ROBOSTAR ROBOT

N1 Series

程序说明书

- ☐ 操作说明书
- ☐ 操作及运用说明书
- ☑ 程序说明书
- ☐ Uni-Host 说明书
- ☐ GAIN 设置
- ☐ 报警代码说明书

Robostar

www.robostar.co.kr

ROBOSTAR ROBOT

N1 Series

PROGRAMMING MANUAL

- □ INSTRUCTION MANUAL
- □ OPERATION MANUAL
- √ PROGRAMMING MANUAL
- □ UNI-HOST MANUAL
- GAIN SETUP MANUAL
- ALARM CODE MANUAL

Robostar

www.robostar.co.kr

配置升级，不另作

关于质量保证

本公司产品质量严格，全部产品保修期均为一年。保修期内出现的因机器本身故障或者在正常使用情况下因机器设计和制造上的问题发生的故障均免费维修。

以下几种情况，不在免费维修范围内：

(1) 超过保修期。

(2) 因用户或是第三方的不正当修理、改造、移动等造成的故障。

(3) 因使用本公司以外的零配件及润滑脂引发的故障。

(4) 因火灾、地震、台风、水灾等灾害导致的机器故障。

(5) 因粪尿及进水等外部环境引起的非机器配置故障。

(6) 耗材消耗引起的故障。

(7) 未按照产品使用说明书的要求进行定期检查造成的故障。

(8) 机器维修以外的费用和本公司无关。

ROBOSTAR 地址及联系方法

- 总公司及第一厂
  京畿道安山市常绿区沙四洞 119-38
  119-38, Sasa-dong, Sangnok-gu,
  Ansan-City, Gyeonggi-do, Republic of
  South Korea (426-220)

- 第二工厂
  京畿道水原市劝善区古素洞 945
  960, Gosaek-dong, Gwonseon-gu,
  Suwon-City, Gyeonggi-do, Republic of
  South Korea (441-813)

售后服务及产品资讯
-业务咨询
TEL. 031-400-3600
FAX. 031-419-4249
-客户中心
TEL. 1588-4428

Robostar

www.robostar.co.kr

使用说明书的构成

本产品使用说明书如下。初次使用本产品请认真阅读产品使用说明书。

- **操作说明书**

  对控制器全面说明。对控制器概要，安装及控制器与外部机器的连接方法的说明

- **操作及运用说明**

  对控制器使用方法全面说明。同时，说明如何设定参数，编辑程序，如何启动.

- **程序说明书**

  关于ROBOSTAR ROBOT 程序 RRL (Robostar Robot Language)。
  对RRL (Robostar Robot Language)程序语言及依据RRL进行的编程方法的说明

- **Unihost说明**

  对本公司的在线电脑程序—Uni-host说明.

- **GAIN 设置说明书**

  试运行时必要的 GIAN 设置方法和 GIAN值的变更所应答的说明

- **报警代码说明书**

  控制器运行中发生的报警情况，分析原因及措施的说明.

目录

第 1 章 概要 1-1

1.1 概要 1-1

第 2 章 命令语一览 2-1

2.1 命令语画面显示 2-1
2.2 命令语一览表 2-2
2.2.1 关于程序控制命令语 2-2
2.2.2 关于ROBOT动作命令语 2-3
2.2.3 关于输入输出命令语 2-4
2.2.4 关于ROBOT动作条件命令语 2-5
2.2.5 变数说明 2-6
2.2.6 关于程序控制命令语 2-6
2.2.7 系统变数 2-8
2.2.8 常数 2-8
2.2.9 运算符 2-9
2.2.10 字符串 2-10

第 3 章 命令语说明 3-1

3.1 MAIN, EOP (程序 开始/结束) 3-1
3.1.1 程序使用例题 3-1
3.2 FOR, NEXT (重复执行文) 3-2
3.2.1 程序使用例题 3-2
3.3 WHILE, ENDWL (条件反复执行命令文) 3-4
3.3.1 程序使用例题 3-4
3.3.2 条件式使用例题 3-5
3.4 IF, ENDIF (条件分歧命令语) 3-6
3.4.1 程序使用例题 3-7
3.4.2 IF(条件式) THEN 3-8
3.5 LABL, GOTO (分歧命令语) 3-9
3.5.1 程序使用例题 3-9
3.6 SUBR, RET (副执行文命令语) 3-11
3.6.1 程序使用例题 3-11
3.7 CALL, JCALL (呼叫命令语) 3-12

3.7.1 程序使用例题 *3-13*
3.8 STOP, EXIT (ROBOT 及 JOB 停止命令语) 3-14
3.8.1 程序使用例题 *3-14*
3.9 JMOV (PTP 移动命令语) 3-15
3.9.1 程序使用例题 *3-15*
3.10 LMOV (CP 移动命令语) 3-17
3.10.1 让TOOL保持一定的方向提取(SCARA) *3-18*
3.10.2 程序使用例题 *3-19*
3.11 CMOV (圆形插补移动命令语) 3-20
3.11.1 程序使用例题 *3-20*
3.12 AMOV (圆弧插补移动命令语) 3-21
3.12.1 程序使用例题 *3-21*
3.13 IMOV, IMOV2 (增量移动命令语) 3-22
3.13.1 程序使用例题 *3-23*
3.14 JNTSYN (PTP同步模式命令语) 3-24
3.14.1 示例程序使用 *3-25*
3.15 EECH (终端装置选择命令语) 3-26
3.15.1 程序使用示例 *3-28*
3.16 TIMOV (TOOL 坐标系标准增量移动命令语) 3-29
3.17 HMOV (原点移动命令语) 3-30
3.17.1 程序使用例题 *3-30*
3.18 PMOV (PALLETZING 移动) 3-31
3.18.1 提取时保持移动的TOOL方向(SCARA) *3-32*
3.18.2 Pallet 作业 例 *3-33*
3.18.3 程序使用例题 (1) *3-36*
3.18.4 程序使用例题 (2) *3-38*
3.18.5 程序使用例题 (3) *3-41*
3.19 PASS 3-43
3.19.1 说 明 *3-43*
3.19.2 程序使用例题 (1) *3-43*
3.19.3 程序使用例题 (2) *3-44*
3.20 WITH, ENDWT (同时处理命令语) 3-45
3.20.1 程序使用例题 (1) *3-45*
3.21 OUT, POUT (外部输出命令语) 3-47
3.21.1 程序使用例题 *3-49*
3.22 IN, PIN (外部输入命令语) 3-50
3.22.1 程序使用例题 *3-52*

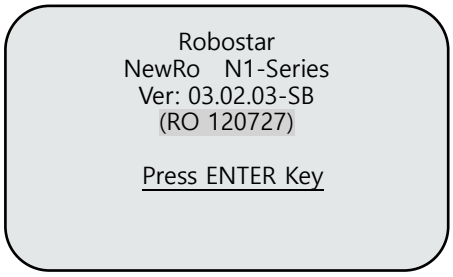
3.23 CIN,CBIN,CWIN,CDIN,CFIN(输入FIELDBUS 命令语)....3-53
3.23.1 程序使用例题....*3-54*
3.24 COUT,CBOUT,CWOUT,CDOUT,CFOUT(输出FIELDBUS 命令语)....3-55
3.24.1 程序使用例题....*3-56*
3.25 VEL (设置轴移动速度命令语句)....3-57
3.25.1 程序使用例题....*3-57*
3.26 ACC, DEC (设定加减速命令语)....3-58
3.26.1 程序使用例题....*3-59*
3.27 FOS, PFOS, (生成连续轨迹命令语)....3-60
3.27.1 程序使用例题....*3-61*
3.28 SFOS(生成连续轨迹及等速 命令语)....3-63
3.28.1 连续运动速度 文件模式....*3-65*
3.28.2 程序使用例题....*3-66*
3.29 SVON, SVOF (伺服 ON/OFF 命令语)....3-67
3.29.1 程序使用例题....*3-67*
3.30 DLAY (延迟时间命令语)....3-69
3.30.1 程序使用例题....*3-69*
3.31 OFFS (指定偏差命令语)....3-70
3.31.1 程序使用例题....*3-70*
3.32 LIMT (限制轴命令语)....3-71
3.32.1 程序使用例题....*3-71*
3.33 PLUP (PULL UP 动作设定命令语)....3-72
3.33.1 程序使用例题....*3-73*
3.34 TOOL (限制轴命令语)....3-74
3.34.1 程序使用例题....*3-75*
3.35 FIX (限制轴命令语)....3-76
3.35.1 程序使用例题....*3-77*
3.36 FORM (指定 ROBOT ARM形状命令语)....3-78
3.36.1 程序使用例题....*3-78*
3.37 TRQ (冲突感知命令语)....3-79
3.37.1 程序使用例题....*3-79*
3.38 TQL (限制轴输出扭矩 命令语)....3-80
3.38.1 程序使用例题....*3-80*
3.39 INPOS (指定目标到达命令语)....3-81
3.39.1 程序使用例题....*3-81*
3.40 MINIT(MAPPING初始化命令语)....3-82
3.41 MSTART(MAPPING检查命令语)....3-83

3.42 MAPPING传感器输入端口关联 3-84
3.43 MREAD(MAPPING位置数据更新命令语) 3-86
3.43.1 程序使用举例 *3-86*
3.44 RSTATE (确认 ROBOT状态 命令语) 3-88
3.44.1 程序使用例题 *3-89*
3.45 RERROR (确认报警代码命令语) 3-90
3.45.1 程序使用例题 *3-90*
3.46 GFTOGP (把 GF 保存到GP的命令语) 3-91
3.46.1 程序使用例题 *3-91*
3.47 REMCMD (SYSTEM COMMAND) 3-92
3.47.1 程序使用例题 *3-92*
3.48 变数 3-93
3.48.1 变数种类 *3-93*
3.48.2 使用方法 *3-94*
3.49 整数型(INT, I, II), 实数型 (REAL, F) 变数 3-95
3.49.1 LOCAL 变数 *3-95*
3.49.2 程序使用例题 *3-95*
3.49.3 整数型 GLOBAL 变数 I, II *3-96*
3.49.4 程序使用例题 *3-96*
3.49.5 实数型 GLOBAL 变数 F *3-97*
3.49.6 程序使用例题 *3-97*
3.50 POSITION 变数 3-98
3.50.1 POS 变数 *3-98*
3.50.2 POINT 变数 *3-99*
3.50.3 程序使用例题 *3-99*
3.51 系统变数 (CNT, TMR, MVR, HERE) 3-100
3.51.1 CNT, TMR 变数 *3-100*
3.51.2 程序使用例题 *3-100*
3.51.3 MVR 变数 *3-102*
3.51.4 程序使用例题 *3-103*
3.51.5 HERE 变数 *3-104*
3.51.6 程序使用例题 *3-104*
3.52 常数 3-105
3.52.1 程序使用例题 *3-105*
3.53 计算式 3-106
3.53.1 分配计算式 *3-106*
3.53.2 数学计算式 *3-107*

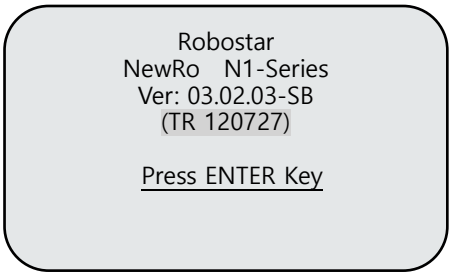
3.53.3 比较计算式 3-108
3.53.4 逻辑计算式 3-108
3.53.5 BIT 计算式 3-108
3.54 内置函数 3-109
3.55 ASC 3-110
3.55.1 程序使用例题 3-110
3.56 CHR 3-111
3.56.1 程序使用例题 3-111
3.57 FLUSH 3-112
3.57.1 程序使用例题 3-112
3.58 FTOS 3-113
3.58.1 程序使用例题 3-113
3.59 HTOS 3-114
3.59.1 程序使用例题 3-114
3.60 SLEFT 3-115
3.60.1 程序使用例题 3-115
3.61 SLEN 3-116
3.61.1 程序使用例题 3-116
3.62 SMID 3-117
3.62.1 程序使用例题 3-117
3.63 SPOS 3-118
3.63.1 程序使用例题 3-118
3.64 SRIGHT 3-119
3.64.1 程序使用例题 3-119
3.65 STRIN 3-120
3.65.1 程序使用例题 3-120
3.66 STROUT 3-121
3.66.1 程序使用例题 3-121
3.67 SVAL 3-122
3.67.1 程序使用例题 3-122

● N1 控制器版本

N1 控制器根据机器人类型的不同分为两个运用软件，各自的运用软件在N1控制器电源ON状态时在Teach Pendant画面中确认。



Scara机器人、直角机器人、DeskTop、机器人操作软件(RO)



半导体运送用机器人操作软件(TR)

● 根据控制器版本变更事项

VER.	RO(Robot)	TR(Transfer Robot)
Global INT	500EA(0~499)	2000EA(0~1999)
Global FLOAT	500EA(0~499)	2000EA(0~1999)
Local Point	2000EA(0~1999)	2000EA(0~1999)
Global Point	1024EA(0~1023)	15000EA(0~14999)

本手册中说明的内容是以RO(Robot)版本为标准，如果使用TR(Transfer Robot)版本而变更的部分请参照上表。

注意
- ▶ 机器人运行前必须确认操作软件和机器人类型。
- ▶ 所使用的机器人和N1控制器的操作软件需一致才可能正常操作。

第 1 章 概要

1.1 概要

1. ＲＯＢＯＴ语言Robostar的专用语在系统内部制作ＲＯＢＯＴ程序使用的命令语
2. ＲＯＢＯＴ语言大体分 4 GROUPE

- FLOW GROUPE
  程序的条件或无条件分歧，重复次数，呼叫副程序等设置的命令语构成
- MOVE GROUPE
  ＲＯＢＯＴ的动作条件(PTP, 补间, Palletzing 等)的设置命令语构成
- I/O GROUPE
  外部输入输出控制(16bit, 1bit)的命令语构成
- COND(=Condition) GROUPE
  速度，延时等设置的命令语构成

使用合适的命令语编写所需要的程序

第 2 章　命令语一览

2.1　命令语画面显示

F1　F2　F3　F4

GROUPE	选择键	FLOW	MOVE	I/O	COND
选择键		F1	F2	F3	F4
画面移动	F1	MAIN	JMOV	OUT	VEL
	F2	FOR	LMOV	POUT	FOS
	F3	IF	CMOV	IN	DLAY
	F4	WHILE	AMOV	PIN	INT
	F1	EOP	IMOV	CIN	ACC
	F2	NEXT	PMOV	CWIN	PFOS
	F3	ELSE	HMOV	COUT	PLUP
	F4	ENDWL		CWOUT	REAL
	F1	SUBR	SVON	SYS	FORM
	F2	RET	SVOF		TOOL
	F3	CALL	WITH		PCLR
	F4	ENDIF	ENDWT		POS
	F1	LABL	MVR		LIMT
	F2	GOTO	HERE		OFFS
	F3	JCALL	PASS		CNT
	F4		PCNT		INPOS
	F1	STOP			TMR
	F2	EXIT			NO
	F3				LEFT
	F4				RIGHT
	F1				FIX
	F2				TRQ
	F3				TQL
	F4				XCHG
	F1				DEC
	F2				
	F3				
	F4				

2.2 命令语一览表

2.2.1 关于程序控制命令语

命令语	功 能	形 式	使用例题
MAIN	开始程序	MAIN : EOP	MAIN VEL 200 JMOV P1 JMOV P2 EOP
EOP	结束程序		
FOR	重复运行STEP	FOR <变数>=<初始值> TO <结束值> [BY<增量>] : NEXT *** 变数只使用整数型变数**	: FOR A=1 TO 5 BY 2 VEL 200 JMOV P1 NEXT :
NEXT			
WHILE	条件重复运行	WHILE<条件式> : ENDWL	: WHILE IN0==1 JMOV P1 JMOV P2 ENDWL :
ENDWL			

IF	条件判断运行	IF<条件式> THEN
		:
		(ELSE)
		:
		ENDIF
			:
			IF IN3==1 THEN
			GOTO A0
			ELSE
			JMOV P1
			ENDIF
			:

ENDIF			
LABL	指定分歧点位置	LABL <LABL名> : GOTO <LABL名>	: LABL A1 JMOV P1 JMOV P2 GOTO A1 :
GOTO	运行分歧文		
SUBR	指定副执行文	SUBR<副执行文名> : RET	: SUBR HON GOTO A0 ELSE JMOV P1 RET :
RET	副执行文复位		
CALL	呼叫副执行文	CALL <副执行文名>	
JCALL	呼叫 JOB文件	JCALL<JOB 文件名>	
STOP	停止 ROBOT动作	STOP	
EXIT	停止 JOB运行	EXIT	

2.2.2 关于ROBOT动作命令语

命令语	功 能	形 式	使用例题
JMOV	从现在的位置向 目标点 PTP移动	JMOV <点位变数>	MAIN VEL 200 JMOV P1 LMOV P2 EOP
LMOV	从现在的位置向 目标点 直线插补 移动	LMOV <点位变数>	
CMOV	沿着过现在位置, 经由点1,经由点2 的圆移动	CMOV <经由点1> <经由点2>	: VEL 200 CMOV P1 P2 JMOV P3 AMOV P4 P5 :
AMOV	沿着过现在位置, 经由点1,目标点 的圆弧移动	AMOV <经由点1> <目标点>	
IMOV	从现在位置向目 标点PTP 增量移 动	IMOV <点位变数> <点位变数>值→ 轴值增量	: JMOV P1 IMOV P2 IMOV2 P3 :
IMOV2	从现在位置向目 标点直线插补增 量移动	IMOV <点位变数> <点位变数>值→ 轴值增量	
PMOV	在现在位置执行指定的 PALLETIZING移	PMOV <作业PALLET编号> <作业基准点> <作业PALLET编号>的数据值→在参数里设定	PMOV P1 P10
HMOV	从现在位置向指 定的HOME位置 PTP 移动	HMOV <HOME位置编号> <位置编号>的数据值→在参数里设定	HMOV 1
JNTSYN	PTP同步模式选择	JNTSYN<模式>	JNTSYN 1
EECH	终端装置选择	EECH<终端装置>	EECH 1
TIMOV	以TOOL坐标系为标准， 根据增量移动量移动相应 得直线插值量。	TIMOV<位置变量>	TIMOV CURR
SVON	SERVO ON	SVON → 모든 축 해당 SVON <지정 축> → 1(X), 2(Y), 3(Z), 4(W)	SVON SVOF 2 SVON 2
SVOF	SERVO OFF	SVOF → 모든 축 해당 SVOF <지정 축> → 1(X), 2(Y), 3(Z), 4(W)	

WITH
 	
ENDWT
	ROBOT动作中同
	时处理下一个命
	令
		WITH
		:
		ENDWT
			:
			WITH
			JMOV P1
			MVR=0
			WHILE MVR<60
			IF IN1==1 THEN
			OUT0==1
			ENDIF
			ENDWL
			ENDWT
			:

PASS	指定 PALLET 作业通过	PASS <pallet NO> <通过的加工品顺序>	PASS 1 3 PASS 2 1

2.2.3 关于输入输出命令语

命令语	功 能	形 式	使用例题
OUT	以1BIT单位把指定的BIT编号输出ON(=1),OFF(=0)	OUT<BIT输出PORT编号>=<0/1> [脉冲有效时间][→] [脉冲有效时间] 单位 ：10ms [脉冲有效时间]无 ： 输出信号一直有效 如果过[脉冲有效时间] 复位到以前状态 [→] ： 脉冲有效时间为周期的输出有周期的波形	OUT0=1 OUT0=1 100 OUT0=1 100 →
POUT	以PORT单位向指定的PORT输出指定的值.	POUT<输出PORT编号>=<输出指定值> <输出PORT 编号> **：** 在**0 ~2** 中选择 **0 → OUT0 ~ OUT15** **1 → OUT16 ~ OUT31** **2 → OUT32 ~ OUT47**	POUT0=0 POUT0=20 POUT1=0H000F
IN	以1BIT单位读入指定的BIT编号ON(=1),OFF(=0)值。	IN<BIT输入PORT编号>=<0/1> -等待到指定的BIT输入PORT条件被满足 <变数>=IN<输入BIT 编号> -指定的BIT输入状态值保存到变数中	INT AA IN10=1 AA=IN0
PIN	以PORT单位读入指定输入PORT值	<变数>=PIN<输入PORT编号> **<输入PORT编号> ：** 在**0 ~2**中选择 **0 → IN0 ~ IN15** **1 → IN16 ~ IN31** **2 → IN32 ~ IN47**	INT AA AA=PIN0
CIN	以1BIT单位读入指定的FIELDBUS 输 入 BIT编号 ON(=1), OFF(=0) 值	<变数>=CIN<输入BIT编号> -指定的BIT输入状态值保存到变数中	INT AA AA=CIN0
CBIN **CWIN** **CDIN**	以PORT单位，读入指定的FIELDBUS 输入PORT值	<变数>=CWIN<输入PORT编号>	INT AA AA=CWIN0
COUT	以1BIT单位向指定的FIELDBUS 输出 ON(=1), OFF(=0)	COUT<输出BIT编号>=<0/1>	COUT=1
CBOUT **CWOUT** **CDOUT**	以PORT单位，向指定的FIELDBUS 输出PORT值	CWOUT<输出PORT编号>=<指定输出值>	CWOUT=0HFF0FFFF
SYS	系统控制命令	运行以预约的系统命令语	

2.2.4 关于ROBOT动作条件命令语

命令语	功 能	形 式	使用例题
VEL	设定轴移动速度的百分比(%)	VEL <百分比(%)> 轴移动速度=额定速度 X 0.001 **额定速度是 MOTION 参数中设置**	MAIN VEL 200 ACC 70 DEC 70 JMOV P1 VEL 1000 LMOV P2 ACC 100 DEC 100 EOP
ACC	设定加速时间的百分比(%)	ACC <百分比值(%)> 加速时间=额定加速时间X百分比值X0.01 **额定加速时间是 MOTION 参数中设置**	
DEC	设定加速时间的百分比(%)	DEC <百分比值(%)> 减速时间=额定减速时间X百分比值X 0.01 **额定减速时间是 MOTION 参数中设置**	
FOS	机器人移动到目标点之前，变更轨迹向下一个目标点移动。	FOS <距离比例(%)> 距离比例是全体移动距离的百分比 适用于JMOV, LMOV, AMOV	FOS 5 JMOV P1 FOS 0(해제)
PFOS	轴前段移动到目标点之前，变更轨迹向下一个目标点移动	PFOS <距离(mm)> 适用于LMOV	PFOS 5 LMOV P1 PFOS 0(해제)
SPOS	轴前段移动到目标点之前，变更轨迹向下一个目标点移动	SFOS<距离>,<速度>,<圆弧角度> 适用于LMOV, AMOV, CMOV	SFOS 10 0 0 LMOV P0 LMOV P1 SFOS 0 0 0(해제)
DLAY	设定延时时间	DLAY <延时> 延时 单位10ms (如果延时 500，则延时 5s)	JMOV P1 DLAY 20 JMOV P2
OFFS	把目标点按指定的值移动.	OFFS <偏差值> **偏差值在 MDI模式下输入** (例) P100 → X:0, Y:-100, Z:10, W:0	JMOV P1 OFFS P100 JMOV P2
LIMT	限制各轴的移动范围	LIMT <轴位置值(最小)> <轴位置值(最大)>	JMOV P1 LIMT P3 P4 JMOV P2
PLUP	设定为Pull up动作的Z轴位置值	PLUP <Z轴位置值> 只适用于Z轴	PLUP 5 JMOV P1 PLUP 0(해제)
FORM	ROBOT动作时指定ROBOT ARM的形状	FORM <形态> 只适用于水平多关节机器人 <形状>：在LEFT, RIGHT, NO 中选择	FORM RIGHT JMOV P1 FORM NO(해제
TOOL	选择安装在ROBOT上的工具	TOOL <工具编号> **在参数里设定对应于工具编号的工具形状数据** <工具编号>：0 ~ 3中选择	TOOL1 JMOV P1 TOOL 0 JMOV P2

命令语	功 能	形 式	使用例题
PCLR	初始化PMOV的 PALLETIZING计 数器	PCLR <PALLET编号> PALLET 编号 : 可以使用0 ~99 清空则计数器(XCNT,YCNT,ZCNT)值变为1。	MAIN PCLR 1 PCLR 2 PMOV P1 P10
INPOS	设定ROBOT动作时 到达目标点的程度	INPOS <到达目标点的程度> 到达程度以 Pulse值设定。	INPOS 10 JMOV P2 INPOS 0(해제) JMOV P3
FIX	在补间动作中选择W轴	FIX (0/1) 0 : W轴旋转, 1 : W轴固定 如果没有特别指定则，W轴是固定	FIX 0 LOMV P1 FIX 1
TRQ	感应冲突时发生限制扭矩 报警	TRQ <轴编号> <扭矩限制值> 如果各轴超出扭矩限制值时发证报警	TRQ 1 50 LMOV P1 LMOV P2
TQL	限制各轴的最大推力扭矩 值	TQL <轴编号> <扭矩限制值> 限制扭矩来启动各轴	TRQ 1 50 LMOV P1 LMOV P2
MINIT	Mapping功能初始化	MINIT<传感器类型>	MINIT 1
MSTART	开始Mapping功能	MSTART<传感器输入引脚><轴><GP Index>	MSTART 3 3 GP10
MREAD	更新Mapping数据	MREAD<限定时间>	MREAD 500

2.2.5 变数说明

命令语	功 能	形 式	使用例题
INT	整数型变数说明	INT <变数名>, REAL <变数名>, POS <变数名>, DEFSTR <变数名> 变数名在 6为以内 可以由大写字母，数字， 组成 变数名的第一个必须是字母。	MAIN INT N REAL A,B POS MM, XA
REAL	实数型变数说明		
POS	位置型变数说明		
DEFSTR	字符串型变数说明		

2.2.6 关于程序控制命令语

命令语	功 能	形 式	使用例题
	编写 Point 文件时， 定义Point 变数	变数名参照 POINT 命令	

2.2.7 系统变数

命令语	功 能	形 式	使用例题
CNTn **TMR0** **TMR1**	系统定义变数 (计数器变数,计时器变数)	CNT<脉冲输入PORT编号>=<初始值> TMR0=<初始值> ▶从输入脉冲输入PORT编号开始计数器变数例分配值，以后每输入脉冲就计数。 ▶从输入整数值开始计时器变数分配值，按在系统参数里定义的时间间隔增加1。	CNT0=2 TMR0=0 TMR1=-50
MVR	MOVE RATE	MVR <机器人移动区间的百分比>	IF MVR<50 THEN
HERE	现在轴角度值变数	HERE 保存现在轴角度值的变数	AP=HERE

2.2.8 常数

命令语	功 能	形 式	使用例题
	表示整数,实数,2进整数,16进整数	[0H/0B]<数字> 在数字前没有0H 或则10进数 0H：16进数, 0B：2进数	

2.2.9 运算符

命令语	功 能	形 式	使用例题
+, -, *, /, %	一般运算符 (加，减，乘，除，modulus)		A = B*C
( )	优先运算符		A = B/(A+C)
=	代入计算式		A = B
&,|, ~, ^, <<, >>	IT计算式 (BAND, BOR, Complement, BXOR, left shift, right shift)		
&&, ||, ^^, !	逻辑计算式 (AND, OR, XOR, Negation)		
>, <, >=, <=, !=, ==	比较计算式 (大于,等于或大于,小于或等于，小于，不等于，等于)		
ABS	绝对值	ABS(-10.5)等于 10.5	
DEG	把弧度转换为角度	DEG(3.1416) 等于180.0	
RAD	把角度转换为弧度	RAD(180.0) 等于3.1416	
POW	次方函数	POW(2,4) 等于16	
RND	四舍五入，把实数变为整数	RND(14.8) 等于15	
LOG	自然对数函数		
SQRT	平方根	SQRT(16) 等于4	
SIN	正弦	SIN(RAD(30)) 等于0.5	
ASIN	反正弦函数	ASIN(0.5) 等于0.4794	
COS	余弦函数	COS(0) 等于1.0	
ACOS	反余弦函数	ACOS(0.5) 等于1.0472	
TAN	正切函数	TAN(RAD(45)) 等于1.0	
ATAN	反正切函数	ATAN(-1.0) 等于-0.7854	
ATAN2	2数反正切函数	ATAN2(Y,X) ATAN2(1,-1) 等于2.3562	

2.2.10 字符串

命令语	功 能	形 式	使用例题
ASC	字符串的第一个文字要变换字符元素	整数型变数=ASC(字符串)	INT AA AA=ASC("ABC")
CHR	把整数变换字符	字符串变数=CHR(整数)	DEFSTR AA AA=CHR(65)
FLUSH	输入、输出、 BUFFER Clear	FLUSH< 选择 Clear BUFFER > ▶ 选择范围: 1~3 1: 输入BUFFER Clear 2: 输出BUFFER Clear 3: 输入、输出、BUFFER Clear	FLUSH3
FTOS	把整数或实数变换字符串	字符串变数=FTOS(整数 or 实数) ▶整数 1234 变换字符串 "1234"	DEFSTR AA AA=FTOS(1234)
HTOS	把整数变换16进制字符串	字符串变数=HTOS(整数) ▶ 把整数 10变换 16进制字符串"A"	DEFSTR AA AA=HTOS(10)
SLEFT	左侧部分的字符串输出	字符串变数=SLEFT(字符串, 数字) ▶字符串的左侧开始输出数字前的字符保存在字符串变数	DEFSTR AA AA=SLEFT("ABCDE",3)
SLEN	返还字符串长度	整数型变数=SLEN(字符串)	INT LEN LEN=SLEN("ABCDE",3)
SMID	字符串指定的位置开始输出字符串	字符串变数=SMID(字符串, 指定位置, 输出字符数) ▶字符串位置是从 0 开始	DEFSTR AA AA=SMID("ABCDEF",2,3)
SPOS	字符串1 内只返还字符串2 的位置	整数型变数= SPOS(字符串1,字符串2)	INT PP PP=SPOS("ABCDEF","B") ▶"B"的位置返还 1
SRIGHT	右侧部分字符串输出	字符串变数=SRIGHT(字符串 , 数字) ▶字符串的友侧开始输出数字前的字符保存在字符串变数	DEFSTR AA AA=SRIGHT("ABCDE",3)
STRIN	输入区分字符串	字符串变数=STRIN(TIMEOUT 时间) ▶IMEOUT 时间单位 ms	DEFSTR AA AA=STRIN(1000)
STROUT	输出字符串	整数型变数= STROUT(字符串) ▶在整数型中保存不能传输的字符	INT RT RT=STROUT("ABCDEF")
SVAL	字符串变换数字	变数=SVAL(字符串)	INT VA REAL VB VA=SVAL("1234") VB=SVAL("0.123")

第 3 章 命令语说明

3.1 MAIN, EOP (程序 开始/结束)

功 能 程序开始及结束.

形 式 MAIN
...
EOP

说 明 1) **MAIN- EOP，必须一起使用，** 形成主程序区间。在一个JOB文件只能使用一次

2) EOP(End Of Program)表示主程序的结尾，运行JOB时遇到此命令则结束程序运行。但是，以JCALL命令运行的JOB是，完成运行后回到呼叫的JOB

3) **EOP 以后，可以继续编写必要的子程序(SUBR-RET)。**

3.1.1 程序使用例题

MAIN	.....	**开始 Program**
VEL 100	.....	轴移动速度
WHILE 1	.....	开始执行条件重复
JMOV P0	.....	移动到 P0
JMOV P1	.....	移动到 P1
CALL ON	.....	呼叫副程序
ENDWL	.....	结束执行条件重复
EOP	.....	**结束 Program**
SUBR ON	.....	指定副程序
OUT0=1 200	.....	以 1Bit 单位输出指定的 Bit
OUT1=1 200	.....	以 1Bit 单位输出指定的 Bit
OUT2=1 200	.....	以 1Bit 单位输出指定的 Bit
RET	.....	返回副程序

3.2 FOR, NEXT (重复执行文)

功 能　满足变数值为止一直重复执行文

形 式　FOR <变数>=<初始值> TO <结束值> (BY <增量>)
…
NEXT

用 语　<变数> : 使用被宣言为整数型的变数名. 例) INT A,B,C,AA,...
<初始值> : 运行FOR区间之前设定在变数的整数值.
<结束值> : 为了限制FOR区间的运行次数而设定的整数值
<增量值> : 为了有规则地增加变数值而设定的整数值

说 明　1) **FOR-NEXT，必须一起使用**.
2) 初始值设定在整数型变数后，一直满足结束值为止反复运行FOR区间.
3) 每运行一次FOR区间，则整数型变数的值按增量值增加，没有设定增量值时增加1.
4) 结束值必须大于初始值，增量值必须大于1.
(初始值<=结束值, 增量值>=1)

3.2.1 程序使用例题

1) 利用FOR区间，计算从1到10之间的奇数的合保存在整数型变数SUM.

```
MAIN
INT AA, SUM ..... 宣言整数型变数AA,SUM
SUM=0 ..... 初始化整数型变数
FOR AA=1 TO 10 BY 2 ..... AA=1,3,5,7,9 运行5次；
SUM=SUM+AA ..... 运行5次后运行NEXT以后的命令
NEXT
EOP
```

2) 利用FOR区间，反复运动20次点位 P1 ↔ P2之间

```
MAIN
INT J            ..............  宣言整数型变数J
J=0              ..............  初始化整数型变数
FOR J=1 TO 20    ..............  反复动作20次后移动到NEXT下一行
JMOV P1          ..............  JMOV P1
JMOV P2          ..............  JMOV P2
NEXT
EOP
```

3) 在Position变数保存提取的点位后移动

```
MAIN
INT A            ..............  宣言整数型变数 A
POS AP(11)       ..............  宣言位置型变数AP(11) →AP(0),AP(1), --- AP(10)
FOR A=1 TO 10    ..............  反复运行 FOR 10回后移动到 NEXT
AP(A)=P(A)       ..............  在点位变数 AP(I)保存 P(1),P(2),P(3) - - - P(10)
NEXT
EOP
```

4) 在Position变数保存提取的点位后移动

```
MAIN
INT J,K          ..............  宣言整数型变数 J , K
POS TMP,AP(11)   ..............  宣言位置型变数AP(11) →AP(0),AP(1), --- AP(10)
FOR K=1 TO 10    ..............  反复运行 FOR 10回后移动到 NEXT
AP(K)=P(K)       ..............  在点位变数 AP(K)保存 P(1),P(2),P(3) - - - P(10)
NEXT
IF P100.3==1.0 THEN
FOR J=1 TO 10                    反复运行 FOR 10回后移动到 NEXT
TMP=AP(J)        ..............  AP(J)=AP(I)=P(1) ~ P(10)
TMP.3=TMP.3+10                   点位变数 TMP的Z轴值加"10"(+).
AP(J)=TMP
NEXT
FOR K=1 TO 10    ..............  反复运行 FOR 10回后移动到 NEXT
JMOV AP(K)                       保存的 AP(1) ~ AP(10)做 PTP 移动
DLAY 100
NEXT
EOP
```

3.3 WHILE, ENDWL (条件反复执行命令文)

功 能　满足条件时区间反复运行

形 式　WHILE <条件式>
…
ENDWL

用 语　<条件式> : 是指判断语句真假的逻辑计算式或比较计算.

说 明　1) WHILE语句,满足条件**(条件式结果为真,或0以外的值时)的期间**一直运行WHILE区间。.
2) **WHILE - ENDWL， 必须一起使用**.
3) **条件式的结果一直为真时，无限反复运行**.

CAUTION

输入命令语时注意事项
(O) WHILE_((IN0==1)&&(IN1==0))
(O) WHILE_IN0==1&&IN1==0
(O) WHILE_IN0==1
&&IN1==0

3.3.1 程序使用例题

1) 点位 P0 ↔ P1之间反复移动

```
MAIN
VEL 100
WHILE 1          ..............   无限反复运行WHILE区间
JMOV P0          ..............   P0 ↔ P1 之间反复移动
JMOV P1          ..............
ENDWL
EOP
```

3.3.2 条件式使用例题

WHILE(条件式)		说 明
常 数	WHILE 1	无限反复运行WHILE ~ ENDWL区间
输 入	WHILE IN0==1	输入信号为IN0=1期间反复运行区
	WHILE ((IN0==1)&&(IN1==0))	输入信号 IN0=1, IN1=0 同时满足两个条件式期间反复运行区间
	WHILE ((IN0==1)‖(IN1==0))	输入信号 IN0=1, IN1=0 只满足两个条件中一个条件期间反复运行区间
	WHILE PIN0==0H000F	输入PORT0的 IN0,IN1,IN2,IN3都是"1"期间 反复运行区间
	WHILE PIN0==32	输入PORT0的输入信号状态IN4为 "1"期间 反复运行区间
输 出	WHILE OUT0==1	输出 OUT0为 "1"期间反复运行区间
	WHILE POUT0==0H000F	输出PORT0的 OUT0,OUT1,OUT2,OUT3都是"1"期间 反复运行区间
	WHILE POUT0== 0B0000000011111111	输出PORT0的 OUT0,OUT1,OUT2,OUT3,OUT4, OUT5,OUT6,OUT7 都是"1"期间反复运行期间
变 数	WHILE TMP==1	变数TMP为 "1"期间反复运行区间
	WHILE MVR>10	MVR变数大于 "10"期间反复运行区间
	WHILE TMR>10	TMR(TIMER)变数值大于"10"期间反复运行区间
	WHILE CNT>10	CNT(COUNT)变数值大于"10"期间反复运行区间
	WHILE AP10.1==1.0	POSITION变数值(X轴值)等于"1"期间 反复运行区间

3.4 IF, ENDIF (条件分歧命令语)

功 能　条件式的条件判断运行.

形 式　IF <条件式> THEN
…
(ELSEIF) <条件式> THEN
…
(ELSE)
…
ENDIF

用 语　<条件式> : 是指判断语句真假的逻辑计算式或比较计算式..

说 明　1) **IF - ENDIF，必须一起使用**.
2) 条件式的计算结果为真(或0以外的值)，则运行THEN以后的行，或假(或0)则运行ELSE以后的行.
3) ELSE语句是编程人员可以**选择性地使用**.
4) IF 区间内，可以重复使用 IF区间。这时，要注意IF和ENDIF在逻辑上成对.

※ IF 个数和 ENDIF 个数不成对时，发生 "Syntax Error".

※ ELSE 或 ELSEIF 个数最多不能超过 10 个.

⚠CAUTION

输入命令时注意事项
(O) IF_((IN0==1)&&(IN1==0))
(O) IF_IN0==1&&IN1==0
(O) IF_IN0==1
　　&&IN1==0

3.4.1 程序使用例题

1) 根据输入IN0的状态决定要移动的点位

```
MAIN
VEL 100
IF IN0==1 THEN
JMOV P1 ..... 输入IN0=1时,到点P1 做 PTP 移动
ELSE
JMOV P2 ..... 输入IN0=0时,到点P2 做 PTP 移动
ENDIF
EOP
```

2) 输入Port "1"(IN0 ~15), IN0的状态检查

```
MAIN
INT AA, BB
AA=PIN1 ..... 把Port"1"(IN0~15)状态保存到变数AA
IF (AA==0HFFFF) THEN AA值判断 (IN0~15 是否全部 1 )
STOP ..... "真(IN0~15都是1)" 时, ROBOT 停止
ELSE
BB=IN0 ..... "假"时, 把IN0的状态保存到BB
IF (BB==1) THEN 判断 是否 BB 等于 1
DLAY10 .....
JMOV P0 ..... "真(IN0=1)" 时,移动到点P0
ENDIF
ENDIF
EOP
```

3) 下面程序是同样的内容.

(但, IN0~IN3中假设同时出入的只有一个时的内容)

```
    :
LABL A0
IF IN0==1 THEN
GOTO B0
ENDIF
IF IN1==1 THEN
GOTO C0
ENDIF
IF IN2==1 THEN
GOTO D0
ENDIF
IF IN3==1 THEN
GOTO A0
ENDIF
    :
```

=

```
    :
LABL A0
IF IN0==1 THEN
GOTO B0
ELSE
IF IN1==1 THEN
GOTO C0
ELSE
IF IN2==1 THEN
GOTO D0
ELSE
IF IN3==1 THEN
GOTO A0
ENDIF
ENDIF
ENDIF
ENDIF
    :
```

3.4.2 IF(条件式) THEN

IF(条件式)		说 明
输 入	IF_IN0==1_THEN	输入 IN0是否为 "1" ?
	IF_((IN0==1)&&(IN==0))_THEN	输入 IN0=1, IN1=0 是否同时满足两个条件?
	IF_((IN0==1)‖(IN==0))_THEN	输入 IN0=1, IN1=0 是否至少满足两个条件当中一个?
	IF_PIN0==0H000F_THEN	输入PORT0的IN0,IN1,IN2,IN3是否都为"1"?
	IF_PIN0==32_THEN	输入PORT0的输入信号状态IN4是否为 "1"?
输 出	IF_OUT0==1_THEN	输出OUT0是否为"1"?
	IF_POUT0==0H000F_THEN	输出PORT0的 OUT0,OUT1,OUT2,OUT3是否都为"1"?
	IF_POUT0== 0B0000000011111111_THEN	输出PORT0的 OUT0,OUT1,OUT2,OUT3, OUT4, OUT5,OUT6,OUT7 是否都为"1"?
变 数	IF_TMP==1_THEN	变数TMP是否为"1"?
	IF_MVR>10_THEN	MVR变数值是否大于"10"?
	IF_TMR>10_THEN	TMR(TIMER)变数值是否大于"10"?
	IF_CNT>10_THEN	CNT(COUNT)变数值是否大于"10"?
	IF_AP10.1==1_THEN	POSITION 变数值(X轴值)是否等于"1"?

3.5 LABL, GOTO (分歧命令语)

功 能　　指定分歧点位置，执行文分歧移动命令语.

形 式　　LABL <LABEL名>
GOTO < LABEL名>

用 语　　< LABEL名> : 由英文字母和数字组成，8位以下。
但是,不能使用表示作业点的"P0,P1....P999".
例) LABL UNCLAMP1 **(O)**, GOTO CHECK2 **(O)**
LABL P1 **(X)**, GOTO P100 **(X)**

说 明　　1) GOTO命令是用于需要无条件分歧到指定行时使用，在一个区间里可以使用。
例如, 不能从MAIN区间分歧到SUBR区间,
从SUBR区间分歧到其他的SUBR区间.
2) 重复使用IF区间或FOR， WHILE区间时可以从里面分歧到外面
但不能从外面分歧到里面.

3.5.1 程序使用例题

1) 反复移动两点 P1,P2之间

```
MAIN
VEL 100
LABL A0
JMOV P1
JMOV P2
GOTO A0      无条件分歧到"LABL A0"
EOP`
```

=

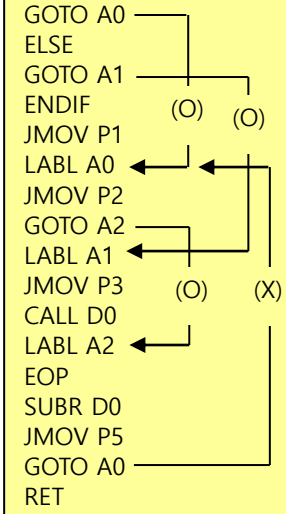
```
MAIN
VEL 100
WHILE 1
JMOV P1
JMOV P2
ENDWL
EOP
```

2) 根据输入IN0的状态分歧

```
MAIN
VEL 100
JMOV P0
IF IN0==1 THEN
GOTO A0          .....   输入 IN0= 1时无条件分歧到  LABL A0
ELSE
GOTO A1          .....   或无条件分歧到 LABL A1
ENDIF
JMOV P1
LABL A0          .....   LABL A0 分歧点
JMOV P2
GOTO A2          .....   无条件分歧到 LABL A2
LABL A1          .....   LABL A1 分歧点
JMOV P3
LABL A2          .....   LABL A2 分歧点
EOP
```

CAUTION

输入命令语时注意事项



3.6 SUBR, RET (副执行文命令语)

功 能　程序中可编程指定反复动作范围，必要时可呼叫

形 式　SUBR <副执行文名>
…
RET

用 语　<副执行文名> : 由英文和数字组成，8位以下.
但是, 不能使用表示作业点的"P0,P1, - - - ,P999.

说 明
1) **SUBR - RET ，必须一起使用**
2) SUBR 区间只能用**CALL命令呼叫**.
3) R RET 表示结束 SUBR区间的运行，回到主程序**CALL以后STEP**.
4) CALL语句,在MAIN或SUBR区间都可以使用，也可以回归呼叫(Recursive Call).
5) SUBR区间必须在 **MAIN-EOP区间后面编写.**

3.6.1 程序使用例题

```
MAIN
VEL 100
JMOV P0
CALL A0          .....   CALL 副执行文 "A0"
CALL A1          .....   CALL 副执行文 "A1"
EOP

SUBR A0
JMOV P1
RET              .....   结束 SUBR A0 区间,
                         恢复到 CALL 之后的Step
SUBR A1
JMOV P2
RET              .....   结束 SUBR A1 区间,
                         恢复到 CALL 之后的Step
```

3.7 CALL, JCALL (呼叫命令语)

功 能　副执行文, 呼叫 JOB文件.

形 式　CALL <副执行文>
JCALL <JOB文件名>

用 语　<副执行文> : 由英文和数字组成，8位以下.
但是, 不能使用表示作业点的"P0,P1, - - - ,P999.
<JOB 文件名> : 要呼叫的 JOB 程序扩展(.JOB)以外的文件名.

说 明　※ **CALL**
1) 要使用CALL语句，必须存在被呼叫的子**程序(SUBR)**.
2) CALL语句的**副程序名和副程序**的语句要一样.
3) 在SUBR区间遇到RET则移动到**CALL以后的STEP**.
※ **JCALL**
1) JCALL是把JOB文件用作副程序.
2) 在被呼叫的JOB遇到 EOP则回到呼叫的程序JCALL以后
3) 编程时JCALL不应有呼叫回归.

可以再归呼叫 (Recursive Call).
注) CALL 再归呼叫**最多 8次** → 超过时 "CALL DEPTH ERROR"
JCALL 再归呼叫**最多 3次** → 超过时 "JCALL DEPTH ERROR"

CAUTION

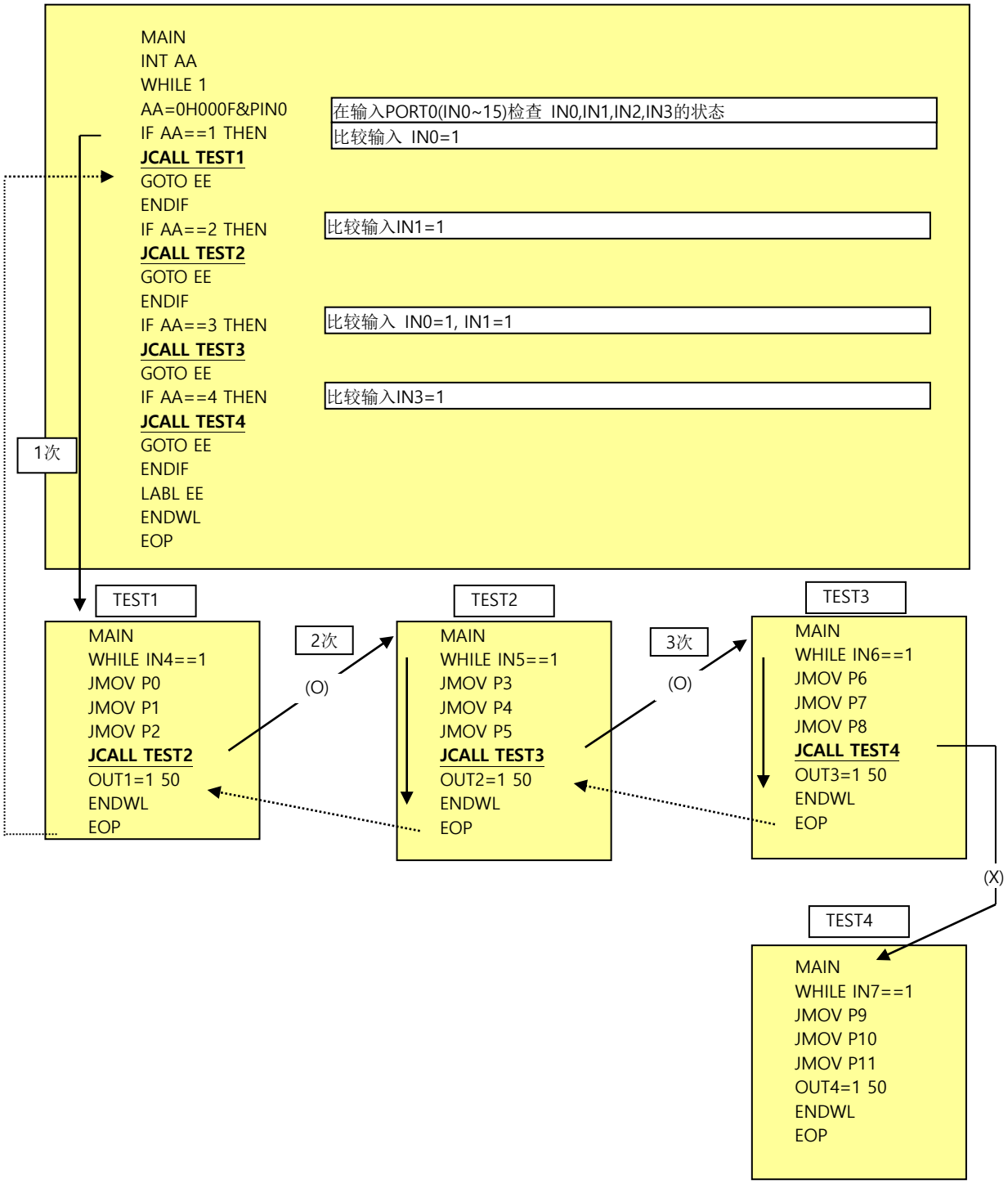
输入命令语时注意事项

```
MAIN
VEL 100
LABL A0
JMOV P1
JMOV P2
CALL A0
GOTO A0
EOP
SUBR A0
OUT1=1
IN1=1
RET
```

如果(X) LABL名和SUBR名相同则 ERROR.

3.7.1 程序使用例题

1) 根据输入PORT0的状态，在"TEST1" ~"TEST4"当中选择一个JOB后JCALL.



3.8 STOP, EXIT (ROBOT 及 JOB 停止命令语)

功 能　停止 ROBOT动作(STOP), 停止执行 JOB(EXIT)

形 式　STOP
EXIT

说 明　1) STOP是**停止移动中的机器人**动作,然后运行下一步命令
即,停止后一直运行JOB
2) EXIT是停止JOB的运行.
这是控制器上发生 **"EXIT Instruction"** 报警.

3.8.1 程序使用例题

1) 移动到 Point 后 输出 OUT1的 ON, OFF

```
MAIN
VEL 500
JMOV P1
MVR=0          ..........    "初始化"移动区间百分比"
WITH           ..........
JMOV P2        ..........    向点P2移动(只有JMOV可以)
WHILE MVR<60   ..........    在移动区间的60% 以内
IF IN0==1 THEN ..........    如果IN0=1,则跳到 "LABL BB"
GOTO BB
ENDIF
ENDWL
OUT0=1
ENDWT

LABL BB
STOP           ..........    停止机器人移动动作
ENDWT          ..........    结束 WITH (必须插入)
JMOV P3
OUT1=1
EOP
```

3.9 JMOV (PTP 移动命令语)

功 能　从现在位置向目标点做插补移动

形 式　JMOV P<编号>

JMOV GP<编号>

JMOV <位置型变数>

用 语　<编号> : 设定提取的位置坐标编号.

P : 分别 使用 JOB LOCAL POINT

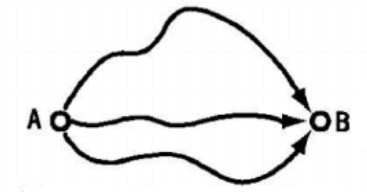
GP : 共同使用的 GLOBAL POINT

VER.	RO(Robot)	TR(Transfer Robot)
Local Point	0 ≤ 编号 ≤ 1999	0 ≤ 编号 ≤ 1999
Global Point	0 ≤ 编号 ≤ 1023	0 ≤ 编号 ≤ 1499

<位置型变数> : 设定提取的位置坐标编号.

说 明　PTP(Point to Point)是 **点到点的移动** .

移动路径跟ROBOT形态有关， **不限于直线运动**.



动作路径不限于直线.

3.9.1 程序使用例题

1) 移动到提取点位的位置坐标

```
MAIN
VEL 100
JMOV P10          .....    向点P10进行PTP移动
JMOV P11          .....    向点P11进行PTP移动
JMOV P100         .....    向点P100进行PTP移动
EOP
```

2) 利用位置型变数的移动 (1)

```
MAIN
POS XA                              ..............  设定位置型变数 XA
XA=<400.0,50.0,10.4,10.0,0,0,1>     ..............  指定XA的值 <X,Y,Z,W,EX1,EX2,FORM>
VEL 100
JMOV P0                             ..............  向Point 0位置坐标作PTP移动
JMOV XA                             ..............  向XA位置坐标作PTP移动
JMOV P100                           ..............  向Point 100位置坐标作PTP移动
EOP
```

3) 利用位置型变数的移动 (2)

```
MAIN
POS AP,AP1,AP2                      ..............  设定位置型变数 AP,AP1,AP2
AP=P10 +<10,10,10,10,0,0>           ..............  在点位P10保存各轴值加10的值
AP1=P11 -<10,10,10,10,0,0>          ..............  在点位P11保存各轴值减10的值
AP2=P100                            ..............  在点位 AP2中保存 P100值
AP2.3=P100.3+10                     ..............  在点位P100保存Z轴值加10的值
JMOV AP                             ..............  到 AP位置坐标 PTP 移动
JMOV AP1                            ..............  到 AP1位置坐标 PTP 移动
JMOV AP2                            ..............  到 AP2位置坐标 PTP 移动
EOP
```

3.10 LMOV (CP 移动命令语)

功 能　从现在位置向目标点作作直线插补移动

形 式　LMOV P<编号>

LMOV GP<编号>

LMOV <位置型变数>

用 语　<编号> : 设定提取位置坐标编号.

P : 分别 使用 JOB LOCAL POINT

GP : 共同使用的 GLOBAL POINT

VER.	RO(Robot)	TR(Transfer Robot)
Local Point	0 ≤ 编号 ≤ 1999	0 ≤ 编号 ≤ 1999
Global Point	0 ≤ 编号 ≤ 1023	0 ≤ 编号 ≤ 1499

<位置型变数> : 设定提取的位置坐标编号.

说 明　CP控制时到达目标位置间做直线插补运动.

CAUTION

- ▶ 执行 CP控制有时ROBOT的位置不同而做不了动作.
  这时控制器发生 "Unreachable Point" 或 "Inverse Error" 等 Alarm.
- ▶ 特别是使用,"FOS","FIX","TOOL" 命令时注意.
- ▶ SCARA Robot时维持当前 FORM做差不运动
- ▶ 程序的第一个动作命令语要使用 PTP控制.

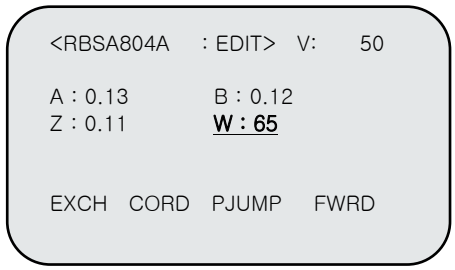
CAUTION

在 SCARA Robot中 W轴座差不动作

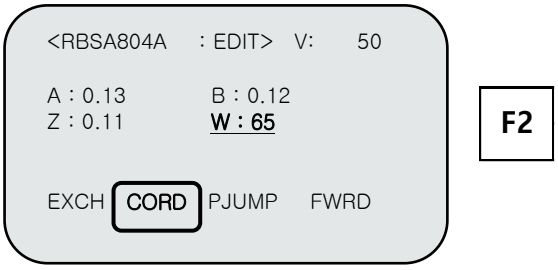
- ▶ 按机械部 Sticker的值正确设定参数 "OFFS","LENG"值.
- ▶ 有TOOL OFFSET时, W轴插补动作有2种.
  W轴姿势固定 (机器人命令语 "FIX 1")和 W轴姿势旋转 (机器人命令语 "FIX 0")
  (参照FIX 命令)
- ▶ T有TOOL OFFSET时, 注意作业点的提取.
  W轴姿势固定: 让TOOL保持一定的方向提取
  W轴旋转 : 正确设定TOOL OFFSET后, 选定 TOOL 及 提取点位.
- ▶ 选TOOL或提取点位时, 建议尽量固定W轴姿势后工作.
- ▶ 特别是W轴姿势旋转是在连续的插补动作中划圆弧时, 会出现机械部跟不上控制器的指令的情况。(Unreachable Point) .

3.10.1 让TOOL保持一定的方向提取(SCARA)

Step 1.

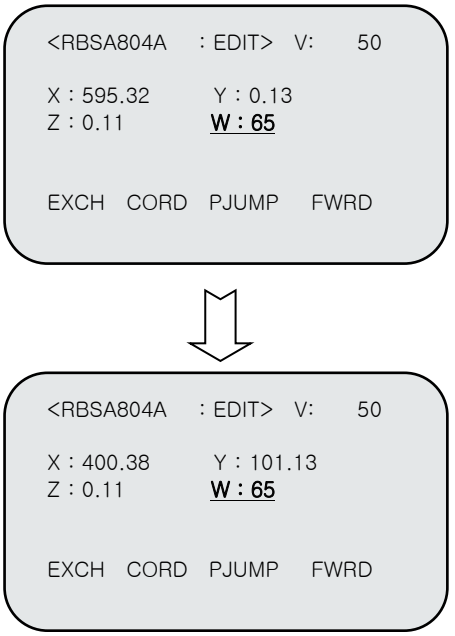


参照提点位说明书转到点位画面..

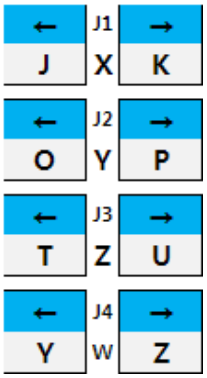


F2

点击 「F2」使圆通坐标
→变更到直角.



在直角坐标中做 JOG动作时
保持 W轴姿势移动.



3.10.2 程序使用例题

1) 移动到提取的点位

```
MAIN
VEL 100
JMOV P10 ..... 向点 P10 做 PTP 移动
LMOV P11 ..... 向点 P11 做 CP 移动
LMOV P100 ..... 向点 P10 做 CP 移动
EOP
```

2) 利用位置型变数的移动 (1)

```
MAIN
POS XA ..... 指定位置型变数 XA
XA=<400.0,50.0,10.4,10.0,0,0,1> ..... 指定XA的值 <X,Y,Z,W,EX1,EX2,FORM>
VEL 100
JMOV P0 ..... 向点 P10 做 PTP 移动
LMOV XA ..... 向 XA位置坐标做 CP 移动
LMOV P100 ..... 向点 P10 做 CP 移动
EOP
```

3) 利用位置型变数的移动 (2)

```
MAIN
POS AP,AP1,AP2 ..... 指定位置型变数 AP,AP1,AP2
AP=P10 +<10,10,10,10,0,0> ..... 在点位P10保存各轴值加10的值
AP1=P11 -<10,10,10,10,0,0> ..... 在点位P11保存各轴值减10的值
AP2=P100 ..... 在点位 AP2中保存 P100值
AP2.3=P100.3+10 ..... 在点位P100保存Z轴值加10的值
JMOV AP ..... 到 AP位置坐标 PTP 移动
LMOV AP1 ..... 到 AP1位置坐标 CP 移动
LMOV AP2 ..... 到 AP2位置坐标 CP 移动
EOP
```

3.11 CMOV (圆形插补移动命令语)

功 能　沿着从现在位置出发过经由点1，经由点2，回到现在位置的圆移动

形 式　CMOV_P<经由点1编号>_P<经由点2 编号>

CMOV GP<经由点1编号>_P<经由点2 编号>

CMOV <经由点1 位置型变数> P<经由点2 位置型变数>

用 语　<编号> : 设定提取的位置坐标编号.

P : 分别 使用 JOB LOCAL POINT

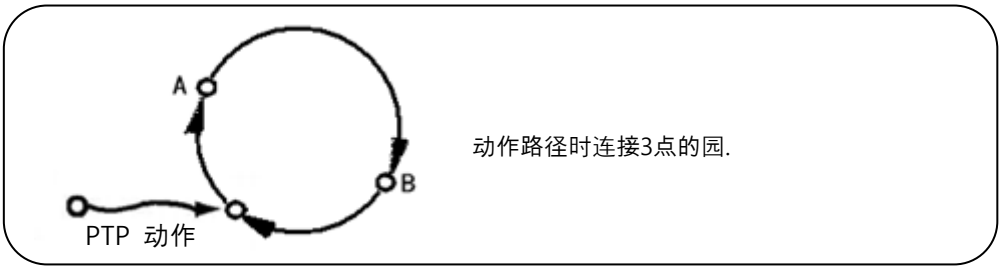
GP : 共同使用的 GLOBAL POINT

VER.	RO(Robot)	TR(Transfer Robot)
Local Point	0 ≤ 编号 ≤ 1999	0 ≤ 编号 ≤ 1999
Global Point	0 ≤ 编号 ≤ 1023	0 ≤ 编号 ≤ 1499

<位置型变数> : 是指被设定为POS型的变数名

说 明　以圆插补动作移动到指定的位置.

作圆插补动作时需要3个坐标.



动作路径时连接3点的园.

CAUTION

- ▶ 如果提取点位有错误则Auto RUN中出现“Unreachable Point”, ”Inverse Error”..
- ▶ 特别是使用, ”FOS”,”FIX”,”TOOL” 命令时注意.
- ▶ W轴插补运动参照考 “LMOV”命令语.
- ▶ 不能单独使用 CMOV, AMOV必须跟例题一样添加 JMOV.

3.11.1 程序使用例题

1) 向提取的位置坐标移动

```
MAIN
VEL 100
JMOV P1          .....    向点P1作PTP移动
CMOV P2 P3       .....    作经过点 P2,P3的圆插补运动
EOP
```

3.12 AMOV (圆弧插补移动命令语)

功 能　沿着从现在位置经过经由点1到目标点的圆弧移动

形 式　AMOV_P<经由点1 编号>_P<目标点>

AMOV GP<经由点1编号>_P<经由点2 编号>

AMOV <经由点1 位置型变数> P<经由点2 位置型变数>

用 语　<编号> : 设定提取的位置坐标编号.

P : 分别 使用 JOB LOCAL POINT

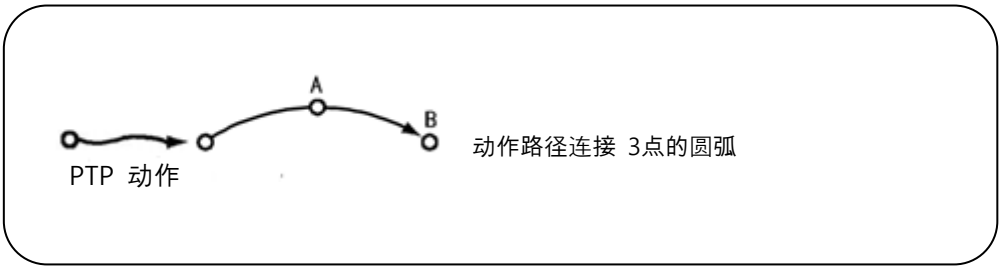
GP : 共同使用的 GLOBAL POINT

VER.	RO(Robot)	TR(Transfer Robot)
Local Point	0 ≤ 编号 ≤ 1999	0 ≤ 编号 ≤ 1999
Global Point	0 ≤ 编号 ≤ 1023	0 ≤ 编号 ≤ 1499

<位置型变数> : 是指被设定为POS型的变数名.

说 明　以圆弧插补动作移动到指定的位置.

作圆弧插补动作时需要3个坐标.



CAUTION

- ▶ 如果提取点位有错误则Auto RUN中出现“Unreachable Point”, ”Inverse Error”..
- ▶ 特别是使用, ”FOS”,”FIX”,”TOOL” 命令时注意.
- ▶ W轴插补运动参照夸 “LMOV”命令语.
- ▶ 不能单独使用 CMOV, AMOV必须跟例题一样添加 JMOV.

3.12.1 程序使用例题

1) 向提取的位置坐标移动

```
MAIN
VEL 100
JMOV P1          .....    向点P1作PTP移动
AMOV P2 P3       .....    作经过点 P2,P3的圆弧插补运动
EOP
```

3.13 IMOV, IMOV2 (增量移动命令语)

功 能　从当前位置坐轴插补增量移动

IMOV – PTP 移动

IMOV2 – CP 移动

形 式　IMOV P<编号>

IMOV GP<编号>

IMOV <位置型变数>

用 语　<编号> : 设定提取的位置坐标编号.

P : 分别 使用 JOB LOCAL POINT

GP : 共同使用的 GLOBAL POINT

VER.	RO(Robot)	TR(Transfer Robot)
Local Point	0 ≤ 编号 ≤ 1999	0 ≤ 编号 ≤ 1999
Global Point	0 ≤ 编号 ≤ 1023	0 ≤ 编号 ≤ 1499

<位置型变数> : 是指被设定为POS型的变数名

说 明　1) 按指定的位置做相对运动..

2) IIMOV是以现在位置为基准移动指定的位置..

IMOV

动作路径不限制于直线

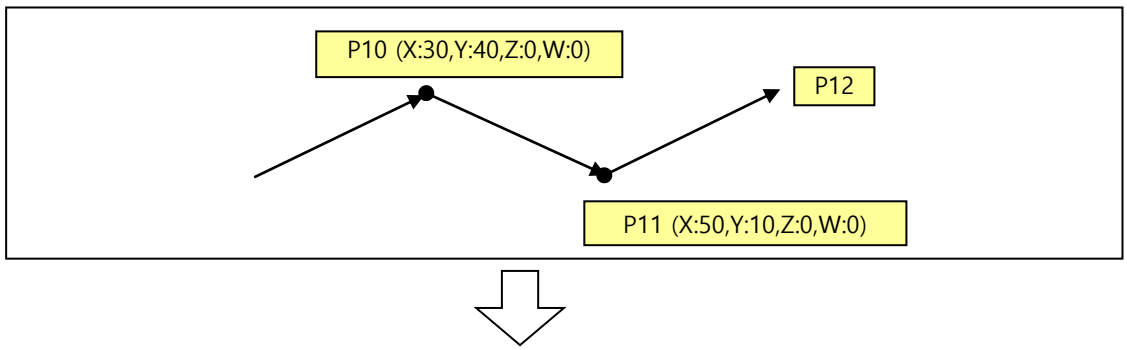
IMOV2

动作路径是直线

CAUTION

- 为了作IMOV动作，保存提取的位置坐标是以参数“RANG”以内的值保存，此值是以原点(ORIGIN)为基准设定的位置值
- 因而以以前位置为基准移动(IMOV)时有可能发生“Range over error”
- 这时请调整参数“RANG”设定
- MOV移动时以以前位置为基准，所以随以前位置的到位（ In Position）程度有可能发生位置偏差
  这时在程序里加“INPOS”,”DLAY”语句，使机器人正确地到达以前位置以后进行IMOV

3.13.1 程序使用例题



1) IMOV 使用 (使用点位P1)

```
MAIN
VEL 100
JMOV P10          .....   向点P10作PTP移动
JMOV P11                  向点P11作PTP移动
INPOS 10          .....   使用INPOS或DLAY → 防止位置偏差.
IMOV P1           .....   以点P11为基准，向提取点P1 PTP相对移动
EOP
```

2) IMOV 使用 (使用位置型变数AP)

```
MAIN
POS AP            .....   指定位置型变数 AP
AP=<20.0,-30.0,0,0.0,0,0,1>  .....  指定 AP的值 <X,Y,Z,W,EX1,EX2,FORM>
VEL 100
JMOV P10          .....   向点P10作PTP移动
IMOV AP           .....   以点P10为基准，PTP相对移动 AP值
JMOV P100         .....   向点P100作PTP移动
EOP
```

3) IMOV2 使用

```
MAIN
VEL 100
LMOV P10          .....   向点P10作CP移动
LMOV P11                  向点P11作CP移动
INPOS 10          .....   使用INPOS或DLAY → 防止位置偏差..
IMOV2 P1          .....   以点P11为基准，向提取点P1 CP相对移动
EOP
```

4) IMOV2使用 (使用位置型变数AP)

```
MAIN
POS AP            .....   指定位置型变数 AP
AP=<20.0,-30.0,0,0.0,0,0,1>  .....  指定 AP的值 <X,Y,Z,W,EX1,EX2,FORM>
VEL 100
LMOV P10          .....   向点P10作CP移动
IMOV2 AP          .....   以点P10为基准，CP相对移动 AP值
LMOV P100         .....   向点P100作CP移动
EOP
```

3.14 JNTSYN (PTP同步模式命令语)

功能 选择PTP模式

JNTSYN 命令语适用于 ROSEP Robot版本及 Ver 03.00.00版本或更高版本。

形式 JNTSYN <形式>

0： 不使用PTP同步模式

1： 使用PTP同步模式

说明

1) 当在奇点附近做直线插补运动时由于在起点附近速度突然变化而发生Over Speed警报。为解决这种问题ROSEP机器人类型的情况同步启动PTP运动（JMOV），使直线运动通过奇点。

2) 如果设定成"JNTSYN 0", 可与一般的JMOV同时工作。

3) 如果设定成"JNTSYN 1"，会使J2、J4、J5的移动量同步到 J1的移动量，做过起点的直线运动。

4) 比较每个轴的移动距离，确认是否成立同步条件。

如果W1、W2不满足同步条件，从同步式中移除。

A, B不满足同步条件会发生警报(1422: PTP Sched. Error)。

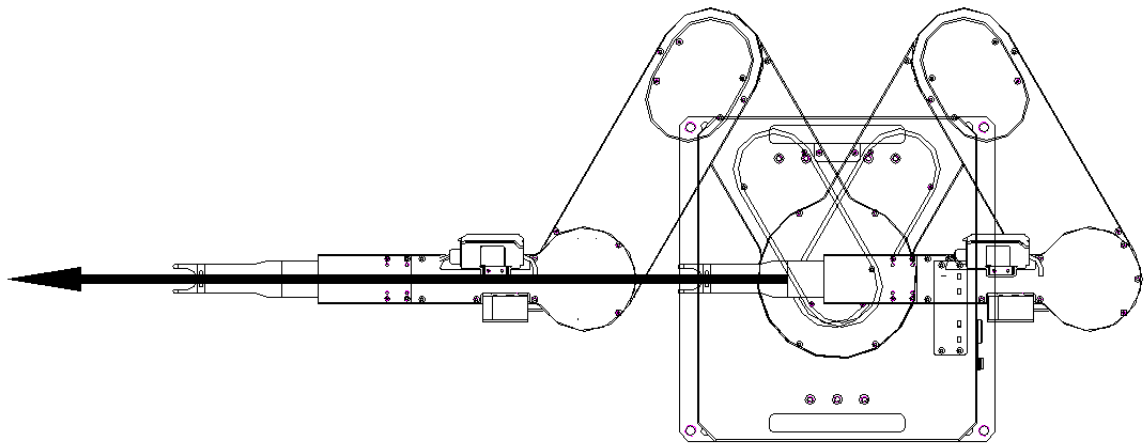
例) P1 =<-25, 230, 0, -115, -115, 0>, P2 =<25, 130, 0, -65, -65, 0>

P2.1 - P1.1= 50 = delta_TH1

P2.2 - P1.2= -100 = -2 * delta_TH1

P2.4 - P1.4= 50 = delta_TH1

P2.5 - P1.5= 50 = delta_TH1



[ROSEP ROBOT TYPE 通过奇点]

3.14.1示例程序使用

MAIN	.....	Program 开始
VEL 10	.....	轴移动速度
JMOV P1	.....	往P1=<-25, 230, 0, -115, -115, 0>移动
WHILE 1	.....	重复条件开始
JNTSYN 1	.....	使用PTP同步模式
JMOV P2	.....	P2=<25, 130, 0, -65, -65, 0>로 이동
JMOV P1	.....	P1=<-25, 230, 0, -115, -115, 0>로 이동
JNTSYN 0	.....	不使用PTP同步模式
ENDWL	.....	终止条件循环
EOP	.....	终止Program

CAUTION

- ▶ 通过JMOV 同步模式做直线运动时，J1(A)轴和J2(B)轴的移动量需变为两倍。
  移动量不是两倍的情况会发生 "PTP Sched. Error " 警报。
- ▶ 如果J1(A)轴、J4(W1)轴、J5(W2)轴的移动量不同，则忽略同步模式并向轴改变移动。

3.15 EECH (终端装置选择命令语)

功能　选择终端装置(W1, W2)

EECH 命令语只适用于ROSEP Robot版本及Ver 03.00.00版本或更高版本。

形式　EECH <形式>

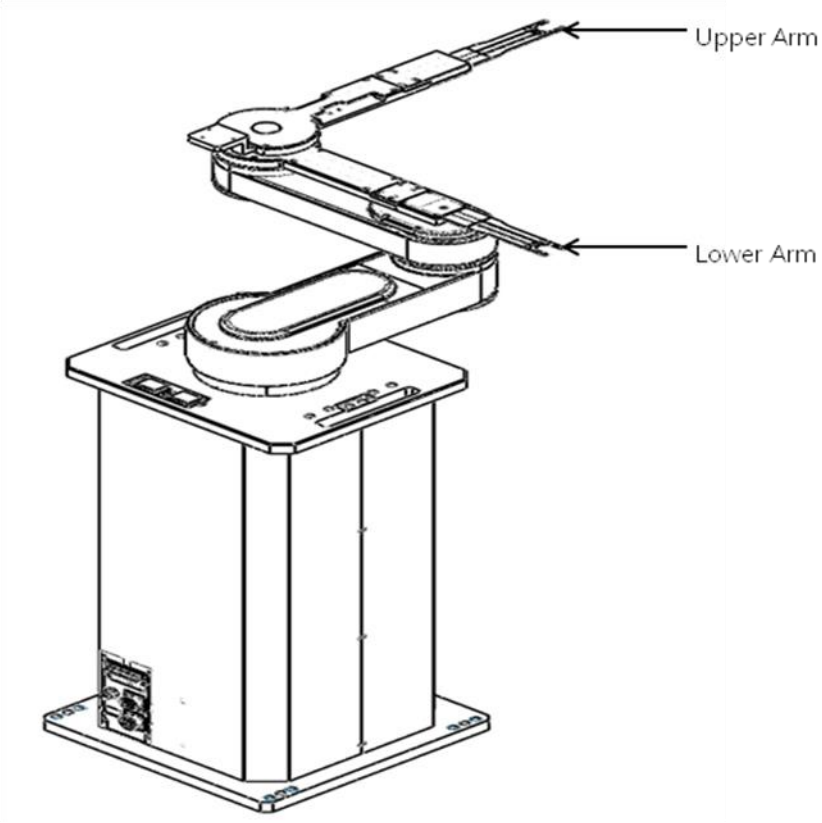
1: 选择第一个W1轴及TOOL 0号

2: 选择第二个W2轴及TOOL 1号

说明　1) 当ROSEP Robot的情况终端装置由两个装置构成，XY坐标变换时选择应用的终端装置的命令语。

2) 如果设定为"EECH 1"，则选择成机器人的W1轴和终端装置的Tool 0。

3) 如果设定为"EECH 2"，则选择成机器人的W2轴(Upper Arm)和终端装置的Tool 1。



[选择 ROSEP ROBOT Arm]

3.15.1程序使用示例

MAIN	.....	Program 开始
VEL 10	.....	设定移动速度
WHILE 1	.....	开始执行条件重复
JMOV P0	.....	PTP移动到Point P0
JMOV P1	.....	PTP移动到Point P1
EECH 1	.....	选择机器人的W1终端装置
LMOV P2	.....	CP移动到Point P2
LMOV P1	.....	CP移动到Point P1
JMOV P3	.....	PTP移动到Point P3
EECH 2	.....	选择机器人的W2终端装置
LMOV P4	.....	CP移动到Point P4
LMOV P3	.....	CP移动到Point P3
ENDWL	.....	终止条件重复
EOP	.....	终止 Program

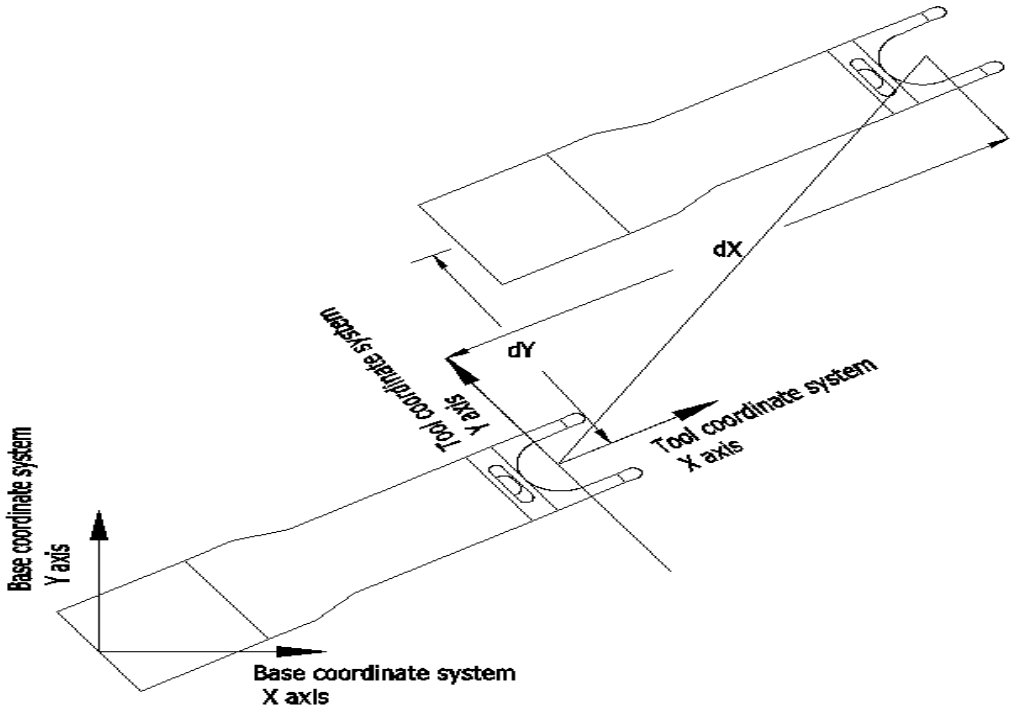
3.16 TIMOV (TOOL 坐标系标准增量移动命令语)

功能 **在当前位置中以TOOL坐标系为标准，根据增量做直线插补运动。**
TIMOV命令语只适用于ROSEP Robot版本及Ver 03.00.00版本或更高版本。

形式 TIMOV 位置变量

<位置变量> ：以POS型定义声明的变数名称。
因为以XY坐标为标准而移动，所以位置变量名称从X坐标开始。
例) POS XP, XCUR

说明 TIMOV是以TOOL 坐标系为标准从当前的位置中移动到相对指定的位置（dx、dy、dz）。



CAUTION

- 为了TIMOV的Point Teaching中，当储存位置坐标时，虽然储存是以参数"RANG"以内的值，但这值是以原点（ORIGIN）为标准设定的位置值。
- 因此当机器人以之前Point为标准移动时会出现"Range over error"。
- 请调"RANG"参数设定。
- TIMOV是以先前的位置为标准而移动，所以会根据先前位置的In Position程度的不同会发生位置偏差。
  这时需要在作业程序中添加"INPOS" ''DLAY",使机器人准确回到原来的位置后，更正程序直到形成TIMOV。

3.17 HMOV (原点移动命令语)

功 能　从现在位置向指定的HOME位置移动

形 式　HMOV <整数型变数或编号>

用 语　<编号> : 在参数设定的原点坐标值编号.
(0 ≤ 编号 ≤ 3)
<整数型变数> : 用 INT 指定整数型的变数.

说 明　1) 在 ORG 参数设定的 HOME , 原点坐标做 PTP.
2) 原点坐标值编号是0号~3号(HMOV 0, HMOV 1, HMOV 2, HMOV 3) , 有4个值

CAUTION

- **HMOV** 移动时 **ORG 参数的 SEQ设定的顺序做原点**
- **按HMOV 1 ~ HMOV 3移动时, 移动轴的移动顺序与原点设定顺序移动无关作PTP移动**.

3.17.1 程序使用例题

1) 向提取的位置坐标移动

```
MAIN
VEL 100
JMOV P1          .....    向点P1作PTP移动
HMOV 0           .....    原点执行 SEQ移动到 HOME 0
HMOV 1           .....    向 HOME 1 做 PTP 移动
EOP
```

3.18 PMOV (Palletzing 移动)

功 能　从现在位置进行指定的 PALLETIZING动作

形 式　PMOV <作业PALLET编号>_<作业基准点>

用 语　<作业PALLET编号> : 输入在参数模式设定的PALLET编号.
(0 ≤ 编号 ≤ 99)
<作业基准点> : PALLET位置，4个提取的点位中输入第一个点位编号
(0 ≤ 编号 ≤ 1999)
例) PMOV P0 P1 的时: 基准点(P1), X方向(P2), Y方向(P3), Z方向(P4)

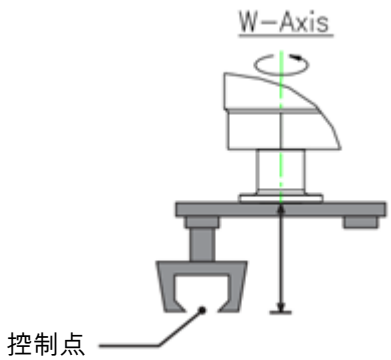
说 明　1) **以PTP动作移动到PALLETIZING作业位置**.
2) PMOV运行是，向在参数计数器(CNT)里指定的PALLET上的位置.
3) 要运行PALLETIZING作业必须有两个内容.
A. 输入在参数模式的PALLET信息
- **最初输入信息之前请初始化PALLET数据.**
B. 提起PALLET上的4个点

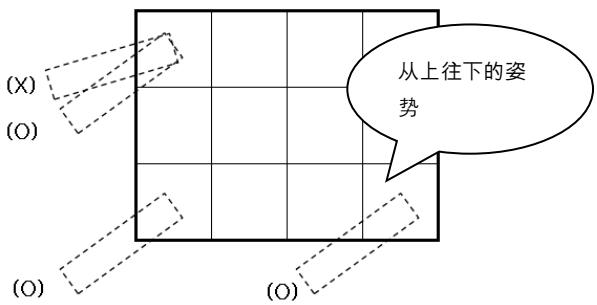
CAUTION

▶ 如果机械部是SCARA时，必须正确输入A,B Arm的 "OFFSET"值.
▶ Pallet 作业顺序是在承诺书中 TYPE 决定.
(参照 使用说明书.)

CAUTION

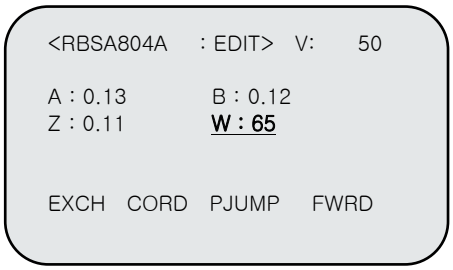
▶ 有旋转轴 (W轴)的机器人， 在Z轴末端(End effect)安装有机械手(HAND)时,
PALLET提取方法如下手臂姿势不变的状态下提点位.



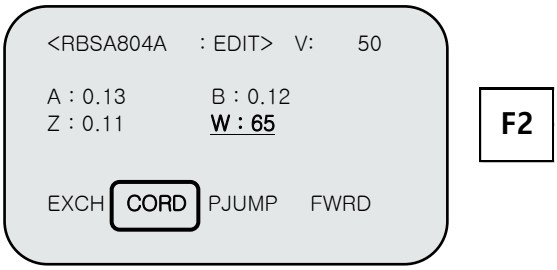


3.18.1提取时保持移动的TOOL方向(SCARA)

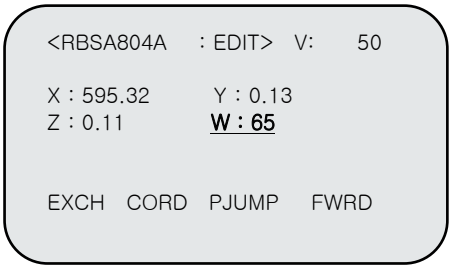
Step 1.



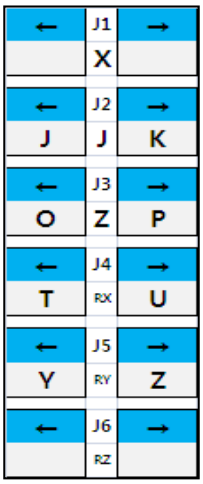
参照提点位说明书转到点位画面.

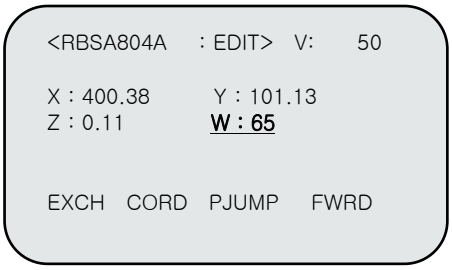


点击 「F2」使圆通坐标
→变更到直角.



在直角坐标中做 JOG动作时
保持 W轴姿势移动.

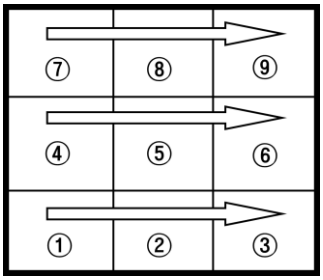




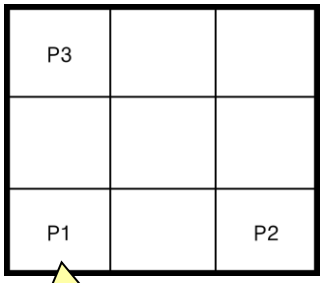
3.18.2 Pallet 作业 例

作业 1 ⇨ 命令语 ：PMOV P1 P1 → 基准点 ：P1
X方向 ：P2
Y方向 ：P3
Z方向 ：P4

①作业顺序 → 提取点位



如果要按照左图的顺序进行作业，则提取点位顺序是?



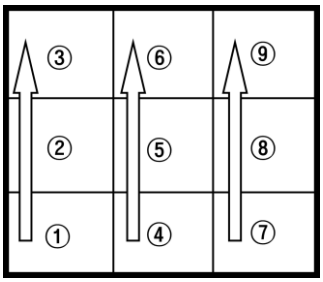
②参数设定中设定DATA

```
Pallet NO = 01
1.XWM=03    3.ZWM=01
2.YWM=03
```

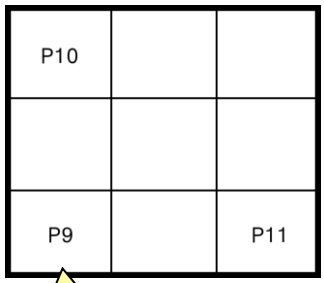
▶ P4 点位?
-. 1层时: P4=P1
-. 2层以上时:
P4和 P1，只有Z轴值不同

作业 2 ⇨ 命令语 ：PMOV P1 P9 → 基准点 ：P9
X方向 ：P10
Y方向 ：P11
Z方向 ：P12

①作业顺序 → 提取点位



如果要按照左图的顺序进行作业，则提取点位顺序是?



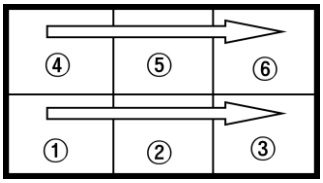
②参数设定中设定DATA

```
Pallet NO = 02
1.XWM=03    3.ZWM=01
2.YWM=03
```

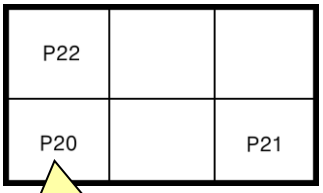
▶ P12 点位?
-. 1层时: P12=P9
-. 2层以上时:
P12和 P9. 只有Z轴值不同

作业 3 ⇨ 命令语 ：PMOV P20 P20 → 基准点 ：P20
X方向 ：P21
Y方向 ：P22
Z方向 ：P23

①作业顺序 → 提取点位



如果要按照左图的顺序进行作业，则提取点位顺序是?



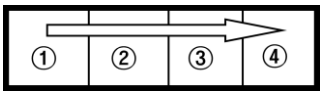
▶ P23 点位?
-. 1层时: P23=P20
-. 2层以上时:
P23和P20，只有轴值不同

②参数设定中设定DATA

```
Pallet NO = 20
1.XWM=03    3.ZWM=01
2.YWM=02
```

作业 4 ⇨ 命令语 ：PMOV P9 P1 → 基准点 ：P1
X方向 ：P2
Y方向 ：P3
Z方向 ：P4

①作业顺序 → 提取点位

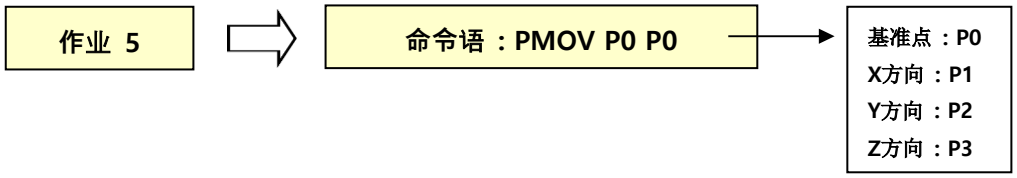


如果要按照左图的顺序进行作业，则提取点位顺序是?

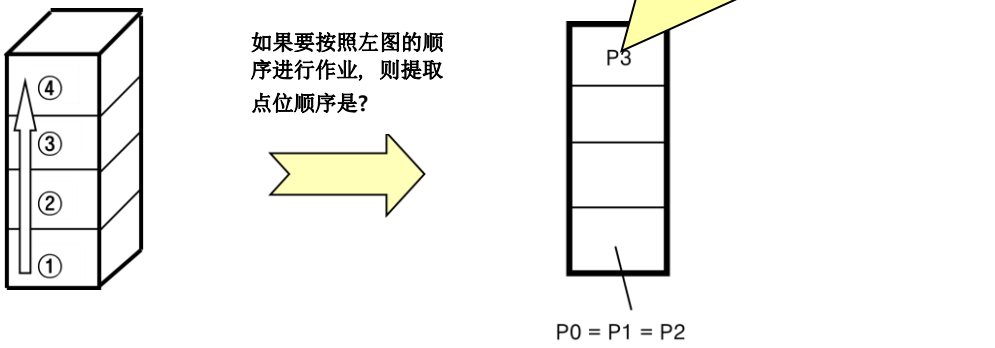
▶ P4 点位?
-. 1层时: P4=P1
-. 2层以上时:
P4和 P1. 只有 Z轴值不同

②参数设定中设定DATA

```
Pallet NO = 99
1.XWM=04    3.ZWM=01
2.YWM=01
```



①在一个地方装4层时



②参数设定中设定DATA

```
    Pallet NO = 0
1.XWM=01    3.ZWM=04
2.YWM=01
```

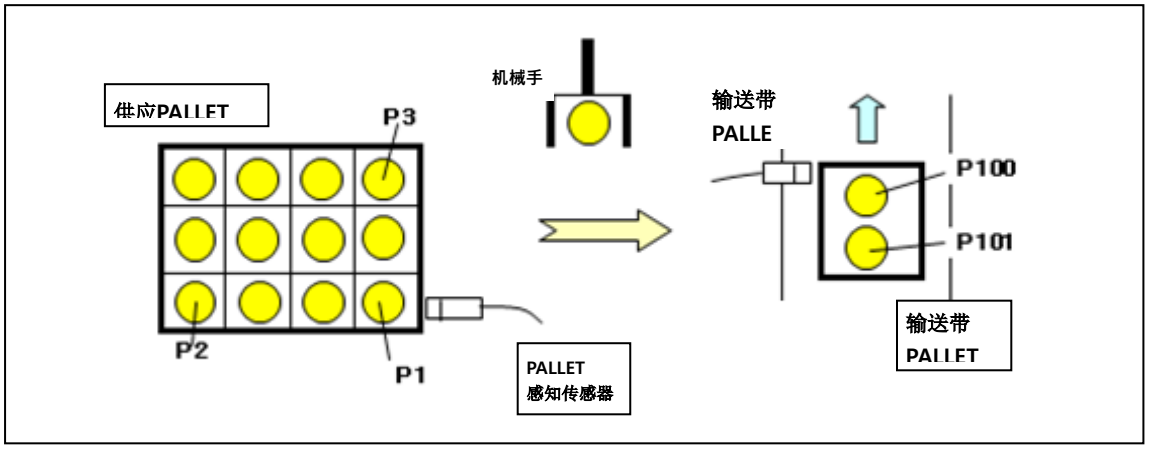
CAUTION

PALLET作业模式 NO. 1 ~ 3请参照使用说明书的"第5章 关于参数模式"的内容(从ROM版本 Ver 2.0A 开始适用).

3.18.3程序使用例题 (1)

1) 作业内容

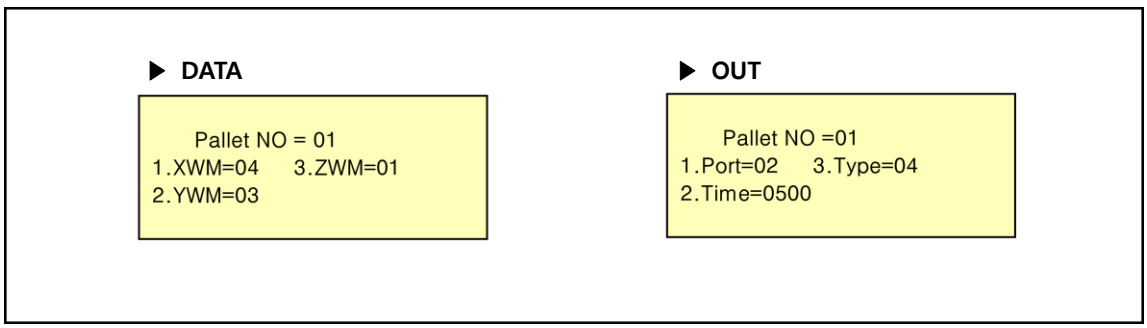
→ 机器人抓住供应PALLET的零件，给输送带PALLET供应零件.



2) 用户(USER) I/O

IN		OUT	
0	感知供应 PALLET	0	开始机械手抓取
1	感知输送带 PALLET	1	开始放开机械手
2	感知机械手抓取	2	完成 PALLET 作业
3	感知机械手放开	3	完成一个周期
4		4	

3) 在参数设定PALLET数据实例



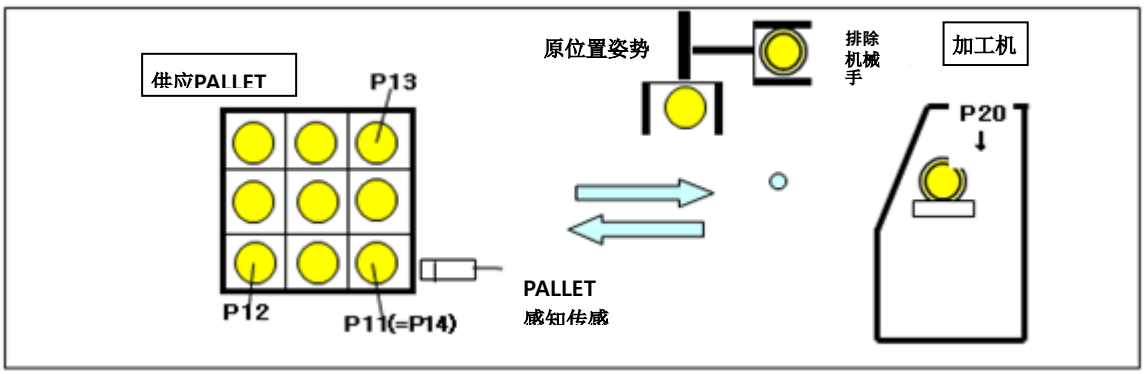
4) 编程

```
MAIN
VEL 500          .....   把速度设定为50%
PLUP 5           .....   把Z轴PULL UP设定为5mm
POUT0=0          .....   输出 ALL OFF
CALL HOFF        .....   放开机械手
LABL A0
IN0=1            .....   等待到PALLET到位
CALL PALT        .....   呼叫PALLET作业子程序
IN1=1            .....   等待到输送带PALLET到位
JMOV P100        .....   向输送带位置P100移动
CALL HOFF        .....   放开机械手
CALL PALT        .....   呼叫PALLET作业子程序
JMOV P101        .....   向输送带位置P101移动
CALL HOFF        .....   放开机械手
OUT3=1 100       .....   完成一个周期
GOTO A0          .....   跳到LABL A0
EOP

SUBRPALT         .....   抓取PALLET零件的作业
PMOV P1 P1
CALL HON
DLAY 100
RET
SUBR HOFF        .....   放开机械手
OUT0=0
OUT1=1
IN3=1
RET
SUBR HON         .....   抓取机械手
OUT1=0
OUT0=1
```

3.18.4程序使用例题 (2)

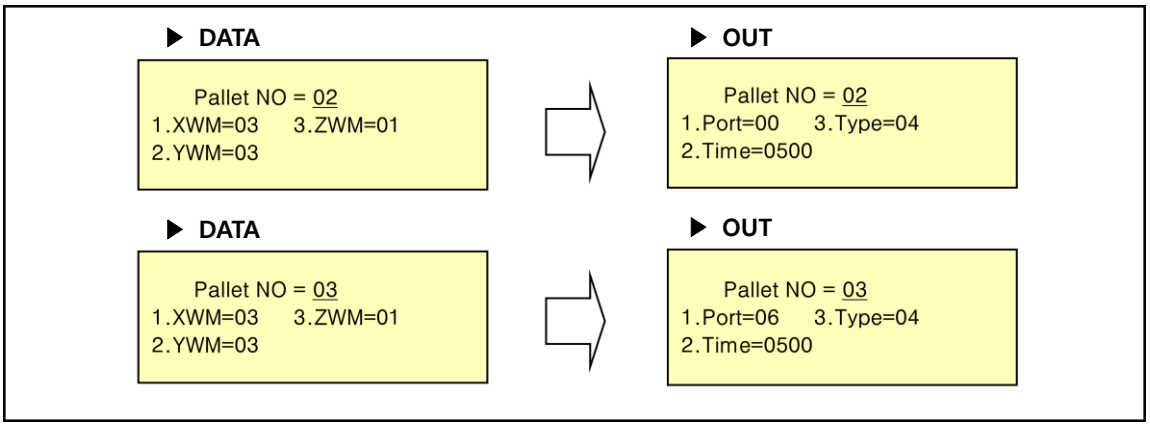
1) 作业内容 → 机器人抓取零件供应PALLET的零件后供给加工机, 然后把加工完成的零件重新装到供应PALLET



2) 用户(USER) I/O

IN		OUT	
0	供应 PALLET 开始作	0	完成 PALLET 加工品排出作业
1	加工机开始作	1	完成给加工机供应加工品(开始加工)
2	感知供应机械手抓	2	开始供应机械手抓取
3	感知供应机械手放	3	开始供应机械手放开
4	感知排出机械手抓	4	开始排出机械手抓取
5	感知排出机械手放	5	开始排出机械手放开
6	感知机械手原位	6	完成 PALLET 加工品供应完成
7	感知机械手 180 度旋	7	开始机械手原位置
8		8	开始机械手 180 度旋转

3) 在参数设定PALLET数据



4) 编程

```
MAIN
POS AP                  .....   指定 POS变数 AP
AP=<0,0,-100,0,0,0>
VEL 500                 .....   把速度设定为50%
PLUP 5                  .....   把Z轴PULL UP设定为5mm
JMOV P1                 .....   移动到等待位置
POUT0=0                 .....   全部 初始化输出信号
CALL H0                 .....   机械手原位置(0度)
CALL HOFF1              .....   供应机械手放开
CALL HOFF2              .....   排出机械手放开

LABL A0                 .....   设定LABLE A0
IN0=1                   .....   PALLET作业开始信号等待
CALL DEPALT             .....   呼叫子程序"DEPALT"
JMOV P1                 .....   等待位置移动
PLUP 0                  .....   解除Z轴PULL UP
IN1=1                   .....   加工机作业开始信号等待
CALL H180               .....   呼叫子程序 "H180"
JMOV P20                .....   移动加工机侧点 P20→P21
JMOV P21                .....
CALL HON2               .....   排出机械手抓取
JMOV P20                .....   移动到加工机侧上端点位P20
CALL H0                 .....   机械手原位置(0度)
JMOV P21                .....   移动到加工机侧下端点位P21
CALL HOFF1              .....   供应机械手放开
JMOV P20                .....   移动到点P20→P1
JMOV P1                 .....
OUT1=1 100              .....   加工开始
CALL H180               .....   呼叫子程序 "H180"
CALL PALT               .....   呼叫子程序 "PALT"
IMOV AP                 .....   相对移动 (Z轴 100mm 上升)
CALL H0                 .....   机械手原位置(0度)
GOTO A0                 .....   跳到LABEL A0
EOP

SUBR DEPALT             .....   "DEPALLETIZING 作业" 子程序
PLUP 5
PMOV P2 P11
CALL HON1
DLAY 100
```

```
SUBR PALT          ..............   "PALLETIZING 作业" 子程序
PLUP 5
PMOV P3 P11
CALL HOFF2
DLAY 100
RET

SUBR HOFF1         ..............   "供应机械手放开" 子程序
OUT2=0
OUT3=1
IN3=1
RET

SUBR HON1          ..............   "供应机械手抓取" 子程序
OUT3=0
OUT2=1
IN2=1
RET

SUBR HOFF2         ..............   "排出机械手放开" 子程序
OUT4=0
OUT5=1
IN5=1
RET

SUBR HON2          ..............   "排出机械手抓取" 子程序
OUT5=0
OUT4=1
IN4=1
RET

SUBR H0            ..............   "机械手原位置(0度) 作业" 子程序
OUT8=0
OUT7=1
IN6=1

SUBR H180          ..............   "机械手旋转(180度) 作业" 子程序
OUT7=0
OUT8=1
IN7=1
```

3.18.5程序使用例题（3)

1）作业内容

进行多个PALLET作业时，电源OFF/ON也也可以记住作业中的PALLET编号，可以继续进行作业.

编程时，宣言整数型变数(INT 变数名)，在此变数读入并保存PIN9 值，则以前进行的PALLET编号被保存。
在程序内部比较保存在此参数的PALLET编号编写程序.

▶ 例）使用 2个PALLET作业

```
MAIN
INT PALNO          .............   定义整数型变数 "PALNO"
PLUP 5
VEL 300
PALNO=PIN9         .............   把PIN9的值保存在PALNO
IF PALNO=1 THEN                    如果PALNO的值是 "1"
GOTO PAL1                          则NO.1 PALLET在作业
ELSE
IF PALNO=2 THEN                    如果PALNO的值是"2"
GOTO PAL2                          则NO.2 PALLET在作业
ENDIF
ENDIF

LABL LOOP
LABL PAL1
PMOV P1 P10
IF OUT1=0 THEN                     如果 NO.1 PALLET 作业没有完成
GOTO PAL1                          则继续作业 NO.1 PALLET
ELSE
GOTO PAL2
ENDIF

LABL PAL2
PMOV P2 P20
IF OUT2==0THEN                     如果 NO.2 PALLET 作业没有完成
GOTO PAL2                          则继续作业 NO.2 PALLET
ELSE
GOTO LOOP
ENDIF
EOP
```

▶设定参数PALLET数据

3. OUT

Pallet No = 1 1. Port : 01 2. Time : 0200 3. Type : 04	Pallet No = 2 1. Port : 02 2. Time : 0200 3. Type : 04

▶ 电源ON/OFF以后，如上面程序先宣言整数型变数后把PIN9保存到此变数则可以记住以前作业中的PALLET编号

3.19 PASS

命令语	功 能	形 式
PASS	作PALLET作业时 跳过指定的序号	PASS_<PALLET NO>_<要跳过的加工品序号>

3.19.1说 明

▶ 有关命令语 : PMOV

▶ 1 PALLET PASS 使用数 : 20个

→ 如果使用20个以上则发生 "E150 (Passing PLT over)".

▶ 一个JOB能使用PASS命令的PALT数是最多5个

→ 参照 程序例(2)内容

3.19.2程序使用例题 (1)

1) 作业内容

→ 要按右图作PALLT 作业.

只对有标记的地方进行作业

1	2	3
4	5	6
7	8	9

2) 编辑程序 编程

```
MAIN
VEL 200
PASS 1 1
PASS 1 3
PASS 1 7
PASS 1 9
LABL A0
PLUP 10
PMOVP1 P10
DLAY 10
GOTO A0
EOP
```

不对PALLET NO.1DE 1,3,7,9号加工品进行作业

3.19.3程序使用例题 (2)

```
MAIN
VEL 200
PLUP 10
LABL A0
PASS 1 1
PASS 1 3          不对PALLET NO.1的 1, 3 号加工品进行加工
PMOV P1 P10
DLAY 10
PASS 2 2
PASS 2 4          不对PALLET NO.2的 2, 4号加工品进行加工
PMOV P2 P10
DLAY 10
PASS 3 5
PASS 3 6          不对PALLET NO.3的 5, 6号加工品进行加工
PMOV P3 P10
DLAY 10
PASS 4 1
PASS 4 4          不对PALLET NO.4的 1, 4号加工品进行加工。
PMOV P4 P10
DLAY 10
PASS 5 2
PASS 5 3          不对PALLET NO.5的 2, 3号加工品进行加工
PMOV P5 P10
DLAY 10
GOTO A0
EOP
```

3.20 WITH, ENDWT (同时处理命令语)

功 能　机器人动作时同时处理下一句命令.

形 式　WITH

…

ENDWT

说 明　1) 运行WITH句内的最初 **JMOV 命令**同时并列处理以后的命令.

2) 使用MVR 变数，把机器人移动距离按百分比分好后，在需要的地点处理条件式.

CAUTION

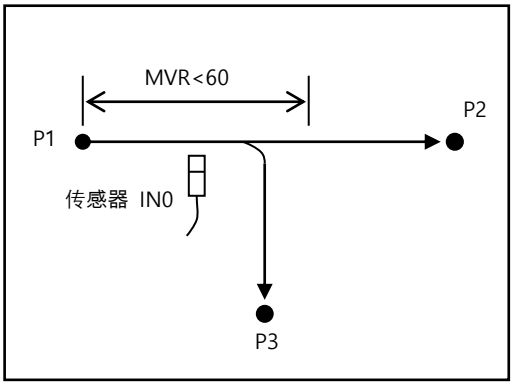
在WITH句内以条件满足退出 WITH ~ ENDWT区间时,
跳过的最初STEP必须插入ENDWT.

3.20.1程序使用例题 (1)

1) 作业内容 →

从P1移动到P2时输入 INO 信号为ON则停止移动，向P3移动

(但，传感器输入信号只感知从 P1到P2为止的移动距离的 60%)



2) 用户(USER) I/O

IN		OUT	
0	感知传感	0	P2 到位完成
1		1	P3 到位完成

3) 编程

```
MAIN
VEL 500
JMOV P1
MVR=0          ..........   初始化移动区间百分比
WITH           ..........
JMOV P2        ..........   向点P2移动(只可以JMOV)
WHILE MVR<60   ..........   移动区间达到60%左右
IF IN0==1 THEN ..........   如果 IN0=1则跳到 "LABL BB"
GOTO BB
ENDIF
ENDWL
OUT0=1
ENDWT

LABL BB
STOP           ..........   机器人停止运动
ENDWT          ..........   结束WITH语句（必须插入）
JMOV P3
OUT1=1
EOP
```

3.21 OUT, POUT (外部输出命令语)

功 能　以 BIT单位或 PORT单位输出指定的值.

形 式　OUT<BIT输出PORT编号>=<0 或 1>_[脉冲有效时间] [→ 或 ~]
OUT<(整数型变数)>=<0 或 1>
POUT<输出PORT编号>=<数据>
<变数>=POUT<输出PORT编号>

用 语　<BIT输出PORT编号> : 设定BIT输出PORT编号. (0 ≤ BIT输出编号 ≤ 95)
1) 用户输出(User Output) : OUT0 - OUT15
2) 扩展1 输出(Option Output1) : OUT16 - OUT47 → 安装1个扩展 I/O Card 时适用
3) 扩展2 输出(Option Output2) : OUT48 - OUT79 → 安装2个扩展 I/O Card 时适用
<脉冲有效时间> : 对应BIT输出PORT指定的有效时间内输出0/1值的(脉冲时间)时间

[→ 或 ~] :
-. 输出以脉冲有效时间为周期的周期波形
-. 脉冲输出,周期波形输出最多可以使用32点(User,Option1,Option2中 1个).
-. 不能重复使用Port。(例) User OUT 16点 + Option1 OUT 16点)
1) "脉冲输出"和"周期波形输出"是与机器人移动同时.
即,下一行是移动命令(MOVE)时移动的同时把把输出BIT ON或OFF.
2) "脉冲输出"和"周期波形输出"是在参数设定要使用的Port.
设定方法请参照 " 参数设定说明书.

<输出PORT编号> : 设定输出PORT编号. (0 ≤ 输出PORT编号 ≤ 4)
-. 0 时 : 指定OUT0 - OUT15
-. 1 时 : 指定OUT16 - OUT31
-. 2 时 : 指定OUT32 - OUT47
-. 3 时 : 指定OUT48 - OUT63
-. 4 时 : 指定OUT64 - OUT79
<数据> : 指输出到对应输出PORT的值(2进数, 16进数)
例1) POUT0=10 时 (10(10进制) => 0000 0000 0000 1010 (2进制))
输出PORT 1 和 输出PORT 3 ON(1) 其他输出PORT OFF (0).
例2) POUT0=0H0F0F 时 (0H0F0F (16进制) => 0000 1111 0000 1111 (2进制))
输出PORT 1 和 输出PORT 3 ON(1) 其他输出PORT OFF (0)

说 明 **OUT<BIT输出PORT编号>=<0 或 1>_[<脉冲有效时间>]_[→]**

1) 指定的脉冲有效时间中，向对应BIT输出PORT输出0/1值.
2) 如果没有脉冲有效时间则一直有效，如果一过了脉冲有效时间复位到以前状态.
3) 脉冲有效时间的单位是10ms

OUT<(整数型变数)>=<0 或 1>

在"BIT输出PORT编号"可以使用整数型变数.

POUT<输出PORT编号>=<数据>

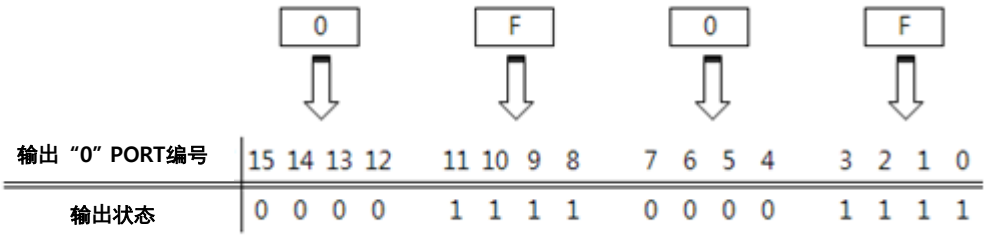
把数据值输出到对应输出PORT(16BIT).

⚠CAUTION

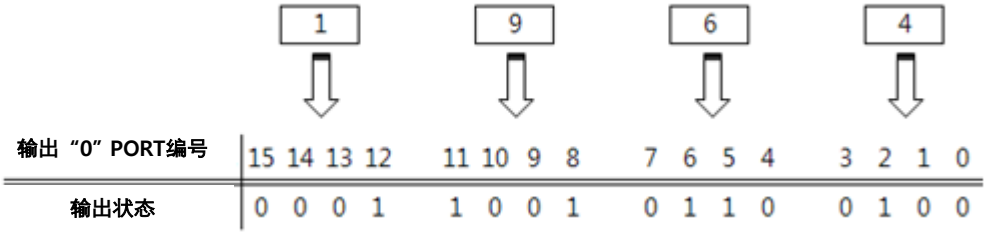
▶ 请注意OUT,POUT和 "BIT输出PORT编号"," 输出PORT编号"之间不要有空白(Blank).
例) OUT_15=1,POUT_0=0H00FF (Syntax Error 发生)

▶ 输出PORT编号数据是从右到左 0,1,2 ~ 15 .
▶ **内部接点不被脉冲输出控制.**
▶ 扩展 I/O 版是得购买.

POUT0=0H0F0F



POUT0=0H1964



3.21.1程序使用例题

1) 脉冲输出，数据输出

```
MAIN
VEL 100
JMOV P10
OUT11=1 100          ..............    输出 PORT 11号 100ms  ON(1)
LMOV P100
POUT0=0H1004         ..............    输出PORT 0~15号 0H1004(16进制)值输出
EOP
```

2) 把BIT输出PORT编号使用为变数

```
MAIN
INT A                ..............    指定整数型变数 A
FOR A=0 TO 10        ..............    反复执行 0 ~ 10
OUT(A)=0             ..............    OUT0,OUT1 ~ OUT10 的输出全部 OFF(0)
NEXT
EOP
```

3.22 IN, PIN (外部输入命令语)

功 能　指定的 BIT单位或PORT单位的输入接收值

形 式　IN<BIT输入PORT编号>=<0 或1>

PIN<输入PORT编号>=<整数值或变数>

<变数>=IN<输入BIT编号>

<变数>=PIN<输入PORT编号>

用 语　<BIT输入PORT编号> : 设定BIT输入PORT编号。 (0 ≤ BIT输入编号 ≤ 95)

1) 用户输入(User Input) : IN0 - IN15

2) 扩展1 输入(Option Input1) : IN16 - IN47 → 安装1个扩展 I/O Card 时适用

3) 扩展2 输入(Option Input2) : IN48 - IN79 → 安装2个扩展 I/O Card 时适用

<输入PORT编号> : 设定输入PORT编号

PIN, WIN 时 : 16bit 绑定(word).

-. 0 时 : 指定IN0 - IN15

-. 1 时 : 指定IN16 - IN31

-. 2 时 : 指定IN32 - IN47

-. 3 时 : 指定IN48 - IN63

-. 4 时 : 指定IN64 - IN79

说 明　**IN<BIT输入PORT编号>**

把指定的BIT输入PORT的 ON/OFF状态(1或 0)保存到 IN<BIT输入PORT编号>变数 .

PIN<输入PORT编号>

读入指定的输入PORT的值(16BIT)后保存到 PIN<输入PORT编号>变数.

IN<BIT输入PORT编号>=< 0/1 >

等待到指定的BIT输入PORT变为 ON/OFF状态(1或 0).

PIN<输入PORT编号>=<16BIT值>

等待到指定的输入PORT的值成为指定的值(16BIT).

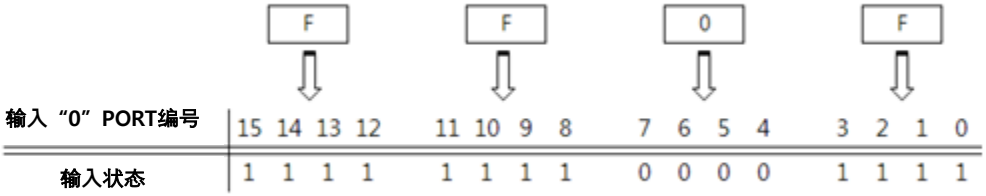
⚠CAUTION

▶ 请注意IN,PIN和 “BIT输入PORT编号”,” 输入PORT编号”之间不能有空白(Blank).
例) AA=IN_15, AA=PIN_0 (发生Syntax Error)

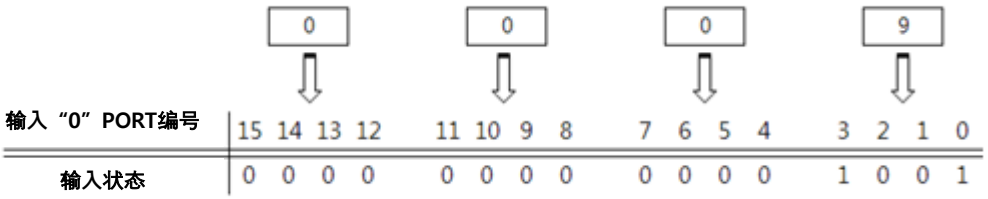
▶ **内部接点不被脉冲输出控制.**

▶ 扩展 I/O 版是得购买.

PIN0=0HFF0F

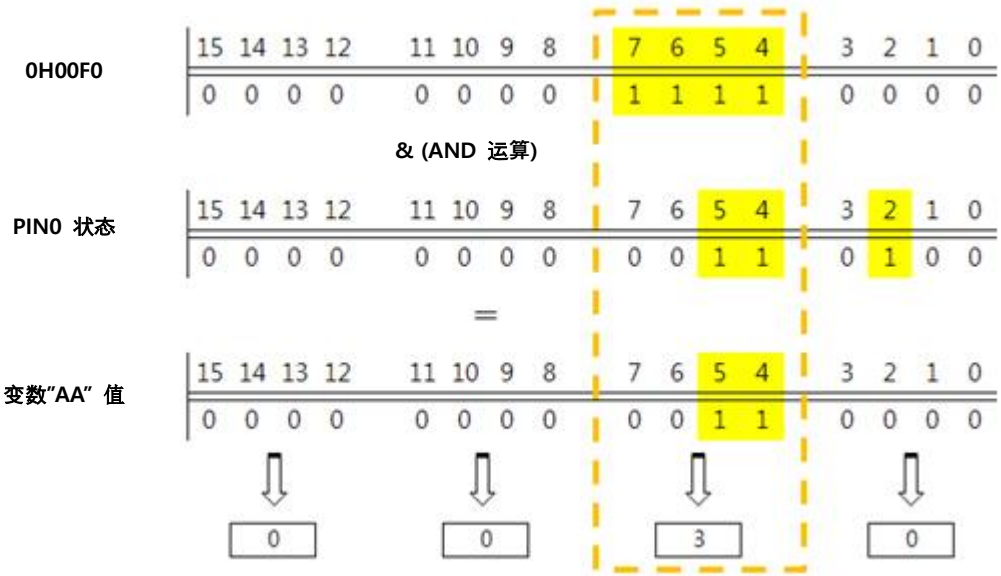


PIN0=0H0009



输入 "0"号PORT的输入中只想检查 "IN4~IN7" 之间的信号?

AA=0H00F0&PIN0



3.22.1程序使用例题

1) 等待输入，Port输入整数变数处理移动选择

```
MAIN
INT D1                  ..............  定义整数型变数D1
VEL 100
WHILE 1
JMOV P0
IN1=1                   ..............  等待到IN1为 ON
IF IN0==1 THEN          ..............  BIT输入PROT 0号 ON(1)时
D1=PIN0                 ..............  输入PORT  0~15 的状态保存到 D1
ELSE
PIN0=0HFFFF             ..............  输入PORT  0~15 全部 ON(1)
ENDIF
IF D1==0HFF0F THEN      ..............  D1的状态跟（ PIN0的状态）  0HFF0F比较
JMOV P1
ELSE
JMOV P2
ENDIF
EOP
```

2) 使用 BCD 单元的 MODEL 比较

- BCD 输入是 IN0 ~ IN7输入使用例题

```
MAIN
INT M1,M2,MD            ..............  指定整数型变数  M1,M2,MD
WHILE 1
M1=PIN0 & 0H000F        ..............  在 M1保存 PIN0和 0H000F的比较
M2=PIN0&0H00F0          ..............  在 M2保存 PIN0和 0H000F的比较
M2=(M2>>4)*10           ..............  M2的值转到 1位置后乘 10)
MD=M1+M2                ..............  运算的 M1和 M2的值相加
IF MD==1 THEN
JMOV P0
ELSE
IF MD==11 THEN
JMOV P1
ENDIF
ENDIF
EOP
```

CAUTION

▶ PORT输入时 16进制运算， BCD 单元输入 10控制器识别16,
这时以上转到运算后 10进制正确的变化才行.

3.23 CIN,CBIN,CWIN,CDIN,CFIN(输入FIELDBUS 命令语)

功 能　从 FIELDBUS接收指定的BIT单位或PORT单位的输入值

形 式　Bit 领域 命令语

<整数型 变数>=CIN<输入BIT编号>

<整数型 变数>=CIN<整数型 变数>

<整数型 变数>=CBIN<输入PORT编号>

Word 领域 命令语

<整数型 变数>=CWIN<输入PORT编号>

<整数型 变数>=CDIN<输入PORT编号>

<实数型 变数>=CFIN<输入PORT编号>

用 语　Bit 领域: 1Bit 单位读取和写入的领域.

<输入BIT编号>: 设定输入 BIT编号 (0≤输入BIT编号≤127)

1) System 领域 : CIN0~CIN31(不能使用)
2) User 领域 : CIN32 ~ CIN127

<输入PORT编号>: 设定输入PORT 编号 .(0≤输入PORT编号≤15)

1) System 领域 : CBIN0~CBIN3(不能使用)
2) User 领域 : CBIN4~CBIN15
   - PORT编号 0: CIN0 ~ CIN7
   - PORT编号 1: CIN8 ~ CIN15
   - PORT编号 2: CIN16 ~ CIN23
   - PORT编号 4: CIN24 ~ CIN31
   - PORT编号 5: CIN32 ~ CIN39

Word 领域: Word(16Bit)单位读取和写入的领域.

<输入PORT编号>: 设定输入PORT 编号.(0≤输入PORT编号≤15).

1) User 领域: CWIN0 ~ CWIN15
2) CDIN<PORT编号>: Word 领域值以Double Word(32Bit)单位接收(整数).
   - PORT编号 0: CWIN0 ~ CWIN1
   - PORT编号 1: CWIN2 ~ CWIN3
   - PORT编号 2: CWIN4 ~ CWIN5
3) CFIN<PORT编号>: Word 领域值以Double Word(32Bit)单位接收(整数).
   - PORT编号 0: CWIN0 ~ CWIN1
   - PORT编号 1: CWIN2 ~ CWIN3
   - PORT编号 2: CWIN4 ~ CWIN5

说 明　CIN<输入BIT编号>

指定的 BIT 输入PORT的 ON/OFF状态(1或0)保存在 IN<BIT输入PORT编号>变数.

CIN<整数型 变数>

"输入BIT编号"中可以使用整数型变数

CBIN<输入PORT编号>

读取指定输入PORT的值(8Bit)保存在 CBIN<输入PORT编号>变数.

CWIN<输入PORT编号>

读取指定输入PORT的值(Word)保存在 CWIN<输入PORT编号>变数.

CDIN<输入PORT编号>

读取指定输入PORT的值(Double Word)保存在 CDIN<输入PORT编号>变数.

CFIN<输入PORT编号>

读取指定输入PORT的值(Double Word)保存在 CFIN<输入PORT编号>变数.

CAUTION

- 注意 CIN,CBIN,CWIN,CDIN,CFIN 和 “输入BIT编号”,” 输入PORT编号”键不能空白(Blank).
  例) AA=CIN_15, AA=CBIN_0 (发生 Syntax Error)
- 输入PORT编号的数据值从右侧到左侧递增 0,1,2 ~ 15 .
- FIELDBUS 命令语只适用于 PARA->PUB->HWCONF->COMM->FieldBus->MAP的USER_PEF.

3.23.1程序使用例题

```
MAIN
INT D1                  .....   指定整数型变数 D1
VEL 100
WHILE 1
JMOV P0
IF CIN50==1 THEN        .....   BIT 输入 PORT 51编号 ON(1) 时
D1=CBIN5                .....   输入PORT 40~47 状态保存在 D1
ELSE
D1=CBIN6                .....   输入PORT 48~45状态保存在 D1
ENDIF
IF D1==0H0F THEN        .....   D1状态跟 0H0F比较
JMOV P1
ELSE
JMOV P2
ENDIF
EOP
```

3.24 COUT,CBOUT,CWOUT,CDOUT,CFOUT(输出FIELDBUS 命令语)

功 能　从 FIELDBUS输出指定的BIT单位或PORT单位值

形 式　Bit 领域 命令语

COUT<输出BIT编号>=<0或1>

COUT<整数型变数>=<0或1>

CBOUT<输出PORT编号>=<数据>

Word 领域 命令语

CWOUT<输出PORT编号>=<数据>

CDOUT<输出PORT编号>=<数据>

CFOUT<输出PORT编号>=<数据>

用 语　Bit 领域: 1Bit 单位读取和写入的领域.

<输出BIT编号>: 设定输出 BIT编号 (0≤输出BIT编号≤127)

1) System 领域 : COUT0~COUT31(不能使用)
2) User 领域 : COUT32 ~ COUT127

<输出PORT编号>: 设定输出PORT 编号 .(0≤输出PORT编号≤15)

1) System 领域 : CBOUT0~CBOUT3(不能使用)
2) User 领域 : CBOUT4~CBOUT15
   - PORT编号 0: COUT0 ~ COUT7
   - PORT编号 1: COUT8 ~ COUT15
   - PORT编号 2: COUT16 ~ COUT23
   - PORT编号 4: COUT24 ~ COUT31
   - PORT编号 5: COUT32 ~ COUT39

Word 领域: 1Word(16Bit)单位读取和写入的领域.

<输出PORT编号>: 设定输出PORT 编号.(0≤输出PORT编号≤15).

1) User 领域: CWOUT0 ~ CWOUT15
2) CDOUT<PORT编号>: Worad 领域值以 Double Word(32Bit)单位(整数)输出.
   - PORT编号 0: CWOUT0 ~ CWOUT1
   - PORT编号 1: CWOUT2 ~ CWOUT3
   - PORT编号 2: CWOUT4 ~ CWOUT5
3) CFOUT<PORT编号>: Worad 领域值以 Double Word(32Bit)单位(整数)输出.
   - PORT编号 0: CWOUT0 ~ CWOUT1
   - PORT编号 1: CWOUT2 ~ CWOUT3
   - PORT编号 2: CWOUT4 ~ CWOUT5

说 明　COUT<输出BIT编号>=<0或1>

把 ON/OFF状态(1或0)以对应的输出BIT 编号输出.

COUT<整数型변수>=<0或1>

"输出BIT编号"中可以使用整数型 变数.

CBOUT<输出PORT编号>=<数据>

数据(8Bit)值以对应 输出PORT 输出.

CWOUT<输出PORT编号>=<数据>

数据(1Word)值以对应 输出PORT 输出.

CDOUT<输出PORT编号>=<数据>

数据(2Word)值以对应 输出PORT 输出.

CFOUT<输出PORT编号>=<数据>

数据(2Word)值以对应 输出PORT 输出.

CAUTION

- 注意 CIN,CBIN,CWIN,CDIN,CFIN 和 “输出BIT编号”,” 输出PORT编号”键不能空白(Blank).
  例) AA=CIN_15, AA=CBIN_0 (发生 Syntax Error)
- 输出PORT编号的数据值从右侧到左侧递增 0,1,2 ~ 15.
- FIELDBUS 命令语只适用于 PARA->PUB->HWCONF->COMM->FieldBus->MAP的USER_PEF.

3.24.1程序使用例题

1) Bit 领域 输出

```
MAIN
INT A                ..............   指定整数型 变数 A
FOR A=32 TO 42       ..............   32 ~ 42重复执行
COUT(A)=0            ..............   OUT32,OUT33 ~ OUT42的输出都 OFF(0)
NEXT
EOP
```

2) Word 领域 输出

```
MAIN
POS CURR
WHILE 1
CURR=HERE            ..............   当前 ROBOT坐标保存在 CURR
CFOUT0=CURR.1        ..............   CURR 1轴值以 FIELDBUS (CWOUT0,CWOUT1) 输出
CFOUT1=CURR.2        ..............   CURR 2轴值以 FIELDBUS (CWOUT2,CWOUT3) 输出
ENDWL
EOP
```

3.25 VEL (设置轴移动速度命令语句)

功 能　　设定轴移动速度的百分比(%)

形 式　　VEL <速度>

用 语　　<速度> : 设定机器人的移动速度。(0 ≤ <速度> ≤ 100%)
机器人移动速度 = Mv (各轴的最大速度)*0.01*速度
(0 ≤ <移动速度> ≤ 最大 RPM).

说 明　　1) 是机器人移动时的速度，最大速度设定为100
2) 程序里没有设定速度，则默认 参数 INIT_V中设定的速度.
3) 可以使用有速度值的变数(整数型变数).

⚠CAUTION

▶ 速度值是 100.0% .
VEL 1000时，以 100%的速度移动. (VEL 100 => 10%)
▶ 超出机械部最大 RPM 运行，则发生噪音及损坏的危险.
必须确认机械部的铭牌后使用.

3.25.1程序使用例题

1) 按作业位置变更速度

```
MAIN
WHILE 1
VEL 1000          ..............   ROBOT 移动速度 MV x 0.001 x 1000
JMOV P0
JMOV P1
VEL 200           ..............   ROBOT 移动速度 MV x 0.001 x 200
LMOV P2
LMOV P3
VEL 1000          ..............   ROBOT 移动速度 MV x 0.001 x 1000
JMOV P0
ENDWL
EOP
```

3.26 ACC, DEC (设定加减速命令语)

功 能　设定加速时间的百分比(%)

形 式　ACC_<加速时间比率>

DEC _<减速时间比率>

用 语　<加减速时间比率> : (50 ≤ <加减速时间比率> ≤ 200%)

-. 在程序里增加或减少加减速时间

-. 加减速时间 = At(加减速时间)*0.01*加减速时间比率(%)

(0.5'最大加减速时间 ≤<加减速时间> ≤ 2'最大加减速时间)

说 明

1) 设定机器人的加减速时间.
2) 如果在程序内部没有设定加减速时间，则自动设定为100%.
3) 可以使用具有加减速比率值的变数(整数型变数)
4) 适用于所有轴。(SCARA：A,B,Z,W轴适用)

例）参数 At ：A轴(0.3秒)，B轴(0.3秒)，Z轴(0.2秒)，W轴(0.5秒)
ACC(DEC) 50 执行
At：A轴(0.15秒)，B轴(0.15秒)，Z轴(0.1秒)，W轴(0.25秒)
▶ 参数值本身不被修改，只在JOB程序里适用

5) 参数加减速时间(At)恢复时使用 "ACC(DEC) 100" 适用于下一个步骤.
(参照 程序例题)

CAUTION

▶ 参数里设定的值是根据机械部设定的最佳值.

▶ ACC/DEC命令语的加减速时间比率比100小的时候，有可能发生机械部的噪音或振动请注意.

3.26.1程序使用例题

1) 根据作业位置变更加减速时间

- 参数中 At = 0.3s 时的例题

```
MAIN
WHILE 1
VEL 500
ACC 50          ..............  加减速设定 0.15s
JMOV P1
ACC 200         ..............  加减速设定 0.6s
LMOV P2
LMOV P3
ACC 100         ..............  加减速设定 0.3s
JMOV P0
ENDWL
EOP
```

3.27 FOS，PFOS，(生成连续轨迹命令语)

功 能　到达目标点之前向下一个目标点改变轨迹

形 式　FOS_<距离比率>

PFOS_<距离>

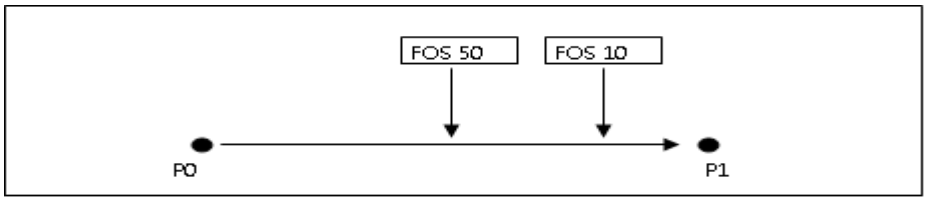
用 语　<距离比率> : (0 ≤ <距离比率> ≤ 100)

-.设定机器人到达指定位置之前移动到下个位置或输出时的始点.

-.距离比率是对两点(Point)之间移动距离的百分比设定.

-. 除了 JMOV以外的模式中不能大于 50%

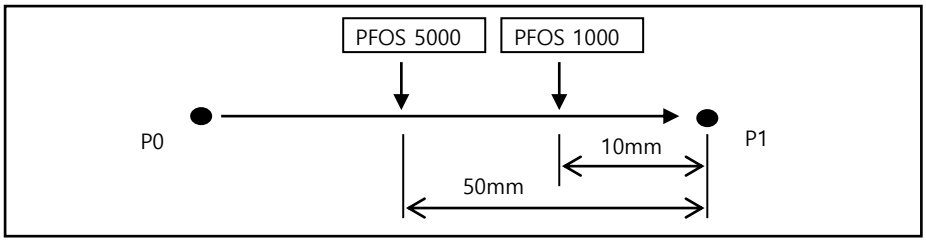
(0 ≤ <距离比率> ≤ 50)



上图表示从P0移动到P1时，根据 FOS值向下一个移动或输出的始点.

<距离> : 设定机器人到达指定位置之前移动到下个位置或输出时的距离

(0 ≤ 距离 ≤ 999mm) 单位: 0.01mm



上图表示从P0到 P1时，根据PFOS值，向下一个位置移动或输出的始点

.

CAUTION

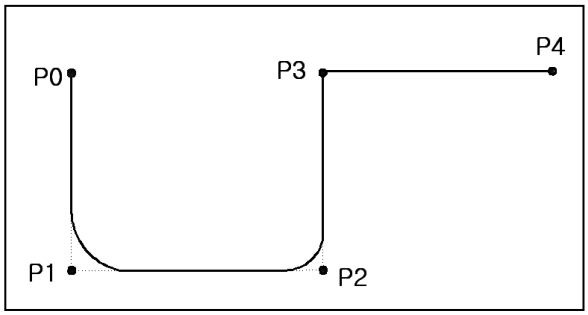
- ▶ 做连续差不移动提点位时，两点间距离不能小于 5mm
  (但 100mm/s 移动速度时)
- ▶ 移动中的轨迹 FOS 比率小于下一个位置整体长度时 "Too much FOS"发生Alarm .
- ▶ 移动轨迹 PTP时轨迹变更差不移动时轨迹插入 (园,弧或直线).

说 明

1) 如果设定FOS 或 PFOS，根据MOVE命令移动时到达指定的位置之前(设定的距离比率或距离值）向下一个位置.
2) OUT 命令时，在移动中可以输出.
3) PFOS时只适用于 **(LMOV, CMOV, AMOV)**.
4) 要在作业程序里取消FOS或PFOS，则使用"FOS 0"或 **"PFOS 0",**从2个STEP以后有效.

```
MAIN
FOS 10
LMOV P0
LMOV P1
FOS 0
LMOV P2
LMOV P3
EOP
```

FOS 0 命令从 LMOV P3 开始有效.



CAUTION

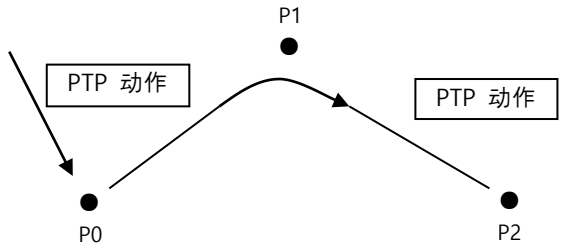
- 因使用FOS,PFOS, 可以进行连续的插补动作.
- 作圆弧(AMOV)或圆(CMOV)插补动作时，如果点位错误则在 Auto RUN中发生"Unreachable Point","Inverse Error".
- 特别是,使用于轨迹小的 AMOV或 CMOV 时注意.

3.27.1程序使用例题

1) PTP 动作 (JMOV)

```
MAIN
VEL 100
JMOV P0
FOS 5
JMOV P1
FOS 0
JMOV P2
EOP
```

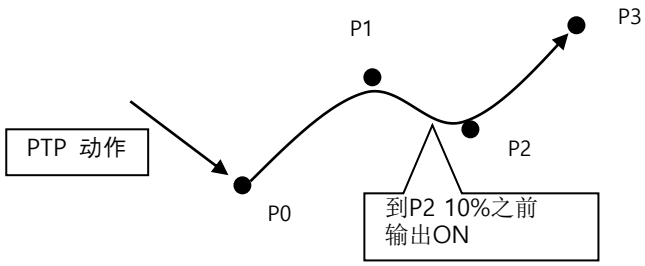
FOS 5 (到点P1 之前 5%)



2) 机器人移动中输出信号 (OUT, POUT) On,Off

```
MAIN
VEL 100
JMOV P0
FOS 10
LMOV P1
LMOV P2
OUT0=1
FOS 0
JMOV P3
EOP
```

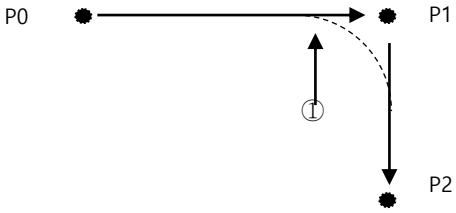
FOS 10



3) 机器人移动中处理下一行命令

```
MAIN
WHILE 1
VEL 100
JMOV P0
FOS 10
IF IN0==1 THEN
JMOV P1
ELSE
IN1=1
JMOV P2
ENDIF
FOS 0
ENDWL
EOP
```

FOS 10



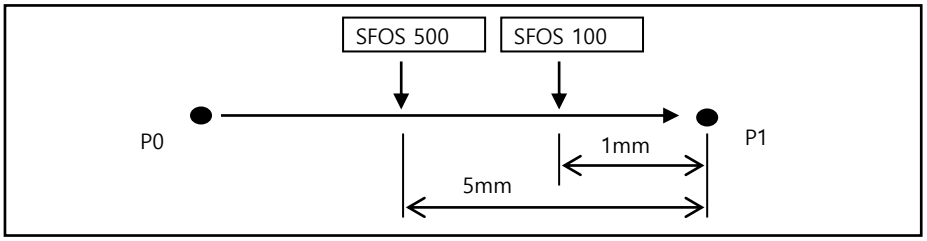
在①位置
到P1 10%之前处理IF命令
即，如果输入 "IN0"有信号，则按虚线的轨迹移动

3.28 SFOS(生成连续轨迹及等速 命令语)

功 能　轴前段到达目标点之前向下一个目标点变更轨迹跟 FOS 相同, 适用 SFOS 命令语的开始以 SFOS 设定的基准值维持等速运动的特性.

形 式　SFOS <距离><速度级别><圆弧角度>

用 语　<距离> : ROBOT 到达指定位置前移动下一个位置或输出时的距离设定
(0 ≤ <距离> ≤ 999mm) 单位: 0.01mm

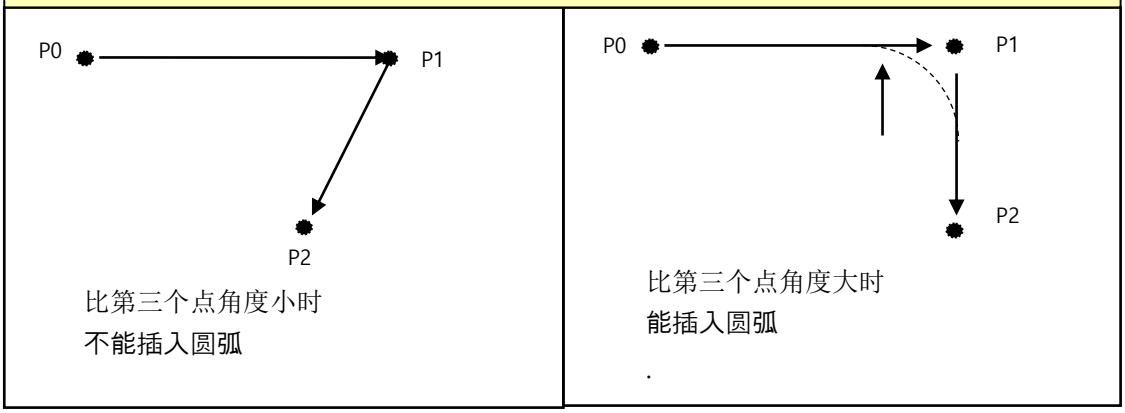


上图是 P0到 P1 移动的时候各 SFOS值 移动到下一个位置或输出时的开.

<速度级别>: SFOS 命令语是设定适用移动区间内的速度.
设定值越高速度越快.(0≤ <速度级别> ≤ 10)
设定值 0时: 跟级别 5相同

<圆弧角度>: 输入连接两点的轨迹和轨迹间插入圆弧角度.
输入角度值小时不能插入圆弧, 大时默认第一个值插入圆弧.
设定值 0度时: 跟 1度值相同
设定值 180度时: 轨迹间不能插入圆弧, 不会发生 Too much fos 报警
设定 180度时轨迹变形及ROBOT移动中发生震动.

SFOS 10 0 80



说 明

1) 设定 SFOS 后, MOVE 命令会到达指定位置前向下一个位置移动（距离值）.
2) **MOVE 命令语键不能使用 DLAY 命令语.**
3) 连续差不移动时没有加减速可以做连续运动(等速移动).
4) **SFOS 插补只适用于(LMOV, CMOV, AMOV) 运动.**
5) 为了作业程序中适用 SFOS 之前状态恢复 输入**"SFOS 0 0 0**
, 适用于下一个命令语.

```
MAIN
SFOS 10 0 0
LMOV P0
LMOV P1
SFOS 0 0 0
LMOV P2
LMOV P3
EOP
```

SFOS 0 命令语
适用于LMOV P3 .

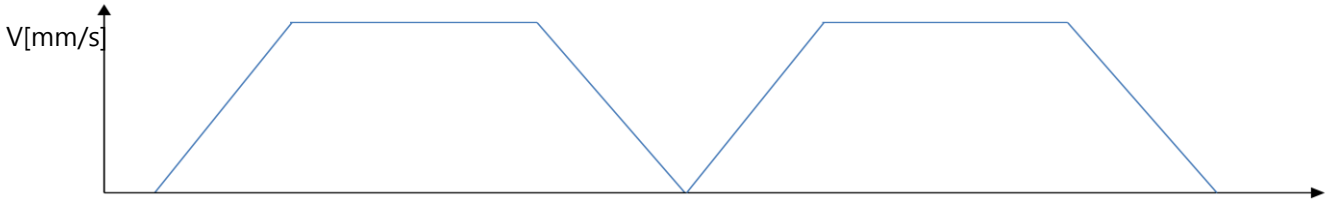
CAUTION

- ▶ 使用 SFOS 使做连续插补动作.
- ▶ 圆弧(AMOV) 或 园(CMOV) 动作时 , 错误 Point Teaching . Auto RUN中发生 "Unreachable Point","Inverse Error" .
- ▶ 特定,使用小的 AMOV或 CMOV 命令语时注意.

3.28.1 连续运动速度 文件模式

1) FOS 0时

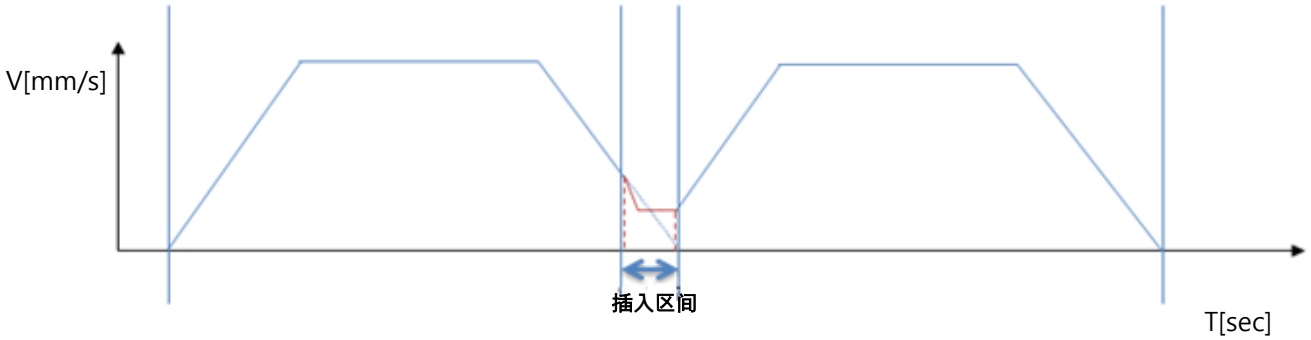
没有 FOS 关联命令语时轨迹连接部分速度为 0.



[FOS命令语 未适用]

2) FOS 10 & PFOS 10

使用 FOS 命令语插入区间(圆弧)的速度是维持插入区间进入的速度.

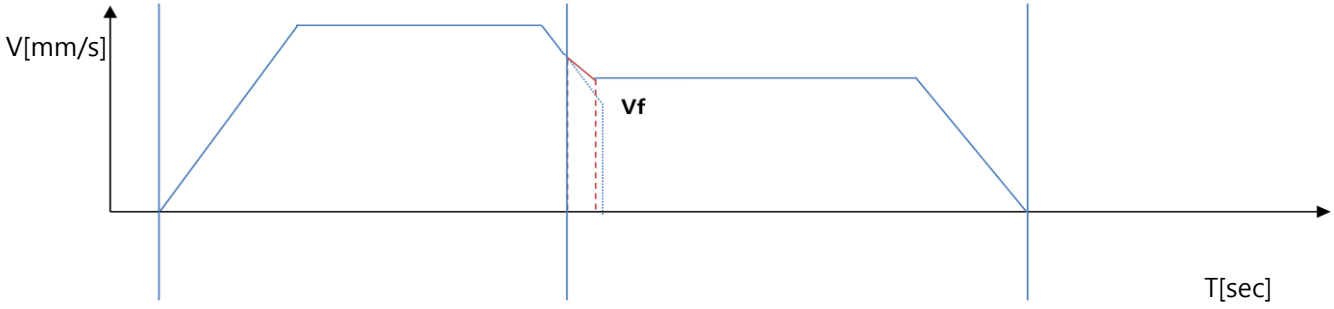


[FOS, PFOS 命令语适用]

3) SFOS 10 0 0

SFOS 设定值和速度级别是下一个轨迹决定最高速度(Vf), 维持移动速度

됩니다.



[SFOS 命令语适用]

CAUTION

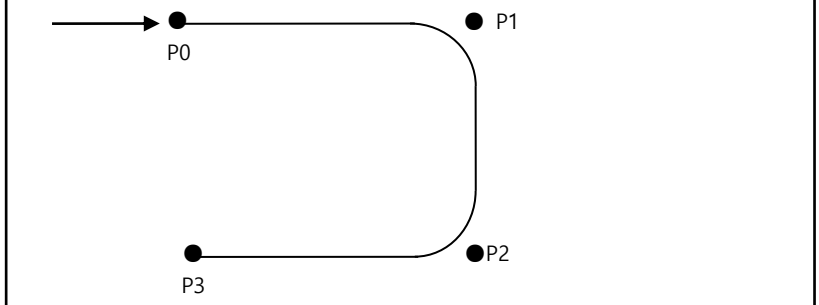
▶ 连接轨迹太短不能维持 vf时, 不能维持等速. 这时降低速度级别, 设定 SFOS 量在区间内的最小长度.

3.28.2程序使用例题

1) 直线插补运动(LMOV)

```
MAIN
VEL 100
LMOV P0
SFOS 150 0 70
LMOV P1
LMOV P2
SFOS 0 0 0
LMOV P3
EOP
```

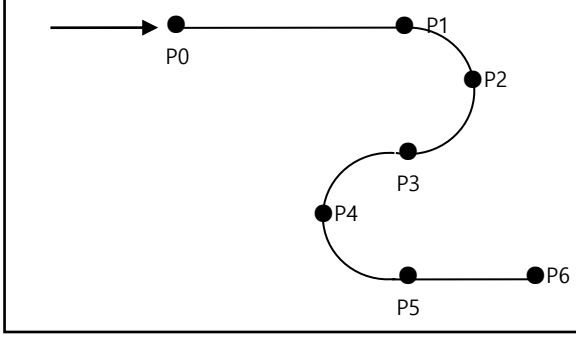
SFOS 150 0 70
P1, P2 到达 15mm 之前向下一个位置移动



2) 圆弧插补运动(AMOV)

```
MAIN
VEL 100
JMOV P0
LMOV P1
SFOS 150 0 180
AMOV P2 P3
AMOV P4 P5
SFOS 0 0 0
LMOV P6
EOP
```

SFOS 150 0 180(0)
P1, P2 到达 1.5mm之前向下一个位置移动



3.29 SVON, SVOF (伺服 ON/OFF 命令语)

功 能　伺服 ON, 伺服 OFF

形 式　SVON　：所有轴伺服 ON
SVON [<指定轴>]　：指定轴伺服 ON
SVOF　：所有轴伺服 OFF
SVOF [<指定轴>]　：指定轴伺服 OFF

说 明　1) SVOF :
-.机器人动作中Servo为 OFF，转换到可以用手移动机器人
-.现在移动中的所有轴(AXIS)为 Servo OFF
2) SVON :
-.把机器人 Servo ON, 与SVOF有关
-. SVOF后没有 SVON状态下运行机器人则
以Dry RUN (机器人 Body不运动，JOB Program 1Step运行的方式) 方式运行.
-. 现在Servo OFF的所有轴 Servo ON
3) SVON(SVOF) <指定轴> : 把指定的单个轴 Servo ON/OFF.. (范围:1~5)
SCARA ROBOT → A轴 : 1, B轴 : 2, Z轴 : 3, W轴 : 4
直角 ROBOT → X轴 : 1, Y轴 : 2, Z轴 : 3, W轴 : 4

3.29.1程序使用例题

1) PTP 动作 (JMOV)

```
MAIN
VEL 100
JMOV P0
IF IN1==1 THEN
SVOF 4 ..... 把第4轴 SERVO OFF
ENDIF
JMOV P1 ..... 第4轴SERVO OFF 状态下向点 P1 移动
IF IN1==0 THEN
SVON 4 ..... 把第4轴 SERVO ON
ENDIF
JMOV P2 ..... 所有轴是SERVO ON 状态下向点 P2移动
ENDWL
EOP
```

2) PTP 动作 (JMOV)

```
MAIN
VEL 500
JMOV P0
MVR=0
WITH
JMOV P1
WHILE MVR<100
IF IN0==1 THEN
STOP                ..............    向P1移动时如果 IN1为 "0"
SVOF                                  停止机器人后SERVO OFF
GOTO L1
ELSE
EXIT
ENDIF
ENDWL

LABL L1
ENDWT
IN0=1                                 输入IN0为 "1"后
SVON                ..............    SERVO ON
JMOV P2
EOP
```

CAUTION

▶ 如果在子程序使用SVON, SVOF, 会发生 "Syntax Error".

例)

```
MAIN
VEL 200
IN0=1
JMOV P0
CALL A0
EOP

SUBR A0
SVOF
OUT1=1
IN1=1
SVON
RET
```

修改程序 ⇒

```
MAIN
VEL 200
IN0=1
JMOV P0
SVOF
CALL A0
SVON
EOP

SUBR A0
OUT1=1
IN1=1
RET
```

3.30 DLAY (延迟时间命令语)

功 能　　设定延迟时间

形 式　　DLAY <延迟时间>

用 语　　<延迟时间> :设定延迟时间使用. (1=10ms)
(1 ≤ <延迟时间> ≤ 999)

说 明　　1) 以数字设定时间.
2) 设定时间的范围是 0.01~9.99秒.
3) 可以使用具有时间值的变数(整数型变数)

3.30.1程序使用例题

1) PTP 动作 (JMOV)

```
MAIN
VEL 100
JMOV P10
DLAY 10          .....          100ms(=0.1秒) 时间内在 P10等待
LMOV P100
EOP
```

3.31 OFFS (指定偏差命令语)

功 能　把目标点移动指定的值

形 式　OFFS P<编号>
OFFS GP<编号>
OFFS <位置型编号>

用 语　<编号> : 设定已提点位的坐标编号.
P – JOB 分别使用的 LOCAL POINT (0 ≤ 编号 ≤ 1999)
GP – 共同使用的 GLOBAL POINT (0 ≤ 编号 ≤ 254)
<位置型 变数> : 是定 POS型变数名.

说 明　1) 在JOB 程序任一点，按各轴直角坐标保存 OFFSET值.
运行OFFS命令后，有移动命令(MOV)时按OFFSET值增减移动.
2) 在运行的JOB内有效.
MAIN 程序的 OFFS值在 Job CALL(JCALL)的程序里页有效。

适用偏差的位置移动.

CAUTION

- JOB程序中保存的 POINT参数“RANG”以内位置 POINT Teaching时, 不会发生“Range OVER” Error.
  但， JOB 程序运行时因 OFFS发生“Range OVER” Error 报警
- Point Teaching 时用 ANGLE 坐标输入.
- OFFS 解除时分配输入, <0,0,0,0,0,0> 值.

3.31.1程序使用例题

P11 <10.23, 22.37, 7.96, 120.00, 0, 0>
P12 <30.23, 22.37, 20.96, 200.00, 0, 0>
P100 <100, 100, 0, 0, 0, 0>

PTP 动作 (JMOV) <SCARA ROBOT>

```
MAIN
VEL 100
JMOV P10        .....   向点 P10 做 PTP 移动
OFFS P100       .....   OFFSET 适用于 保存在 P100 的值
LMOV P11        .....   P11 + P100 的位置做 CP移动 <110.23,122.37,7.96,120.00, 0, 0>
LMOV P12        .....   P12 + P100 的位置做 CP移动 <130.23, 122.37, 20.96,200.00,0, 0>
EOP
```

3.32 LIMT (限制轴命令语)

功 能　　限制各轴的移动范围

形 式　　LIMT P<编号1> P(编号2)
LIMT <位置型变数> (位置型变数)

用 语　　<编号1> : 把负(-) 限制值设定在JOB程序的POINT.
(0 ≤ <编号> ≤ 999)
(编号2) : 把正(+) 限制值设定在JOB程序的POINT.
(0 ≤ (编号) ≤ 999)

说 明　　1) 在JOB程序内设定软件限制.
2) 只在运行的JOB程序里有效.
MAIN程序的LIMT值在Job CALL(JCALL)的程序里也有效.

3.32.1程序使用例题

1) PTP 动作 (JMOV) <SCARA ROBOT>

```
MAIN
VEL 100
JMOV P0
LIMIT P100 P101
JMOV P1
EOP
```

在JOB程序内的软件限位如下.
A: -100 度 ~ +100度
B: -100 度 ~ +100度
Z: 0 ~ 100 mm
W:-300 度 ~ +300度

在水平多关节机器人把限位设定在 P100, P101时

P100 位置: (A:-100.00 B:-100.00 Z:000.00 W:-300.00 E1:-100.00 E2:-100.00)
P101 位置: (A:100.00 B:100.00 Z:100.00 W:300.00 E1:100.00 E2:100.00)

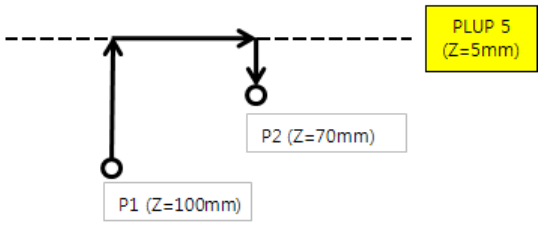
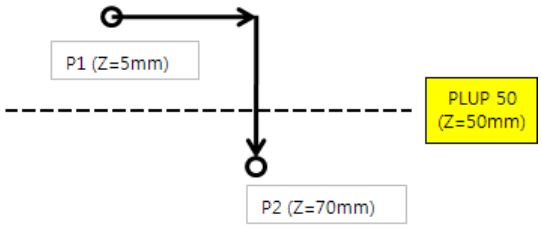
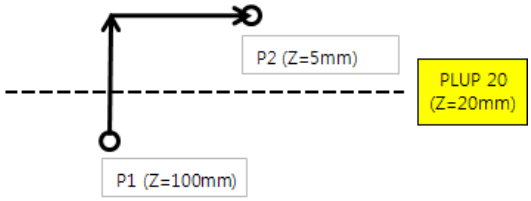
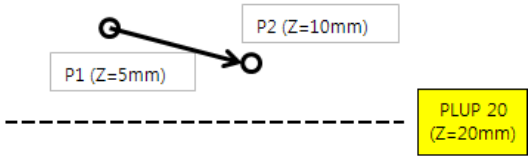
3.33 PLUP (Pull up 动作设定命令语)

功 能　　设定Pull up动作时的Z轴值

形 式　　PLUP <Z轴值>

用 语　　<Z轴值> : 设定Z轴PULL UP值.
(0 ≤ <Z轴值> ≤ Z轴行程(Stroke))

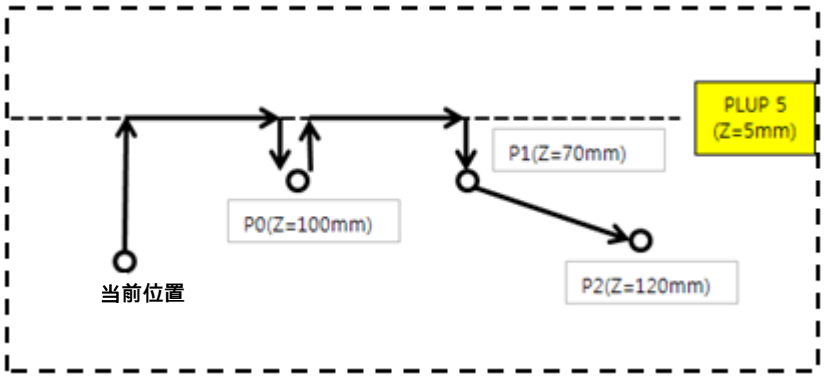
说 明　　1) **只适用于有Z轴的机器人**. 2个以上Z轴时只适用于第一个Z轴.
2) **只适用于PTP移动 (JMOV,PMOV,IMOV) , 插补移动时无效**.
3) Z轴PULL UP值可以使用具有PULL UP 值的变数(整数型变数) .
4) 解除Z轴PULL UP值使用"PLUP 0"命令,从下一行开始有效。
5) 随Z轴PULL UP值的机器人动作如下.

区 分	ROBOT 动作	说 明
1. P1,P2的 Z轴值都比PULL UP值大时		1. 在P1把Z轴上升到PULL UP值 2. 除了Z轴以外的其他轴以P2的PULLUP位置移动 3. Z轴下降到 P2点
2. P1的Z轴值比PULL UP值小, P2的 Z轴值比PULL UP值大时		1. 除了Z轴以外的其他轴以P2的PULLUP位置移动 2. Z轴下降到 P2点
3. P1的Z轴值比PULL UP值大, P2的 Z轴值比PULL UP值小时		1. 在P1把 Z轴上升到P2的 Z轴位置 2. 移动到 P2位置
4. P1,P2的Z轴值都比PUL UP值小时		**与PLUP命令无关从P1直接移动到P2**

3.33.1程序使用例题

1) 向提点位位置坐标移动

```
MAIN
VEL 100
PLUP 5          ..............   把Z轴PULL UP设定为5mm
JMOV P0
JMOV P1
PLUP 0          ..............   解除Z轴PULL UP设定
JMOV P2
EOP
```



3.34 TOOL (限制轴命令语)

功 能　选择安装在ROBOT的 TOOL使用

形 式　TOOL <TOOL编号>

用 语　<TOOL编号> : 设定在参数保存"TOOL"偏差值(dx,dy,dz)的TOOL编号
(0 ≤ <TOOL编号> ≤ 3)

说 明　1) 设定作业时使用的TOOL编号
2) TOOL的形状在系统参数设定.

CAUTION

▶ 在水平多关节机器人, TOOL 0的偏差值成为操作板JOG 移动的基准.
▶ 如果作业程序里不使用" TOOL命令", 则适用 "TOOL 0"偏差值。.
▶ 建议把 "TOOL 0"用于TOOL偏差值初始化用, TOOL 偏差值应用适用TOOL1 ~ 3中的一个.

CAUTION

▶ 在使用有偏差值的TOOL时的作业点位提取
1) 设定正确的TOOL偏差值 (参照参数)
2) 固定W轴(FIX 1) 或旋转(FIX 0)中设定一个后提取作业点位

CAUTION

▶ 在有连续插补动作(例 Sealing 或 Dispensing)的作业中, 以TOOL 末端提取点位时, A(X),B(Y)轴可能有急剧的轨迹变化.
▶ 这时请注意点位的提取.

■ TOOL 制作及固定
1. TOOL能满足在"LMOV 命令"说明的W轴姿势固定
2. 使TOLL偏差在dx,dy 中, 只存在一个方向

3.34.1程序使用例题

1) 使用 TOOL 做插补移动

```
MAIN
VEL 300
TOOL 1                    使用(TOOL1) 进行插补动作
FIX 0
LMOV P1
LMOV P2
OUT0=0
FIX1
TOOL 0                    没有偏差的变更到TOOL 0
EOP
```

3.35 FIX (限制轴命令语)

功 能　在插补动作选择W轴的动作

形 式　FIX <形式>

用 语　<形式>

1 →机器人W轴姿势固定 (固定 W轴姿势)

0 → 机器人机械手姿势旋转 (W轴旋转)

说 明　1) 机器人作插补动作时，固定或旋转W轴的姿势.

2) 作PTP动作时与FIX设定无关向提取的点位移动.

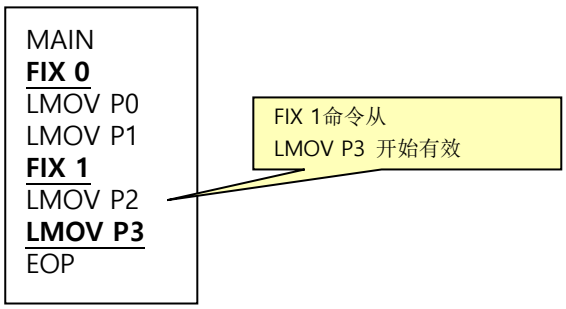
3) 在"FIX 0" 设定为姿势旋转，则旋转轴（W轴)向提取点的W轴坐标值作插补移动.

4) 在"FIX 1" 设定为姿势固定，则机器人的旋转轴(W轴)保持起始点的姿势作插补移动

5) 在"FIX 1" 设定为姿势固定，目标点的旋转轴(W轴)坐标值有可能最初提取的点位不同.

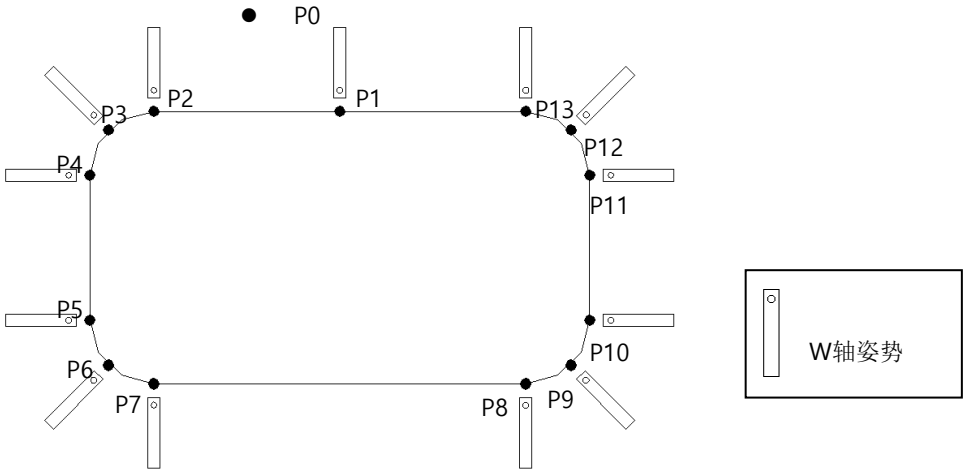
6) 如果在作业程序不使用FIX命令，则适用于设定"FIX 1"值.

7) 在作业程序要回到使用FIX以前的状态，使用"FIX 1" 从命令的第2STEP开始有效.



3.35.1程序使用例题

旋转W轴进行插补动作




```
MAIN
VEL 100
JMOV P0
FIX 0            ..............   W轴姿势旋转
LMOV P2
AMOV P3 P4
LMOV P5
AMOV P6 P7
LMOV P8
AMOV P9 P10
LMOV P11
AMOV P12 P13
FIX 1            ..............   固定 W轴姿势
LMOV P1
EOP
```

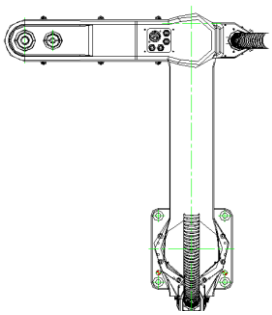
3.36 FORM (指定 ROBOT ARM形状命令语)

功 能　机器人动作时设定机器人臂形状

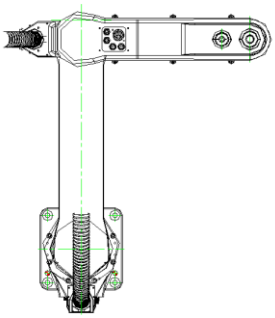
形 式　FORM <形状>

用 语　<形状> : 在水平多关节机器人移动时设定臂(A,B Arm)的形状
(LEFT, RIGHT,NO)

说 明
1) **只适用水平多关节机器人(SCARA)**
2) 设定机器人移动时的ARM的形状.
3) 形状有3种(LEFT, RIGHT, NO)



FORM RIGHT



FORM LEFT

CAUTION

- ▶ 只在 JMOV 命令语中变更机器人姿势.
- ▶ 水平多关节机器人在一个圆柱坐标系有2种.
(LEFT FORM, RIGHT FORM)
- ▶ 用 LEFT FORM 保存点位，但程序指定 "FORM RIGHT"时
忽略保存的 POINT 姿势以 RIGHT FORM 移动.
- ▶ FORM 命令语只适用于 JMOV.

3.36.1程序使用例题

1) 向提点位位置坐标移动

```
MAIN
VEL 100
JMOV P10
FORM LEFT          .....    ARM 形状选 LEFT
JMOV P11                    向 P11 移动
FORM RIGHT         .....    ARM 形状 RIGHT
JMOV P11                    向 P11 移动
EOP
```

3.37 TRQ (冲突感知命令语)

功 能　　冲突感知时扭矩限制报警

形 式　　TRQ_<指定轴>_<扭矩限制>

用 语　　<指定轴> : 1 → A(X)轴, 2 → B(Y)轴, 3 → Z轴, 4 → W轴,
<扭矩限制> : 范围 → 1~300[%] (300%表示电机额定扭矩的3倍。)

说 明　　1) 在机器人动作中指定各轴的扭矩限制，如果超过其值，超过参数里TOL设定的时间动作时发生 Torque Limit Alarm. (参照 参数模式)
2) 把扭矩限制值设定为240以上时，虽然扭矩值被限制单不发生报警.

3.37.1程序使用例题

```
MAIN
VEL 500
FOS 10
JMOV P0
WHILE 1
LMOV P1
TRQ 1 50 ..... 把A(X)轴的扭矩限制设定为50%
TRQ 2 50 ..... 把B(Y)轴的扭矩限制设定为50%
LMOV P2
LMOV P1
LMOV P3
ENDWL
EOP
```

向 P2, P1, P3 移动中收到外部冲击超出参数设定的 TOL 时间发生报警

3.38 TQL (限制轴输出扭矩 命令语)

功 能　　限制各轴最大输出扭矩值.

形 式　　TRQ <指定轴> <扭矩限制>

用 语　　<指定轴> : 1 → A(X)轴, 2 → B(Y)轴, 3 → Z轴, 4 → W轴,
<扭矩限制> : 范围 → 1~300[%] (300% 电机额定扭矩的3 倍)

说 明　　1) 通电瞬间扭矩允许额定扭矩的 3倍, 这时考虑电机或机械强度等发生问题以致于**限制最大扭矩**.
2) 扭矩不够在额定时间内不能到达目标位时发生 "In Position Error".

3.38.1程序使用例题

```
MAIN
VEL 500
FOS 10
JMOV P0
WHILE 1
LMOV P1
TQL 1 50      ..............  限制 A(X)轴的最大输出扭矩是额定的 50%
TQL 2 50      ..............  限制 B(Y)轴的最大输出扭矩是额定的 50%
LMOV P2
LMOV P1
LMOV P3
ENDWL
EOP
```

P2, P1, P3 移动时设定 TRQ的最大输出扭矩以移动.
扭矩不够时发生报警

3.39 INPOS (指定目标到达命令语)

功 能　机器人动作时表示到达目标点程度.

形 式　INPOS <指定目标点到达程度>

用 语　<指定目标点到达程度> : 设定机器人是否到达目标点的许可值.
(单位 : 脉冲量)

说 明
1) 如果使用INPOS命令语，机器人执行移动命令后所有轴没有到达许可值以内则不进行下一个命令.
2) 如果机器人的现在位置值和目标提取点的差距在INPOS命令设定的脉冲(Pulse)以内，则认为到达目标点进行下一个命令
3) INPOS命令的解除使用"INPOS 0"，从下一行开始有效.

CAUTION

▶ 使用 INPOS 命令时，正确的确定当前位置可能发生 DELAY ROBOT 移动 Cycle Time 增加.
▶ 执行 FOS 动作或 WITH 命令中 INPOS 命令不适用.

3.39.1程序使用例题

1) 使用 TOOL做差不移动

```
MAIN
VEL 100
JMOV P10
INPOS 100          .....   到达完成许可值设定 100 PULSE
JMOV P1                    向 P1,P2位置移动后，等待到各轴的
JMOV P2                    现在值和最初提取的点位差距在10Pulse以内
INPOS 0            .....   解除 INPOS 命令
JMOV P3
EOP
```

3.40 MINIT(MAPPING初始化命令语)

功能 **Mapping功能初始化命令语**

MINIT 命令语只适用于Ver 03.00.00版本或更高版本

形式 MINIT <传感器类型>

用语 <传感器类型>: 设定适用于Mapping功能的传感器类型。

设定值1: NC传感器的情况

设定值2: NO传感器的情况

说明

1) 初始化与Mapping关联的控制器内部变数。
2) 设定Mapping时使用的传感器类型。

CAUTION

- Mapping功能是以MINIT, MSTART, MREAD 3个命令组合运行一个功能。
- 为了正确的运行Mapping动作，需按顺序进行Mapping初始化(MINIT)、Mapping开始(MSTART)、Mapping读取数据(MREAD)。
  如果没有会发生"Latch Sequence Err"警报。

3.41 MSTART(MAPPING检查命令语)

功能 **Mapping开始命令语**
此命令只适用于ROSEP Robot及Ver 03.00.00版本或更高版本。

形式 MSTART <指定轴> <GP号码> <传感器输入PIN>

用语 <指定轴>：赋值给执行Mapping时的轴

<GP号码>: 赋值储存Mapping Data时的Global Point开始号码

<传感器输入PIN>：赋值使用Mapping时的传感器输入PIN号码

说明 1) 设定执行Mapping时所需要的传感器输入PIN号码，指定轴以及储存与Mapping有关联的Global Point起点数据。
2) Mapping工作中检测到的上升Edge数和下降Edge数储存到Global Point起点位置中。
3) Mapping检测最多为50个。

CAUTION

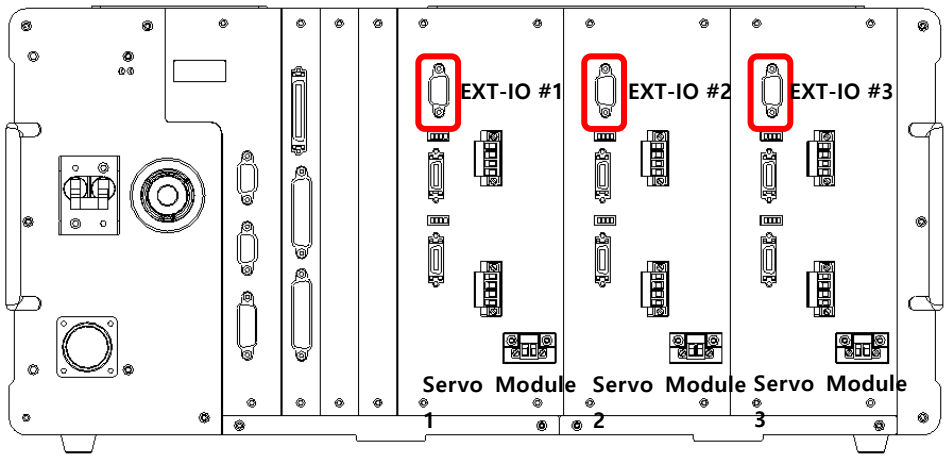
▶ 为使用Mapping功能，传感器输入PIN码和轴指定号码必须对应到同一个Servo Module。
否则无法使用Mapping功能。

CAUTION

▶ Mapping功能是由MINIT, MSTART, MREAD等3个命令组合成一起来执行一个功能。
▶ 为正确的执行Mapping动作需要按顺序执行Mapping 初始化MINIT)、Mapping 开始(MSTART)、Mapping 读取数据(MREAD)。
否则会出现"Latch Sequence Err"警报。

3.42 Mapping传感器输入端口关联

为正确使用Mapping功能，Servo Module的传感器输入号码与驱动轴相对应。
各自的Servo Module拥有两个传感器输入PIN，与下表"Servo Module传感器输入号码"相对应。



	Servo Module 1	Servo Module 2	Servo Module 3
IN 1	1	3	5
IN 2	2	4	6

[Servo Module传感器输入号码]

PIN 号码	功能
1	-
2	-
3	-
4	-
5	-
6	-
7	-
8	P24V(输出)
9	IN1(输入)
10	IN2(输入)
11	G24V(输出)
12	-
13	-
14	-
15	-

[Servo Module EXT IO PIN功能]

例) Scara机器人Z轴Mapping功能使用情况

Scara机器人Z轴的情况连接到Servo Module2，Mapping数据从GP100开始储存，传感器连接到IN1 的情况。

MSTART 3 GP100 3
Z轴 GPNT INDEX 传感器 号码

3.43 MREAD(MAPPING位置数据更新命令语)

功能 **Mapping数据更新命令语。**
此命令语只适用于ROSEP Robot及Ver 03.00.00版本或更高版本。

形式 MREAD <限制时间>

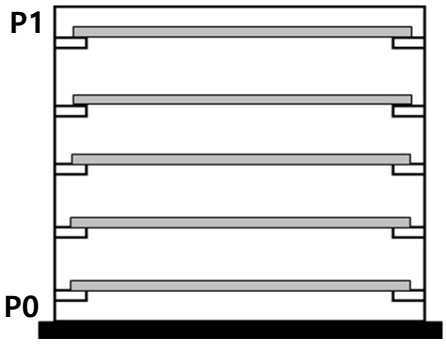
用语 <限制时间> ：读取Mapping数据时限制时间。(单位 ：ms)

说明
1) 把Mapping传感器中检测到的位置数据，按顺序储存到MSTART中设定的Global Point开始号码中。
2) 最小设定时间设定为预期检测Point个数*20ms以上。

3.43.1程序使用举例

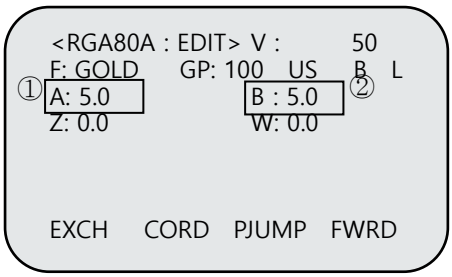
1) 测定在cassette中的玻璃厚度

```
MAIN            .....   Program开始
JMOV P0         .....   移动到待机位置P0
MINIT 1         .....   Mapping初始化
MSTART 3 GP100 3 .....  选择传感器输入PIN 3号和Z轴
                        Mapping数据从GP100开始
JMOV P1         .....   移动到P1位置
MREAD 500       .....   Mapping数据储存到GPNT
EOP                     Program终止
```



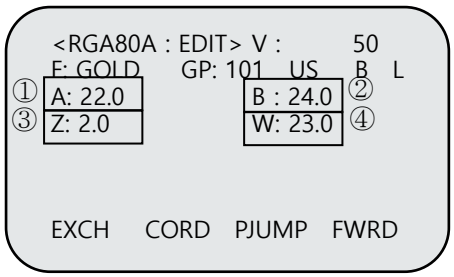
CAUTION

1) Mapping数据储存方法说明



[上升下降Edge数储存画面]

① 上升Edge数
② 下降Edge数



① 上升Edge时位置值
② 下降Edge时位置值
③ 两个位置偏差
④ 连个位置的平均位置值

在上例Cassette中有5个玻璃的情况，GP100的第一个轴传感器的上升Edge数输入为5，第二个轴传感器的下降Edge数输入为5。

当Mapping执行完后遇到MREAD命令时，储存GP101第一个轴出现的第一个上升Edge发生的位置，GP101第二个轴出现的第一个下降Edge发生的位置。

GP101第三个轴储存下降Edge发生位置和上升Edge发生位置之间的偏差，GP101第四个轴储存下降Edge发生位置和上升Edge发生位置总和的平均值。

以相应的形式储存GP101~GP105的Mapping数据。

3.44 RSTATE (确认 ROBOT状态 命令语)

功 能　　读取 ROBOT状态的命令语

形 式　　<整数型变数>=RSTATE(Channel, STATE INDEX)

说 明

INDEX	STATE NAME	内　　容
0	ALL	返还全部状态情报
1	ALARM	返还报警状态
2	READY	返还运行准备状态
3	ORIGIN	返还原点完成状态원 (ABS Type 时始终 ON)
4	INPOSITION	到达目标位置完成
5	RUN(JOB 运行中)	返还 JOB RUNNING 状态
6	SERVO ON	返还 SERVO MOTOR ON/OFF 状态
7	STOP(JOB 停止中)	返还 JOB STOP 状态
8	RUN MODE	Manual Run mode
9	SYSTEM MODE	System mode
10	TEACH MODE	Point Teaching mode
11	EMG	发生紧急停止时 ON
12	PSEL	JOB LOAD 完成.
13	Reserved	预约的领域.
14	Reserved	
15	MOVING(ROBOT运行中)	返还对应Channel Robot 移动中的状态
16	TIMEOUT	返还 String command Time out 状态
17	1 AXIS MOVING FLAG	返还对应 Channel 第一轴移动中的状态
18	2 AXIS MOVING FLAG	返还对应 Channel 第二轴移动中的状态
19	3 AXIS MOVING FLAG	返还对应 Channel 第三轴移动中的状态
20	4 AXIS MOVING FLAG	返还对应 Channel 第四轴移动中的状态
21	5 AXIS MOVING FLAG	返还对应 Channel 第五轴移动中的状态
22	6 AXIS MOVING FLAG	返还对应 Channel 第六轴移动中的状态
23	Reserved	预约的领域..
.....	.....	
31	Reserved	

3.44.1程序使用例题

1) 确认 CH1的状态分别输出 I/O

```
MAIN
INT TEMP
TEMP=RSTATE(1,0)          .....   确认 CH1的全部状态
OUT10=(TEMP>>5)&0H01      .....   SVON 状态输出
OUT9=(TEMP>>4)&0H01       .....   JOB RUN 状态输出
OUT8=TEMP&0H01            .....   Alarm 状态输出
EOP
```

2) 分别读取 CH2的状态输出 I/O

```
MAIN
OUT10= RSTATE(2,6)        .....   CH2 SVON 状态输出
OUT9= RSTATE(2,5)         .....   CH2 JOB RUN 状态输出
OUT8= RSTATE(2,1)         .....   CH2 Alarm 状态输出
EOP
```

3.45 RERROR (确认报警代码命令语)

功 能　　读取 ROBOT报警代码的命令语

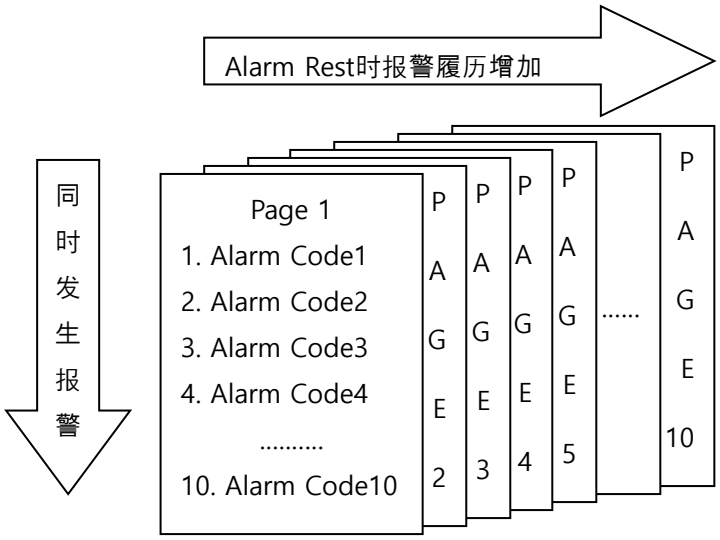
形 式　　<整数型变数>=RERROR(PAGE NO, INDEX)

说 明　　PAGE NO：报警履历编号

- Alarm History的履历编号, 1 Page能保存 10个 Index.
- Page 编号为 1是最近的报警履历, 10是最久的报警履历编号.

INDEX: ：同时发生的报警表编号

- 同时发生的 Alarm Code最多 10个.
- 没有 Alarm code时返还 0.



3.45.1程序使用例题

1) Alarm Code Read

```
MAIN
INT ERR,ERR1
WHILE 1
ERR=RERROR(1,1)        .....    确认 Page1的第一个 Alarm code
IF ERR != 0 THEN       .....    如果有 Alarm Code
ERR1=RERROR(1,2)       .....    确认 Page1的第二个 Alarm code
ENDIF
ENDWL
EOP
```

3.46 GFTOGP (把 GF 保存到GP的命令语)

功 能　把 GF(Global Float)值保存到 GP(Global Point) 变数的命令语

形 式　GFTOGP <Global Float index> <Global Point index>

用 语　Global Float Index : 0~499
Global Point Index : 0~1023

	RO(Robot)	TR(Transfer Robot)
Global FLOAT	500EA(0~499)	2000EA(0~1999)
Global Point	1024EA(0~1023)	15000EA(0~14999)

说 明　1) 指定的 GF变数开始编号和指定要保存的 GP编号.从开始编号有 6个 GF Data 保存到指定的 GP.

3.46.1程序使用例题

1) ROBOT 当前位置保存在 GP(Global Point)变数

```
MAIN
POS CURR
CURR=HERE ..... ROBOT当前位置坐标 Read
F10=CURR.1 ..... ROBOT 第一轴坐标保存在 F10
F11=CURR.2 ..... ROBOT 第二轴坐标保存在 F11
F12=CURR.3 ..... ROBOT 第三轴坐标保存在 F12
F13=CURR.4 ..... ROBOT 第四轴坐标保存在 F13
F14=CURR.5 ..... ROBOT 第五轴坐标保存在 F14
F15=CURR.6 ..... ROBOT 第六轴坐标保存在 F15
GFTOGP F10 GP50 ..... F10~F15的值保存到 GP50
EOP
```

3.47 REMCMD (SYSTEM COMMAND)

功 能 System Command

形 式 REMCMD <Channel> <Command>

说明 Channel: 执行 Command的 Channel

Command :

command	名 称	说 明
1	Servo ON	对应 Channel的 ROBOT Servo ON.
2	Servo OFF	对应 Channel的 ROBOT Servo OFF.
3	PSEL	选择 JOB程序 LOAD .
4	START	启动选择 JOB.
5	STOP	停止启动中的 JOB.
6	RESET	清零报警.
7	EMG STOP	ROBOT 急停.

3.47.1程序使用例题

1) 启动 CH1 JOB 和 启动 ROBOT Power ON

```
MAIN
INT CHECK
REMCMD 1 3 ..... CH1的 JOB文件 LOAD.
CHECK=RSTATE(1,12) ..... 确认选择 JOB LOAD 完成情报
IF CHECK==1 THEN
REMCMD 1 1 ..... CH1的 ROBOT Servo ON
CHECK=RSTATE(1,6) ..... 确认 CH1的 SERVO ON/OFF 状态
IF CHECK==1 THEN
REMCMD 1 4 ..... CH1的 JOB RUN.
ENDIF
ENDIF
EOP
```

3.48 变数

3.48.1 变数种类

变数分为一般变数,POSITION变数,系统变数

		LOCAL	GLOBAL
一般变数	整数型	INT	I(编号)
	实数型	REAL	F(编号)
POSITION 变数		POS, P(编号)	GP(编号)
系统变数		CNT,TMR,MVR,HERE	

-. LOCAL 变数 – 只适用于当前运行的程序内的变数

1) INT ：是整数型变数,用于数学计算，输入/输出Data 临时保存
2) REAL ：是实数型变数用于数学计算, Point Data 计算等
3) POS ：是Position变数, 区分为单纯变数和排列变数
4) CNT ：计数从输入PORT进来的脉冲输入后(UP COUNT)保存
5) TMR ：参数设定值在初始化的值增加1
6) MVR ：移动两个 Point 之间时, 对全体距离的移动距离的百分比
7) HERE ：保存现在轴的角度值的变
8) P(编号) ：使用者保存提点位的变数 (0 ≤ 变数 ≤ 1999)

-. GLOBAL 变数 – 全部程序及频道共同使用的变数

1) I：用整数型变数 INT 指定 LOCAL 变数同样的功能 (0 ≤ 变数 ≤ 499)
2) F：用实数型变数 REAL 指定 LOCAL 变数同样的功能 (0 ≤ 变数 ≤ 499)
3) GP：使用者保存提点位的变数 (0 ≤ 变数 ≤ 1023)

	RO(Robot)	TR(Transfer Robot)
Global INT	0 ≤ 编号 ≤ 499	0 ≤ 编号 ≤ 1999
Global FLOAT	0 ≤ 编号 ≤ 499	(0 ≤ 编号 ≤ 1999
Local Point	0 ≤ 编号 ≤ 1999	0 ≤ 编号 ≤ 1999
Global Point	0 ≤ 编号 ≤ 1023	0 ≤ 编号 ≤ 14999

3.48.2使用方法

-. 使用位置

  -. 在程序里使用的一般变数必须在程序开头"MAIN" 后宣言

  -. 子程序(Subroutine)或程序开头以外的地方不能宣言

-. 组成

  -. 一般变数的名称是有英文和数字组成的8位数

  -. 变数名的第一个字必须是英文字母或 '_'. (数字不能用)
  但, **"P", "GP", "I", "F"** 等 Point 变数及 GLOBAL 变数重复，所以不能单独使用

-. 初始化

  -. 变数是初始化以后使用.

  -. CNT, TMR 变数是使用变数之前的STEP进行初始化

  -. MVR 变数是开始 MOVE 命令则倍初始化为0

3.49 整数型(INT, I, II), 实数型 (REAL, F) 变数

3.49.1 LOCAL 变数

功 能　在程序内部指定可使用的整数型，实数型变数

形 式　INT <变数名>,<变数名>,---

REAL <变数名>,<变数名>,---

用 语　<变数名> :

-. 有英文字母和数字组成的8位数.

-. 变数名的第一个字必须是英文字母或 '_'

说 明　1) 这些变数都要在主程序的开头宣言,在子程序那不能宣言.

2) 宣言为INT的变数是整数型变数。(整数值范围: $-2.14 \times 10^{9} \sim 2.14 \times 10^{9}$)

3) 宣言为REAL的变数是实数型变数 (实数值范围: $\pm 8.4 \times 10^{-37} \sim \pm 3.4 \times 10^{38}$)

⚠ CAUTION

- **"P", "GP", "I", "F"** 等 Point 变数及 GLOBAL 变数重复，所以不能单独使用.
  (**"Syntax Error" , "Duplicated Symbol"** 等 ALARM)
- 实数型变数和整数型变数混合运算时，变更到整数型.
  9.5 + **10** = 19 **(实数和整数运算发生误差)**
  9.5 + **10.0**=19.5 **(变更实数 + 实数运算使用)**

3.49.2 程序使用例题

1) 整数型,实数型变数使用

```
MAIN
INT J,BOUT          ..............  指定整数型变数 J 和 BOUT
REAL FF1            ..............  指定实数型变数 FF1
WHILE 1
BOUT=0H000F
J=1                 ..............  初始化变数
FF=10.5
FOR J=0 TO 10       ..............  向 AP1 位置移动
JMOV P1             ..............  向 AP2 位置移动
JMOV P2
POUT0=BOUT          ..............  使用整数型变数 BOUT 输出
NEXT
FF1=FF1+100.0                       实数型变数运算
ENDWL
EOP
```

3.49.3 整数型 GLOBAL 变数 I, II

功 能　　指定 INT 整数型变数相同全部程序中共同使用的变数.

形 式　　I<整数>　　：可以使用的整数范围 0~499 .

II<整数>　　：可以使用的整数范围 0~499

→ I和 <整数>, II和 <整数> 间不能有空白 (BLANK) .

→ I和 (整数), I和 (整数变数) 间不能有空白 (BLANK) .

说 明　　1) 跟 LOCAL 变数不同在程序中不指定就可以使用.

2) II<整数> 是重复使用 **I<整数>相同** .

例) I35 = 20 时， II35 = I20 .

3) 程序中直接变更 I<整数> 或 II<整数>, I(整数), I(变数)的值.

4) VEL, ACC, DEC, FOS, PFOS, PLUP, DLAY, INPOS 中可以使用.

例) VEL I20 : I20 中保存的值设定速度 (Velocity) .

5) IF - (ELSE) - ENDIF 区间和 WHILE-ENDWL 区间可以使用.

例) IF I7 > A0 THEN　　例) WHILE II49 ≤ 100

※ **FOR-NEXT 区间不能使用.**

6) I<整数> 型是 Point 转译变数(Global Point Variable)中可以使用.

例) I9 = 3 时， PI9 = P3 .

※ **PII<整数> 形式不能使用.**

3.49.4程序使用例题

```
MAIN
INT L,M,H          ..........  指定整数型 变数 L,M,H
L=I0               ..........  L=300 分配
M=I1               ..........  M=600 分配
H=I2               ..........  H=1000 分配
VEL H              ..........  指定速度 H(1000)
JMOV P0
WHILE I10<I11      ..........  满足 I10<I11 时
FOS I4             ..........  FOS I4 => FOS 10
LMOV PI25          ..........  向 P(I25) 做 CP 移动
VEL L              ..........  指定速度 L(300)
AMOV PI26 PI27     ..........  P(I26) 和 P(I27)间经由的圆弧插补
LMOV PI28          ..........  向 P(I28) 做 CP 一定
I25=I25+10
I26=I26+10         ..........     移动 1回后 , I25,26,27值每增加 4
I27=I27+10
I10=I10+1          ..........  脉冲计数器 I10 增加 1
ENDWL
EOP
```

I 变数值
I0 = 300
I1 = 600
I2 = 1000
I10 = 1
I11 = 4
I25 = 1
I26 = 2
I27 = 3
I28 = 4

3.49.5实数型 GLOBAL 变数 F

功 能　　指定 REAL 实数型变数有相同的功能且适用于全部程序的变数.

形 式　　F<整数>　　：可以使用的整数范围 0～499 .

→ F和 <整数> 间不能有空白 (BLANK) .

说 明　　1) 跟 LOCAL 变数不同在程序中不指定就可以使用.

2) 程序中直接变更 F<整数> .

3) IF - (ELSE) - ENDIF 区间和 WHILE-ENDWL 区间可以使用.

例) IF F7 > A0 THEN　　例) WHILE F49 ≤ 100

※FOR-NEXT 区间不能使用.

CAUTION

▶ 实数型变数和整数型变数混合运算时，变更到整数型.

9.5 + **10** = 19　**(实数和整数运算发生误差)**

9.5 + **10.0**=19.5 **(变更实数 + 实数运算使用**

3.49.6程序使用例题

```
MAIN
POS XP
XP=P1
FOR I0=0 TO 10
XP1.1=F0        .....   对应位置型变数 XP1的 X轴(第一轴)的 F0值
LMOV XP         .....   向 XP的位置 CP 移动
F0=F0+10.0      .....   F0 值每增加 10.0
NEXT
EOP
```

3.50 POSITION 变数

3.50.1 POS 变数

功 能　指定位置型变数 (单纯变数及排列变数단순변수 및 배열변수)

形 式　单纯变数 → POS <变数名>, <变数名>, ...

排列变数 → POS <变数名>(大小), {<变数名>(大小)}, ...

用 语　<<变数名> :

-.变数名是由英文字母和数字组成的8位数及排列

-.变数的第一字必须是英文字母或.("P","GP","I","F" 文字单独使用除外)

(大小) : 排列变数的大小以括号内的数字(整数)表示.

指定排列变数括号内的数字表示使用个数.

例) POS AA(3) 指定时 – 可以使用的变数 AA(0), AA(1), AA(2) 3个.

说 明

1) **MAIN 开始时指定变数**, 子程序内不能指定.
2) 宣言位POS的变数值是各轴的角度，导程值，直角坐标值
   -. **X开始的变数值**(例: XA, XA1, ...) → 直角坐标值变数
   -. **X以外的字母开始的变数** → 各轴的角度，导程值, 各轴的增量变数
3) 宣言POS的变数的各成分值可以使用为<变数名>.<编号>.
   例) POS XP1 → XP1.1 XP1.2 XP1.3 XP1.4...
4) 角度值 POS 变数最多由6种成分,
   变数名(单纯变数,排列变数)=
   <1轴成分,2轴成分,3轴成分,4轴成分，5轴成分，6轴成分> 形式初始化.
5) 直角坐标值 POS 变数必须由7各成分,
   变数名 (但存变数,排列变数) =
   <X轴成分, Y轴成分, Z轴成分, 4轴成分, 5轴成分, 6轴成分, Arm Form值>
   形式初始化. → 只适用于机械部 **SCARA ROBOT**.
   -. Arm Form 值 : 0 → LEFT form, 1 → RIGHT form, 2 → NO form
6) 角度值 POS变数和直角坐标值POS变数可以互相之间进行计算及指定, 这时按保存的值的形式变换.

⚠CAUTION

- **"P", "GP", "I", "F"** 等 Point 变数及 GLOBAL 变数重复，所以不能单独使用.
  (**"Syntax Error"，"Duplicated Symbol"** 等 ALARM
- **成分值 ARMFORM 没有时，被判断角度值 XP转换到 XY 坐标值**
- 5,6轴是考虑扩张不使用的轴. 输入**0.0值**.

3.50.2 **POINT 变数**

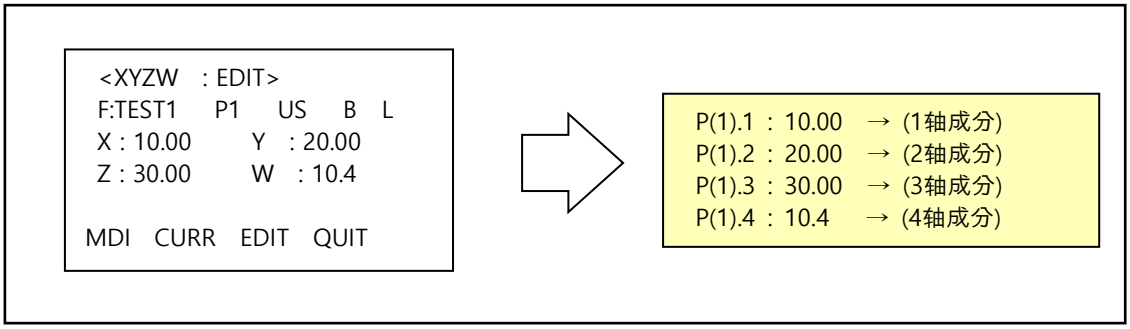
功 能　Job Editor中使用者使用 Teaching 的所有Point

形 式　LOCAL变数 → P(编号)　(0 ≤ 编号 ≤ 1999)

GLOBAL变数 → GP(编号)　(0 ≤ 编号 ≤ 1023)

说 明　1) 与POS一样最初不作变数定义.

2) Point的轴成分值可以使用为<点位编号><.轴编号>.



3.50.3程序使用例题

1) 适用 POINT变数, POSITION变数的减速点位

```
MAIN
POS MM,XA,XB                    ..... 指定位置型变数 XA , MM , XB
MM=<10.1,10.2,10.3,10.4,0,0>    ..... MM初始化
XA=<<400.0,50.0,10.0,0.0,0.0,0.0,1>  ..... XA初始化
XB=XA                           ..... 把 XB值跟 XA值一样初始化
XB.3=XA.3-40.0                  ..... 把 XB的 Z(3轴)值减去 40mm
VEL 1000
JMOV P0                         ..... 向 P0 位置 PTP 移动
LMOV XB                         ..... 向 XB 位置坐标 CP 移动 (减速点)
VEL 300
LMOV XA                         ..... 向 XA 位置坐标 CP 移动
IMOV2 MM                        ..... 位置型变数 MM值增量移动
VEL 1000
JMOV P0
EOP
```

CAUTION

▶ **因变数名，成分值也转换保存.**

XP=<角度值> :转换直角坐标值保存.

XP=<直角坐标值>: 成分值保存.

AP=<角度值>:成分值保存.

AP=<直角坐标值>:转换直角坐标值保存.

3.51 系统变数（CNT, TMR, MVR, HERE）

3.51.1 CNT, TMR 变数

功 能　CNT：保存脉冲输入计数器.

TMR：保存时间值

形 式　CNT<脉冲输入 BIT 编号>=<初始值>

TMR0=<初始值>

TMR1=<初始值>

用 语　<脉冲输入 BIT 编号>：要接受输入的输入Bit 编号

<初始值>：计数器(CNT), 计时器(TMR)开始的整数值

说 明　1) 系统变数不宣言使用.

2) CNT 变数是脉冲输入Bit决定的瞬间保存 "0"，
以后每次输入脉冲（10m 以上脉冲）时就计数. (0 ≤ 输入范围 ≤ 65,535)

3) TMR 变数是输入整数值的瞬间开始分配值，按在系统参数定义的时间间隔
增加1.
设定值适用 参照参数模式的 **TMR**.

CAUTION

▶ CNT和脉冲输入PORT编号之间 ,TMR和 "0" 或 "1" 之间不要有空白(BLANK)
.
例) CNT_0=0 (X), TMR_1= -100 (X)

3.51.2 程序使用例题

1) CNT, TMR 变数使用

```
MAIN
VEL 100
CNT0=0          ..............  初始化计数器(CNT)
TMR1=-100       ..............  初始化计时器(TMR) (0秒)
WHILE CNT0<20   ..............  等待输入BIT "0"号有20个脉冲输入LOOP
JMOV P0
IF(TMR1>0) THEN ..............  如果计时器1秒以上
GOTO TEST                      (-100 ~> 0 : 1ms 发生100个)100个=1s
ENDIF
DLAY 100
JMOV P1
ENDWL
LABL TEST
EOP
```

3.51.3MVR 变数

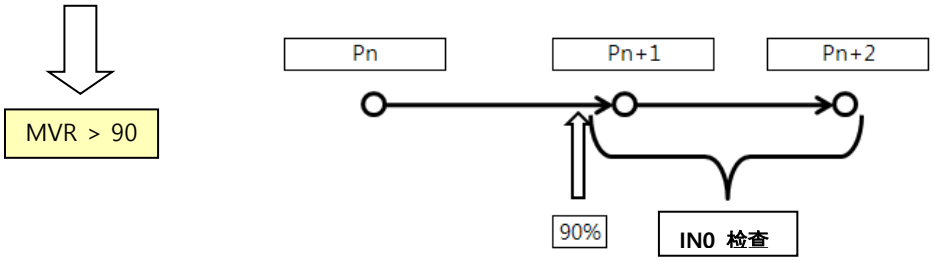
功 能　保存对全体距离的移动距离的百分比.

形 式　MVR <对全体距离的移动距离的百分比>

说 明　1) 移动 Point (Pn,Pn+1)之间时对全部距离的移动距离的百分比
即, MVR = (移动距离/全部距离) *100
2) 用于机器人动作中条件分歧及I/O并列处理
3) 把移动区间分为100单位必须与WITH命令一起使用
4) 2个 Point (Pn,Pn+1)之间的距离短，并高速运行时请注意 MVR值的设定.

CAUTION

▶ **从Pn Point向 Pn+1 Point移动时，90%以后确认输入 "IN0"时**



CAUTION

移动中有信号输入时使用读入现坐标 READ (HERE 命令语)功能

-. 机器人移动中有信号输入时，停止后现坐标读入功能随机器人移动速度有可能有偏差

-.这时请降低机器人的速度.

3.51.4程序使用例题

1) 移动中根据输入Bit状态修改移动轨迹.

```
MAIN
VEL 100
FOS 10
JMOV P100
MVR=0
WITH
JMOV P0
WHILE MVR<100          ..............  到P0 全体距离的
IF MVR>10 THEN         ..............  移动10%位置
OUT1=1 50              ..............  输出0.5秒内保持"1(ON)"
ENDIF
IN1=1
GOTO L1
ENDWL
LABL L1
STOP
ENDWT
JMOV P1
EOP
```

2) 从P1点出发过了80%以后，根据输入"IN0"的信号输入向P2点移动

```
MAIN
VEL 100
WHILE IN1==1
PLUP 10
JMOV P0
MVR=0
WITH
JMOV P1
WHILE MVR<100          ..............  从P1点出发过了80%以后
IF (MVR>80)&&(IN0==1)  ..............  检查输入IN0是否为 "1"
THEN
PLUP 0
JMOV P22
GOTO BB
ENDIF
ENDWL
ENDWT
IN0=1
PLUP 0
JMOV P2
LABL BB
ENDWT
OUT1=1 50
ENDWL
EOP
```

3.51.5HERE 变数

功 能　保存当前位置的变数

形 式　HERE

HERE<整数>

用 语　<整数>：Channel 编号 (1 或 2)

-. 适用于 Channel间 读取 ROBOT位置.

说 明　**1) 机器人的现在位置值以角度值或导程值保存的变数**

2) 保存有其他 Channel 的当前位置.

3.51.6程序使用例题

1) 利用HERE变数，READ现在位置

```
MAIN
VEL 100
POS CURR
JMOV P0
MVR=0
WITH
JMOV P1
WHILE MVR<100
IF MVR>50 THEN                    向P1移动中过了移动距离50%
CURR=HERE                ..... 把现在位置保存到位置型变数 CURR
GOTO LLL
ENDIF
ENDWL
LABL LLL
ENDWT
JMOV P2
JMOV CURR
EOP
```

2) 利用 HERE<n>变数 READ其他Channel 当前位置

```
MAIN
VEL 100
POS AP
JMOV P0
WHILE AP.1<10.0
AP=HERE2                  ..... 保存 2 Channel 当前位置
DLAY 10
ENDWL
OUT0=1
JMOV P1
EOP
```

3.52 常数

功 能 16进数, 2进数 表示.

形 式 0H<数字>

0B<数字>

用 语 <数字> : 数字前面没有0H 或0B则表示10进数

* 0H : 16进数

* 0B : 2进数

说 明 1) 0H<数字>是16进数, 数字可以是8位(32BIT).

2) 0B<数字>是2进数, 数字可以是16位(16BIT).

3) 如果没有0H或0B,则表示 10进整数或实数.

3.52.1程序使用例题

```
MAIN
INT AA
VEL 100
WHILE 1
AA=PIN0 & 0H00FF                    输入PORT0 的值跟 16进数00FF 比较
IF AA==1 THEN
POUT0=0B00110100                    输出PORT0的 2进数 00110100
JCALL MD1
ENDIF
IF AA==1 THEN
POUT0=0B01110100                    输出PORT0的 2进数 01110100
JCALL MD2
ENDIF
ENDWL
EOP
```

3.53 计算式

3.53.1分配计算式

命令语	功 能
=	计算分配计算式的右边后，把其结果按照左边变数醒变换后保存 **(例: IN0=1**)

	整数型	实数型	位置型	计数器型	计时器型
整数型	整数型	整数型	**不能**	整数型	整数型
实数型	实数型	实数型	**不能**	实数型	实数型
位置型	**不能**	**不能**	位置型	**不能**	**不能**
计数器型	整数型	整数型	**不能**	整数型	整数型
计时器型	整数型	整数型	**不能**	整数型	整数型

3.53.2 数学计算式

命令语	功 能
*,/,+,-,%	不同种类的计算结果如下表 表中 '不能'表示不可以互相计算

▶ **+, - 计算式**

	整数型	实数型	位置型	计数器型	计时器型
整数型	整数型	实数型	**不能**	整数型	整数型
实数型	实数型	实数型	**不能**	实数型	实数型
位置型	**不能**	**不能**	位置型	**不能**	**不能**
计数器型	整数型	整数型	**不能**	整数型	整数型
计时器型	整数型	整数型	**不能**	整数型	整数型

▶ *** 计算式**

	整数型	实数型	位置型	计数器型	计时器型
整数型	整数型	实数型	位置型	整数型	整数型
实数型	实数型	实数型	位置型	实数型	实数型
位置型	位置型	位置型	**不能**	位置型	位置型
计数器型	整数型	实数型	位置型	整数型	整数型
计时器型	整数型	实数型	位置型	整数型	整数型

▶ **/ 计算式**

	整数型	实数型	位置型	计数器型	计时器型
整数型	整数型	实数型	**不能**	整数型	整数型
实数型	实数型	实数型	**不能**	实数型	实数型
位置型	位置型	位置型	**不能**	位置型	位置型
计数器型	整数型	实数型	**不能**	整数型	整数型
计时器型	整数型	实数型	**不能**	整数型	整数型

▶ **% 计算式**

只能在整数之间计算，在除式去掉商数把余数作为结果值

例) AA=10%3 时 **AA的值等于 1**

3.53.3 比较计算式

命令语	功 能
>, <, ≤, ≥, ==, !=	-.主要以整数型和实数型为比较对象，在 IF 语句和 WHILE 语句的条件式使用. -.比较计算的结果是逻辑值(true,false 或 0/1)

3.53.4 逻辑计算式

命令语	功 能
&&, ||, !	逻辑计算式只把逻辑值作为对象 即把比较计算的结果作为被计算式. 逻辑计算的结果是逻辑值

3.53.5 BIT 计算式

命令语	功 能
&, | <<, >> ~	逻辑计算式 移动计算式 1 的补数

3.54 内置函数

函 数	功 能	使用例
EXP(X)	指数 e^x	EXP(3) 是 20.085
LOG(X)	常用对数 $\log_{10}X$	LOG(100) 是 2
LN(X)	自然对数 $\log_eX$	LN(15)是 2.708
SQRT(X)	平方根 $\sqrt{}$	SQRT(16)是 4
POW(x,y)	x^y	POW(2,10)是 1024
ABS(K)	整数 K 的绝对值	ABS(-12)是 12
RND(X)	四舍五入 实数 X	RND(97.62)是 98
SIN(X)	sine, x 单位 : radian	SIN(RAD(30))是 0.5
COS(X)	cosine, x 单位 : radian	COS(0)是 1
TAN(X)	tangent, x 单位 : radian	TAN(RAD(45))是 1.0
ASIN(X)	arcsine, - π/2~π/2 值返还	DEG(ASIN(0.5))是 30
ACOS(X)	arccosine, 0 ~ π 返还	DEG(ACOS(0.5))是 60
ATAN(X)	arctangent, -π/2 ~π/2 值返还	DEG(ATAN(1.0))是 45
ATAN2(Y,X)	第 2 arctangent, -π~π 值返还	DEG(ATAN2(-1,-1))是 -135
DEG(X)	把 radian 转换 angle 值	DEG(3.1416)是 180.0
RAD(X)	把 angle 转换 radian 值	RAD(180.0)是 3.1416

3.55 ASC

功 能　字符串 的第一个文字返还字符端口

形 式　整数型 变数 = ASC(字符串)

用 语　字符串变数

- 使用 DEFSTR指定的变数, 使用字符串所用.

字符串

- 表示字符串变数或字符串常数. 字符串常数是用这个符号 " "集中的集合.
- EX) "ABCDEF"

字符端口

- 对英文字的数字(整数).

  Ex) 字符 'A'是 整数(65) , 字符 'B'是 整数(66).
- 参照 ASC 端口表.

说 明　**1)** 返还在 ASC中输入的字符串的第一个字符.

3.55.1程序使用例题

```
MAIN
DEFSTR   AA
INT BB
AA = "XYZW"  ..........  对应字符串变数 AA的 "XYZW"
BB = ASC(AA)  ..........  字符串变数 AA中第一个字符 "X"返还到字符端口(88)
EOP
```

3.56 CHR

功 能　　ASC 端口范围的整数返还到字符

形 式　　字符串变数 = CHR(整数)

用 语　　字符串变数

- 使用 DEFSTR指定的变数, 使用字符串所用.

整数

- ASC 端口值内的整数(0 ~ 127)

说 明　　**1)** ASC中输入的整数返还到对应的 ASC 端口.

3.56.1程序使用例题

```
MAIN
DEFSTR   AA
AA = CHR(65)          .....   字符串变数 AA的 ASC端口 65所对应的 "A"
EOP
```

3.57 FLUSH

功 能　输入,，输出，缓冲清零

形 式　FLUSH(整数)

用语　PORT: 选择COM Port(1~2)

- 1: 选择RS-232 Port
- 2: 选择RS-485 Port
- 3: 清除输入、输出缓冲区

整数: 选择清除缓冲区(1~3)

- 1: 清除输入缓冲区
- 2: 清除输出缓冲区
- 3: 清除输入、输出缓冲区

说 明　**1)** Serial通信传送接收使用的输入，输出，缓冲清零..

3.57.1程序使用例题

```
MAIN
DEFSTR  AA
AA = STRIN(1000)        ..........   以 Serial 通信接收输入，保存到字符串 AA
FLUSH(1)                ..........   输入，缓冲清零
EOP
```

3.58 FTOS

功 能　把整数或实数返还字符串

形 式　字符串变数 = FTOS(整数 or 实数)

用 语　字符串变数

- 使用 DEFSTR指定的变数, 使用字符串所用.

整数 or 实数

- 返还字符串的整数或实数

说 明　**1)** 整数或实数值的数据转换到字符串.

EX) FTOS(1234) → "1234"

3.58.1程序使用例题

```
MAIN
DEFSTR   AA,BB
AA = FTOS(1234)        ..........   把 1234转换字符串"1234"后， 带入字符串变数 AA
BB = FTOS(-123.456)    ..........   把-123.456转换字符串"-123.456"后，带入字符串变数 BB
EOP
```

3.59 HTOS

功 能　把整数转换 16进数的字符串

形 式　字符串变数= HTOS(整数)

用 语　字符串变数

- 使用 DEFSTR指定的变数, 使用字符串所用.

整数

- 转换到 16进数字符串的整数

说 明　**1)** HTOS中输入的整数转换到 16进数的字符串.

3.59.1程序使用例题

```
MAIN
DEFSTR  AA
AA = HTOS(10)          .....    把整数 10转换到 16进数字符串("A")带入到字符串 AA
EOP
```

3.60 SLEFT

功 能　　输出输入字符串的左侧部分

形 式　　字符串 变数 = SLEFT(字符串, 整数)

用 语　　字符串变数

- 使用 DEFSTR指定的变数, 使用字符串所用.

字符串

- 表示字符串变数或字符串常数.
- 字符串常数是用这个符号 ”“集中的集合.

整数

- 输出字符数

说 明　　**1)** 在输入字符串的左侧开始输出输入整数

3.60.1程序使用例题

```
MAIN
DEFSTR   AA,BB
AA = “XYZW’          ..................   在字符串变数 AA带入 “XYZW”
BB = SLEFT(AA,2)     ..................   字符串变数 AA的左侧开始转换 2字符(“XY”)
EOP
```

3.61 SLEN

功 能　　返还输入的字符串长度

形 式　　整数型变数=SLEN(字符串)

用 语　　字符串

- 表示字符串变数或字符串常数. 字符串常数是用这个符号 " "集中的集合.
- EX) "ABCDEF"

说 明　　**1)** 返还输入的字符串长度.

3.61.1程序使用例题

```
MAIN
DEFSTR  AA
INT LL
AA = "XYZW"          ..........    在字符串变数 AA带入 "XYZW"
LL = SLEN(AA)        ..........    把字符串变数 AA的长度 (4)带入到整数型变数 LL
EOP
```

3.62 SMID

功 能　在字符串中指定的位置开始输出指定的字符串

形 式　字符串 变数 = SMID(字符串,指定位置, 整数)

用 语　字符串变数

- 使用 DEFSTR指定的变数, 使用字符串所用.

字符串

- 表示字符串变数或字符串常数. 字符串常数是用这个符号 " "集中的集合.

指定位置

- 是字符串输出的开始位置, 字符串的开始是 0.

整数

- 从指定位置读取的字符数

说 明　**1)** 在指定位置输出使用者规定的输入整数.

3.62.1程序使用例题

```
MAIN
DEFSTR   AA, BB
AA = "ABCDEFG"        .....    在字符串变数 AA带入 "ABCDEFG"
BB = SMID(AA,2,3)     .....    在字符串变数 AA的 第2位置开始转换字符 3个 ("CDE")
EOP
```

3.63 SPOS

功 能　　字符串1中返还字符串2 配合的开始位置

形 式　　整数型变数 = SPOS(字符串1, 字符串2)

用 语　　字符串

- 表示字符串变数或字符串常数. 字符串常数是用这个符号 " "集中的集合.
- EX) "ABCDEF"

说 明　　**1)** 字符串1中返还字符串2 配合的开始位置.

3.63.1程序使用例题

```
MAIN
DEFSTR   AA, BB
INT PP
AA = "XYZW"          ..........   在字符串变数 AA代入 "XYZW"
BB = "Z"             ..........   在字符串变数 BB代入 "Z"
PP = SPOS(AA, BB)    ..........   在字符串变数 AA中返还跟 BB一致的位置
EOP
```

3.64 SRIGHT

功 能　　输出输入字符串右侧部分

形 式　　字符串 变数 = SRIGHT(字符串, 整数)

用 语　　字符串变数

- 使用 DEFSTR指定的变数, 使用字符串所用.

字符串

- 表示字符串变数或字符串常数. 字符串常数是用这个符号 " "集中的集合..

整数

- 输出的字符数

说 明　　**1)** 在输入字符串的右侧开始输出输入整数

3.64.1程序使用例题

```
MAIN
DEFSTR   AA, BB
AA = "ABCDEF"          ..........    在字符串变数 AA代入 "ABCDEF"
BB = SRIGHT(AA,2)      ..........    在字符串变数 AA的右侧开始返还 2个字符("EF")
EOP
```

3.65 STRIN

功 能　读取通过系列通信进入的字符串

形 式　字符串 变数 = STRIN(整数)

用 语　字符串变量

使用DEFSTR来指明声明的变量，为了区分字符串使用。

PORT: 选择COM Port(1~2)

- 1: 选择RS-232 Port
- 2: 选择RS-485 Port

整数

- Time Out时间.(单位: ms)

根据Time Out时间等待串行输入，如果Time Out时间期间数据无法进入，系统区检查Time Out状态，然后运行step命令。

说 明　**1)** 等待使用者是定的时间并且读取通过系列通信的输入.

3.65.1程序使用例题

```
MAIN
DEFSTR   PACKET
INT LEN, OVERTM
PACKET = STRIN(1000)        ..............  通过系列通信的字符串输入(没有输入时等待 1s)
LEN = SLEN(PACKET)          ..............  保存输入字符串的长度
SS1=SYS1
OVERTM=(SS1>12)             ..............  读取是否 TIMEOUT
&0H01
IF OVERTM==0&&LEN>0         ..............  不是 TIMEOUT, 输入字符串存在时
THEN
.............
ENDIF
EOP
```

3.66 STROUT

功 能　　通过系列通信输出字符串

形 式　　整数型变数 = STROUT(字符串)

用 语　　字符串

- 表示字符串变数或字符串常数. 字符串常数是用这个符号 " "集中的集合..

PORT: 选择COM Port(1~2)

- 1: 选择RS-232 Port
- 2: 选择RS-485 Port

说 明　　**1)** 被输入字符串通过系列通信输出,返还没有传送的字符数

EX) "ABCDEFGH"的字符传送成功时返还值为 0，没有传送成功时返还值为 8

3.66.1程序使用例题

```
MAIN
DEFSTR   SENDPK
INT STRRET
SENDPK="ABCDEF"          .....   SENDPK 字符串变数中代入 "ABCDEF"
STRRET=                  .....   系列通信 PORT输出字符串("ABCDEF")
STROUT(SENDPK)
IF STRRET !=0 THEN       .....    确认传送失败
.....
ENDIF
EOP
```

3.67 SVAL

功 能　　把字符串转换数字

形 式　　(整数型变数 or 实数型变数) = SVAL(字符串)

用 语　　字符串

- 表示字符串变数或字符串常数. 字符串常数是用这个符号 " "集中的集合..

说 明　　**1)** 输入字符串转换到整数或实数数据.

3.67.1程序使用例题

```
MAIN
INT AI
REAL BR
AI=SVAL("1234")          ..........  把字符串 "1234"转换成 整数 1234
BR=SVAL("12.234")        ..........  把字符串 "12.234"转换成 实数 12.234
EOP
```

Rev.	修正日期	内容	修正者	S/W Version
V.1	2012.04.09	初版 印刷		
V.2	2013.05.30	添加到 CMD	Kimjs	Ver. 03.02.10

N1 ROBOT CONTROLLER

CONTROLLER MANUAL

FIRST EDITION JULY 2011

ROBOSTAR CO, LTD

ROBOT R&D CENTER