < 偏移角度 >］

说　明

< 模型编号 >

指定登录的模型的编号。（0 ～ 99）

<X 坐标 >

指定登录模型的原点 X 坐标。（16 ～ 485）

<Y 坐标 >

指定登录模型的原点 Y 坐标。（16 ～ 463）

< 宽度 >

指定登录模型的宽度。（10 ～ 256）

< 高度 >

指定登录模型的高度。（10 ～ 255）

< 偏移 X>

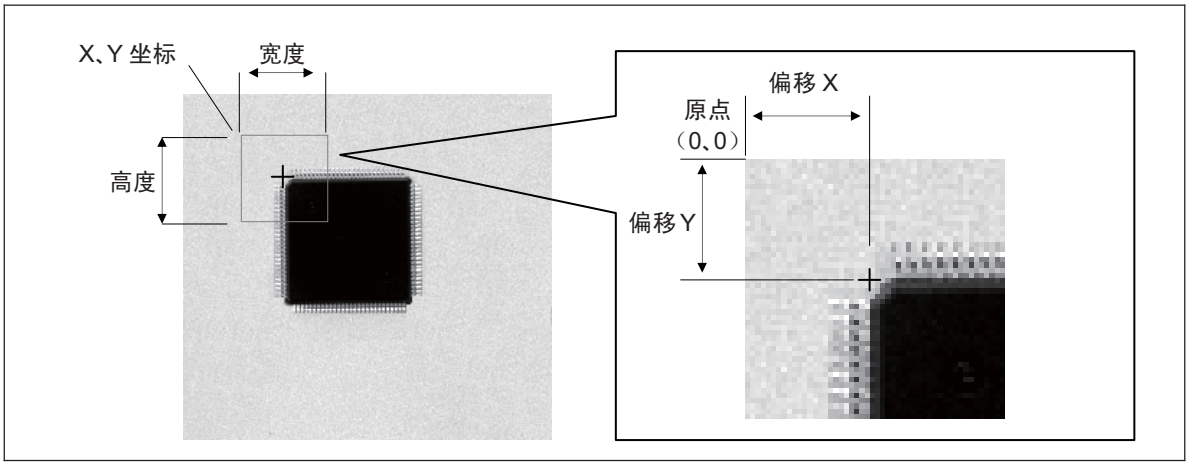
指定来自原点的偏移 X。（-511 ～ +511）

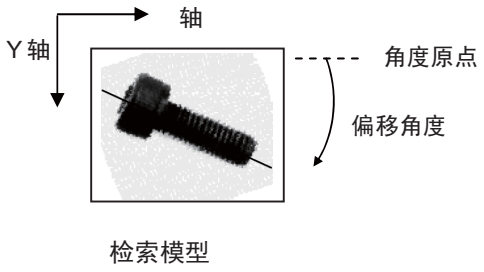
< 偏移 Y>

指定来自原点的偏移 Y。（-511 ～ +511）

< 偏移角度 >

指定来自角度原点的偏移角度。（-360 ～ 360）［V1.4 以上版本］





在［偏移角度］上指定来自角度原点的偏移值。如图所示，登录在检索模型上的目标对角度原点如果在有偏移时指定［偏移角度］，则在 [SHMODEL] 测量结果［角度］上能够反映偏移值。

注意

- 登录模型如果不向内侧离开画面端 16 像素以上，就不能登录。
- 基准坐标是指在模型检索时将查出点指定设立在何处。
- 登录模型需要一定程度的辉度分布和特征。辉度分布过于平坦的和细微变化过多时，不能登录。
- 指定已经被登录的编号时，要进行覆盖。
- 命令执行之后，可以通过 VISSTATUS 知道可否登录。
- 在执行本命令时，由于将处理画面 3 号作为工件区域使用，所以处理画面 3 号的数据不被保证。此外，处理画面 3 号 (VISWORKPLN 3) 不能处理。

	VISSTATUS (n)
n	项目
0	执行结果 0 = 正常 -1 = 指定范围错误 -2 = 全部清除 -3 = 均一型号 -4 = 复杂 1, 2 = 检索时间大 3, 4 = 角度特定不可 ［ V1.4 以上版本］
1	未使用 0
2	执行时间

- 本命令在机械手控制器中需要 µVision 卡 （选件）。
- 执行本命令时，请勿将电源 OFF。判断下一步电源 ON 时没有正常结束，将与视觉有关的信息进行初始化。

相关项目

SHREFMODEL、 SHDISPMODEL、 SHCLRMODEL、 SHMODEL

应用示例

```
VISSCREEN 1,0,1                          ' 在描绘画面 0 号上进行即时描绘。
VISCLS 0                                 '
VISRECT 100,100,100,100                  '
CAMIN 1                                  ' 在存放存储器上获取照相机影像。
SHDEFMODEL 1,100,100,100,100,50,50       ' 登录模型。
I1 = VISSTATUS(0)                        '
VISLOC 10,10                             '
IF I1 = 0 THEN VISPRINT " 登录结束 " ELSE VISPRINT " 不能登录。 "
VISCAMOUT 1                              '
```

SHREFMODEL（语句）

功　能

参照登录模型数据。

格　式

SHREFMODEL （< 模型编号 >，< 项目 >）

说　明

< 模型编号 >

指定参照模型编号。（0 ～ 99）

< 项目 >

指定参照模型的数据类型。（0 ～ 8）

0: 状态 （有没有模型的登录有 = 0 没有 = -1）
1: 登录模型的宽度
2: 登录模型的高度
3: 登录模型的偏移 X
4: 登录模型的偏移 Y
5: 登录模型的文件大小
6: 可以登录的文件大小 （剩余容量）
7: 登录模型的偏移角度 ［ V1.4 以上版本］
8: 互换模式（ 1: 没有角度测量用、2: 有角度测量用、3: 没有角度测量 + 有角度测量用）［V1.4 以上版本］

相关项目

SHDEFMODEL

应用示例

```
VISSCREEN 1,0,1                          ’
VISCLS 0                                 ’
IF SHREFMODEL(1,0) = 0 THEN              ’ 确认是否有登录。
  VISLOC 10,1                            ’
  VISPRINT " 宽度 =" ;  SHREFMODEL (1,1) ’ 显示宽度。
  VISLOC 10,2                            ’
  VISPRINT " 高度 =" ;  SHREFMODEL (1,2) ’ 显示高度。
  VISLOC 10,3                            ’
  VISPRINT "X=";SHREFMODEL(1,3)          ’ 显示基准 X 坐标。
  VISLOC 10,4                            ’
  VISPRINT "Y=";SHREFMODEL(1,4)          ’ 显示基准 Y 坐标。
  VISLOC 10,5                            ’
  VISPRINT " 大小 =" ;  SHREFMODEL (1,5) ’ 显示文件大小。
  VISLOC 10,6                            ’
  VISPRINT " 剩余 =" ;  SHREFMODEL (1,6) ’ 显示剩余容量。
END IF                                   ’
```

SHCOPYMODEL（语句）

功　能

复制登录模型。

格　式

SHCOPYMODEL < 复制原件模型编号 >， < 复制位置模型编号 >

说　明

< 复制原件模型编号 >

指定复制的模型的编号。（0 ～ 99）

< 复制位置模型编号 >

指定新模型编号。（0 ～ 99）

注意

- 复制原件和复制位置编号相同时，则发生错误。
- 没有复制原件的文件时，或者剩余容量不足时，不进行复制。
- 以下是用处理结果获取函数可以得到的数据。

	VISSTATUS (n)
n	项目
0	执行结果 0 = 正常 -1 = 异常
1	unknown
2	unknown

- 本命令在机械手控制器中需要 μVision 卡 （选件）。
- 执行本命令时，请勿将电源 OFF。判断下一步电源 ON 时没有正常结束，将与视觉有关的信息进行初始化。

相关项目

SHDEFMODEL、 SHREFMODEL

应用示例

```
VISSCREEN 1,0,1                 '
VISCLS 0                        '
SHCOPYMODEL 2,0                 ' 将模型 2 号的信息复制在 0 号上。
VISLOC 10,10                    '
I1 = VISSTATUS(0)               '
IF I1 = 0 THEN VISPRINT " 复制结束 " ELSE VISPRINT " 复制失败 "
```

SHCLRMODEL（语句）

功　能

消除登录模型。

格　式

SHCLRMODEL < 模型编号 >

说　明

< 模型编号 >

指定消除的模型编号。（0 ～ 99）

注意

- 在不存在指定的模型编号时，不进行任何动作。
- 本命令在机械手控制器中需要 μVision 卡 （选件）。
- 执行本命令时，请勿将电源 OFF。判断下一步电源 ON 时没有正常结束，将与视觉有关的信息进行初始化。

相关项目

SHDEFMODEL、 SHREFMODEL

应用示例

```
VISSCREEN 1,0,1                    ’
VISCLS 0                           ’
SHCLRMODEL 1                       ’ 消除登录模型 1 号。
I1 = SHREFMODEL(1,0)               ’
VISLOC 10,10                       ’
IF I1 = -1 THEN VISPRINT " 消除结束 " ELSE VISPRINT " 消除失败 "
```

SHDISPMODEL（语句）

功　能

将登录模型显示在画面上。

格　式

SHDISPMODEL < 模型编号 >， < 显示位置画面 >， <X 坐标 >， <Y 坐标 >

说　明

< 模型编号 >

指定显示的模型编号。（0 ～ 99）

< 显示位置画面 >

指定显示时的存放存储器编号。（0 ～ 3）

<X 坐标 >

指定登录模型的原点 X 坐标。（0 ～ 511）

<Y 坐标 >

指定登录模型的原点 Y 坐标。（0 ～ 479）

注意

- 在不存在指定的模型编号时，不进行任何动作。
- 本命令在机械手控制器中需要 µVision 卡 （选件）。

相关项目

SHDEFMODEL

应用示例

```
VISSCREEN 0,0,1                 ' 在存放存储器 0 号上进行即时描绘。
VISPLNOUT 0                     ' 将存放存储器 0 号显示在监视器上。
SHDISPMODEL 1,0,100,100         ' 将模型 1 号显示在画面上。
```

SHMODEL（语句）

功　能

检索模型。

格　式

SHMODEL < 视窗编号 >， <X 坐标 >， <Y 坐标 >,< 模型编号 >， < 一致度 >
[，< 模式 > [，< 个数 > [，< 起始角度 > [，< 结束角度 >]]]]

说　明

< 视窗编号 >

指定视窗的编号。（0 ～ 511）

<X 坐标 >

指定 X 坐标。（0 ～ 511）

<Y 坐标 >

指定 Y 坐标。（0 ～ 479）

< 模型编号 >

指定检索的编号编号。（0 ～ 99）

< 一致度 >

指定检索的模型的一致度。（0 ～ 100）

< 模式 >

指定检索时的模式。

0: 像素单位的检索

1: 辅助像素单位的检索

省略时，指定 0。

< 个数 >

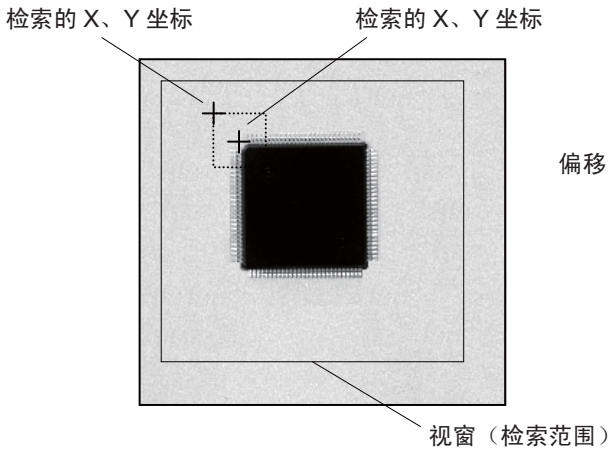
指定一块儿检索的个数。 (1 ～ 49) 省略时，指定 1。

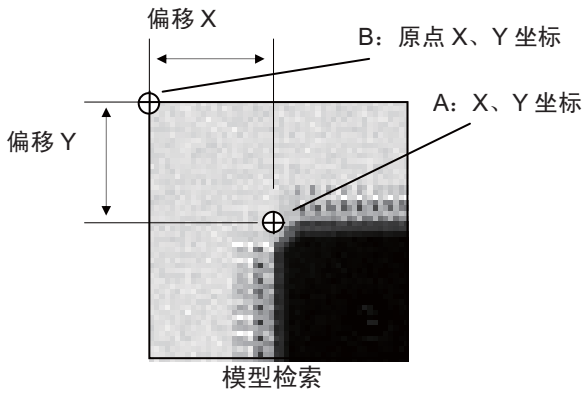
< 起始角度 >

指定检索的目标的旋转范围 （起始角度）。 (-360 ～ 360) 省略时，指定 0。［V1.4 以上版本］

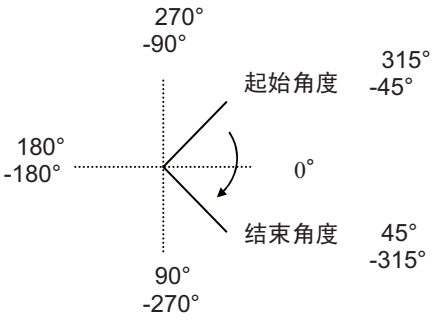
< 结束角度 >

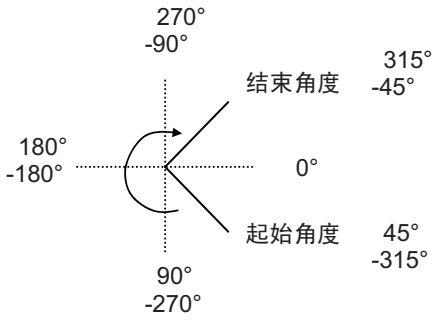
指定检索目标的旋转范围 （结束角度）。 (-360 ～ 360) 省略时，指定 0。［V1.4 以上版本］





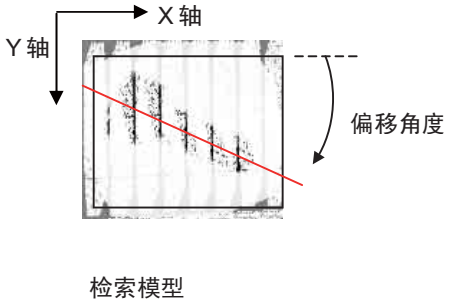
测量结果将检索模型的A位置存放在［X, Y坐标值］。此外，将检索模型的原点B的位置存放在［原点X, Y坐标值］。

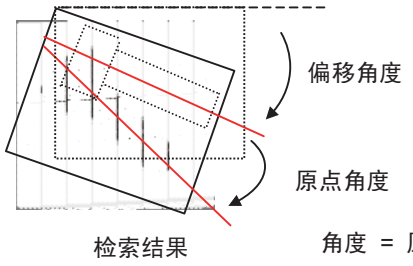




从［起始角度］到［结束角度］的范围内，将原点角度所有的目标进行角度测量。如左图所示，即使 (1)、(2) 哪一个的条件，角度测量的范围也是相同的。

(1)［起始角度］= -45°，［结束角度］= 45°　(2)［起始角度］= -45°，［结束角度］= -315°





角度 = 原点角度 + 偏移角度

将检索模型上所登录的图像和偏离角度存放在测量结果［原点角度］中。此外，将检索模型上所登录的［偏移角度］与［原点角度］相加的结果存放在测量结果［角度］中。

注意

- 需要用 SHDEFMODEL 对检索的模型进行登录。
- 设定 2 以上的个数时，如果不能进行设定个数检索，则只将可以检测的结果存放。
- 检索有时需要花费时间，但在限制时间之内检索没有结束时，将会超时。设定用计算机进行。（初始值 2 秒）
- 根据目标画面的状态，如果对一致度设定过高，则有时会不能检索。在这种情况下，请将一致度向下进行调整。
- 在视窗上指定处理范围。
- 指定的视窗的位置超出画面时，执行结果发生错误。
- 所能指定的视窗的形状只是矩形角度为 0°。在其他的视窗情况下，则发生错误。
- 处理目标成为 VISWORKPLN 所指定的画面。
- 在执行本命令时，由于将处理画面 3 号作为工件区域使用，所以处理画面 3 号的数据不被保证。处理画面 3 号 (VISWORKPLN 3) 不能进行处理。
- 以下是用处理结果获取函数可以得到的数据。

	VISSTATUS (n)
n	项目
0	执行结果 0 = 正常 -1 = 型号未登录 -2 = 失败 -3 = 超时
1	个数
2	执行时间

	VISGETNUM （a、 b）
b	a = 0 ～ 49
0	个数
1	X 坐标值
2	Y 坐标值
3	一致度
4	角度 [V1.4]
5	原点 X 坐标值 [V1.4]
6	原点 Y 坐标值 [V1.4]
7	原点角度 [V1.4]
8	Unknown
9	Unknown

- 本命令需要 μVision 卡 （选件）。
- 检索模型大小与视窗大小相同时，只计算该位置上的一致度。
- 如果设定为 ［起始角度］= 0，［结束角度］= 0，则不进行角度测量。
- 如果将 ［起始角度］和 ［结束角度］设定为 0 以外的相同数值，则进行和 ［起始角度］= 0、［结束角度］= 360 相同的角度测量。

相关项目

WINDMAKE、 VISWORKPLN、 VISGETNUM、 VISSTATUS、 SHDEFMODEL

应用示例

```
VISSCREEN 1,0,1                  ' 在描绘画面 0 号上进行即时描绘。
VISCLS 0                         '
VISPLNOUT 0                      '
WINDMAKE R,1,512,480,0,2         ' 将视窗 1 号设定在矩形视窗上。
CAMIN 1                          ' 在存放存储器上获取照相机影像。
VISWORKPLN 0                     ' 将目标指定在存放存储器 0 号上。
SHMODEL 1,0,0,1,80               ' 从画面检索和模型 1 号一致度 80% 的部分。

I1 = VISSTATUS(0)                '
VISLOC 10,10                     '
VISDEFCHAR 1,1,3                 '
VISPRINT I1                      '
IF I1 = 0 THEN                   '
  VISPRINT VISGETNUM(0,1),VISGETNUM(0,2)
  VISCROSS VISPOSX(0),VISPOSY(0)
END IF                           '
```

SHDEFCORNER（语句）

功　能

设定棱角检索的条件。

格　式

SHDEFCORNER < 距离 >， < 间隔 >， < 宽度 >， < 高度 >

说　明

< 距离 >

指定棱角检索的凸缘棱角边的距离。（1 ～ 10）

< 间隔 >

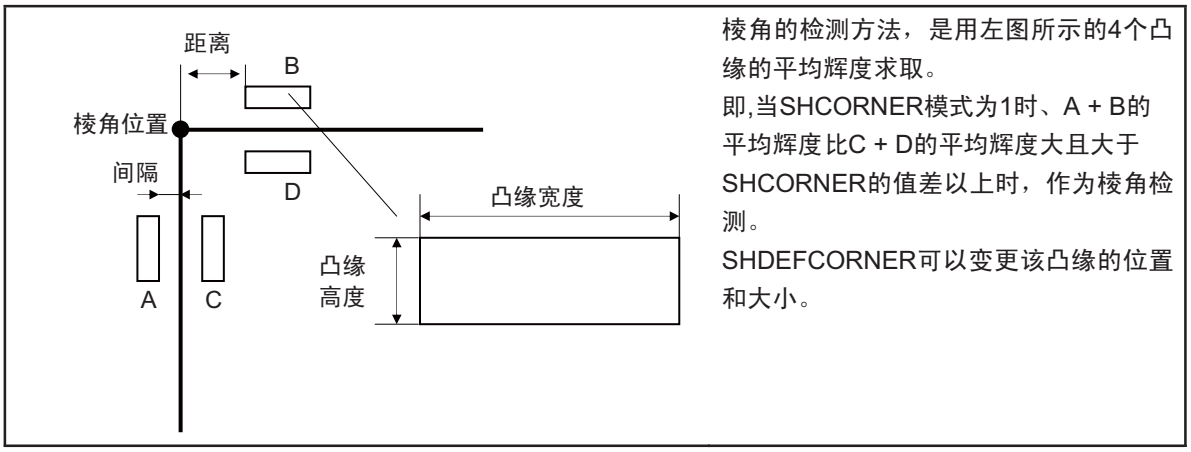
指定棱角检索的凸缘的间隔。（1 ～ 10）

< 宽度 >

指定棱角检索的凸缘的宽度。（1 ～ 10）

< 高度 >

指定棱角检索的凸缘的高度。（1 ～ 10）



棱角的检测方法，是用左图所示的4个凸缘的平均辉度求取。
即,当SHCORNER模式为1时、A + B的平均辉度比C + D的平均辉度大且大于SHCORNER的值差以上时，作为棱角检测。
SHDEFCORNER可以变更该凸缘的位置和大小。

注意

- 本命令是暂时设定，不伴随初始值的永久变更。
- 本命令在机械手控制器中需要 μVision 卡 （选件）。
- 如果增大间隔，则检测时对外紊乱强，检测精度会降低。
- 出厂时的初始值被设定为 : 距离 = 4、间隔 = 1、宽度 = 3、高度 = 3。

相关项目

SHCORNER

应用示例

SHDEFCORNER 4,1,3,3 ’

SHCORNER（语句）

功　能

检索棱角。

格　式

SHCORNER < 视窗编号 >， <X 坐标 >， <Y 坐标 >， < 等级差 >， < 模式 >

说　明

< 视窗编号 >

指定视窗的编号。（0 ～ 511）

<X 坐标 >

指定 X 坐标。（0 ～ 511）

<Y 坐标 >

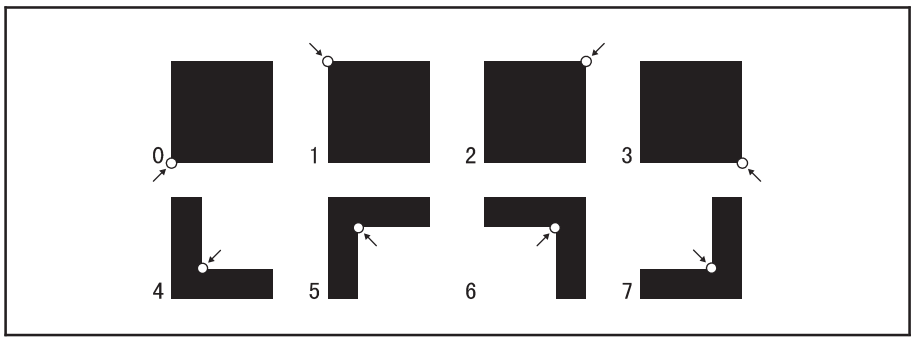
指定 Y 坐标。（0 ～ 479）

< 等级差 >

指定棱角检索的查出等级。（0 ～ 255）

< 模式 >

指定棱角检索的目标。（0 ～ 7）



注意

- 在视窗上指定处理范围。
- 指定的视窗的位置超出画面时，执行结果发生错误。
- 所能指定的视窗的形状只是矩形角度为 0°。在其他的视窗情况下，则发生错误。
- 处理目标成为 VISWORKPLN 所指定的画面。
- 在执行本命令时，由于将处理画面 3 号作为工件区域使用，所以处理画面 3 号的数据不被保证。此外，处理画面 3 号 (VISWORKPLN 3) 不能处理。
- 以下是用处理结果获取函数可以得到的数据。

	VISSTATUS (n)
n	项目
0	执行结果 0 = 正常 -1 = 异常
1	个数
2	执行时间

	VISGETNUM （a、 b）
b	a = 0 ～ 511
0	个数
1	X 坐标值
2	Y 坐标值
3	Unknown
4	等级
5	Unknown
6	Unknown
7	Unknown
8	Unknown
9	Unknown

- 在查出过程中发生超时时，成为此时所查出的棱角数。
- 本命令在机械手控制器中需要 μVision 卡 （选件）。

相关项目

WINDMAKE、 VISWORKPLN、 VISGETNUM、 VISSTATUS、 SHDEFCORNER

应用示例

```
VISSCREEN 1,0,1                    ’在描绘画面 0 号上进行即时描绘。
VISCLS 0                           ’
WINDMAKE R,1,512,480,0,2           ’将视窗 1 号设定在矩形视窗上。
CAMIN 1                            ’在存放存储器上获取照相机影像。
VISWORKPLN 0                       ’将目标指定在存放存储器 0 号上。
SHCORNER 1,0,0,180,0               ’检索黑色的左下棱角。
I1 = VISSTATUS(0)                  ’
IF I1 = 0 THEN                     ’
  I2 = VISSTATUS(1)                ’
  IF I2 <> 0 THEN                  ’
    FOR I1 = 0 TO I2-1             ’
      VISCROSS VISGETNUM(I1,1),VISGETNUM(I1,2)
    NEXT I1                        ’
  END IF                           ’
END IF                             ’
```

SHDEFCIRCLE（语句）

功　能

设定圆检索的条件。

格　式

SHDEFCIRCLE < 间隔 >， < 宽度 >

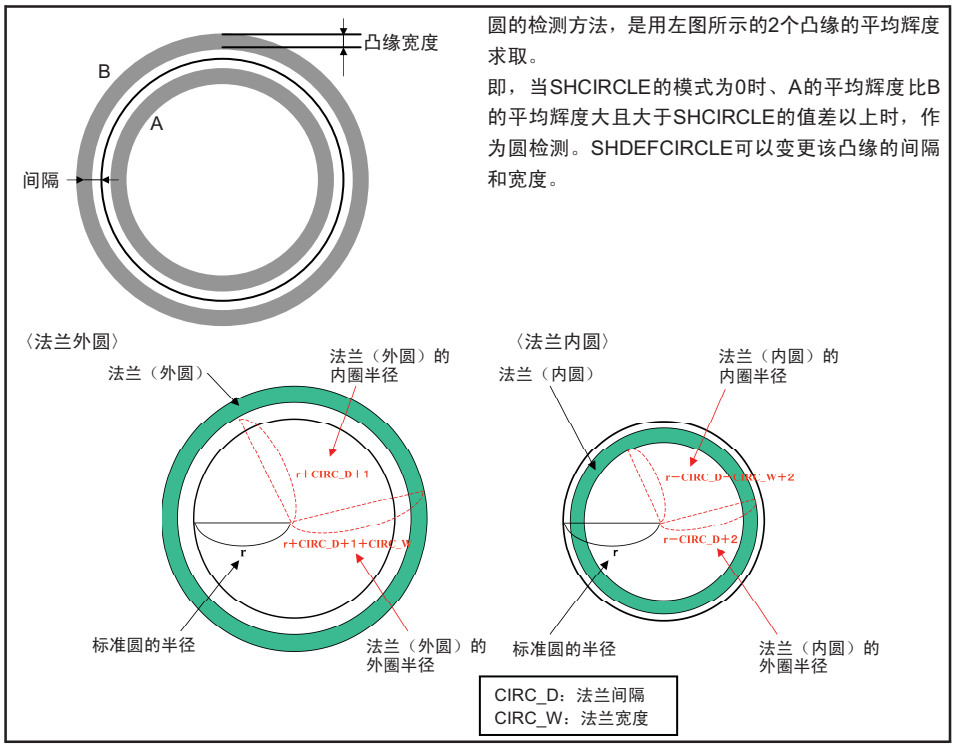
说　明

< 间隔 >

指定圆检索的凸缘的间隔。（1 ～ 10）

< 宽度 >

指定圆检索的凸缘的宽度。（1 ～ 10）



注意

- 本命令是暂时设定，不伴随初始值的永久变更。
- 本命令在机械手控制器中需要 μVision 卡 （选件）。
- 如果增大间隔，则检测时对外紊乱强，检测精度会降低。
- 出厂时的初始值被设定为 : 间隔 = 1、宽度 = 3。

相关项目

SHCIRCLE

应用示例

SHDEFCIRCLE 1,3

SHCIRCLE（语句）

功　能

检索圆。

格　式

SHCIRCLE < 视窗编号 >， <X 坐标 >， <Y 坐标 >， < 半径 >， < 等级差 >， < 模式 >

说　明

< 视窗编号 >

指定视窗的编号。（0 ～ 511）

<X 坐标 >

指定 X 坐标。（0 ～ 511）

<Y 坐标 >

指定 Y 坐标。（0 ～ 479）

< 半径 >

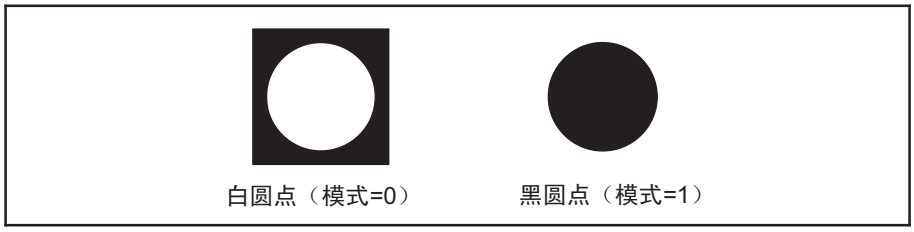
指定检索的圆的半径。（3 ～ 240）

< 等级差 >

指定圆检索的检测等级。（0 ～ 255）

< 模式 >

指定圆检索的目标。（0 或者 1）



注意

- 在视窗上指定处理范围。
- 指定的视窗的位置超出画面时，执行结果发生错误。
- 所能指定的视窗的形状只是矩形角度为 0°。在其他的视窗情况下，则发生错误。
- 处理目标成为 VISWORKPLN 所指定的画面。
- 在执行本命令时，由于将处理画面 3 号作为工件区域使用，所以处理画面 3 号的数据不被保证。此外，处理画面 3 号 (VISWORKPLN 3) 不能处理。
- 以下是用处理结果获取函数可以得到的数据。

	VISSTATUS (n)
n	项目
0	执行结果 0 = 正常 -1 = 异常
1	个数
2	执行时间

	VISGETNUM （a、 b）
b	a = 0 ～ 511
0	个数

	VISGETNUM （a、 b）
b	a = 0 ～ 511
1	X 坐标值
2	Y 坐标值
3	Unknown
4	等级
5	Unknown
6	Unknown
7	Unknown
8	Unknown
9	Unknown

- 本命令在机械手控制器中需要 μVision 卡 （选件）。
- 指定的视窗的检索范围大且半径小时，则有时会发生处理时间增多超时错误。

相关项目

WINDMAKE、 VISWORKPLN、 VISGETNUM、 VISSTATUS、 SHDEFCIRCLE

应用示例

```
VISSCREEN 1,0,1                    ' 在描绘画面 0 号上进行即时描绘。
VISCLS 0
WINDMAKE R,1,512,480,0,2           ' 将视窗 1 号设定在矩形视窗上。
CAMIN 1                            ' 在存放存储器上获取照相机影像。
VISWORKPLN 0                       ' 将目标指定在存放存储器 0 号上。
SHCIRCLE 1,0,0,30,128,1            ' 检索黑色的圆。
I1 = VISSTATUS(0)                  '
VISLOC 10,10                       '
VISPRINT I1                        '
IF I1 = 0 THEN                     '
  I2 = VISSTATUS(1)                '
  IF I2 <> 0 THEN                  '
    FOR I1 = 0 TO I2-1             '
      VISCROSS VISGETNUM(I1,1),VISGETNUM(I1,2)
    NEXT I1                        '
  END IF                           '
END IF                             '
```

21.8 获取结果

VISGETNUM （函数）

功　能

从存放存储器中获取图像处理结果。

格　式

VISGETNUM （< 参数 1>， < 参数 2>）

说　明

< 参数 1>

指定获取的处理结果的编号。（0 ～ 511）

< 参数 2>

指定获取的处理结果的种类。（0 ～ 9）

所取得的值是单精度实数型常量 （ F 型）。

注意

- 处理结果会因事前所进行的图像处理命令，被存放的结果的内容不同。请参照图像处理的命令。
- 结果因为会被保持到下一个图像处理命令被执行为止，所以可以多次获取。
- 事前的图像处理命令，指定没有意图的存放位置时的值成为未知数。
- 本命令在机械手控制器中需要 µVision 卡 （选件）。

相关项目

VISMEASURE、VISPROJ、VISEDGE、VISREADQR、BLOB、BLOBMEASURE、SHMODEL、SHCORNER、SHCIRCLE、 VISPOSX、 VISPOSY

应用示例

```
VISSCREEN 1,0,1                    ' 在描绘画面 0 号上进行即时描绘。
VISCLS 0                           '
WINDMAKE R,1,512,480,0,2           ' 将视窗 1 号设定在矩形视窗上。
CAMIN 1                            ' 在存放存储器上获取照相机影像。
VISPLNOUT 0                        '
VISWORKPLN 0                       ' 将目标指定在存放存储器 0 号上。
BLOB 1,0,0,0,128                   ' 执行标注。
I1 = VISSTATUS(0)                  '
IF I1 = 0 THEN                     '
  I2 = VISSTATUS(1)                '
  IF I2 <> 0 THEN                  '
    FOR I1 = 0 TO 15               '
      VISDEFCHAR 1,1,0             '
      VISLOC 10,10+I1              '
      VISPRINT "X=";VISGETNUM(I1,1),VISGETNUM(I1,2)
    NEXT I1                        '
  END IF                           '
END IF                             '
```

VISGETSTR（函数）

功　能

获取编码识别结果。

格　式

VISGETSTR （< 开头字符编号 >，< 字符数 >）

说　明

< 开头字符编号 >

指定获取的字符的开头编号。（1 ～ 611）

< 字符数 >

指定获取的字符数。（1 ～ 240）

注意

- 获取的字符的长度是未知时，请参照各个命令的处理结果。
- 本命令在机械手控制器中需要 μVision 卡 （选件）。

相关项目

VISREADQR、 VISGETNUM

应用示例

```
VISSCREEN 1,0,1                    ’
VISCLS 0                           ’
WINDMAKE R,1,512,480,0,2           ’ 将视窗 1 号设定在矩形视窗上。
CAMIN 1                            ’ 在存放存储器上获取照相机影像。
VISPLNOUT 0                        ’
VISREADQR 1,0,0,0                  ’QR 编码的读取
I1 = VISSTATUS(0)                  ’
VISPRINT I1,VISSTATUS(1)           ’
IF I1 = 0 THEN                     ’
  VISLOC 10,10                     ’
  VISPRINT VISGETSTR(1,VISSTATUS(1))
END IF                             ’
```

VISPOSX（函数）

功　能

从存放存储器中取得图像处理结果 （ X 坐标）。

格　式

VISPOSX （< 参数 >）

说　明

< 参数 >

指定获取的处理结果 （ X 坐标）的编号。（0 ～ 511）
所取得的值是单精度实数型常量 （ F 型）。

注意

- 与将 VISGETNUM 的参数 1 指定为 1 的情况相同。
  VISGETNUM （n， 1）= VISPOSX （n）
- 处理结果会因事前所进行的图像处理命令，被存放的结果的内容不同。请参照图像处理的命令。
- 结果因为会被保持到下一个图像处理命令被执行为止，所以可以多次获取。
- 事前的图像处理命令，指定没有意图的存放位置时的值成为未知数。
- 本命令在机械手控制器中需要 μVision 卡 （选件）。

相关项目

VISMEASURE、VISPROJ、VISEDGE、VISREADQR、BLOB、BLOBMEASURE、SHMODEL、SHCORNER、SHCIRCLE、 VISGETNUM、 VISPOSY

应用示例

```
VISSCREEN 1,0,1                  ' 在描绘画面 0 号上进行即时描绘。
VISCLS 0                         '
WINDMAKE R,1,512,480,0,2         ' 将视窗 1 号设定在矩形视窗上。
CAMIN 1                          ' 在存放存储器上获取照相机影像。
VISPLNOUT 0                      '
VISWORKPLN 0                     ' 将目标指定在存放存储器 0 号上。
BLOB 1,0,0,0,128                 ' 执行标注。
I1 = VISSTATUS(0)                '
IF I1 = 0 THEN                   '
  I2 = VISSTATUS(1)              '
  IF I2 <> 0 THEN                '
    FOR I1 = 0 TO 15             '
      VISDEFCHAR 1,1,0           '
      VISLOC 10,10+I1            '
      VISPRINT "X=";VISPOSX(I1),"Y=";VISPOSY(I1)
      VISCROSS VISPOSX(I1),VISPOSY(I1)
    NEXT I1                      '
  END IF                         '
END IF                           '
VISCAMOUT 1                      '
```

VISPOSY（函数）

功　能

从存放存储器中取得图像处理结果 （ Y 坐标）。

格　式

VISPOSY （< 参数 >）

说　明

< 参数 >

指定获取的处理结果 （ Y 坐标）的编号。（0 ～ 511）
所取得的值是单精度实数型常量 （ F 型）。

注意

- 与将 VISGETNUM 的参数 1 指定为 2 的情况相同。
  VISGETNUM （n， 2）= VISPOSY （n）
- 处理结果会因事前所进行的图像处理命令，被存放的结果的内容不同。请参照图像处理的命令。
- 结果因为会被保持到下一个图像处理命令被执行为止，所以可以多次获取。
- 事前的图像处理命令，指定没有意图的存放位置时的值成为未知数。
- 本命令在机械手控制器中需要 μVision 卡 （选件）。

相关项目

VISMEASURE、VISPROJ、VISEDGE、VISREADQR、BLOB、BLOBMEASURE、SHMODEL、SHCORNER、SHCIRCLE、 VISGETNUM、 VISPOSX

应用示例

```
VISSCREEN 1,0,1                  ' 在描绘画面 0 号上进行即时描绘。
VISCLS 0                         '
WINDMAKE R,1,512,480,0,2         ' 将视窗 1 号设定在矩形视窗上。
CAMIN 1                          ' 在存放存储器上获取照相机影像。
VISPLNOUT 0                      '
VISWORKPLN 0                     ' 在存放存储器 0 号上指定目标。
BLOB 1,0,0,0,128                 ' 执行标注。
I1 = VISSTATUS(0)                '
IF I1 = 0 THEN                   '
  I2 = VISSTATUS(1)              '
  IF I2 <> 0 THEN                '
    FOR I1 = 0 TO 15             '
      VISDEFCHAR 1,1,0           '
      VISLOC 10,10+I1            '
      VISPRINT "X=";VISPOSX(I1),"Y=";VISPOSY(I1)
      VISCROSS VISPOSX(I1),VISPOSY(I1)
    NEXT I1                      '
  END IF                         '
END IF                           '
VISCAMOUT 1                      '
```

VISSTATUS（函数）

功　能

监视各个命令的处理结果。

格　式

VISSTATUS(< パラメータ >)

说　明

< パラメータ >

指定获取的数据。（0 ～ 2）

0: 执行结果状态

1: 辅助数据

2: 处理时间

所取得的值是单精度实数型常量 （ F 型）。

注意

- 获取的结果会由于在此前所执行的命令而不同。请参照各个命令的说明。
- 处理时间，所有命令成为目标。可以取得此前所执行的命令的处理时间。
- 本命令在机械手控制器中需要 μVision 卡 （选件）。

相关项目

VISMEASURE、VISPROJ、VISEDGE、VISREADQR、BLOB、BLOBMEASURE、SHMODEL、SHCORNER、SHCIRCLE

应用示例

```
VISSCREEN 1,0,1                  ' 在描绘画面 0 号上进行即时描绘。
VISCLS 0                         '
WINDMAKE R,1,512,480,0,2         ' 将视窗 1 号设定在矩形视窗上。
CAMIN 1                          ' 在存放存储器上获取照相机影像。
VISPLNOUT 0                      '
VISWORKPLN 0                     ' 将目标指定在存放存储器 0 号上。
BLOB 1,0,0,0,128                 ' 执行标注。
I1 = VISSTATUS(0)                '
I2 = VISSTATUS(1)                '
F1 = VISSTATUS(2)                '
VISLOC 10,9                      '
VISPRINT " 执行结果 = "； I1； " 标注数 = "； I2," 执行时间 （秒）= "； F1
IF I1 = 0 THEN                   '
  IF I2 <> 0 THEN                '
    FOR I1 = 0 TO I2 -1          '
      VISCROSS VISGETNUM(I1,1),VISGETNUM(I1,2)
    NEXT I1                      '
  END IF                         '
END IF                           '
VISCAMOUT 1                      '
```

VISREFCAL（函数）

功　能

获取 CAL （视觉－机械手坐标转换）数据。

格　式

VISREFCAL （< 群组编号 >，< 数据编号 >）

说　明

< 群组编号 >

指定使用哪个编号的 CAL 数据群。（0 ～ 31）

< 数据编号 >

指定 CAL 数据编号。

［Ver. 1.5 以前版本 : 0 ～ 11]

［Ver. 1.6 以前版本 : 0 ～ 38]

数据编号	0	1	2	3	4	5	6	7	8	9	10	11
内容	NX	OX	AX	RX	NY	OY	AY	RY	NZ	OZ	AZ	RZ

数据编号	12	13	14	15	16	17	18	19	20
内容	Vx1	Vy1	Vz1	Vx2	Vy2	Vz2	Vx2	Vy2	Vz2

数据编号	21	22	23	24	25	26	27	28	29	30	31	32
内容	Rx1	Ry1	Rz1	Rrx1	Rry1	Rrz1	Rx2	Ry2	Rz2	Rrx2	Rry2	Rrz2

数据编号	33	34	35	36	37	38
内容	Rx3	Ry3	Rz3	Rrx3	Rry3	Rrz3

0-11: 转换表数据

12-20: 视觉坐标数据

21-38: 机械手坐标数据

注意

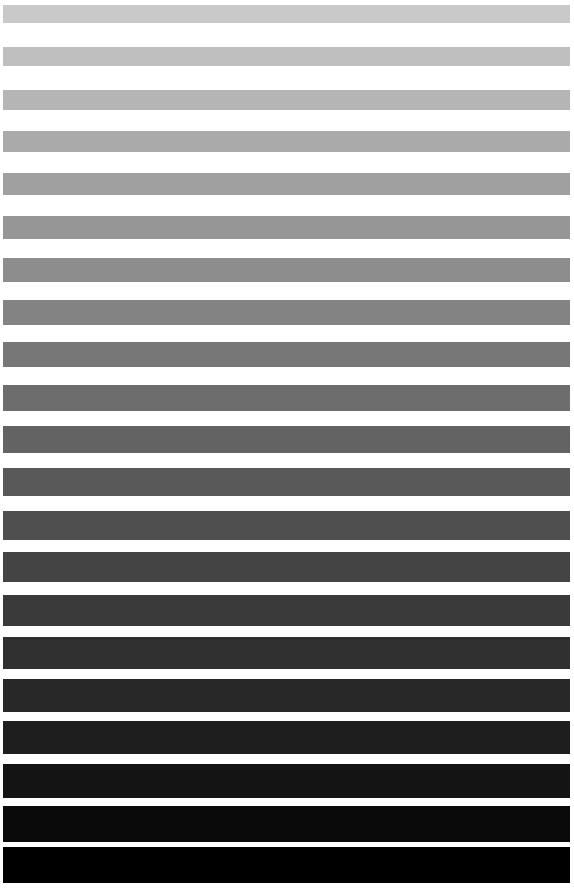
- CAL 数据群可以存储 32 群组。
- 从计算机上进行 CAL 数据的设定。
- 本命令在机械手控制器中需要 μVision 卡 （选件）。

相关项目

应用示例

```
VISSCREEN 1,0,1                    ' 在存放存储器 0 号上进行即时描绘。
VISCLS 0                           '
FOR I1 = 0 TO 11                   '
  VISLOC 10,10+I1                  '
  VISPRINT "Data";I1;"=";VISREFCAL(0,I1)
NEXT I1                            '
```

第 22 章
附录



第 22 章　附录

22.1 字符编码表

表附录1-1　字符编码表

	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
00																
10																
20		!	"	#	$	～	&	’	(	)	*	+	,	-	.	/
30	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
40	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
50	P	Q	R	S	T	U	V	W	X	Y	Z	[	¥	]	^	_
60	‘	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
70	p	q	r	s	t	u	v	w	x	y	z	{	|	}	～	・
80																
90																
A0	・	。	「	」	、	・	ヲ	ァ	ィ	ゥ	ェ	ォ	ャ	ュ	ョ	ッ
B0	-	ア	イ	ウ	エ	オ	カ	キ	ク	ケ	コ	サ	シ	ス	セ	ソ
C0	タ	チ	ツ	テ	ト	ナ	ニ	ヌ	ネ	ノ	ハ	ヒ	フ	ヘ	ホ	マ
D0	ミ	ム	メ	モ	ヤ	ユ	ヨ	ラ	リ	ル	レ	ロ	ワ	ン	゛	゜
E0																
F0																

22.2 机械手形态

(1)　16种类型的形态

6 轴机械手，对于工具前端的 1 个位置和姿势 (X、Y、Z、RX、RY、RZ)，如以下所述，在臂、肘、手腕、第 6 轴、第 4 轴上可以采取不同的形态。

在附录图 2-1～2-5 中，对于臂、肘、手腕、第 6 轴、第 4 轴的各自形态的不同进行说明。

如果将这些形态进行组合，则可以发现对于 1 个位置和姿势可以采取 32 种形态。将该形态的组合列于附录表 2-1 中。此外，在附录图 2-6 中，列出了 VS 系列机械手的臂、肘、手腕形态的 8 种组合的示例。

附录表2-1　机械手形态

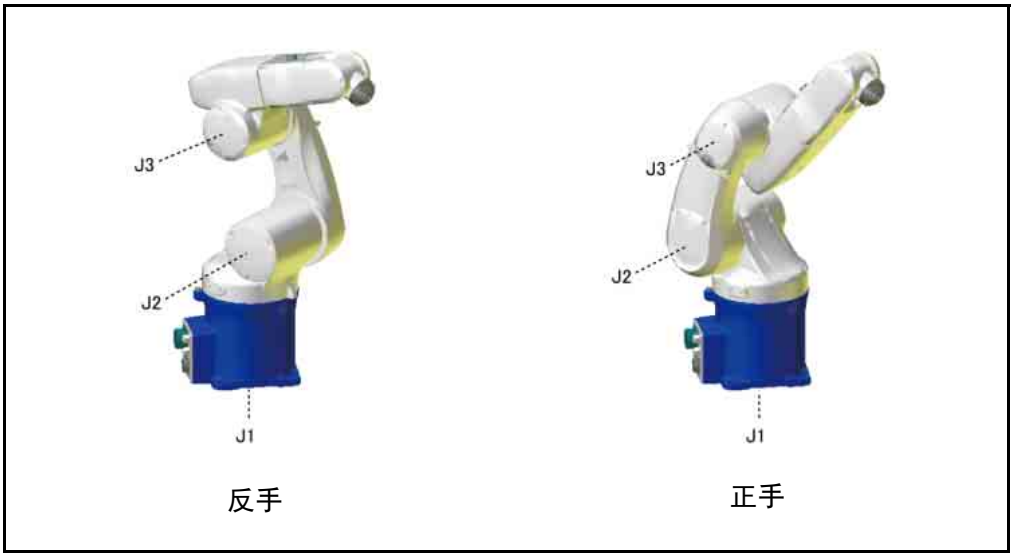
值	第4轴形态	第6轴形态	手腕形态	肘形态	臂形态
0	SINGLE 4	SINGLE	FLIP	ABOVE	RIGHTY
1	SINGLE 4	SINGLE	FLIP	ABOVE	LEFTY
2	SINGLE 4	SINGLE	FLIP	BELOW	RIGHTY
3	SINGLE 4	SINGLE	FLIP	BELOW	LEFTY
4	SINGLE 4	SINGLE	NONFLIP	ABOVE	RIGHTY
5	SINGLE 4	SINGLE	NONFLIP	ABOVE	LEFTY
6	SINGLE 4	SINGLE	NONFLIP	BELOW	RIGHTY
7	SINGLE 4	SINGLE	NONFLIP	BELOW	LEFTY
8	SINGLE 4	DOUBLE	FLIP	ABOVE	RIGHTY
9	SINGLE 4	DOUBLE	FLIP	ABOVE	LEFTY
10	SINGLE 4	DOUBLE	FLIP	BELOW	RIGHTY
11	SINGLE 4	DOUBLE	FLIP	BELOW	LEFTY
12	SINGLE 4	DOUBLE	NONFLIP	ABOVE	RIGHTY
13	SINGLE 4	DOUBLE	NONFLIP	ABOVE	LEFTY
14	SINGLE 4	DOUBLE	NONFLIP	BELOW	RIGHTY
15	SINGLE 4	DOUBLE	NONFLIP	BELOW	LEFTY
16	DOUBLE 4	SINGLE	FLIP	ABOVE	RIGHTY
17	DOUBLE 4	SINGLE	FLIP	ABOVE	LEFTY
18	DOUBLE 4	SINGLE	FLIP	BELOW	RIGHTY
19	DOUBLE 4	SINGLE	FLIP	BELOW	LEFTY
20	DOUBLE 4	SINGLE	NONFLIP	ABOVE	RIGHTY
21	DOUBLE 4	SINGLE	NONFLIP	ABOVE	LEFTY
22	DOUBLE 4	SINGLE	NONFLIP	BELOW	RIGHTY
23	DOUBLE 4	SINGLE	NONFLIP	BELOW	LEFTY
24	DOUBLE 4	DOUBLE	FLIP	ABOVE	RIGHTY
25	DOUBLE 4	DOUBLE	FLIP	ABOVE	LEFTY
26	DOUBLE 4	DOUBLE	FLIP	BELOW	RIGHTY
27	DOUBLE 4	DOUBLE	FLIP	BELOW	LEFTY
28	DOUBLE 4	DOUBLE	NONFLIP	ABOVE	RIGHTY
29	DOUBLE 4	DOUBLE	NONFLIP	ABOVE	LEFTY
30	DOUBLE 4	DOUBLE	NONFLIP	BELOW	RIGHTY
31	DOUBLE 4	DOUBLE	NONFLIP	BELOW	LEFTY

① 臂形态

臂形态，由第1轴、第2轴、第3轴的值决定。

臂形态，可以采取左臂系反手 (LEFTY)、右臂系正手 (RIGHTY) 的两种形态。

（J1～J6表示各个轴。）



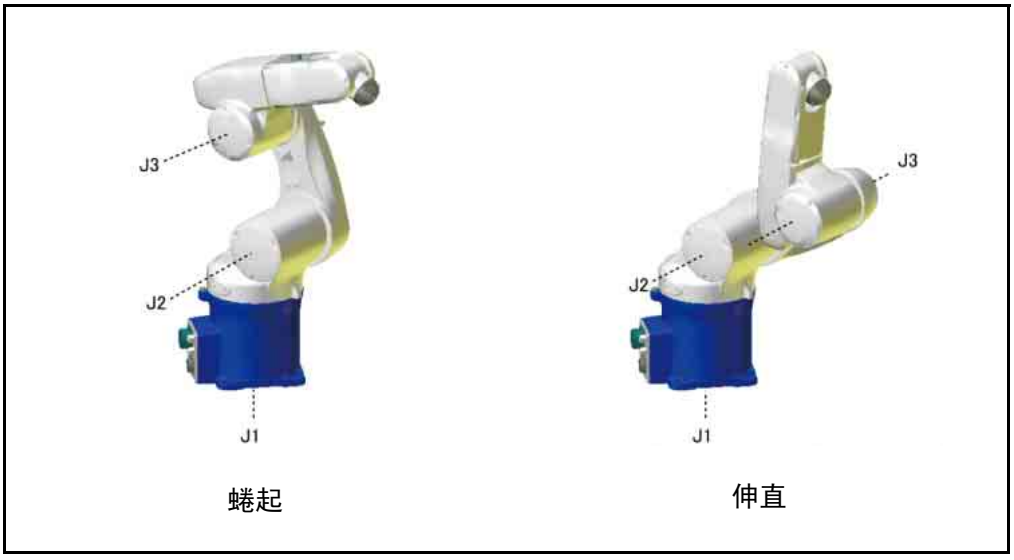
附录图2-1　臂形态

② 肘形态

肘形态由第2轴、第3轴的值决定。

肘的形态可以采取蜷起 (ABOVE) 和伸直 (BELOW) 的2种类型。

（J1～J6表示各个轴。）



附录图2-2　肘形态

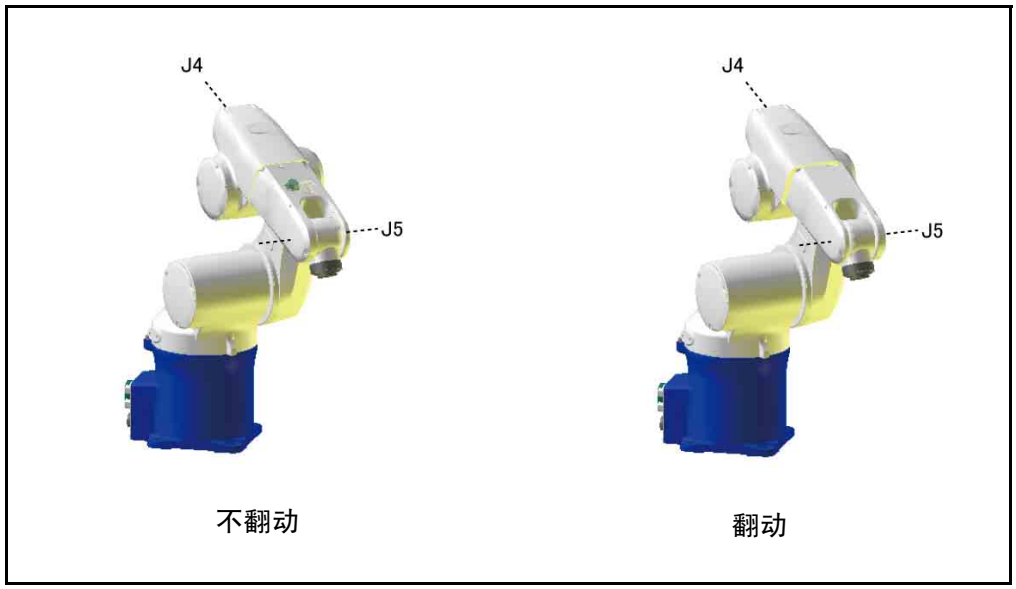
③ 手腕形态

手腕形态由第4轴、第5轴的值决定。

手腕的形态可以采取翻动 (FLIP) 和不翻动 (NONFLIP) 的2种类型。

不翻动就是从翻动形态不改变手腕部位的姿势而让第4轴转动180度的形态。

（J1～J6表示各个轴。）



附录图2-3　手腕形态

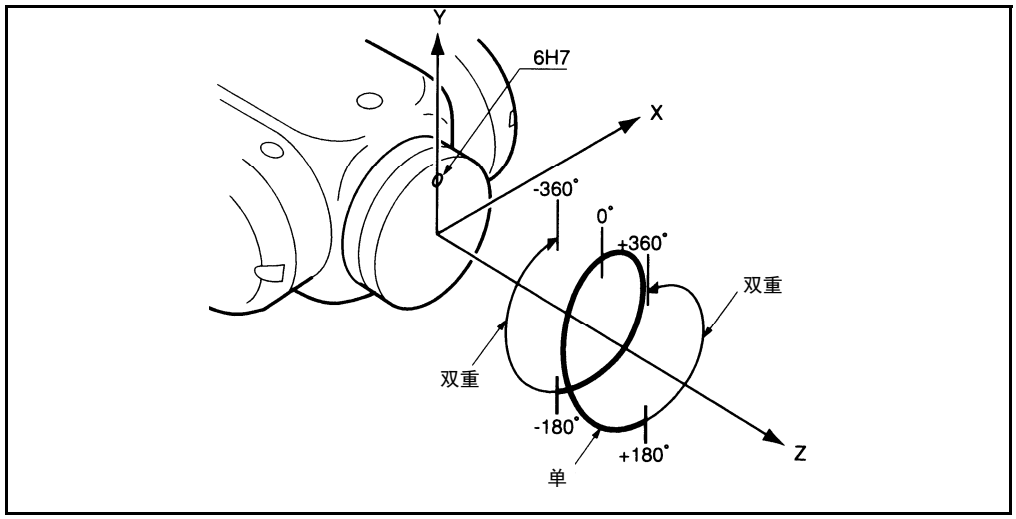
④ 第6轴形态

第6轴形态、由第6轴的值决定。

第6轴的形态可以采取单 (SINGLE) 和双 (DOUBLE) 的2种类型。

第6轴的旋转角度θ_6为 ($-180 < \theta_6 \leqq 180$) 时是单,为（$180 < \theta_6 \leqq 360$或者 $-360 < \theta_6 \leqq -180$）时是双。

在$\theta_6 = 180°$和$\theta_6 = 181°$时形态改变。将位置数值进行数值变更时,请注意形态的设定。（打算修正为$\theta_6 = 181°$，但形态不被修正时则成为$\theta_6 = -179°$。）



附录图2-4　第6轴形态

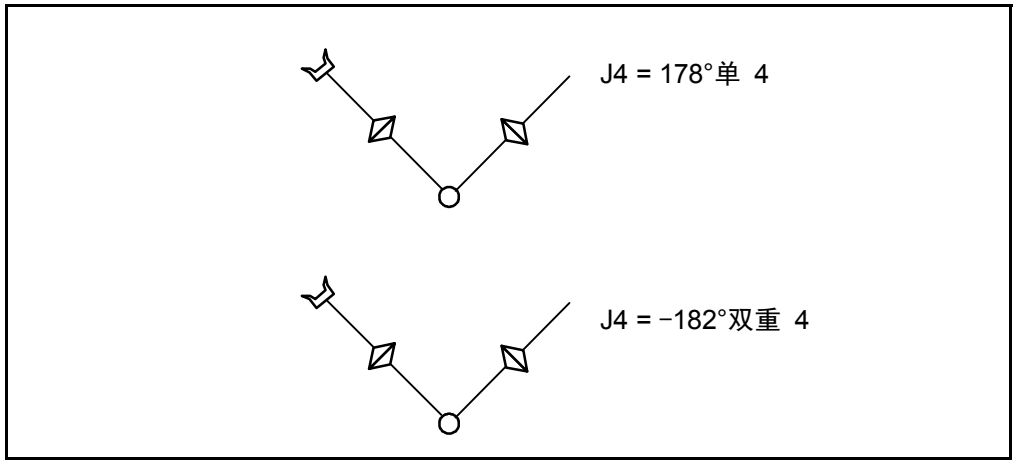
⑤ 第4轴形态

第4轴形态、由第4轴的值决定。

第4轴的形态可以采取单4 (SINGLE 4) 和双4 (DOUBLE) 的2种类型。

第4轴的旋转角度θ_4为 ($-180 < \theta_4 \leqq 180$) 时是单4，为（$180 < \theta_4 \leqq 185$或者 $-185 < \theta_4 \leqq -180$）时是双4。

在$\theta_4 = 180°$和$\theta_4 = 181°$时形态改变。将位置数值进行数值变更时,请注意形态的设定。（打算修正为$\theta_4 = 181°$，但形态不被修正时则成为$\theta_4 = -179°$。）



附录图2-5　第4轴形态

形态的种类1
"反手、蜷起、不翻动"



形态的种类2
"反手、蜷起、翻动"



形态的种类3
"反手、伸直、不翻动"



形态的种类4
"反手、伸直、翻动"



形态的种类5
"正手、蜷起、不翻动"



形态的种类6
"正手、蜷起、翻动"



形态的种类7
"正手、伸直、不翻动"

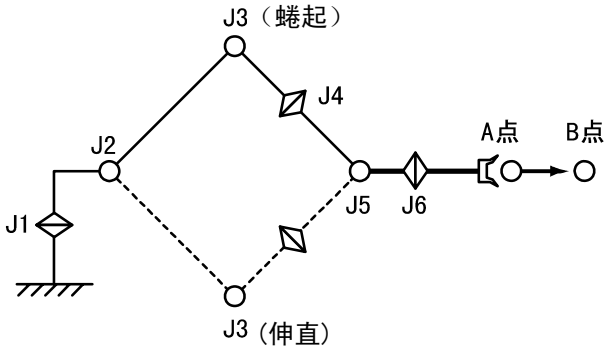


形态的种类8
"正手、伸直、翻动"



附录图2-6 手腕、肘、手掌形态的组合示例

⚠注意：执行CP动作指令时，在出发点时的位置和姿势即使相同，但如果臂、肘、手腕的形态不同，各个轴也会进行不同的动作移向目标位置。
因此，根据在出发点上的形态不同时的CP动作进行移动，请事前确认对各个轴的设备等是否有干扰，之后再实施。
但是，在工具前端所看到的移动路径，即使形态改变也是相同的。

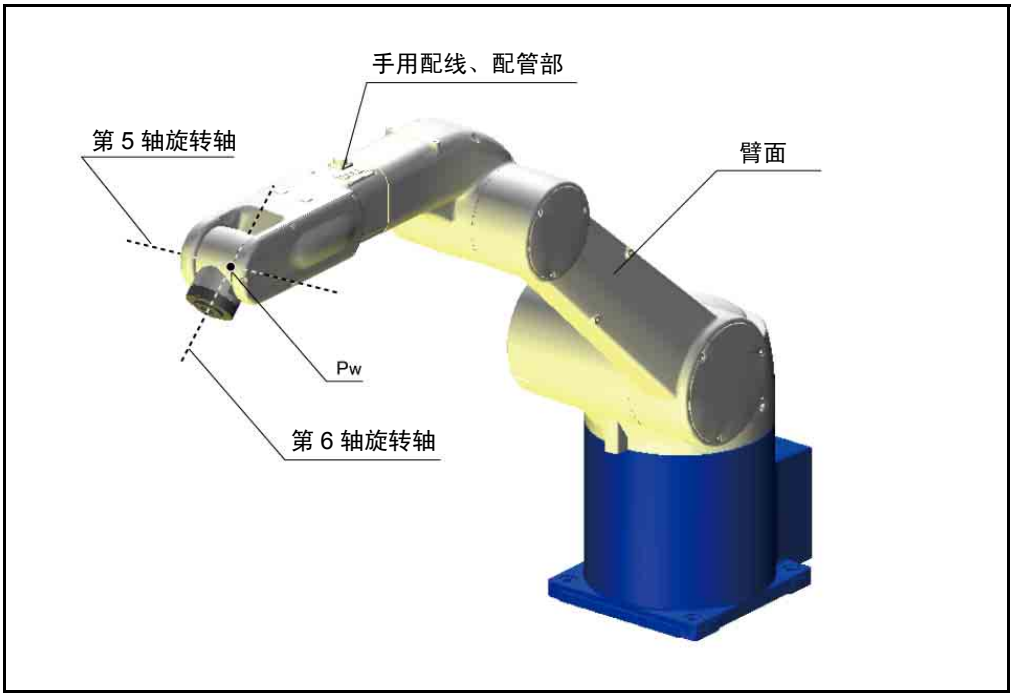


注意：机械手的结构上，对于任意的1个位置和姿势，并不都能采取所有16种的形态。
根据位置和姿势，有时只能采取例如反手、蜷起、不翻动的形态。
（但是，在通常的动作范围内，大多只能采取反手、蜷起、不翻动或者反手、蜷起、翻动的2种形态。此外，4轴形态大多是单4时。）

(2)　形态的边界

就臂、肘、手腕、第6轴的各种形态的边界条件进行说明。

使用第5轴的旋转轴和第6轴的旋转轴交叉点 "Pw" 的位置进行手腕、肘、手掌的边界的判断。（参照附录图2-7）



附录图2-7　"Pw" 的位置

各种形态的边界位置称为特异点。

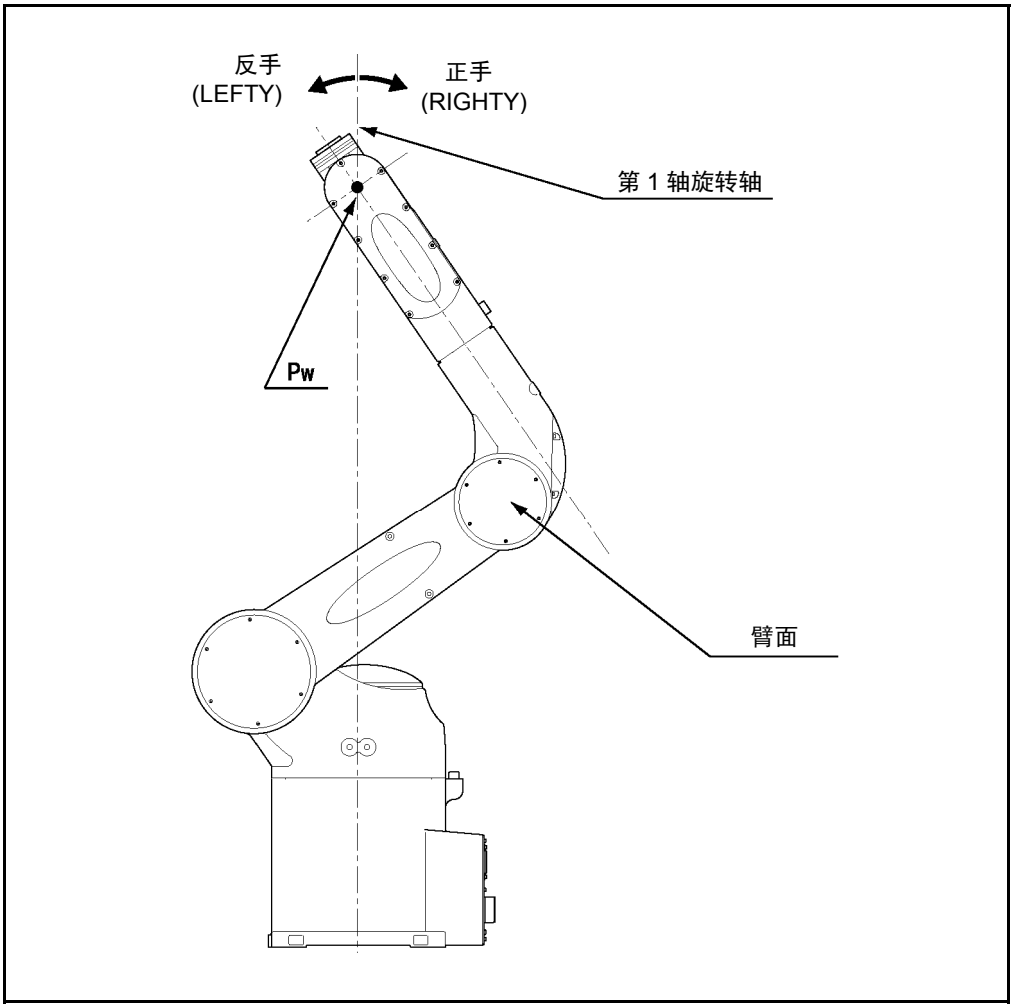
像MOVE、APPROACH、DEPART进行CP动作的指令（p.3-11 "3.3　插补控制"）是不能通过特异点附近的。

如果特异点在轨道附近，会发生错码6080号（速度超程）或者错码6070号（软件限位超程），并停止。

① 反手、正手

对于臂面从垂直方向看到的 "Pw" 的位置，在第1轴旋转轴的左边时为反手、在右边时为正手。

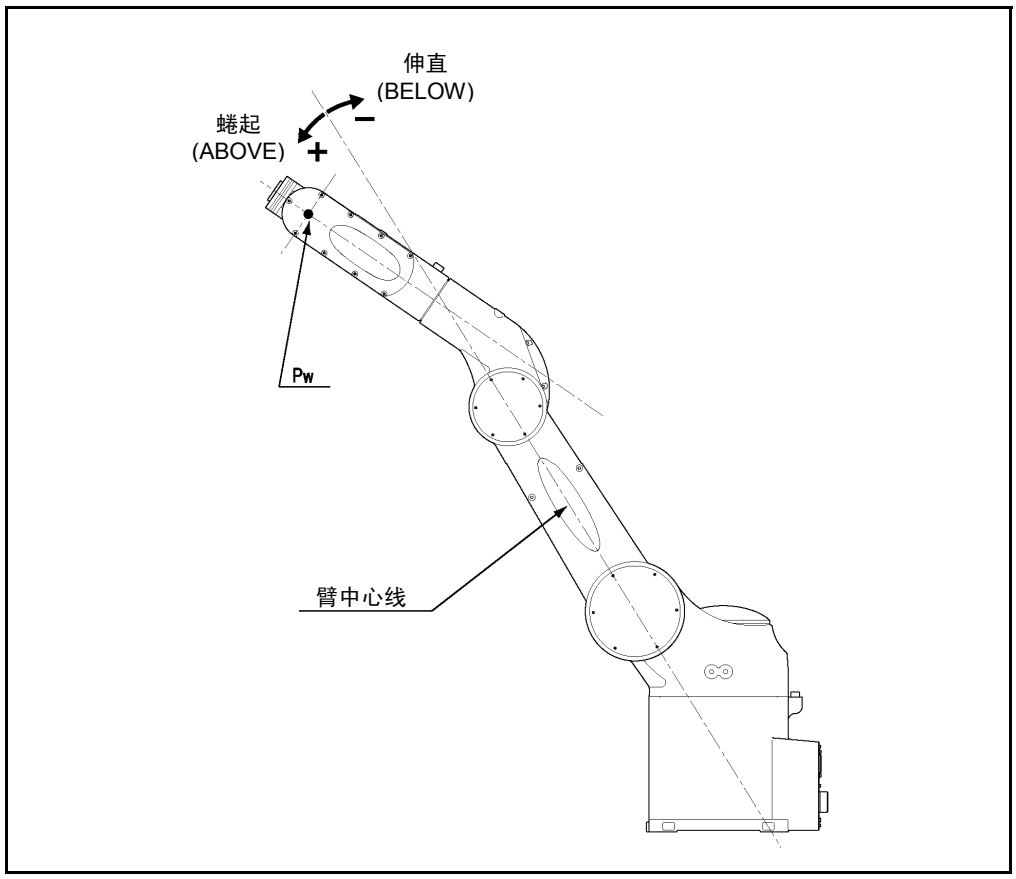
注意："Pw" 的位置在第1轴的旋转轴的上方时，则成为在2个形态的边界上，这个位置叫做特异点。



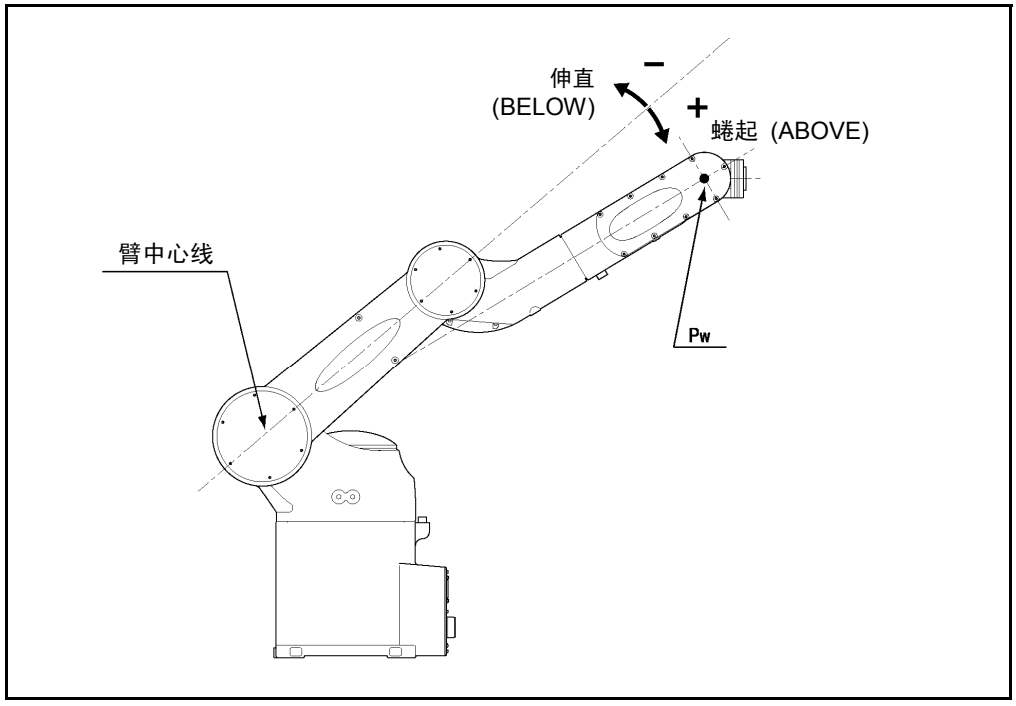
附录图2-8　反手、正手的边界

② 蜷起、伸直

对于臂中心线，"PW" 的位置在 + 侧时是蜷起，在－侧时是伸直。



附录图2-9　蜷起、伸直的边界（反手时）

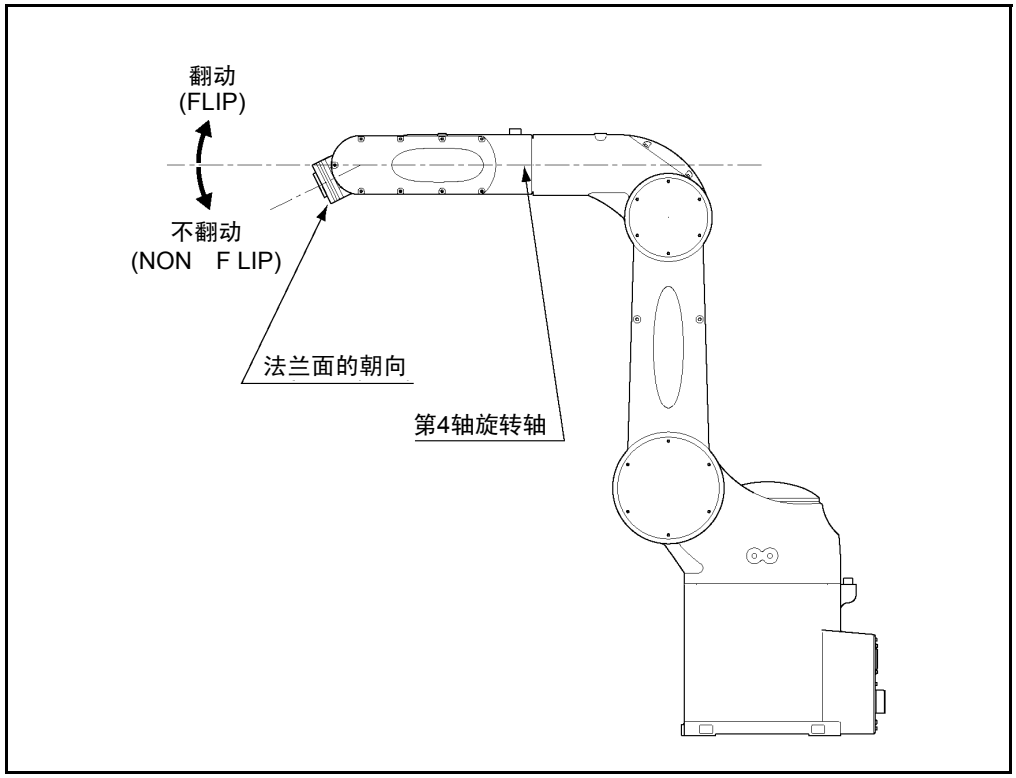


附录图2-10　蜷起、伸直的边界（正手时）

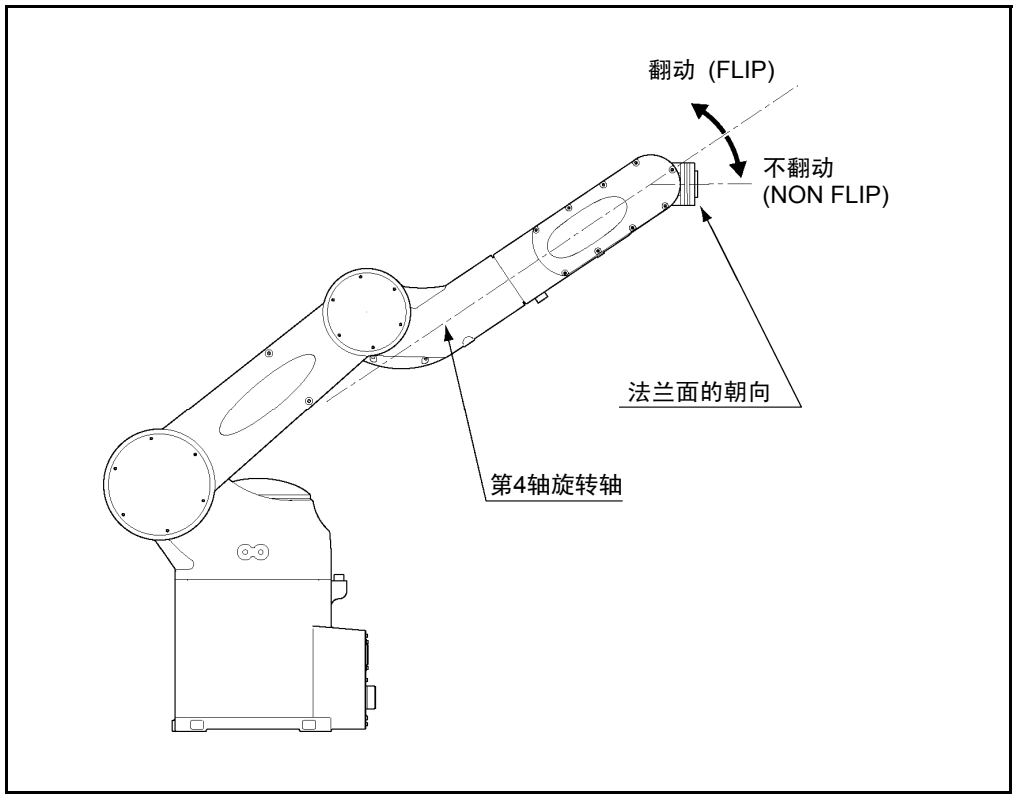
③ 翻动、不翻动

以第4轴的旋转轴为基准，法兰面朝上时是翻动，朝下时是不翻动。

（附录图2-11和2-12是朝下）



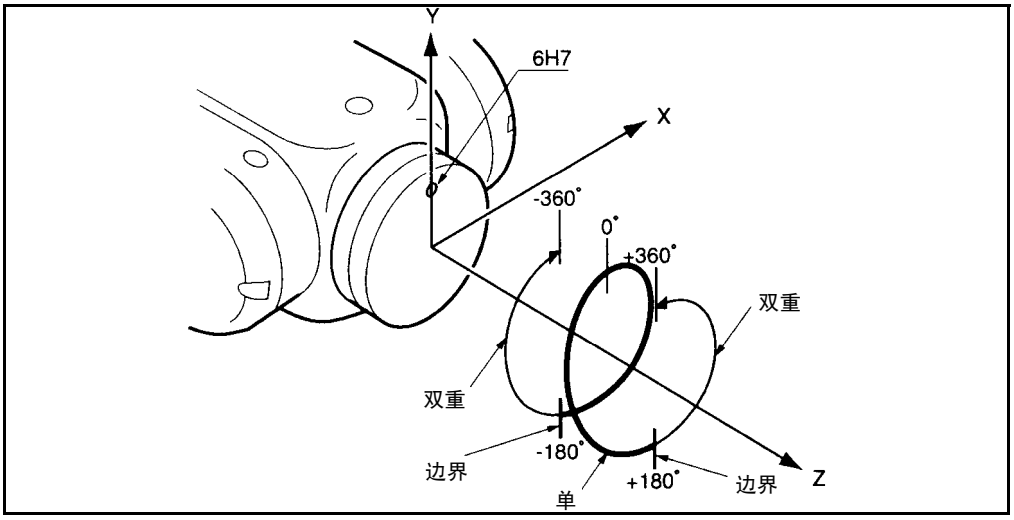
附录图2-11　翻动、不翻动的边界（反手时）



附录图2-12　翻动、不翻动的边界（正手时）

④ 单、双重

第6轴的旋转角度θ6为 (–180 < θ_6≦180) 时是单,为（180 < θ_6 ≦ 360或者 –360 < θ_6 ≦ –180）时是双。



附录图2-13　单、双重的边界

22.3 环境设定值

表编号	宏名称	内容
1	cnfSYS	系统参数表
2	cnfARM	轨道生成参数表
3	cnfVIS	视觉参数表
4	cnfPAC	PAC 参数表
5	cnfSRV	伺服参数表
6	cnfSPD	使用条件参数表
7	cnfITP	解释程序参数表
8	cnfDIO	DIO 参数表
9	cnfCOM	通信参数表

备注①：宏名称被定义在<pacman.h>上。在GETENV、LETENV命令上使用宏名称时，如下所示，请包含该文件。

```
#INCLUDE      <pacman.h>
```

②：任意表的要素编号的宏名称，均可在WINCAPSⅢ 进行创建。例如，在创建轨道生成参数表的宏定义文件时，请按以下所示包含该文件。

```
#INCLUDE      "arm_cnf.h"
```

22.4 使用条件参数

多功能教导器的使用条件视窗（操作路径：［F2 臂］－［F6 插补功能］－［F7 使用条件］），以及WINCAPSⅢ的使用条件制表键（操作路径：在［项目(P)］－［参数(T)…］－［使用条件］制表键）上显示的参数一览。

编号	显示	出厂时值	电源接通时	内容	备注
7	最佳可搬运质量设定模式	0	0	0：OFF 1：只限 PTP 动作 2：只限 CP 动作 3：CP、PTP 均有效 （参照编程手册 "4.6 最佳可搬运质量设定功能"）	通过 aspChange（）可以变更
8	地面设置 / 悬挂设定	0	电源 OFF 时的值	0：地面设置 1：悬挂 2：壁挂	只有 6 轴机械手需要设定
9	前端负荷质量(g)	根据机型而不同	电源 OFF 时的值	工具和工件相加的质量	通过 aspACLD 可以变更
10	前端负荷重心位置 X (mm)	0	电源 OFF 时的值	工具和工件相加负荷的重心位置的 X 分量（参照编程手册 "4.6 最佳可搬运质量设定功能"）	通过 aspACLD 可以变更
11	前端负荷重心位置 Y (mm)	80	电源 OFF 时的值	工具和工件相加负荷的重心位置的 Y 分量（参照编程手册 "4.6 最佳可搬运质量设定功能"）	
12	前端负荷重心位置 Z (mm) <4 轴机械手 Ver.1.9 以上版本> 负荷惯量 ($kgcm^2$)	100	电源 OFF 时的值	工具和工件相加负荷的重心位置的 Z 分量（参照编程手册"4.6 最佳可搬运质量设定功能"）	
13~20	停止时允许脉冲宽度 (J1～J8)	20	20	动作命令的@E 指定时的对象轴（1 轴～8 轴）结束范围（脉冲数）	通过 mvSetPulseWidth（）可以变更
21	动作结束超时(ms)	5600	5600	指定动作命令的@E 时，在指定时间以内没有收敛在指定脉冲以内时，就会造成检查命令超时错误。	通过 mvSetTimeOut（）可以变更
22	控制 LOG 记录模式	1	电源 OFF 时的值	控制 LOG 的记录数 设定值：1～24 记录数：1250 个 × （设定值）	大量使用程序、变量而加多记录个数电源 ON 时有时会成为错误。错误发生时请减少个数。

编号	显示	出厂时值	电源接通时	内容	备注
23	控制 LOG 记录间隔	8	电源 OFF 时的值	控制 LOG 的记录间隔 设定值：8、16、24、32 ms	如果被设定为 8 的倍数以外的数值时，会被 8 的倍数修正。
24	重力补偿有效 / 无效设定（6 轴机械手用)	0	电源 OFF 时的值	0：将重力补偿功能无效 1：将重力补偿功能有效。	通过 SetGravity、ResetGravity 可以变更
25	电流限制复位设定	0	电源 OFF 时的值	a) 最下限比特 0 时：在接通电机电源时电流限制被复位。 b) 第 2 比特为 0 时：在接通电机电源时伺服锁定解除被复位。（4 轴机械手专用） c) 第 3 比特为 0 时：在接通电机电源时 PWM 切换解除被复位。（4 轴机械手专用）	请不要变更初始设定值。
26	伺服锁定设定 （4 轴机械手用）	0	电源 OFF 时的值	1：解除伺服锁定。	通过 OffSrvLock、OnSrvLock 可以变更
27	控制方法 （HM / HS-D 机械手用）	0	电源 OFF 时的值	1：将控制方法 P 控制 （参照 H*-D 安装与维护指南、"2.7 控制方法的变更功能"）	请确认 "加速度变更模式" 是在 0 上。
28	设定高可搬运增益 （HM / HS-D 机械手用）	0	电源 OFF 时的值	1：将控制增益变更为高可搬运增益。 （参照 H*-D 安装与维护指南、"2.8 高可搬运增益的设定"）	请确认 "加速度变更模式" 是 0，前端负荷质量 (g) 是 10000。
29	加速度变更模式	0 或者 1	电源 OFF 时的值	0：将增益变更功能有效。 1：将增益变更功能无效。	初始值是 4 轴机械手为 0、6 轴机械手为 1。**请不要变更初始设定值。**
34	电机电源保持功能	1	电源 OFF 时的值	设定自动允许 SW 切换时的电机电源的状态。 0：如果在自动允许 SW 切换时电机电源成 ON 了则将电机电源置为 OFF。 1：即使切换自动允许 SW 电机电源的状态也不改变。	
35	摆线动作设定	0	电源 OFF 时的值	0：将摆线动作无效 1：将摆线动作有效。	通过 Setcycloid、Resetcycloid 可以变更

编号	显示	出厂时值	电源接通时	内容	备注
53～60	增益减少比例 (J1～J8)	机械手 固有值	电源 OFF 时的值	设定对象轴（1～8 轴）的增益减少比例。	在"加速度变更模式"为0, "控制方法"为0, "高可搬运增益设定"为 0 时,是有效的。 **请不要变更初始设定值。**
61～68	高可搬运 10kg 设定 增益减少比例 (J1～J8) （HM / HN 机械手用）	0	电源 OFF 时的值	设定高可搬运增益设定时的对象轴（1～8 轴）的增益减少比例。	在 "加速度变更模式 "为 0, "控制方法 "为 0, "高可搬运增益设定" 为 1 时,是有效的。 **请不要变更初始设定值。**
69	新旧机械选择 （4 轴机械手用）	0	电源 OFF 时的值	0：新式机械（D 系列）的设定值。 1：旧式机械（C 系列）的设定值。	单体购买控制器连接到 HM / HS / HC-C 系列时，请设定为 1。
70	路径再启动时的动作设定	0	电源 OFF 时的值	设定通过 (Pass)路径动作过程中被进行了停止处理后再启动时的动作目标位置。 0：通过 (Pass)路径动作后的目标位置的动作。（默认） 1：通过 (Pass)路径动作前的目标位置的动作。	
71	路径动作结束范围	5	电源 OFF 时的值	为了在再启动时不通过(Pass)路径动作前的目标位置动作的条件。 以从目标位置开始的距离进行设定。	是在指令值水平上从目标位置开始的距离，不是距机械手的手指端位置的距离。 请不要变更初始设定值。
78	阻尼器设定比例 （X 方向） （6 轴机械手用）	10000	10000	力限制时的 X 方向的阻尼器比例	通过 SetDampRate、ResetDampRate 可以设定 不能通过教导器进行变更。 （Ver 1.4 以上版本）
79	阻尼器设定比例 （Y 方向） （6 轴机械手用）	10000	10000	力限制时的 Y 方向的阻尼器比例	
80	阻尼器设定比例 （Z 方向） （6 轴机械手用）	10000	10000	力限制时的 Z 方向的阻尼器比例	

编号	显示	出厂时值	电源接通时	内容	备注
81	阻尼器设定比例 （X 旋转） （6 轴机械手用）	10000	10000	力限制时的 X 旋转的阻尼器比例	通过 SetDampRate、ResetDampRate 可以设定 不能通过教导器进行变更。 （Ver 1.4 以上版本）
82	阻尼器设定比例 （Y 旋转） （6 轴机械手用）	10000	10000	力限制时的 Y 旋转的阻尼器比例	
83	阻尼器设定比例 （Z 旋转） （6 轴机械手用）	10000	10000	力限制时的 Z 旋转的阻尼器比例	
84	依从控制模式选择 （6 轴机械手用）	1	1	最下限比特是 0 时：为依从速度控制模式。 第 2 比特是 1 时：丢失力限制有效设定时的重力补偿修正处理。	通过 SetCompVMode、ResetCompVMode、SetCompControl、SetCompFControl 可以设定 不能通过教导器进行变更。 （Ver 1.4 以上版本）
86	选择防振模式 （6 轴机械手用）	0	电源 OFF 时的值	1：余留振动降低模式	通过 SetVibControl、ResetVibControl 可以设定
87	设定力限制功能 有效 / 无效 （6 轴机械手用）	0	0	1：正在执行力限制	通过 SetCompControl、SetCompFControl、ResetCompControl 可以设定。 不能通过教导器进行变更。 （Ver 1.4 以上版本）
88	选择力限制坐标系 （6 轴机械手用）	0	0	力限制坐标系设定值 0：底座坐标 1：工具坐标 2：工件坐标	通过 SetFrcCoord 可以设定。 不能通过教导器进行变更。 （Ver 1.4 以上版本）
89	力限制比例（+X 方向） （6 轴机械手用）	10000	10000	力限制时的+X 方向力限制比例	通过 SetFrcCoord 可以设定。 不能通过教导器进行变更。 （Ver 1.4 以上版本）
90	力限制比例（+Y 方向） （6 轴机械手用）	10000	10000	力限制时的+Y 方向力限制比例	
91	力限制比例（+Z 方向） （6 轴机械手用）	10000	10000	力限制时的+Z 方向力限制比例	

编号	显示	出厂时值	电源接通时	内容	备注
92	力限制比例（+X 旋转）（6 轴机械手用）	10000	10000	力限制时的+X 旋转力限制比例	通过 SetFrcCoord 可以设定。 不能通过教导器进行变更。 （Ver 1.4 以上版本）
93	力限制比例（+Y 旋转）（6 轴机械手用）	10000	10000	力限制时的+Y 旋转力限制比例	
94	力限制比例（+Z 旋转）（6 轴机械手用）	10000	10000	力限制时的+Z 旋转力限制比例	
95	力限制比例（−X 方向）（6 轴机械手用）	10000	10000	力限制时的−X 方向力限制比例	通过 SetFrcCoord 可以设定。 不能通过教导器进行变更。 （Ver 1.4 以上版本）
96	力限制比例（−Y 方向）（6 轴机械手用）	10000	10000	力限制时的−Y 方向力限制比例	
97	力限制比例（−Z 方向）（6 轴机械手用）	10000	10000	力限制时的−Z 方向力限制比例	
98	力限制比例（−X 旋转）（6 轴机械手用）	10000	10000	力限制时的−X 旋转力限制比例	通过 SetFrcCoord 可以设定。 不能通过教导器进行变更。 （Ver 1.4 以上版本）
99	力限制比例（−Y 旋转）（6 轴机械手用）	10000	10000	力限制时的−Y 旋转力限制比例	
100	力限制比例（−Z 旋转）（6 轴机械手用）	10000	10000	力限制时的−Z 旋转力限制比例	
101	依从设定比例（X 方向）（6 轴机械手用）	10000	10000	力限制时的 X 方向柔度比例	可通过 SetCompRate 设定。 不能用教导器进行变更。 （Ver 1.4 以上版本）
102	依从设定比例（Y 方向）（6 轴机械手用）	10000	10000	力限制时的 Y 方向柔度比例	
103	依从设定比例（Z 方向）（6 轴机械手用）	10000	10000	力限制时的 Z 方向柔度比例	
104	依从设定比例（X 周围）（6 轴机械手用）	10000	10000	力限制时的 X 周围柔度比例	可通过 SetCompRate 设定。 不能用教导器进行变更。 （Ver 1.4 以上版本）
105	依从设定比例（Y 周围）（6 轴机械手用）	10000	10000	力限制时的 Y 周围柔度比例	
106	依从设定比例（Z 周围）（6 轴机械手用）	10000	10000	力限制时的 Z 周围柔度比例	
107	依从偏差允许值（X 方向）（6 轴机械手用）	100	100	力限制时的 X 方向偏差允许值	可通过 SetCompEralw 设定。 不能用教导器进行变更。 （Ver 1.4 以上版本）
108	依从偏差允许值（Y 方向）（6 轴机械手用）	100	100	力限制时的 Y 方向偏差允许值	
109	依从偏差允许值（Z 方向）（6 轴机械手用）	100	100	力限制时的 Z 方向偏差允许值	

编号	显示	出厂值	电源接通时	内容	备注
110	依从偏差允许值（X 周围）（6 轴机械手用）	300	300	力限制时的 X 周围偏差允许值	可通过 SetCompEralw 设定。不能用教导器进行变更。（Ver 1.4 以上版本）
111	依从偏差允许值（Y 周围）（6 轴机械手用）	300	300	力限制时的 Y 周围偏差允许值	
112	依从偏差允许值（Z 周围）（6 轴机械手用）	300	300	力限制时的 Z 周围偏差允许值	
113	力补偿设定（X 方向）（6 轴机械手用）	0	0	力限制时的 X 方向补偿力	可通过 SetFrcAssist 设定。不能用教导器进行变更。（Ver 1.4 以上版本）
114	力补偿设定（Y 方向）（6 轴机械手用）	0	0	力限制时的 Y 方向补偿力	
115	力补偿设定（Z 方向）（6 轴机械手用）	0	0	力限制时的 Z 方向补偿力	
116	力补偿设定（X 周围）（6 轴机械手用）	0	0	力限制时的 X 周围补偿力矩	可通过 SetFrcAssist 设定。不能用教导器进行变更。（Ver 1.4 以上版本）
117	力补偿设定（Y 周围）（6 轴机械手用）	0	0	力限制时的 Y 周围补偿力矩	
118	力补偿设定（Z 周围）（6 轴机械手用）	0	0	力限制时的 Z 周围补偿力矩	
120	最佳可搬运质量初始化设定	0	电源 OFF 时的值	0：可搬运质量模式在电源 ON 时设为 0。（默认） 1：可搬运质量设定模式在电源为 OFF 时不被初始化。电源 OFF 时的值被保持。	（Ver 1.4 以上版本）
121～128	依从轴转矩限制值（J1～J8）（6 轴机械手用）	0	0	力限制时的相应轴（1～8 轴）的电流限制值。	可通过 SetCompJLimit、ResetCompJLimit 设定。不能用教导器进行变更。（Ver 1.4 以上版本）

编号	显示	出厂值	电源接通时	内容	备注
196	J4 制动器锁定选择（VM-6083D, VM-60B1D,VS-E 用）	0	0	制动器解除过程中发生 J4 软件限位超程时的选择 0：J4 制动器锁定 1：制动器不锁定	（Ver1.7 以上版本）
197	CP 动作时的速度设定（注 1）	0	电源 OFF 时的值	0：与原来相同的速度设定 1：TCP 速度（TOOL 前端 CP 速度）为一定的速度设定	（Ver1.8 以上版本）
	注1：在原来的设定中，进行包括手指端的旋转动作在内的动作时，TCP速度（TOOL前端CP速度）按照旋转动作量进行减速。为此，会发生不按指定速度动作或者速度不一定（变速）的现象。 通过将该参数设定为1，可以将TCP速度保持一定。但是，手指端的旋转速度根据动作条件进行增减，指定速度超过规定的界限值时，会在显示警告的同时，TCP速度相对于所指定的速度降低。				
198	TOOL、WORK 复位条件（注 2）	0	电源 OFF 时的值	0：与原来的动作相同（不复位） 1：电源接通时 TOOL、WORK 的坐标系定义、坐标系变更复位	（Ver1.8 以上版本）
	注2：如果切断控制器的电源，原来在程序中所执行的TOOL、WORK的坐标系定义以及变更的TOOL、WORK坐标系将不能被保持。因此，进行同样操作时，下一次接通电源后，需要进行与切断电源之前同样的操作。通过将该参数设定为1，在电源接通时可以将TOOL、WORK坐标系定义以及变更的TOOL、WORK坐标系复位到前一次的状态。				
199	简单教导的圆弧动作允许值	100	100	简单教导时的圆弧动作的位置偏离允许值	通常情况下请不要进行变更。（Ver1.8 以上版本）
200	弓形动作执行标志 (Flag)	0	电源 OFF 时的值	执行 ArchMove 时的弓形形状的设定	可用 SetArchParam 设定（Ver1.9 以上版本）
201	弓形动作上升距离	0	电源 OFF 时的值	ArchMove 时的上升动作过程中，开始横向动作的位置	可用 SetArchParam 设定（Ver1.9 以上版本）
202	弓形动作下降距离	0	电源 OFF 时的值	ArchMove 时的下降动作过程中，开始横向动作的位置	可用 SetArchParam 设定（Ver1.9 以上版本）
233	常时皮带断裂查出设定	0	电源 OFF 时的值	特殊机械手专用的参数	（Ver1.9 以上版本）
234	电流限制时的 ZT 干扰修正	0	电源 OFF 时的值	ZT 轴上有齿轮干扰时设定	通常情况下请不要进行变更。（Ver1.9 以上版本）
235	预约参数	0	电源 OFF 时的值	将来的功能扩展所需的预约参数	（Ver1.9 以上版本）
237	允许清空基准异常	0	电源 OFF 时的值	可以允许从 TP 上将基准位置异常进行清空的设定	通常情况下请不要进行变更。（Ver1.95 以上版本）
238	动作速度限制设定	0	电源 OFF 时的值	特殊机械手专用的参数	通常情况下请不要进行变更。（Ver1.95 以上版本）

编号	显示	出厂值	电源接通时	内容	备注
239	跟踪模式	0	电源 OFF 时的值	在其他用途跟踪设定画面进行设定。 （当前未对应）	在该画面上请不要进行变更。 （Ver1.95 以上版本）
240	识别时的编码器 1 基准位置	0			
241	动作时的编码器 1 基准位置	0			
242	识别时的编码器 2 基准位置	0			
243	动作时的编码器 2 基准位置	0			
244	编码器 1 当前值	0			
245	编码器 2 当前值	0			
246	编码器 1CALDTA				
247	编码器 2CALDTA	0			
248	工件位置检测精度 1	5			
249	跟踪范围上限值 1	20000			
250	跟踪范围下限值 1	-20000			
251	跟踪范围上限值 2	20000			
252	跟踪范围下限值 2	-20000			
253	跟踪起始范围 1（+侧）	20000			
254	跟踪起始范围 1（-侧）	-20000			
255	跟踪起始范围 2（+侧）	20000			
256	跟踪起始范围 2（-侧）	-20000			
257	中断使用功能	0			
258	编码器 1 中断次数	0			
259	编码器 2 中断次数	0			
260	编码器 1 中断设定	0			
261	编码器 2 中断设定	0			
262	编码器 1 中断数据更新	0			
263	编码器 2 中断数据更新	0			
264	编码器 1CALDAT (Z)	0			
265	编码器 2CALDAT (Z)	0			
266	工件查出位置精度 2	5			
267	编码器加减速度检查	0			
268	编码器 1 中断数据设定	0			
269	编码器 2 中断数据设定	0			
270	基准异常查出设定	0			
271	CP 直线高轨迹控制	0			
272	编码器速度加速度异常查出	1			
273	跟踪对象	0			

编号	显示	出厂值	电源接通时	内容	备注
274	索引 1 中心坐标 (X)	100000	电源 OFF 时的值	在其他用途跟踪设定画面进行设定。 （当前未对应）	请不要在该画面上变更。 （Ver1.95 以上版本）
275	索引 1 中心坐标 (Y)	100000			
276	索引 1 中心坐标 (Z)	100000			
277	索引 1 中心半径	100000			
278	索引 2 中心坐标 (X)	100000			
279	索引 2 中心坐标 (Y)	100000			
280	索引 2 中心坐标 (Z)	100000			
281	索引 2 中心半径	100000			
282	索引 1 半径方向跟踪上限值	1000			
283	索引 1 半径方向跟踪下限值	0			
284	索引 2 半径方向跟踪上限值	1000			
285	索引 2 半径方向跟踪下限值	0			
286	索引 1 基准坐标 (X)	0			
287	索引 1 基准坐标 (Y)	0			
288	索引 1 基准坐标 (Z)	0			
289	索引 2 基准坐标 (X)	0			
290	索引 2 基准坐标 (Y)	0			
291	索引 2 基准坐标 (Z)	0			
292	索引 1 姿势跟随	0			
293	索引 2 姿势跟随	0			
294	工件重复确认范围 1	200			
295	工件重复确认范围 2	200			
296	电机指令设定	0	电源 OFF 时的值	电机指令的异常检测设定 0：机械手动作过程中，因执行 MOTOR OFF 指令异常检测 1：机械手动作过程中，因执行 MOTOR OFF 指令异常未检测	（Ver1.98 以上版本）
297	伺服系统数据获取编号	0	电源 OFF 时的值	特殊机械手专用的参数	通常情况下请不要进行变更。 （Ver1.98 以上版本）
307	特异点回避功能有效设定	0	电源 OFF 时的值	特异点回避功能的有效/无效设定 0：无效 1：有效 （参照编程手册 I "特异点回避功能"）	（Ver2.61 以上版本） 可利用 SetSingularAvoid() 进行变更

编号	显示	出厂值	电源接通时	内容	备注
394	挠度修正功能的有效设定	0	电源 OFF 时的值	根据动作位置、姿势、负荷状况，自动修正机械手臂因重力而产生的挠度。 0：无效 1：有效	（Ver3.0 以上版本）
	注：如果在教导要点后使挠度修正有效，教导要点将发生变化。请在教导要点之前使之有效。此外，教导要点也可能因前端负荷质量及前端负荷重心位置而改变。教导要点是让前端负荷质量与前端负荷重心位置准确契合。				
398	特异点回避扩展功能	因机械手的型号而有所差异。 VS-050,060,068,087：1 除此之外的：0	电源 OFF 时的值	在通过动作进行过程中，进行特异点的回避。但是，要想使此功能有效，必须事先使特异点回避功能的有效设定生效（1）。 0：无效 1：有效	（Ver3.2 以上版本）
404	检查教导器时的轨道修正	1	电源 OFF 时的值	让检查教导器时的机械手轨道与自动运转时的轨迹相吻合。 0：有时，检查教导器时的通过动作的轨道与自动运转时的轨道不同。 1：检查教导器时的通过动作按照与自动运转时相同的轨迹动作。	（Ver2.8 以上版本）

22.5 保留字一览

这些保留字不能作为变量名和标签名使用。

A	ABOVE ABS ACCEL ACOS ADDCOMLOG ADDERRLOG ALL AND APPROACH AREA AREAPOS AREASIZE ARRIVE AS ASC ASIN ATN ATN2 AVEC
B	B BELOW BIN BIT BLOB BLOBCOPY BLOBLABEL BLOBMEASURE BREAK BUZZER BYTE
C	C CALCWORKPOS CALL CAMASPECT CAMIN CAMLEVEL CAMMODE CAMTIN CAMTOFF CAMTON CASE CHANGE_BCAP CHANGE_PCAP CHANGECOORD CHANGETOOL CHANGEWORK CHGEXTMODE CHGINTMODE CHR CLEARLOG CLOSECOMLOG CLOSEERRLOG CLRERR CODE COM_DISCOM COM_ENCOM COM_STATE CONTINUERUN CONT COS CREATESEM CURACC CURDEC CUREJNT CUREXJ CUREXTACC CUREXTDEC CUREXTSPD CURFIG CURJACC CURJDEC CURJNT CURJSPD CUROPTMODE CURPOS CURSPD CURTOOL CURTRACKPOS CURTRACKPOSEX CURTRACKSPD CURTRN CURWORK CYCLE
D	D DATE DECEL DEF DEFDBL DEFEND DEFINT DEFIO DEFJNT DEFPOS DEFSNG DEFSTR DEFTRN DEFVEC DEG DEGRAD DELAY DELETESEM DEPART DESTEXJ DESTJNT DESTPOS DESTTRN DETECT DIM DISP_PAGE DIST DO DOUBLE DRAW DRIVE DRIVEA
E	E EJ ELSE ELSEIF EMGSTOP END ENDIF EQJ EQP EQS EQT EQU EQV ERL ERR ERRMSG ERROR EX EXA EXECAL EXIT EXP EXTSPEED
F	F FALSE FIG FIGAPRL FIGAPRP FLIP FLOAT FLUSH FLUSHSEM FOR FORMAT FREE
G	GETCOMLOG GETENV GETERR GETERRLOG GETERRLVL GETJNTDATA GETSRVDATA GETSRVSTATE GIVEARM GIVEARMNOWAIT GIVESEM GIVEVIS GO GOHOME GOSUB GOTO GRASP
H	H HALT HAND HEX HMOVE HOLD HOME
I	I IF IN INIT INPUT INPUTB INPUTB$ INT INTEGER INTERRUPT IO IOBLOCK IS
J	J J2P J2T JOINT JSPEED JACCEL JDECEL
K	KEEP KILL KILLALL
L	L LEFT LEFTY LEN LET LETA LETENV LETF LETJ LETJ1 LETJ2 LETJ3 LETJ4 LETJ5 LETJ6 LETO LETP LETR LETRX LETRY LETRZ LETT LETX LETY LETZ LINEINPUT LINPUTB LOG LOG10 LOOP LPRINTB
M	MAGNITUDE MAX MID MIN MIRROR MOD MOTOR MOVE MPS
N	NEJ NEP NEQ NES NET NEV NEXT NONFLIP NORMTRN NOT
O	OFF ON OR ORD OUT OVEC
P	P P2J P2T PI POSCLR POSE POSITION POSRX POSRY POSRZ POST POSX POSY POSZ POW PRINT PRINTB PRINTDBG PRINTLBL PRINTMSG PRIORITY PROGRAM PTP PVEC
R	R RAD RADDEG RELEASE REM REPEAT RESET RESETAREA RESETVALVE RESULT RESUME RETURN RIGHT RIGHTY RND ROB ROBOT ROTATE ROTATEH RUN RVEC

S	S SEC SELECT SENDKEY SET SETAREA SETVALVE SGN SHADDGROUP SHCIRCLE SHCLRGROUP SHCLRMODEL SHCOPYMODEL SHCORNER SHDEFCIRCLE SHDEFCORNER SHDEFMODEL SHDISPMODEL SHDROPGROUP SHGROUP SHLOADMODEL SHMODEL SHREFGROUP SHREFMODEL SHSAVEMODEL SIN SINGLE SPEED SPRINTF SQR SQRT ST_ASPACLD ST_ASPCHANGE ST_OFFSRVLOCK ST_ONSRVLOCK ST_RESETCOMPCONTROL ST_RESETCOMPERALW ST_RESETCOMPJLIMIT ST_RESETCOMPRATE ST_RESETCOMPVMODE ST_RESETCURLMT ST_RESETDAMPRATE ST_RESETERALW ST_RESETFRCASSIST ST_RESETFRCLIMIT ST_RESETGRAVITY ST_RESETGRVOFFSET ST_RESETZBALANCE ST_SETCOMPCONTROL ST_SETCOMPERALW ST_SETCOMPFCONTROL ST_SETCOMPJLIMIT ST_SETCOMPRATE ST_SETCOMPVMODE ST_SETCURLMT ST_SETDAMPRATE ST_SETERALW ST_SETFRCASSIST ST_SETFRCCOORD ST_SETFRCLIMIT ST_SETGRAVITY ST_SETGRVOFFSET ST_SETZBALANCE STARTDATA STARTLOG STATUS STEP STOP STOPEND STOPLOG STARTTRI STR STRING STRPOS SUB SUSPEND SUSPENDALL SYSSTATE
T	T T2J T2P TACCEL TAKEARM TAKESEM TAKEVIS TAN TDECEL THEN TIME TIMER TINV TLSPEED TO TOOL TOOLPOS TRACKDATAGET TRACKDATAINFO TRACKDATAINITIALIZE TRACKDATANUM TRACKDATASET TRADIUS TRANS TRAVELD TRAVELG TRNS TRUE TS TSMOVE TSPEED TSPIN TSRSTATE TTURN
U	UNTIL
V	V VAL VECTOR VER VISBINA VISBINAR VISBRIGHT VISCAL VISCAMOUT VISCIRCLE VISCLS VISCOPY VISCROSS VISDEFCHAR VISDEFTABLE VISEDGE VISELLIPSE VISFILTER VISGETNUM VISGETP VISGETSTR VISHIST VISLEVEL VISLINE VISLOC VISMASK VISMEASURE VISOVERLAY VISPLNOUT VISPOSX VISPOSY VISPRINT VISPROJ VISPTP VISPUTP VISREADBAR VISREADQR VISRECT VISREFCAL VISREFHIST VISREFTABLE VISSCREEN VISSECT VISSTATUS VISWORKPLN
W	WAIT WAITTRACKMOVE WAITTRACKMOVEEX WDIN WOUT WEND WHILE WINDCLR WINDCOPY WINDDISP WINDMAKE WINDREF WORD WORK WORKPOS WRITE
X	X XOR XY XYH
Y	YZ YZH
Z	ZX ZXH

22.6 旧语言指令对照表(VS)

■动作指令对照表

旧语言指令	PAC 指令	MOVE 宏
MVE J*n*	MOVE P, J*n*	——
MVP J*n*	MOVE P, @P J*n*	——
MVSE P*n*	MOVE L, T*n*	——
MVSP Pn	MOVE L, @P T*n*	——
DRVE (*j1j2j3j4j5j6*)	DRIVE (1*j1*),…	Jn = Jc + (*j1j2j3j4j5j6*) MOVE P, *Jn*
DRVE J*n*	DRIVE (1,JOINT(1,J*n*)),…	*jj = Jc* + (JOINT (1,J*n*)),…,JOINT (6,J*n*) MOVE P, @P J*n*
DRVP (*j1j2j3j4j5j6*)	DRIVE @P (1*j1*),…	*Jn = Jc* + (*j1j2j3j4j5j6*) MOVE P, @P J*n*
DRVP J*n*	DRIVE @P (1,JOINT (1*j1*)),…	*jj = Jc* + (*JOINT (1,Jn)),…,JOINT* (*6,Jn*) MOVE P, *@P jj*
DRWE (*x,y,z*)	DRAW L, (*x,y,z*)	MOVE L, *Pc* + (*x,y,z)*
DRWE P*n*	DRAW L, PVEC (*Tn*)	MOVE L, *Pc* + (POSX (P*n*),POSY (P*n*), POSZ (P*n*))
DRWP (*x,y,z*)	DRAW L, @P (*x,y,z*)	MOVE L, @P *Pc* + (*x,y,z)*
DRWP P*n*	DRAW L, @P PVEC (T*n*)	MOVE L, @P *Pc* + (POSX (*Pn*), POSY (P*n*),POSZ (P*n*))
DEPE *n*	DEPART L,*n*	MOVE L, *Pc* + (0,0,−*n*) H
DEPP *n*	DEPART L, @P *n*	MOVE L, @P *Pc* + (0,0,−*n*) H
APRE *n*	APPROACH L, *Px, n*	MOVE L,*Px* + (0,0,−*n*) H
APRP *n*	APPROACH L, @P *Px, n*	MOVE L,@P *Px* + (0,0,−*n) H*
APRJE *n*	APPROACH P, *Px, n*	MOVE P,*Px* + (0,0,−*n*) H
APRJP *n*	APPROACH P, @P *Px, n*	MOVE P,@P *Px* + (0,0,−*n*) H
ROTE *n*	ROTATEH *n*	MOVE L, *Pc* + (*0,0,0,0,0,n*) H
ROTP *n*	ROTATEH @P *n*	MOVE L, @P *Pc* + (*0,0,0,0,0,n*) H
MVRE P*n2,*P*n3*	MOVE C, T*n2,* T*n3*	——
MVPR P*n2,*P*n3*	MOVE C, T*n2,* @P T*n3*	——

(!) Pc / Jc 为当前位置，*Px / Jx* 为下个目标位置。
(!) DRVE 只在非零的轴上可与 DRIVE 语句的自变量 (argument)并用。
(!) jj 是临时的局部 (Local) 变量。

■速度指定指令对照表

旧语言指令	PAC 指令	备注
ISP *n*	SPEED *n*	
ACC *n*	ACCEL *n,n*	
AACC *n*	ACCEL *n*	
RACC *n*	DECEL *n*	

■转移指令对照表

旧语言指令	PAC 指令	备注
JI *m-n*	IF IO [m]=1 THEN *labeln*	
JZ *m-n*	IF IO [m]=0 THEN *labeln*	
JMP *n*	GOTO *labeln*	
CMP *I s m*	IF *I s m* THEN *labeln*	*s* 为比较记号

旧语言指令	PAC 指令	备注
LABL *n*	*label*n.*	
LABL *n*	*label*n.*	
IPCLR *n*	CALL pltResetAll (*n*)	
INTRPT	INTERRUPT ON	在下个指令后进行 INTERRUPT OFF。
REM	REM|'	
ON *n* ON *n-m*	SET IO [*n*] SET IO [*n* TO *m*]	
OFF *n* OFF *n-m*	RESET IO [*n*] RESET IO [*n* TO *m*]	
ONT *n-m* TIME = *t*	SET IO [*n* TO *m*], t*10	单位是 msec.（注）PAC 时，不进入下个处理。
VON *n* VON *n-m*	SET IO [*n*] SET IO [*n* TO *m*]	
VOFF *n* VOFF *n-m*	RESET IO [*n*] RESET IO [*n* TO *m*]	
ON PLT1END	SET IO [*n*]	参照 DIO 配置表
OFF PLT1END	RESET IO [*n*]	参照 DIO 配置表
ON PLTEND	SET IO [*n*]	参照 DIO 配置表
OFF PLTEND	RESET IO [*n*]	参照 DIO 配置表
INB *L m-n*	CALL ndInb (*L,m,n*)	共用程序库 (Library) 调出
ONB *L m-n*	CALL ndOnb (*L,m,n*)	共用程序库 (Library) 调出
ONB I*x m-n*	CALL ndOnbl (*x,m,n*)	共用程序库 (Library) 调出

■停止指令对照表

旧语言指令	PAC 指令	备注
END	END	
STOP	HOLD	
STOPEND	STOPEND	
TIM *n*	DELAY *n** 10	

■SETI 指令对照表

旧语言指令	PAC 指令	备注
变量 6 轴 P*n*	T [*n*]	
间接参照 I*n*.P	P [*n*]	
$	CURPOS|*	
¥	CURJNT|*	
ISP	CURSPD	
AACC	CURACC	
RACC	CURDEC	
SETI I*x* = N_*n*	CALL pltGetN (*n,x*)	共用程序库 (Library) 调出
SETI I*x* = M_*n*	CALL pltGetM (*n,x*)	共用程序库 (Library) 调出
SETI I*x* = K_*n*	CALL pltGetK (*n,x*)	共用程序库 (Library) 调出
SETI I*x* = N1_*n*	CALL pltGetN1 (*n,x*)	共用程序库 (Library) 调出
SETI I*x* = M1_*n*	CALL pltGetM1 (*n,x*)	共用程序库 (Library) 调出
SETI I*x* = K1_*n*	CALL pltGetK1 (*n,x*)	共用程序库 (Library) 调出

■运算指令对照表

旧语言指令	PAC 指令	备注
+	+	加
-	-	减
*	*	乘
/	/	除
%	MOD	余数
.	*	内积（矢量用）
x	x	外积（矢量用）
ABS (*n*)	ABS (*n*)	
SIN (*n*)	SIN (*n*)	
COS (*n*)	COS (*n*)	
TAN (*n*)	TAN (*n*)	
ATAN (*n*)	ATAN (*n*)	
ATN2 (*y,x*)	ATN2 (*y,x*)	
SQRT (*n*)	SQRT (*n*)	
FWRD (J*n*)	J2P (J[*n*])	
REV2 (P*n*)	P2J (P[*n*])	
TRNS (P*m*,J*k*)	P*m* + (*x,y,z,α,β,γ*) H	参照下页的 "关于坐标系移动函数"
TINV (P*n*)	TINV (T*n*)	
DATE	CALL ndDate	正在检查共用程序库 (Library) 规格
TIME (0)	CALL ndTime	正在检查共用程序库 (Library) 规格
TIME (1)	TIMER	

■通信指令对照表

旧语言指令	PAC 指令	备注
VIS *n*	CALL ndVis (*n*)	共用程序库 (Library) 调出
JF *n-m*	CALL ndJf (*n,k*) IF I[*k*] = 0 THEN *label	共用程序库 (Library) 调出
VSET *n*	CALL ndVset ()	共用程序库 (Library) 调出
VDT	CALL ndVdt	共用程序库 (Library) 调出
VPUT $ VPUT P*n*	CALL ndVput ()	共用程序库 (Library) 调出
VRST	CALL ndVrst	共用程序库 (Library) 调出

■已定义的命令对照表

旧语言指令	PAC 指令	备注
SUB *n*	CALL SUB*n*	SUB*n* 为程序名
PALT *n*	CALL pltMove (*n*)	共用程序库 (Library) 调出
TOOL *n*	CHANGETOOL *n*	

关于坐标系移动函数

在 PAC 指令中，没有旧语言指令的坐标系移动函数（TRNS 指令），但通过位置运算（参照 "7.9.7 位置运算"）可以移动坐标系。
坐标系有采用 P 型变量的方法和采用 T 型变量的方法，各自可移动工件坐标基准和工具坐标基准。
对于 J 型变量移动坐标系时，请用 J2P、J2T 变换为 P 型或者 T 型变量。

(1) 采用 P 型变量的方法

① 以工件坐标系为基准指定相对移动量时

Pn = Pm + (X,Y,Z,Rx,Ry,Rz)

位置从基准位置Pm分别向工件坐标系的X轴、Y轴、Z轴方向移动X (mm)、Y (mm)、Z (mm)，姿势分别围绕X轴、Y轴、Z轴求取旋转Rx（度）、Ry（度）、Rz（度）的位置Pn。机械手的形态为Pm的形态。

② 以工具坐标系为基准指定相对移动量时

Pn = Pm + (X,Y,Z,Rx,Ry,Rz) H

位置从基准位置Pm分别向工件坐标系的X轴（常规矢量）、Y轴（定向矢量）、Z轴（近似矢量）方向移动X (mm)、Y (mm)、Z (mm)，姿势分别围绕X轴（常规矢量）、Y轴（定向矢量）、Z轴（近似矢量）求取旋转Rx（度）、Ry（度）、Rz（度）的位置Pn。机械手的形态为Pm的形态。

(2) 采用 T 型变量的方法

① 以工件坐标系为基准指定相对移动量时

```
DEFTRN LT1,LT2
DEFPOS LP1
LP1 = (0,0,0,Rx,Ry,Rz)
LT1 = P2T (LP1)
LT2 = Tm
LETP LT2 = (0,0,0)
Tn = LT1*LT2
LETP Tn = PVEC (Tm) + (X,Y,Z)
LETF Tn = FIG (Tm)
```

位置从基准位置Tm分别向工件坐标系的X轴、Y轴、Z轴方向移动X (mm)、Y (mm)、Z (mm)，姿势分别围绕X轴、Y轴、Z轴求取旋转Rx（度）、Ry（度）、Rz（度）的位置Tn。机械手的形态为Tm的形态。

② 以工具坐标系为基准指定相对移动量时

```
DEFTRN LT1
DEFPOS LP1
LP1 = (X,Y,Z,Rx,Ry,Rz)
LT1 = P2T (LP1)
Tn = Tm*LT1
LETF Tn = FIG (Tm)
```

位置从基准位置Tm分别向工件坐标系的X轴（常规矢量）、Y轴（定向矢量）、Z轴（近似矢量）方向移动X (mm)、Y (mm)、Z (mm)，姿势分别在X轴（常规矢量）周围、Y轴（定向矢量）周围、Z轴（近似矢量）周围求取旋转Rx（度）、Ry（度）、Rz（度）的位置Tn。机械手的形态为Tm的形态。

（注意）TRNS指令是以工具坐标系为基准的坐标系移动命令，但姿势的旋转方法与P型变量、T型变量所决定的工具坐标系移动命令在以下的点上不同。

	坐标系的姿势旋转方法
Js = (X,Y,Z,Rx,Ry,Rz) Pn = TRNS (Pm,Js)	从基准位置Pm的姿势，沿工具坐标系的X轴（常规矢量）旋转Rx（度）、沿旋转的工具坐标系的Y轴旋转Ry（度）、再沿旋转的工具坐标系的Z轴旋转Rz（度）。
Pn = Pm + (X,Y,Z,Rx,Ry,Rz) H	从基准位置Pm的姿势，沿工具坐标系的X轴（常规矢量）旋转Rx（度）、沿旋转前的工具坐标系的Y轴旋转Ry（度）、再沿旋转前的工具坐标系的Z轴旋转Rz（度）。

与旧语言的TRNS指令等同的姿势旋转，请如下指定。

(1) 采用P型变量时

```
Pn = Pm + (X,Y,Z,0,0,0) H
Pn = Pn + (0,0,0,Rx,0,0) H
Pn = Pn + (0,0,0,0,Ry,0) H
Pn = Pn + (0,0,0,0,0,Rz) H
```

(2) 采用T型变量时

```
DEFTRN LT
DEFPOS LP
LP = (X,Y,Z,0,0,0)
LT = P2T (LP)
Tn = Tm*LT
LP = (0,0,0,Rx,0,0)
LT = P2T (LP)
Tn = Tn*LT
LP = (0,0,0,0,Ry,0)
LT = P2T (LP)
Tn = Tn*LT
LP = (0,0,0,0,0,Rz)
LT = P2T (LP)
Tn = Tn*LT
LETF Tn = FIG (Tm)
```

22.7 版本对应表

在VER$函数上所使用的算式的值和模块的对照情况列于下表。
关于VER$，请参照 p19-3 "19.1系统、VER$（函数）"。

版本对应表

算式的值	模块名称
0	ROM
1	主板
2	DIO 板
3	视觉卡
4	装置网卡
5	多功能教导器
6	操作面板
7	PAC 语言处理模块
8	中间编码处理模块
9	轨道处理模块
10	伺服系统处理模块
11	通信处理模块
12	教导器处理模块
13	视觉处理模块
14	视觉通信处理模块
15	装置网络处理模块
16	PAC 语言规格
17	中间编码规格
18	伺服系统通信规格
19	视觉通信规格
20	ROBOTalk 通信规格
21	教导器通信规格
22	装置网络通信规格

22.8 设定参数表

■Pac Manager-程序

项目名称	宏名称	说明
I 型变量的数	PC_NO_INT	带有编号的长整数型变量的数
F 型变量的数	PC_NO_SNG	带有编号的单精度实数型变量的数
D 型变量的数	PC_NO_DBL	带有编号的双精度实数型变量的数
V 型变量的数	PC_NO_VEC	带有编号的矢量型变量的数
P 型变量的数	PC_NO_POS	带有编号的位置型变量的数
J 型变量的数	PC_NO_JNT	带有编号的关节型变量的数
T 型变量的数	PC_NO_TRN	带有编号的齐次变换型变量的数
S 型变量的数	PC_NO_STR	带有编号的字符串型变量的数
近似比较精度 (*10 ^ 6)	PC_EPSILON	近似比较运算符 (=.) 的允许偏差量 (× 1000000)
摹写编码的删除	PC_NO_TRACE	从编译程序编码上删除调试编码 （0：不删除， 2：全部删除）
循环时间计算编码的删除	PC_NO_CYC_TIME	从编译程序编码中删除循环定时器计算编码 （0：不删除，1：部分删除，2：全部删除）
排列范围检查编码的删除	PC_NO_CHK_INDEX	从编译程序编码中删除排列范围检查编码 （0：不删除，1：删除）
错误中断处理编码的删除	PC_NO_ERR_TRAP	从编译程序编码中删除错误中断处理编码 （0：不删除，1：删除）

■Dio Manager -硬件

项目名称	宏名称	说明
I / O 配置 ■I / O 硬件选择 ■I / O 配置模式的设定	IO_ASSIGN	■I / O 硬件选择 （并行、DeviceNet 子局、DeviceNet 主局、PROFIBUS-DP 子局） ■I / O 的配置模式的设定 （互换模式，标准模式）
奇偶性比特位（0：无效，1：有效）	IO_PARITY	程序启动时的程序 No 选择奇偶的设定 奇偶性比特位（0：无效，1：有效）
HandIO 中断设定 （0：无效，1：有效）	IO_HD_ENABLED	HandIO 中断设定　（0：无效，1：有效）
DeviceNet 输入插槽数	IO_DN_IN_SLOT	DeviceNet 的输入插槽数
DeviceNet 输出插槽数	IO_DN_OUT_SLOT	DeviceNet 的输出插槽数
DeviceNet 准备异常检测 （0：无效，1：有效）	IO_DN_EXERRCHK	DeviceNet 的准备异常检测　（0：无效，1：有效）
专用 I / O 输出模式 （0：无效，1：有效）	IO_DEDICTDIO	专用 I / O 输出模式（0：无效，1：有效）
M-Dnet 异常显示（0：每次，1：第一次）	IO_MDN_ERRDISP	M-Dnet 异常显示（0：每次，1：第一次）
PROFIBUS 节点地址 (1～125)	IO_SPRFI_ADRS	PROFIBUS 节点地址 (1～125)
PROFIBUS 输入设定 (0:8 / 1:12 / 2,16 / 3:20 / 4:32)	IO_SPRFI_INTYPE	PROFIBUS 输入设定 0 : 8 字节 (Byte) Output con 1 : 12 字节 (Byte) Output con 2 : 16 字节 (Byte) Output con 3 : 20 字节 (Byte) Output con 4 : 32 字节 (Byte) Output con
PROFIBUS 输出设定 (0:8 / 1:12 / 2,16 / 3:20 / 4:32)	IO_SPRFI_OUTTYPE	PROFIBUS 输出设定 0 : 8 字节 (Byte) Input con 1 : 12 字节 (Byte) Input con 2 : 16 字节 (Byte) Input con 3 : 20 字节 (Byte) Input con 4 : 32 字节 (Byte) Input con

■Arm Manager -轨道生成

项目名称	宏名称	说明
正方向软件限位 (J1,deg*10 ^ 3)	AM_JPRM_PLIM1	1 轴的正方向软件限位（×1000，单位：度）
正方向软件限位 (J2,deg*10 ^ 3)	AM_JPRM_PLIM2	2 轴的正方向软件限位（×1000，单位：度）
正方向软件限位 (J3,deg*10 ^ 3)	AM_JPRM_PLIM3	3 轴的正方向软件限位（×1000，单位：度）
正方向软件限位 (J4,deg*10 ^ 3)	AM_JPRM_PLIM4	4 轴的正方向软件限位（×1000，单位：度）
正方向软件限位 (J5,deg*10 ^ 3)	AM_JPRM_PLIM5	5 轴的正方向软件限位（×1000，单位：度）
正方向软件限位 (J6,deg*10 ^ 3)	AM_JPRM_PLIM6	6 轴的正方向软件限位（×1000，单位：度）
正方向软件限位 (J7,deg*10 ^ 3)	AM_JPRM_PLIM7	7 轴的正方向软件限位（×1000，单位：度）
正方向软件限位 (J8,deg*10 ^ 3)	AM_JPRM_PLIM8	8 轴的正方向软件限位（×1000，单位：度）
负方向软件限位 (J1,deg*10 ^ 3)	AM_JPRM_NLIM1	1 轴的负方向软件限位（×1000，单位：度）
负方向软件限位 (J2,deg*10 ^ 3)	AM_JPRM_NLIM2	2 轴的负方向软件限位（×1000，单位：度）
负方向软件限位 (J3,deg*10 ^ 3)	AM_JPRM_NLIM3	3 轴的负方向软件限位（×1000，单位：度）
负方向软件限位 (J4,deg*10 ^ 3)	AM_JPRM_NLIM4	4 轴的负方向软件限位（×1000，单位：度）
负方向软件限位 (J5,deg*10 ^ 3)	AM_JPRM_NLIM5	5 轴的负方向软件限位（×1000，单位：度）
负方向软件限位 (J6,deg*10 ^ 3)	AM_JPRM_NLIM6	6 轴的负方向软件限位（×1000，单位：度）
负方向软件限位 (J7,deg*10 ^ 3)	AM_JPRM_NLIM7	7 轴的负方向软件限位（×1000，单位：度）
负方向软件限位 (J8,deg*10 ^ 3)	AM_JPRM_NLIM8	8 轴的负方向软件限位（×1000，单位：度）
RANG (J1,deg*10 ^ 5)	AM_JPRM_RANG1	1 轴的 RANG 值（×100000，单位：度）

■Arm Manager -使用条件

项目名称	宏名称	说明
最佳可搬运质量设定模式	CND_ASPACC	最佳可搬运质量设定模式 （0：通常，1：仅限 PTP 有效，2：仅限 CP 有效， 3：PTP / CP 均有效）
地面设置、悬挂、壁挂设定	CND_SPACE	地面设置、悬挂或壁挂的设定 （0：地面设置，1：悬挂，2：壁挂）
前端负载质量 (g)	CND_LOAD	机械手前端的负荷质量（单位：g）
负荷重心位置 X6 (mm)	CND_LDPOSGX	机械手前端的负荷重心位置的 X 分量（单位：mm）
负荷重心位置 Y6 (mm)	CND_LDPOSGY	机械手前端的负荷重心位置的 Y 分量（单位：mm）
负荷重心位置 Z6 (mm)	CND_LDPOSGZ	机械手前端的负荷重心位置的 Z 分量（单位：mm）
停止时允许脉冲宽度 (J1)	CND_PLSWDTH1	1 轴的停止时允许脉冲宽度（单位：pulse)
停止时允许脉冲宽度 (J2)	CND_PLSWDTH2	2 轴的停止时允许脉冲宽度（单位：pulse)
停止时允许脉冲宽度 (J3)	CND_PLSWDTH3	3 轴的停止时允许脉冲宽度（单位：pulse)
停止时允许脉冲宽度 (J4)	CND_PLSWDTH4	4 轴的停止时允许脉冲宽度（单位：pulse)
停止时允许脉冲宽度 (J5)	CND_PLSWDTH5	5 轴的停止时允许脉冲宽度（单位：pulse)
停止时允许脉冲宽度 (J6)	CND_PLSWDTH6	6 轴的停止时允许脉冲宽度（单位：pulse)
停止时允许脉冲宽度 (J7)	CND_PLSWDTH7	7 轴的停止时允许脉冲宽度（单位：pulse)
停止时允许脉冲宽度 (J8)	CND_PLSWDTH8	8 轴的停止时允许脉冲宽度（单位：pulse)
动作结束超时 (msec)	CND_MVTIMOUT	指定@E 时的动作结束超时时间（单位：msec）

■Vision Manager–一般设定

项目名称	宏名称	说明
照像机 1－快门方式	CA_SHUT1	照像机 1 的快门方式（0：半帧、1：整帧）
照像机 1－输入下限水平	CA_LEVEL_L1	照像机 1 的输入水平下限
照像机 1－输入上限水平	CA_LEVEL_H1	照像机 1 的输入水平上限
照像机 1－是否可以使用照像机	CA_CONNECT1	是否可以使用照像机 1（0：可使用、1：未连接、2：选择、3：无）
照像机 2－快门方式	CA_SHUT2	照像机 2 的快门方式（0：半帧、1：整帧）
照像机 2－输入下限水平	CA_LEVEL_L2	照像机 2 的输入水平下限
照像机 2－输入上限水平	CA_LEVEL_H2	照像机 2 的输入水平上限
照像机 2－是否可以使用照像机	CA_CONNECT2	是否可以使用照像机 2 （0：可使用、1：未连接、2：选择、3：无）
照相机同步方式	CA_SYNC	照像机的同步方式 （0：照像机内部同步。1：照像机外部同步）
ShCorn－距离	SH_CORN_DIST1	ShCorner 命令的测量条件（凸缘和棱角的距离）1～10
ShCorn－间隔	SH_CORN_DIST2	ShCorner 命令的测量条件（凸缘和凸缘的间隔）1～10
ShCorn－宽度	SH_CORN_WIDTH	ShCorner 命令的测量条件（凸缘的宽度）1～10
ShCorn－高度	SH_CORN_HEIGHT	ShCorner 命令的测量条件（凸缘的高度）1～10
ShCirc－间隔	SH_CIRC_DIST	ShCircle 命令的测量条件（凸缘和凸缘的间隔）1～10
ShCirc－宽度	SH_CIRC_WIDTH	ShCircle 命令的测量条件（凸缘宽度）1～10
搜索超时时间 (ms)	SH_TIMEOUT	搜索测量的超时时间
视觉监视器显示位置	VS_DISP_LOC	处理画面在视觉监视器上的显示位置（0：中央、1：左、2：右）

■Log Manager –通信规格

项目名称	宏名称	说明
超时时间 (msec)	COM_RTK_TOUT	通信时的超时时间（单位：msec）
超时发生时的重复次数	COM_RTK_TRETRY	通信超时发生时的重复次数
NAK 发生时的重复次数	COM_RTK_RETRY	NAK（分组异常）发生时的重复次数
RS232C (2) .通信权	COM_RS2_ACCESS	控制器的 RS232C 频道 2 的通信权的设定 （0：不能使用，1：只可读入，2：可读入 / 写入）
RS232C (2) .波特率	COM_RS2_SPEED	控制器的 RS232C 频道 2 的波特率的设定（单位：bps）
RS232C (2) .奇偶 （0：偶数，1：无，2：奇数）	COM_RS2_PARITY	控制器的 RS232C 频道 2 的奇偶的设定（0：偶数，1：无，2：奇数）
RS232C (2) .数据长度	COM_RS2_DATALEN	控制器的 RS232C 频道 2 的数据长度的设定（单位：比特）
RS232C (2) .结束比特 (0:1 比特, 1:1.5 比特, 2:2 比特)	COM_RS2_STOPBIT	控制器的 RS232C 频道 2 的结束比特的设定(0:1 比特,1:1.5 比特,2:2 比特)
RS232C (2) .间隔符 (0:CR, 1:CR+LF)	COM_RS2_DELIMITER	控制器的 RS232C 频道 2 的间隔符的设定 (0:CR, 1:CR+LF)
RS232C (3) .通信权	COM_RS3_ACCESS	控制器的 RS232C 频道 3 的通信权的设定 （0：不能使用，1：只可读入，2：可读入 / 写入）
RS232C (3) .波特率	COM_RS3_SPEED	控制器的 RS232C 频道 3 的波特率的设定（单位：bps）
RS232C (3) .奇偶 （0：偶数，1：无，2：奇数）	COM_RS3_PARITY	控制器的 RS232C 频道 3 的奇偶的设定 （0：偶数，1：无，2：奇数）
RS232C (3) .数据长度	COM_RS3_DATALEN	控制器的 RS232C 频道 3 的数据长度的设定 （单位：比特）
RS232C (3) .结束比特 (0:1 比特, 1:1.5 比特, 2:2 比特)	COM_RS3_STOPBIT	控制器的 RS232C 频道 3 的结束比特的设定(0:1 比特,1:1.5 比特,2:2 比特)
RS232C (3) .间隔符 (0:CR, 1:CR + LF)	COM_RS3_DELIMITER	控制器的 RS232C 频道 3 的间隔符的设定 (0:CR, 1:CR+LF)

项目名称	宏名称	说明
RS232C (4) .通信权	COM_RS4_ACCESS	控制器的 RS232C 频道 4 的通信权的设定 （0：不能使用，1：只可读入，2：可读入 / 写入）
RS232C (4) .波特率	COM_RS4_SPEED	控制器的 RS232C 频道 4 的波特率的设定（单位：bps）
RS232C (4) .奇偶 （0：偶数，1：无，2：奇数）	COM_RS4_PARITY	控制器的 RS232C 频道 4 的奇偶的设定（0：偶数，1：无，2：奇数）
RS232C (4) .数据长度	COM_RS4_DATALEN	控制器的 RS232C 频道 4 的数据长度的设定（单位：比特）
RS232C (4) .结束比特 (0:1 比特, 1:1.5 比特, 2:2 比特)	COM_RS4_STOPBIT	控制器的 RS232C 频道 4 的结束比特的设定(0:1 比特,1:1.5 比特,2:2 比特)
RS232C (4) .间隔符 (0:CR, 1:CR+ LF)	COM_RS4_DELIMITER	控制器的 RS232C 频道 4 的间隔符的设定 (0:CR, 1:CR+LF)
Ethernet.通信权	COM_ET_ACCESS	控制器的 Ethernet 的通信权的设定 （0：不能使用，1：只可读入，2：可读入 / 写入）
Ethernet.IP 地址	COM_ET_IP_ADDR	控制器的 Ethernet 的 IP 地址的设定
Ethernet 辅助网络掩码	COM_ET_SUBNET_MASK	控制器的 Ethernet 的辅助网络掩码的设定
Ethernet 连接地址 1 本地端口	COM_ET_LPORT1	控制器的 Ethernet 连接地址 1 本地端口的设定
Ethernet 连接地址 2 本地端口	COM_ET_LPORT2	控制器的 Ethernet 连接地址 2 本地端口的设定
Ethernet 连接地址 3 本地端口	COM_ET_LPORT3	控制器的 Ethernet 连接地址 3 本地端口的设定
Ethernet 连接地址 4 本地端口	COM_ET_LPORT4	控制器的 Ethernet 连接地址 4 本地端口的设定
Ethernet 连接地址 5 本地端口	COM_ET_LPORT5	控制器的 Ethernet 连接地址 5 本地端口的设定

DENSO 机械手

编程手册Ⅰ　基础知识与指令

初　版　2008 年 1 月
第 2 版　2009 年 4 月
第 3 版　2011 年 9 月

DENSO WAVE INCORPORATED

9N**C

● 未经允许禁止复制或转载本使用说明书的部分或全部内容。
● 本说明书的内容若有变动，恕不另行通知。
● 关于本说明书的内容，在编辑时虽然力求万无一失，若发现不当之处、错误以及遗漏等情况，请与本公司联系。
● 对于使用本说明书所造成的后果及影响，本公司概不负责，敬请谅解。