DENSO机械手

编程手册 II

PAC 共用程序库（Library）

前言

承蒙惠购 DENSO 机械手，深表铭谢。

该产品是汇聚了本公司先进技术的高速度，高精度，高功能的 " 组装用机械手 "。

在使用之前，请详细阅读理解本说明书，以便安全高效地使用本机。

本书说明的对象产品

■配置 RC7 型控制器

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

■选件品

- 视觉设备　μVision 系列

注 1：机械手控制器的版本记载于粘贴在控制器表面的 " 控制器设定表 " 的软件 Ver. 栏中。
此外，还可通过多功能教导器的［主画面］-［F6 设定］-［F6 维护］-［F2 版本］所显示的 ROM 版本栏进行确认。

要求

在使用本产品之前，请务必阅读 " 安全注意事项 "，以便于正确安全地使用 DENSO 机械手。

本书的构成

本书的构成如下所示。

指令一览表

按字母顺序排列的指令一览表

按功能单位排列的指令一览表

PAC 共用程序库 (Library)

关于 WINCAPS Ⅲ上标准配置的共用程序库 (Library) 进行说明。

目录

按照字母顺序排列的指令一览表

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

指令	功能	4 轴	6 轴	视觉装置	说明页
A					
ArchMove	实施弧形运行。	V1.9			7-49
ArchMoveV	实施弧形运行。		V3.2		7-50
arrange_button_pos	指定画面上的按钮配置 （位置、大小等）。	V1.7	V1.7		8-12
arrange_button_size	指定画面上的按钮配置 （位置、大小等）。	V1.7	V1.7		8-12
aspACLD	变更内部负荷条件值。以前端负荷质量 (g)、负荷重心位置 (mm) （注 1）指定所有负荷条件值。	◎	◎		3-1
aspChange	选择最佳搬运质量设定模式的内部模式。	◎	◎		3-2
C					
ClearCollisionForce	将外力最大值初始化。		V2.7		7-65
ClearSrvMonitor	初始化伺服单轴数据监视的获取数据的指针 (Pointer)。（对应 Ver. 1.5 以上版本）	V1.5	V1.5		7-55
D					
dioConstantDistanceIoOut	当轴处于一定角度 （Z 轴、直动轴等则是距离）动作时，反转 I/O 的输出状态。	V3.0	V3.0		6-2
dioSync	与被 DIO 连接的外部设备 （序列器 (sequencer) 等）的同步	◎	◎		6-1
G					
GetCollisionForce	获得外力最大值。		V2.7		7-64
M					
make_LABEL	创建标题 （标签）。	V1.7	V1.7		8-10
make_LED	创建 LED 按钮。	V1.7	V1.7		8-9
make_PARAM_BOX	创建变量按钮 （输入＆显示 BOX）。	V1.7	V1.7		8-10
make_PB	创建 PB 按钮。	V1.7	V1.7		8-9
MotionComp	判断执行动作命令是否完成了。（对应 Ver. 1.5 以上版本）	V1.5	V1.5		7-48
MotionSkip	中断执行中的运行命令。（对应 Ver. 1.5 以上版本）	V1.5	V1.5		7-47
mvResetPulseWidth	将停止时的允许脉冲宽度设为默认值。	◎	◎		7-1
mvResetPulseWidthJnt	将指定的附加轴的停止时允许脉冲宽度设定回默认值。	V1.5	V1.5		7-2
mvResetTimeOut	将动作结束超时值设为默认值。	◎	◎		7-3
mvReverseFlip	4 轴形态反转	◎	◎		7-3
mvSetPulseWidth	设定停止时允许脉冲宽度	◎	◎		7-4
mvSetPulseWidthJnt	设定指定的附加轴的停止时允许脉冲宽度。	V1.5	V1.5		7-5
mvSetTimeOut	设定动作结束超时值	◎	◎		7-6
N					
ndApra	工具坐标系指定的绝对动作 （4 轴机械手专用）	◎			3-14
ndDepa	工具坐标系指定的绝对动作 （4 轴机械手专用）	◎			3-15
ndInb	将指定端口的输入视为 2 进制数并转换为 10 进制数	◎	◎		3-3
ndJf	从外部设备接收 OK / NG，条件分支 （RS232C 输入输出）	◎	◎		3-4
ndOnb	将 10 进制数转换为 2 进制数后通过指定端口输出	◎	◎		3-5
ndOnbI	将 10 进制数转换为 2 进制数后通过指定端口输出	◎	◎		3-6
ndTc	TC 时间设定	◎	V1.2		3-16

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

指令	功能	4 轴	6 轴	视觉装置	说明页
ndTs	设定 TS 时间、低速度	◎	V1.2		3-16
ndVcom	与外部设备的通信动作 （内核）。（RS232C 输入输出）	◎	◎		3-13
ndVdt	存储从外部设备转发来的变量 （RS232C 输入输出）	◎	◎		3-7
ndVis	向外部设备转发指定的 2 位的整数 （RS232C 输入输出）	◎	◎		3-8
ndVput	向外部设备转发位置姿势 （RS232C 输入输出）	◎	◎		3-9
ndVrst	外部设备的初始化 （RS232C 输入输出）	◎	◎		3-10
ndVset	从外部设备接收数据 （RS232C 输入输出）	◎	◎		3-11
ndVType	指定与外部设备的通信协议。（RS232C 输入输出）	◎	◎		3-12
O					
OffPWM	解除指定轴的 PWM 切换控制。（4 轴机械手专用）	◎			7-19
OffSrvLock	解除指定轴的伺服锁定。（4 轴机械手专用）	◎			7-17
OnPWM	将指定的轴进行 PWM 切换控制。（4 轴机械手专用）	◎			7-18
OnSrvLock	将指定轴置为伺服锁定状态。（4 轴机械手专用）	◎			7-16
P					
pltDecCnt	码垛堆积总计数器的递减	◎	◎		4-1
pltGetCnt	码垛堆积总计数器的获取	◎	◎		4-1
pltGetK	码垛堆积设定值 K 的获取	◎	◎		4-2
pltGetK1	码垛堆积计数器 K1 的获取	◎	◎		4-2
pltGetM	码垛堆积设定值 M 的获取	◎	◎		4-3
pltGetM1	码垛堆积计数器 M1 的获取	◎	◎		4-3
pltGetN	码垛堆积设定值 N 的获取	◎	◎		4-4
pltGetN1	码垛堆积计数器 N1 的获取	◎	◎		4-4
pltGetNextPos	下一位置的获取	◎	◎		4-5
pltGetPLT1END	码垛堆积 1 段结束标志 (Flag) 的获取	◎	◎		4-5
pltGetPLTEND	码垛堆积全段结束标志 (Flag) 的获取	◎	◎		4-6
pltIncCnt	码垛堆积总计数器的递加	◎	◎		4-6
pltInit1	码垛堆积初始化模板 1	◎	◎		4-7
pltInitialize	码垛堆积初始化	◎	◎		4-8
pltKernel	码垛堆积动作 （内核）	◎	◎		4-9
pltLetCnt	码垛堆积总计数器的设定	◎	◎		4-10
pltLetK1	码垛堆积计数器 K1 的设定	◎	◎		4-10
pltLetM1	码垛堆积计数器 M1 的设定	◎	◎		4-11
pltLetN1	码垛堆积计数器 N1 的设定	◎	◎		4-11
pltMain1	码垛堆积 模板 1	◎	◎		4-12
pltMain2	码垛堆积 模板 2	◎	◎		4-12
pltMove	标准码垛堆积 模板 1	◎	◎		4-13
pltMove0	标准码垛堆积动作 1	◎	◎		4-14
pltResetAll	所有码垛堆积计数器的清零	◎	◎		4-15
pltResetPLT1END	码垛堆积 1 段结束标志 (Flag) 的清零	◎	◎		4-15
pltResetPLTEND	码垛堆积全段结束标志 (Flag) 的清零	◎	◎		4-16
R					
ResetCollisionJnt	使指定轴的碰撞检测无效。		V2.7		7-63

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

指令	功能	4 轴	6 轴	视觉装置	说明页
ResetCompControl	将力限制功能设为无效。（6 轴专用命令）		V1.4		7-29
ResetCompEralw	将力限制时的工具端的位置、姿势偏差允许值初始化。（6 轴专用命令）		V1.4		7-42
ResetCompJLimit	将力限制时的电流限制值初始化。（力限制特殊功能共用程序库 (Library)）（6 轴专用命令）		V1.4		7-38
ResetCompRate	将力限制时的柔度的比例初始化。（6 轴专用命令）		V1.4		7-34
ResetCompVMode	将力限制时的速度控制模式设为无效。（力限制特殊功能共用程序库 (Library)）（6 轴专用命令）		V1.4		7-40
ResetCurLmt	解除指定轴的电机电流限制。	◎	V1.2		7-13
Resetcycloid	从摆线动作模式转移到通常模式。	◎	V1.4		7-22
ResetCycloidJnt	解除摆线动作模式，转移到通常模式。	V1.5	V1.5		7-23
ResetDampRate	将力限制时的粘性比例初始化。（6 轴专用命令）		V1.4		7-44
ResetEralw	将指定轴的偏差允许值返回到默认值。	◎	V1.2		7-15
ResetFrcAssist	将力限制时的补偿力初始化。（力限制特殊功能共用程序库 (Library)）（6 轴专用命令）		V1.4		7-36
ResetFrcLimit	将力限制比例初始化。（6 轴专用命令）		V1.4		7-32
ResetGravity	将重力平衡设为无效。		V1.2		7-8
ResetGrvOffset	将重力补偿值的修正设为无效。		V1.2		7-10
ResetHighPathAccuracy	使提高 CP 动作 （直线、圆弧、自由曲线）时的动作轨迹的高轨迹控制功能无效。	V2.61	V2.61		7-60
ResetVibControl	从降低残留振动控制模式返回到通常控制模式。	V1.4	V1.4		7-46
S					
set_button_param	指定按钮属性 （类型、颜色、形状等）。	V1.7	V1.7		8-11
SetArchParam	弧形运行的在上升动作中开始横向动作的位置 （弧形起始位置）及在下降动作中结束横向动作的位置 （弧形完成位置）的设定	V1.9			7-51
SetCollisionJnt	使指定轴的碰撞检测有效。		V2.7		7-62
SetCollisionLevel	设定指定轴的碰撞检测等级。		V2.7		7-66
SetCompControl	将力限制功能设为有效。（6 轴专用命令）		V1.4		7-27
SetCompEralw	设定力限制时的工具端的位置、姿势偏差允许值。（6 轴专用命令）		V1.4		7-41
SetCompFControl	将力限制功能设为有效。（6 轴专用命令）		V1.4		7-28
SetCompJLimit	设定力限制时的电流限制值。（力限制特殊功能共用程序库 (Library)）（6 轴专用命令）		V1.4		7-37
SetCompRate	设定力限制时的柔度的比例。（6 轴专用命令）		V1.4		7-33
SetCompVMode	设定力限制时的速度控制模式。（力限制特殊功能共用程序库 (Library)）（6 轴专用命令）		V1.4		7-39
SetCPSpdMode	变更 CP 动作时的 TCP 速度设定。	V1.8	V1.8		7-24
SetCurLmt	限制指定轴的电机电流值。	○	V1.2		7-11
Setcycloid	转移至抑制 PTP 动作结束动作时的过冲量及残留振动的摆线动作模式。	◎	V1.4		7-20
SetCycloidJnt	转移至抑制附加轴结束动作时的超程量及残留振动的摆线动作模式。	V1.5	V1.5		7-21
SetDampRate	设定力限制时的粘性比例。（6 轴专用命令）		V1.4		7-43
SetEralw	变更指定轴的偏差允许值。	◎	V1.2		7-14
SetExtForceDetect	设定外力检测的有效或无效。		V2.7		7-67

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

指令	功能	4 轴	6 轴	视觉装置	说明页
SetForce_HC	是通过 HC 机械手指定 Z 轴推力 （单位 : N）的电流限制共用程序库 (Library)。	◎			7-26
SetForce_HM	HM/HS 机械手指定 Z 轴的推力 （单位 : N）的电流限制共用程序库 (Library)。	◎			7-25
SetFrcAssist	设定力限制时的补偿力。（力限制特殊功能共用程序库 (Library)）（6 轴专用命令）		V1.4		7-35
SetFrcCoord	选择力限制设定坐标系。（6 轴专用命令）		V1.4		7-30
SetFrcLimit	设定力限制比例。（6 轴专用命令）		V1.4		7-31
SetGravity	修正各个关节的静负荷 （重力转矩），设定重力平衡。		V1.2		7-7
SetGrvOffset	通过各个关节的重力转矩修正重力补偿值。		V1.2		7-9
SetHighPathAccuracy	向提高 CP 动作 （直线、圆弧、自由曲线）时的动作轨迹的高轨迹控制功能过渡。	V2.61	V2.61		7-59
SetMonitorCond	设定伺服单轴数据监视功能的监视条件。（对应 Ver. 1.5 以上版本）	V1.5	V1.5		7-52
SetSingularAvoid	使特异点回避功能有效或无效。（6 轴专用）		V2.61		7-61
SetVibControl	置为降低残留振动控制模式。	V1.4	V1.4		7-45
single_button_set	仅创建一个按钮。	V1.7	V1.7		8-11
StartSrvMonitor	开始伺服单轴数据监视。（对应 Ver. 1.5 以上版本）	V1.5	V1.5		7-53
StopSrvMonitor	结束伺服单轴数据监视。（对应 Ver. 1.5 以上版本）	V1.5	V1.5		7-54
T					
tolChange	工具更换	◎	◎		5-1
tolInit1	工具更换初始化模板 1	◎	◎		5-1
tolInitialize	工具更换初始化	◎	◎		5-2
tolKernel	工具更换动作 （内核）	◎	◎		5-2
tolMain1	工具更换 模板 1	◎	◎		5-3
V					
viTran6	将视觉坐标转换为机械手坐标 （6 轴）	◎	◎	◎	9-1
viTran6S	将视觉坐标转换为机械手坐标 （6 轴）				9-2
X					
xdSPLClrTakeArm	变更在 TakeArm 时执行的自由曲线的通过点删除处理的有效、无效。	V2.3	V2.3		7-58
xdSPLPASSNUM	获取自由曲线已经通过的通过点编号。	V2.3	V2.3		7-57
xdWAITSPLINE	等待通过自由曲线所指定的通过点。	V2.3	V2.3		7-56

以功能为单位的指令一览表

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

功能分类	指令	功能	4 轴	6 轴	视觉装置	说明页
以前语言的共用程序库 (Library)	aspACLD	变更内部负荷条件值。以前端负荷质量 (g)、负荷重心位置 (mm) （注 1）指定所有负荷条件值。	◎	◎		3-1
	aspChange	选择最佳搬运质量设定模式的内部模式。	◎	◎		3-2
	ndInb	将指定端口的输入视为 2 进制数并转换为 10 进制数	◎	◎		3-3
	ndJf	从外部设备接收 OK / NG，条件分支（RS232C 输入输出）	◎	◎		3-4
	ndOnb	将 10 进制数转换为 2 进制数后通过指定端口输出	◎	◎		3-5
	ndOnbI	将 10 进制数转换为 2 进制数后通过指定端口输出	◎	◎		3-6
	ndVdt	存储从外部设备转发来的变量（RS232C 输入输出）	◎	◎		3-7
	ndVis	向外部设备转发指定的 2 位的整数（RS232C 输入输出）	◎	◎		3-8
	ndVput	向外部设备转发位置姿势 （RS232C 输入输出）	◎	◎		3-9
	ndVrst	外部设备的初始化 （RS232C 输入输出）	◎	◎		3-10
	ndVset	从外部设备接收数据 （RS232C 输入输出）	◎	◎		3-11
	ndVType	指定与外部设备的通信协议。（RS232C 输入输出）	◎	◎		3-12
	ndVcom	与外部设备的通信动作 （内核）。（RS232C 输入输出）	◎	◎		3-13
	ndApra	工具坐标系指定的绝对动作 （4 轴机械手专用）	◎			3-14
	ndDepa	工具坐标系指定的绝对动作 （4 轴机械手专用）	◎			3-15
	ndTc	TC 时间设定	◎	V1.2		3-16
	ndTs	设定 TS 时间、低速度	◎	V1.2		3-16
码垛堆积的共用程序库 (Library)	pltDecCnt	码垛堆积总计数器的递减	◎	◎		4-1
	pltGetCnt	码垛堆积总计数器的获取	◎	◎		4-1
	pltGetK	码垛堆积设定值 K 的获取	◎	◎		4-2
	pltGetK1	码垛堆积计数器 K1 的获取	◎	◎		4-2
	pltGetM	码垛堆积设定值 M 的获取	◎	◎		4-3
	pltGetM1	码垛堆积计数器 M1 的获取	◎	◎		4-3
	pltGetN	码垛堆积设定值 N 的获取	◎	◎		4-4
	pltGetN1	码垛堆积计数器 N1 的获取	◎	◎		4-4
	pltGetNextPos	下一位置的获取	◎	◎		4-5
	pltGetPLT1END	码垛堆积 1 段结束标志 (Flag) 的获取	◎	◎		4-5
	pltGetPLTEND	码垛堆积全段结束标志 (Flag) 的获取	◎	◎		4-6
	pltIncCnt	码垛堆积总计数器的递加	◎	◎		4-6
	pltInit1	码垛堆积初始化模板 1	◎	◎		4-7

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

功能分类	指令	功能	4 轴	6 轴	视觉装置	说明页
	pltInitialize	码垛堆积初始化	◎	◎		4-8
	pltKernel	码垛堆积动作 （内核）	◎	◎		4-9
	pltLetCnt	码垛堆积总计数器的设定	◎	◎		4-10
	pltLetK1	码垛堆积计数器 K1 的设定	◎	◎		4-10
	pltLetM1	码垛堆积计数器 M1 的设定	◎	◎		4-11
	pltLetN1	码垛堆积计数器 N1 的设定	◎	◎		4-11
	pltMain1	码垛堆积 模板 1	◎	◎		4-12
	pltMain2	码垛堆积 模板 2	◎	◎		4-12
	pltMove	标准码垛堆积 模板 1	◎	◎		4-13
	pltMove0	标准码垛堆积动作 1	◎	◎		4-14
	pltResetAll	所有码垛堆积计数器的清零	◎	◎		4-15
	pltResetPLT1END	码垛堆积 1 段结束标志 (Flag) 的清零	◎	◎		4-15
	pltResetPLTEND	码垛堆积全段结束标志 (Flag) 的清零	◎	◎		4-16
工具操作的共用程序库 (Library)	tolChange	工具更换	◎	◎		5-1
	tolInit1	工具更换初始化模板 1	◎	◎		5-1
	tolInitialize	工具更换初始化	◎	◎		5-2
	tolKernel	工具更换动作 （内核）	◎	◎		5-2
	tolMain1	工具更换 模板 1	◎	◎		5-3
输入输出的共用程序库 (Library)	dioSync	与被 DIO 连接的外部设备 （序列器 (sequencer) 等）的同步	◎	◎		6-1
	dioConstantDistanceIoOut	当轴处于一定角度 （Z 轴、直动轴等则是距离）动作时，反转 I/O 的输出状态。	V3.0	V3.0		6-2
臂动作的共用程序库 (Library)	mvResetPulseWidth	将停止时的允许脉冲宽度设为默认值。	◎	◎		7-1
	mvResetPulseWidthJnt	将指定的附加轴的停止时允许脉冲宽度设定回默认值。	V1.5	V1.5		7-2
	mvResetTimeOut	将动作结束超时值设为默认值。	◎	◎		7-3
	mvReverseFlip	4 轴形态反转	◎	◎		7-3
	mvSetPulseWidth	设定停止时允许脉冲宽度	◎	◎		7-4
	mvSetPulseWidthJnt	设定指定的附加轴的停止时允许脉冲宽度。	V1.5	V1.5		7-5
	mvSetTimeOut	设定动作结束超时值	◎	◎		7-6
	SetGravity	修正各个关节的静负荷 （重力转矩），设定重力平衡。		V1.2		7-7
	ResetGravity	将重力平衡设为无效。		V1.2		7-8
	SetGrvOffset	通过各个关节的重力转矩修正重力补偿值。		V1.2		7-9
	ResetGrvOffset	将重力补偿值的修正设为无效。		V1.2		7-10
	SetCurLmt	限制指定轴的电机电流值。	○	V1.2		7-11
	ResetCurLmt	解除指定轴的电机电流限制。	◎	V1.2		7-13
	SetEralw	变更指定轴的偏差允许值。	◎	V1.2		7-14
	ResetEralw	将指定轴的偏差允许值返回到默认值。	◎	V1.2		7-15
	OnSrvLock	将指定轴置为伺服锁定状态。（4 轴机械手专用）	◎			7-16

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

功能分类	指令	功能	4 轴	6 轴	视觉装置	说明页
	OffSrvLock	解除指定轴的伺服锁定。（4 轴机械手专用）	◎			7-17
	OnPWM	将指定的轴进行 PWM 切换控制。（4 轴机械手专用）	◎			7-18
	OffPWM	解除指定轴的 PWM 切换控制。（4 轴机械手专用）	◎			7-19
	Setcycloid	转移至抑制 PTP 动作结束动作时的过冲量及残留振动的摆线动作模式。	◎	V1.4		7-20
	SetCycloidJnt	转移至抑制附加轴结束动作时的超程量及残留振动的摆线动作模式。	V1.5	V1.5		7-21
	Resetcycloid	从摆线动作模式转移到通常模式。	◎	V1.4		7-22
	ResetCycloidJnt	解除摆线动作模式，转移到通常模式。	V1.5	V1.5		7-23
	SetCPSpdMode	变更 CP 动作时的 TCP 速度设定。	V1.8	V1.8		7-24
	SetForce_HM	HM/HS 机械手指定 Z 轴的推力 （单位 : N）的电流限制共用程序库 (Library)。	◎			7-25
	SetForce_HC	是通过 HC 机械手指定 Z 轴推力（单位 : N）的电流限制共用程序库 (Library)。	◎			7-26
	SetCompControl	将力限制功能设为有效。（6 轴专用命令）		V1.4		7-27
	SetCompFControl	将力限制功能设为有效。（6 轴专用命令）		V1.4		7-28
	ResetCompControl	将力限制功能设为无效。（6 轴专用命令）		V1.4		7-29
	SetFrcCoord	选择力限制设定坐标系。（6 轴专用命令）		V1.4		7-30
	SetFrcLimit	设定力限制比例。（6 轴专用命令）		V1.4		7-31
	ResetFrcLimit	将力限制比例初始化。（6 轴专用命令）		V1.4		7-32
	SetCompRate	设定力限制时的柔度的比例。（6 轴专用命令）		V1.4		7-33
	ResetCompRate	将力限制时的柔度的比例初始化。（6 轴专用命令）		V1.4		7-34
	SetFrcAssist	设定力限制时的补偿力。（力限制特殊功能共用程序库 (Library)）（6 轴专用命令）		V1.4		7-35
	ResetFrcAssist	将力限制时的补偿力初始化。（力限制特殊功能共用程序库 (Library)）（6 轴专用命令）		V1.4		7-36
	SetCompJLimit	设定力限制时的电流限制值。（力限制特殊功能共用程序库 (Library)）（6 轴专用命令）		V1.4		7-37
	ResetCompJLimit	将力限制时的电流限制值初始化。（力限制特殊功能共用程序库 (Library)）（6 轴专用命令）		V1.4		7-38
	SetCompVMode	设定力限制时的速度控制模式。（力限制特殊功能共用程序库 (Library)）（6 轴专用命令）		V1.4		7-39

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

功能分类	指令	功能	4 轴	6 轴	视觉装置	说明页
	ResetCompVMode	将力限制时的速度控制模式设为无效。（力限制特殊功能共用程序库(Library)）（6 轴专用命令）		V1.4		7-40
	SetCompEralw	设定力限制时的工具端的位置、姿势偏差允许值。（6 轴专用命令）		V1.4		7-41
	ResetCompEralw	将力限制时的工具端的位置、姿势偏差允许值初始化。（6 轴专用命令）		V1.4		7-42
	SetDampRate	设定力限制时的粘性比例。（6 轴专用命令）		V1.4		7-43
	ResetDampRate	将力限制时的粘性比例初始化。（6 轴专用命令）		V1.4		7-44
	SetVibControl	置为降低残留振动控制模式。	V1.4	V1.4		7-45
	ResetVibControl	从降低残留振动控制模式返回到通常控制模式。	V1.4	V1.4		7-46
	MotionSkip	中断执行中的运行命令。（对应 Ver. 1.5 以上版本）	V1.5	V1.5		7-47
	MotionComp	判断执行动作命令是否完成了。（对应 Ver. 1.5 以上版本）	V1.5	V1.5		7-48
	ArchMove	实施弧形运行。	V1.9			7-49
	ArchMoveV	实施弧形运行。		V3.2		7-50
	SetArchParam	弧形运行的在上升动作中开始横向动作的位置 （弧形起始位置）及在下降动作中结束横向动作的位置 （弧形完成位置）的设定	V1.9			7-51
	SetMonitorCond	设定伺服单轴数据监视功能的监视条件。（对应 Ver. 1.5 以上版本）	V1.5	V1.5		7-52
	StartSrvMonitor	开始伺服单轴数据监视。（对应 Ver. 1.5 以上版本）	V1.5	V1.5		7-53
	StopSrvMonitor	结束伺服单轴数据监视。（对应 Ver. 1.5 以上版本）	V1.5	V1.5		7-54
	ClearSrvMonitor	初始化伺服单轴数据监视的获取数据的指针 (Pointer)。（对应 Ver. 1.5 以上版本）	V1.5	V1.5		7-55
	xdWAITSPLINE	等待通过自由曲线所指定的通过点。	V2.3	V2.3		7-56
	xdSPLPASSNUM	获取自由曲线已经通过的通过点编号。	V2.3	V2.3		7-57
	xdSPLClrTakeArm	变更在 TakeArm 时执行的自由曲线的通过点删除处理的有效、无效。	V2.3	V2.3		7-58
	SetHighPathAccuracy	向提高 CP 动作 （直线、圆弧、自由曲线）时的动作轨迹的高轨迹控制功能过渡。	V2.61	V2.61		7-59
	ResetHighPathAccuracy	使提高 CP 动作 （直线、圆弧、自由曲线）时的动作轨迹的高轨迹控制功能无效。	V2.61	V2.61		7-60
	SetSingularAvoid	使特异点回避功能有效或无效。（6 轴专用）		V2.61		7-61
	SetCollisionJnt	使指定轴的碰撞检测有效。		V2.7		7-62
	ResetCollisionJnt	使指定轴的碰撞检测无效。		V2.7		7-63
	GetCollisionForce	获得外力最大值。		V2.7		7-64
	ClearCollisionForce	将外力最大值初始化。		V2.7		7-65
	SetCollisionLevel	设定指定轴的碰撞检测等级。		V2.7		7-66

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

功能分类	指令	功能	4 轴	6 轴	视觉装置	说明页
	SetExtForceDetect	设定外力检测的有效或无效。		V2.7		7-67
用于创建操作盘画面的共用程序库 (Library)	make_PB	创建 PB 按钮。	V1.7	V1.7		8-9
	make_LED	创建 LED 按钮。	V1.7	V1.7		8-9
	make_LABEL	创建标题 （标签）。	V1.7	V1.7		8-10
	make_PARAM_BOX	创建变量按钮 （输入＆显示 BOX）。	V1.7	V1.7		8-10
	single_button_set	仅创建一个按钮。	V1.7	V1.7		8-11
	set_button_param	指定按钮属性 （类型、颜色、形状等）。	V1.7	V1.7		8-11
	arrange_button_size	指定画面上的按钮配置 （位置、大小等）。	V1.7	V1.7		8-12
	arrange_button_pos	指定画面上的按钮配置 （位置、大小等）。	V1.7	V1.7		8-12
视觉的共用程序库 (Library)	viTran6	将视觉坐标转换为机械手坐标 （6 轴）	◎	◎	◎	9-1
	viTran6S	将视觉坐标转换为机械手坐标 （6 轴）				9-2

第 1 章　共用程序库 (Library) 的使用方法

要使用共用程序库 (Library) 时，需要启动 WINCAPS Ⅲ的程序库 (Program Bank)，并追加必要的共用程序库 (Library)。

关于程序库 (Program Bank) 的操作方法，请参照 WINCAPS Ⅲ指南。

第 2 章　共用程序库 (Library) 的分类

在 WINCAPS Ⅲ的共用程序库 (Library) 中，按照如下表所示的分类提供了标准程序库 (Program Bank)。

分类 （类别名称）	内容
!DW00 : 原有语言	提供与以前语言的指令同等的功能。
!DW01 : 码垛堆积	提供码垛堆积功能。
!DW02 : 工具操作	提供工具操作相关的功能。
!DW03 : 输出入	提供 DIO、 RS232C 输出入相关的功能。
!DW04 : Arm_ 臂 _ 依从	提供臂动作相关的依从功能。
!DW05 : Arm_ 臂 _ 传送带跟踪	提供臂动作相关的传送带跟踪功能。
!DW06 : Arm_ 臂 _ 碰撞检测	提供臂动作相关的碰撞检测功能。
!DW07 : Arm_ 臂 _ 自由曲线	提供臂动作相关的自由曲线功能。
!DW08 : Arm_ 臂 _ 功能 On/Off	提供臂动作其他功能的 ON/OFF 相关的程序。
!DW09 : Arm_ 臂 _ 其他	提供臂动作相关的其他功能。
!DW10 : 视觉	提供视觉动作相关的功能。
!DW11 : 其他	提供与旧版相对应的功能等。

第 3 章　以前语言的共用程序库 (Library)

是为了提供与装有 RC3 控制器的机械手所使用的以前语言的指令同等功能的共用程序库 (Library)。

aspACLD（共用程序库 (Library)）

功　能

变更内部负荷条件值。以前端负荷质量 (g)、负荷重心位置 (mm)（注 1）指定所有负荷条件值。

格　式

aspACLD（< 前端负荷重量 >、< 前端负荷重心位置 X 坐标 >、< 前端负荷重心位置 Y 坐标 >、< 前端负荷重心位置 Z 坐标 >）（注 1）

注 1: 4 轴机械手的 Ver.1.9 以上版本的情况下

aspACLD（< 前端负荷质量 >、< 前端负荷重心位置 X 坐标 >、< 前端负荷重心位置 Y 坐标 >、< 前端负荷惯量 >）

说　明

前端负荷质量是机械手的第 6 轴所承载的负荷 （工具 + 工件）的质量。单位以 (g) 指定。

负荷重心位置在工具 0 坐标系上指定负荷的重心位置。单位是 (mm)。

负荷惯量是以负荷的重心位置为原点的，绕 Z 轴的惯量。单位是 $kgcm^2$。

用 6 轴机械手的示例进行说明就是，工具 0 坐标系的原点是 6 轴法兰中心， Y 成分为从法兰中心开始至 Φ5H7 (Φ6H7) 销钉方向（定向矢量方向），Z 成分是通过法兰中心与法兰面垂直的方向（近似矢量方向），X 成分是以定向矢量为 Y 轴、近似矢量为 Z 轴时的右手坐标系中的 X 轴方向 （常规矢量方向）。请参照编程手册Ⅰ的 "4.7 ' 使用条件 ' 中的最佳可搬运质量设定功能 "。

前端负荷质量、负荷重心位置 X、负荷重心位置 Y、负荷重心位置 Z （负荷惯量） 4 个值中即使只变更 1 个值时， 4 个值也请全部记述。

负荷条件值的切换需要约 0.1 秒的执行时间。如果频繁地切换负荷条件，则会成为导致动作时间延迟的主要原因。此外，在通过 (Pass) 动作过程中切换负荷条件值时，会变为结束动作，所以请注意不要在障碍物附近的通过 (Pass) 动作过程中变更模式。

宏定义

必须有 <pacman.h> 文件。

相关项目

编程手册Ⅰ的 "4.7 ' 使用条件 ' 中的最佳可搬运质量设定功能 "

注意事项

- 前端负荷质量请用每台机械手上所设定的范围内的整数进行指定。如果指定其以外的数值，则会发生 "60d2 前端负荷设定值超过允许值 "。
- 前端负荷重心位置的输入请满足每台机械手所指定的范围。如果不满足范围，则会发生 "60d2 前端负荷设定值超过允许值 "。
- 内部负荷条件值对于外部负荷条件值请设定在以下范围。如果不满足范围，则会发生 "60d2 前端负荷设定值超过允许值 "。
  0.5 × 外部负荷条件值 ≦ 内部负荷条件值 ≦ 外部负荷条件值

应用示例

```
CALL aspACLD(8500,-50,100,80)        ' 将内部前端负荷条件值设定为前端负荷质量 8500 (g)、
                                     ' 负荷重 ' 心位置 X 成分 -50 (mm)、 Y 成分 100
                                     '(mm)、 Z 成分 80 (mm)。
```

aspChange（共用程序库 (Library)）

功　　能

选择最佳搬运质量设定模式的内部模式。

格　　式

aspChange （< 模式 >）

说　　明

切换最佳可搬运质量设定模式。

< 设定值 >

0 → 无效
1 → 仅限 PTP 有效
2 → 仅限 CP 有效
3 → PTP、 CP 均有效

在模式切换时，需要约 0.1 秒的执行时间。如果频繁地切换模式，则会成为导致动作延迟的主要原因。此外，在通过 (Pass) 动作中切换模式时，会变为结束动作。请注意在障碍物附近的通过 (Pass) 动作中不要变更模式。

宏定义

必须有 <pacman.h> 文件。

相关项目

编程手册 I 的 "4.7 ' 使用条件 ' 中的最佳可搬运质量设定功能 "

注意事项

请用 0 ～ 3 的整数指定 < 模式 >。如果指定其以外的数值，则会发生 "6003 超过了有效的数值范围 "。

应用示例

```
CALL aspChange(1)                    ' 将最佳搬运质量设定模式的内部模式置为 1。
```

ndInb（共用程序库 (Library)）

功　能

将指定端口的输入视为 2 进制数并转换为 10 进制数

格　式

ndInb （< 整数值变量编号 >、< 最低位输入端口编号 >、< 最高位输入端口编号 >）

说　明

提供与以前语言的 INB 命令同等的功能。

读取指定的输入端口的信号的状态，并将其视为 2 进制数转换为 10 进制数。

转换的值赋值整数变量。

相关项目

ndOnb、 ndOnbI

备　注

在 PAC 语言中同等功能可以用 DEFIO 和 IN 命令进行记述。

那种方法比该共用程序库 (Library) 高效，所以请尝试使用 DEFIO 和 IN 命令。

输入端口请置为连续的 16 个端口以内。当指定为 16 个端口以上时，不被处理。

应用示例

```
CALL ndInb(1,552,567)          ' 将输入端口 552 ～ 567 视为 16 比特的 2 进制数
                               ' 转换为 10 进制数，并将结果赋值Ⅰ[1]。
```

ndJf（共用程序库 (Library)）

功　能

从外部设备接收 OK / NG，条件分支 （RS232C 输入输出）

格　式

ndJf （<2 位的整数 >、 < 判断自变量 (argument)>）

说　明

提供与以前语言的 JF 命令同等的功能。

将被指定的 2 位的整数转发至外部设备后，判断来自外部设备的应答结果，并在程序中进行条件分支。

如果来自外部设备的应答为 OK 则进入下一步骤，为 NG 时则进入标签位置。

相关项目

ndVcom、 ndVType

应用示例

```
#include <pacman.h>
PROGRAM PRO1
DEFINT JF_VAL = 0
  FLUSH
  CALL ndVType(1)                ' 设定通信协议排列顺序 （旧 = 0 / 新 = 1）
  CALL  ndVrst                   ' 外部设备的初始化
  CALL  ndVset(0)                ' 将接收数据 (VDT) 清空为 0
  CALL  ndVis(3)                 ' 启动外部设备 （传送 03）
  CALL  ndJf(3、 JF_VAL)         ' 获取外部设备的应答结果 （传送 03）
  IF JF_VAL = TRUE THEN          ' 如果应答为 OK (TRUE) 则接收数据
    CALL   ndVset(3)             ' 从外部设备接收 10 个的数据 （传送 03）
    CALL   ndVdt(pacPOS、 1)     ' 将从外部设备接收的数据赋值变量 (P1)
    CALL   ndVdt(pacJNT、 1)     ' 将从外部设备接收的数据赋值变量 (J1)
    CALL   ndVdt(pacTRN、 1)     ' 将从外部设备接收的数据赋值变量 (T1)
  END IF
END
```

ndOnb（共用程序库 (Library)）

功　　能

将 10 进制数转换为 2 进制数后通过指定端口输出

格　　式

ndOnb （< 整数值 >、 < 最低位输出端口编号 >、 < 最高位输出端口编号 >）

说　　明

提供与以前语言的 ONB 命令同等的功能。

将整数转换为 2 进制数，并从指定的端口输出。

输出Ⅰ型变量值的形式的 ONB 命令使用 ndOnb Ⅰ共用程序库 (Library)。

相关项目

ndInb、 ndOnbI

备　　注

输出端口请置为连续的 16 个端口以内。当指定了超出 16 个端口时、低位端口 > 高位端口时、指定了不能输出的端口时，不被处理。

应用示例

```
CALL ndOnb(15,104,119)          ' 将输出端口 104 ～ 119 视为 16 比特的 2 进制数
                                ' 转换并输出为 10 进制数 15。
```

ndOnbI（共用程序库 (Library)）

功　能

将 10 进制数转换为 2 进制数后通过指定端口输出

格　式

ndOnbI （< 整数值变量编号 >、 < 最低位输出端口编号 >、 < 最高位输出端口编号 >）

说　明

提供与以前语言的 ONB 命令同等的功能。

将整数变量值转换为 2 进制数，并从指定的端口输出。

输出整数值的形式的 ONB 命令使用 ndOnb 共用程序库 (Library)。

相关项目

ndInb、 ndOnb

备　注

输出端口请置为连续的 16 个端口以内。当指定了超出 16 个端口时、低位端口 > 高位端口时、指定了不能输出的端口时，不被处理。

应用示例

```
CALL ndOnbI(1,104,119）          ' 将输出端口 104 ～ 119 视为 16 比特的 2 进制数 ,
                                 ' 转换为 I[1] 的值并输出。
```

ndVdt（共用程序库 (Library)）

功　　能

存储从外部设备转发来的变量 （RS232C 输入输出）

格　　式

ndVdt （< 存放变量型 >、 < 存放变量编号 >）

说　　明

提供与以前语言的 VDT 命令同等的功能。

将从外部设备转发来的数据赋值被指定的存放变量型 (P / J / T) 的存放变量编号。

宏定义

需要 <pacman.h>。

相关项目

ndVcom、 ndVset

应用示例

```
#include <pacman.h>
PROGRAM PRO1
DEFINT JF_VAL = 0
  FLUSH
  CALL ndVType(1)                ' 设定通信协议排列顺序 （旧 = 0 ／ 新 = 1）
  CALL  ndVrst                   ' 外部设备的初始化
  CALL  ndVset(0)                ' 将接收数据 (VDT) 清空为 0
  CALL  ndVis(3)                 ' 启动外部设备 （传送 03）
  CALL  ndJf(3、 JF_VAL)         ' 获取外部设备的应答结果 （传送 03）
  IF JF_VAL = TRUE THEN          ' 如果应答为 OK (TRUE) 则接收数据
    CALL   ndVset(3)             ' 从外部设备接收 10 个的数据 （传送 03）
    CALL   ndVdt(pacPOS、 1)     ' 将从外部设备接收的数据赋值变量 (P1)
    CALL   ndVdt(pacJNT、 1)     ' 将从外部设备接收的数据赋值变量 (J1)
    CALL   ndVdt(pacTRN、 1)     ' 将从外部设备接收的数据赋值变量 (T1)
  END IF
END
```

ndVis（共用程序库 (Library)）

功　能

向外部设备转发指定的 2 位的整数 （RS232C 输入输出）

格　式

ndVis （<2 位的整数 >）

说　明

提供与以前语言的 VIS 命令同等的功能。

确认了外部设备的准备状态之后，自机械手向外部设备转发指定的 2 位的整数。

相关项目

ndVcom、 ndVType

应用示例

```
#include <pacman.h>
PROGRAM PRO1
DEFINT JF_VAL = 0
  FLUSH
  CALL ndVType(1)              ' 设定通信协议排列顺序 （旧 = 0 / 新 = 1）
  CALL  ndVrst                 ' 外部设备的初始化
  CALL  ndVset(0)              ' 将接收数据 (VDT) 清空为 0
  CALL  ndVis(3)               ' 启动外部设备 （传送 03）
  CALL  ndJf(3、 JF_VAL)       ' 获取外部设备的应答结果 （传送 03）
  IF JF_VAL = TRUE THEN        ' 如果应答为 OK (TRUE) 则接收数据
    CALL   ndVset(3)           ' 从外部设备接收 10 个的数据 （传送 03）
    CALL   ndVdt(pacPOS、 1)   ' 将从外部设备接收的数据赋值变量 (P1)
    CALL   ndVdt(pacJNT、 1)   ' 将从外部设备接收的数据赋值变量 (J1)
    CALL   ndVdt(pacTRN、 1)   ' 将从外部设备接收的数据赋值变量 (T1)
  END IF
END
```

ndVput（共用程序库 (Library)）

功　能

向外部设备转发位置姿势 （RS232C 输入输出）

格　式

ndVput （< 存放变量型 >、 < 存放变量编号 >）

说　明

提供与以前语言的 VPUT 命令同等的功能。

< 位置变量编号 > 为负数时 :

向外部设备转发机械手的当前位置坐标 （正在进行工具定义时，工具坐标系的原点位置坐标）和姿势或位置变量的内容。

< 位置变量编号 > 为正数时 :

向外部设备转发通过 < 存放变量型 > 和 < 存放变量编号 > 被指定的位置变量的内容。

宏定义

需要 <pacman.h>。

相关项目

ndVcom、 ndVType

应用示例

```
#include <pacman.h>
PROGRAM PRO65
  FLUSH
  CALL dVType(1)                    ' 设定通信协议排列顺序 （旧 = 0 / 新 = 1）
  CALL ndVrst                       ' 外部设备的初始化
  CALL ndVis(4)                     ' 启动外部设备 （传送 04）
  P1 = (1,2,3,4,5,6,7)
  J1 = (11,12,13,14,15,16)
  T1 = (21,22,23,24,25,26,27,28,29,30)
  CALL ndVput(pacPOS,1)             ' 将变量 "P1" 的数据传送至外部设备
  CALL ndVput(pacJNT,1)             ' 将变量 "J1" 的数据传送至外部设备
  CALL ndVput(pacTRN,1)             ' 将变量 "T1" 的数据传送至外部设备
END
```

ndVrst（共用程序库 (Library)）

功　　能

外部设备的初始化 （RS232C 输入输出）

格　　式

ndVrst

说　　明

提供与以前语言的 VRST 命令同等的功能。
对外部设备指示初始化。

相关项目

ndVcom、 ndVType

应用示例

```
#include <pacman.h>
PROGRAM PRO1
DEFINT JF_VAL = 0
  FLUSH
  CALL ndVType(1)              ' 设定通信协议排列顺序 （旧 = 0 / 新 = 1）
  CALL  ndVrst                 ' 外部设备的初始化
  CALL  ndVset(0)              ' 将接收数据 (VDT) 清空为 0
  CALL  ndVis(3)               ' 启动外部设备 （传送 03）
  CALL  ndJf(3、 JF_VAL)       ' 获取外部设备的应答结果 （传送 03）
  IF JF_VAL = TRUE THEN        ' 如果应答为 OK (TRUE) 则接收数据
    CALL   ndVset(3)           ' 从外部设备接收 10 个的数据 （传送 03）
    CALL   ndVdt(pacPOS、 1)   ' 将从外部设备接收的数据赋值变量 (P1)
    CALL   ndVdt(pacJNT、 1)   ' 将从外部设备接收的数据赋值变量 (J1)
    CALL   ndVdt(pacTRN、 1)   ' 将从外部设备接收的数据赋值变量 (T1)
  END IF
END
```

ndVset（共用程序库 (Library)）

功　能

从外部设备接收数据 （RS232C 输入输出）

格　式

ndVset （<2 位的整数 >）

说　明

提供与以前语言的 VSET 命令同等的功能。

将指定的 2 位的整数自机械手向外部设备转发之后，从外部设备接收数据，并向内部变量进行加算 （赋值）。如果将 <2 位的整数 > 指定为 "0"，则内部变量以 "0" 进行初始化。

相关项目

ndVcom、 ndVType

应用示例

```
#include <pacman.h>
PROGRAM PRO1
DEFINT JF_VAL = 0
  FLUSH
  CALL ndVType(1)                ' 设定通信协议排列顺序 （旧 = 0 / 新 = 1）
  CALL  ndVrst                   ' 外部设备的初始化
  CALL  ndVset(0)                ' 将接收数据 (VDT) 清空为 0
  CALL  ndVis(3)                 ' 启动外部设备 （传送 03）
  CALL  ndJf(3、 JF_VAL)         ' 获取外部设备的应答结果 （传送 03）
  IF JF_VAL = TRUE THEN          ' 如果应答为 OK (TRUE) 则接收数据
    CALL   ndVset(3)             ' 从外部设备接收 10 个的数据 （传送 03）
    CALL   ndVdt(pacPOS、 1)     ' 将从外部设备接收的数据赋值变量 (P1)
    CALL   ndVdt(pacJNT、 1)     ' 将从外部设备接收的数据赋值变量 (J1)
    CALL   ndVdt(pacTRN、 1)     ' 将从外部设备接收的数据赋值变量 (T1)
  END IF
END
```

ndVType（共用程序库 (Library)）

功　能

指定与外部设备的通信协议。（RS232C 输入输出）

格　式

ndVType （<2 位的整数 >）

说　明

指定 ndVis、ndJf、ndVset、ndVrst、ndVput、ndVcom 的通信协议排列顺序。各个共用程序库 (Library) 的初始值被设定为新通信协议排列顺序。

（旧通信协议排列顺序 = 0 ／新通信协议排列顺序 = 1）

相关项目

ndVis、 ndJf、 ndVset、 ndVrst、 ndVput、 ndVcom

应用示例

```
#include <pacman.h>
PROGRAM PRO1
DEFINT JF_VAL = 0
  FLUSH
  CALL ndVType(1)                  ’设定通信协议排列顺序 （旧 = 0 ／新 = 1）
  CALL  ndVrst                     ’外部设备的初始化
  CALL  ndVset(0)                  ’将接收数据 (VDT) 清空为 0
  CALL  ndVis(3)                   ’启动外部设备 （传送 03）
  CALL  ndJf(3、 JF_VAL)           ’获取外部设备的应答结果 （传送 03）
  IF JF_VAL = TRUE THEN            ’如果应答为 OK (TRUE) 则接收数据
    CALL   ndVset(3)               ’从外部设备接收 10 个的数据 （传送 03）
    CALL   ndVdt(pacPOS、 1)       ’将从外部设备接收的数据赋值变量 (P1)
    CALL   ndVdt(pacJNT、 1)       ’将从外部设备接收的数据赋值变量 (J1)
    CALL   ndVdt(pacTRN、 1)       ’将从外部设备接收的数据赋值变量 (T1)
  END IF
END
```

ndVcom（共用程序库 (Library)）

功　能

与外部设备的通信动作 （内核）。（RS232C 输入输出）

格　式

ndVcom （< 功能编码 >、 < 排列变量 >）

说　明

是与外部设备进行通信动作的核心程序。在使用通信共用程序库 (Library) (ndVis、 ndJf、 ndVset、 ndVrst、 ndVput、 ndVType) 时，需要该共用程序库 (Library)。

控制器的软件版本为 Ver.1.2* 之前版本时，请使用 Ver.1.2 互换的类别中的版本。如果使用以前语言的类别中的版本，则会发生编译异常。

控制器软件的版本确认方法请参照操作指南 "5.7［设定（主）］视窗的显示、各个模块的版本信息的显示、［F6 设定］—［F6 维护］—［F2 版本］ "。

注意

该程序不设想用多个任务来执行。

如果是用多个任务来执行，则需要利用信号 (semaphore) 等变更为取得同步。

相关项目

ndVis、 ndJf、 ndVset、 ndVrst、 ndVput、 ndVdt、 ndVType

ndApra（共用程序库 (Library)）

功　能

工具坐标系指定的绝对动作 （4 轴机械手专用）

格　式

ndApra （< 基准位置 >、 <Z 坐标 >、 < 停止精度 >）

说　明

提供与以前语言的 APRA 命令同等的功能。

仅指定基准位置的 Z 坐标，通过 PTP 控制进行移动。根据停止精度的指定，有如下所示的不同。

0: 结束动作
1: 通过 (Pass) 动作
2: 确认根据编码器值向目标位置的到达

想要使用与以前语言的 APRT 命令同等的功能时，参照该共用程序库 (Library) 通过并进偏差计算 （参照程序 1 的 p.7-19 "7.9.7 位置计算 "）计算位置型的值，并请使用 MOVE 命令。

相关项目

ndDepa

应用示例

```
PROGRAM PRO1
  TAKEARM
  CALL  ndApra((P0),400.0,1)        'P0 的 Z 坐标移动至 400 的坐标
  MOVE P,@0 P0                      ' 向 P0 移动
  CALL  ndDepa(400.0,0)             'P0 的 Z 坐标移动至 400 的坐标
  GIVEARM
END
```

ndDepa（共用程序库 (Library)）

功　　能

工具坐标系指定的绝对动作 （4 轴机械手专用）

格　　式

ndDepa （<Z 坐标 >、< 停止精度 >）

说　　明

提供与以前语言的 DEPA 命令同等的功能。

从当前位置仅指定 Z 坐标，通过 PTP 控制进行移动。根据停止精度的不同，有如下所示的不同。

0: 结束动作
1: 通过 (Pass) 动作
2: 确认根据编码器值向目标位置的到达

想要使用与以前语言的 DRET 命令同等的功能时，参照该共用程序库 (Library) 通过 并进偏差计算（参照程序 1 的 p.7 - 19 "7.9.7 位置计算 "）计算位置型的值，并请使用 MOVE 命令。

相关项目

ndApra

应用示例

```
PROGRAM PRO1
  TAKEARM
  CALL  ndApra((P0),400.0,1)        'P0 的 Z 坐标移动至 400 的坐标
  MOVE P,@0 P0                      ' 向 P0 移动
  CALL  ndDepa(400.0,0)             'P0 的 Z 坐标移动至 400 的坐标
  GIVEARM
END
```

ndTc（共用程序库 (Library)）【Ver.1.2 以上版本】

功　能

TC 时间设定

格　式

ndTc （<TC 时间 >）

说　明

提供与以前语言的 TC 指令同等的功能。
设定 TC 时间。
可以在 0 秒～ 600 秒之间进行设定。出厂时被设定为 60 秒。
如果设定为 0 秒，则提供与以前语言的 TC OFF 同等的功能。

宏定义

必须有 <pacman.h> 文件。

相关项目

ndTs

ndTs（共用程序库 (Library)）【Ver.1.2 以上版本】

功　能

设定 TS 时间、低速度

格　式

ndTS （<TS 时间 >、 < 低速度 >）

说　明

提供与以前语言的 TS 命令同等的功能。
设定 TS 时间、低速度。
可以在 3 秒～ 30 秒之间设定 TS 时间。出厂时被设定为 5 秒。
可以在 1% ～ 10% 之间设定低速度。出厂时被设定为 10%。

宏定义

必须有 <pacman.h> 文件。

相关项目

ndTc

第 4 章　码垛堆积的共用程序库 (Library)

对码垛堆积功能的共用程序库 (Library) 进行如下说明。

pltDecCnt（共用程序库 (Library)）

功　能

码垛堆积总计数器的递减

格　式

pltDecCnt （< 码垛堆积编号 >）

说　明

将被指定的码垛堆积编号的码垛堆积总计数器从当前值加 -1。
通过操作码垛堆积计数器，可以做成锯齿型及拔牙型的码垛堆积。

相关项目

pltGetCnt、 pltIncCnt、 pltKernel、 pltLetCnt

应用示例

```
CALL pltDecCnt(1)                    ’将码垛堆积编号为 1 号的总计数器减 1。
```

pltGetCnt（共用程序库 (Library)）

功　能

码垛堆积总计数器的获取

格　式

pltGetCnt （< 码垛堆积编号 >、 <Cnt 保存整数变量编号 >）

说　明

获取被指定的码垛堆积编号的码垛堆积总计数器。

相关项目

pltDecCnt、 pltIncCnt、 pltKernel、 pltLetCnt

应用示例

```
CALL pltGetCnt(1、 0)                ’将当前的计数器赋值Ｉ [0]。
CALL pltLetCnt(1、 I[0] - 1)         ’将计数器退 1。
```

pltGetK（共用程序库 (Library)）

功　能

码垛堆积设定值 K 的获取

格　式

pltGetK （< 码垛堆积编号 >、 <K 保存整数变量编号 >）

说　明

获取被指定的码垛堆积编号的设定值 K。

相关项目

pltGetK1、 pltLetK1、 pltKernel

应用示例

```
CALL pltGetK(1,10)          ' 将码垛堆积编号为 1 号的码垛堆积设定值 K
                            ' 赋值Ⅰ [10]。
```

pltGetK1（共用程序库 (Library)）

功　能

码垛堆积计数器 K1 的获取

格　式

pltGetK1 （< 码垛堆积编号 >、 <K1 保存整数变量编号 >）

说　明

获取被指定的码垛堆积编号的计数器 K1。

相关项目

pltGetK、 pltLetK1、 pltKernel

应用示例

```
CALL pltGetK1(1,10)         ' 将码垛堆积编号为 1 号的码垛堆积计数器 K1
                            ' 赋值Ⅰ [10]。
```

pltGetM（共用程序库 (Library)）

功　能

码垛堆积设定值 M 的获取

格　式

pltGetM （< 码垛堆积编号 >、 <M 保存整数变量编号 >）

说　明

获取被指定的码垛堆积编号的设定值 M。

相关项目

pltGetM1、 pltLetM1、 pltKernel

应用示例

```
CALL pltGetM(1,10)          ' 将码垛堆积编号为 1 号的码垛堆积设定值 M
                            ' 赋值Ⅰ [10]。
```

pltGetM1（共用程序库 (Library)）

功　能

码垛堆积计数器 M1 的获取

格　式

pltGetM1 （< 码垛堆积编号 >、 <M1 保存整数变量编号 >）

说　明

获取被指定的码垛堆积编号的计数器 M1。

相关项目

pltGetM、 pltLetM1、 pltKernel

应用示例

```
CALL pltGetM1(1,10)         ' 将码垛堆积编号为 1 号的码垛堆积计数器
                            ' 赋值Ⅰ [10]。
```

pltGetN（共用程序库 (Library)）

功　能

码垛堆积设定值 N 的获取

格　式

pltGetN （< 码垛堆积编号 >、 <N 保存整数变量编号 >）

说　明

获取被指定的码垛堆积编号的设定值 N。

相关项目

pltGetN1、 pltLetN1、 pltKernel

应用示例

```
CALL pltGetN(1,10)          ' 将码垛堆积编号为 1 号的码垛堆积设定值 N
                            ' 赋值Ⅰ [10]。
```

pltGetN1（共用程序库 (Library)）

功　能

码垛堆积计数器 N1 的获取

格　式

pltGetN1 （< 码垛堆积编号 >、 <N1 保存整数变量编号 >）

说　明

获取被指定的码垛堆积编号的计数器 N1。

相关项目

pltGetN、 pltLetN1、 pltKernel

应用示例

```
CALL pltGetN1(1,10)         ' 将码垛堆积编号为 1 号的码垛堆积计数器 N1
                            ' 赋值Ⅰ [10]。
```

pltGetNextPos（共用程序库 (Library)）

功　能

下一位置的获取

格　式

pltGetNextPos （< 码垛堆积编号 >、 < 下一位置的获取位置型变量编号 >）

说　明

获取下一位置。此外，在执行该程序库时码垛堆积计数器被递加。
因此，只获取而不动作等情况，需要通过 pltDecCnt 等递减。

相关项目

pltMove、 pltKernel

应用示例

```
CALL pltGetNextPos(1,10)          ' 将码垛堆积编号为 1 号的下一位置赋值Ⅰ[10]。
```

pltGetPLT1END（共用程序库 (Library)）

功　能

码垛堆积 1 段结束标志 (Flag) 的获取

格　式

pltGetPLT1END （< 码垛堆积编号 >、 <1 段结束标志 (Flag) 保存整数编号 >）

说　明

将码垛堆积 1 段结束标志 (Flag) 存放在Ⅰ［<1 段结束标志 (Flag) 保存整数编号 >］。

相关项目

pltGetPLTEND、 pltResetPLT1END、 pltKernel

应用示例

```
CALL pltGetPLTIEND(1,10)          ' 将码垛堆积编号为 1 号的 1 段结束标志 (Flag)
                                  ' 赋值Ⅰ[10]。
```

pltGetPLTEND（共用程序库 (Library)）

功　能

码垛堆积全段结束标志 (Flag) 的获取

格　式

pltGetPLTEND （< 码垛堆积编号 >、 < 全段结束标志 (Flag) 保存整数编号 >）

说　明

将码垛堆积全段结束标志 (Flag) 存放在Ⅰ［< 全段结束标志 (Flag) 保存整数编号 >］。

相关项目

pltGetPLT1END、 pltResetPLTEND、 pltKernel

应用示例

```
CALL pltGetPLTEND(1,10)          ' 将码垛堆积编号为 1 号的全段结束标志 (Flag)
                                 ' 赋值Ⅰ[10]。
```

pltIncCnt（共用程序库 (Library)）

功　能

码垛堆积总计数器的递加

格　式

pltIncCnt （< 码垛堆积编号 >）

说　明

将被指定的码垛堆积编号的码垛堆积总计数器从当前值加 +1。

通过操作码垛堆积计数器，可以做成锯齿型及拔牙型的码垛堆积。

相关项目

pltDecCnt、 pltGetCnt、 pltKernel、 pltLetCnt

应用示例

```
CALL pltIncCnt(1)                ' 将码垛堆积编号为 1 号的码垛堆积总计数器加 1。
```

pltInit1 （共用程序库 (Library)）

功　　能

码垛堆积初始化模板 1

格　　式

pltInit1

说　　明

这是 pltInitialize 的使用示例。

自变量 (argument) 的含义请参照 pltInitialize 的自变量 (argument) 信息。

相关项目

pltInitialize

pltInitialize（共用程序库 (Library)）

功　能

码垛堆积初始化

格　式

pltInitialize （< 码垛堆积编号 >、< 横拆分数 >、< 纵拆分数 >、< 积层数 >、< 接近长度 >、< 脱离长度 >、< 台架高度 >、< 台架 4 角 1P 型变量编号 >、< 台架 4 角 2P 型变量编号 >、< 台架 4 角 3P 型变量编号 >、< 台架 4 角 4P 型变量编号 >）

说　明

是定义码垛堆积动作的初始化程序。

仅对码垛堆积计数器进行初始化时，使用 pltResetAll 共用程序库 (Library) 会更有效率。

注意

如果不执行 pltInitialize 就不能执行码垛堆积动作。

宏定义

mcApprVal : 赋值接近长度的 F 型变量编号

McDepVal : 赋值脱离长度的 F 型变量编号

McPltH : 赋值台架高度的 F 型变量编号

注意

当这些定义与用户使用的变量编号相冲突时，请重新进行宏定义。

相关项目

pltKernel

应用示例

```
CALL pltInitialize(0,4,3,1,50,50,50,52,53,54,55)
                                    ' 码垛堆积编号 : 0、横拆分数 : 4、纵拆分数 : 3、积层数 : 1
                                    ' 接近长度 : 50mm、脱离长度 : 50mm、台架高度 : 50mm
                                    ' 台架 4 角 1: P [52]、 2: P [53]、 3: P [54]、 4: P [55]，
                                    ' 以此进行初始化。
```

pltKernel（共用程序库 (Library)）

功　能

码垛堆积动作 （内核）

格　式

pltKernel （< 码垛堆积编号 >、 < 行动编号 >、 < 行动自变量 (argument)>、 < 错误编号 >）

说　明

是码垛堆积动作的核心程序。在使用其他所有的码垛堆积共用程序库 (Library) 时，需要该共用程序库 (Library)。

注意

该程序不设想用多个任务来执行。如果是用多个任务来执行，则需要利用信号 (semaphore) 等变更为取得同步。

宏定义

mcPaltMax: 最大台架数

只重新定义该宏就可以简单地增加台架数。

注意

如果增加台架数，局部 (Local) 变量区域将被消耗掉该增加部分。

相关项目

pltInitialize

备　注

- 基本上该共用程序库 (Library) 仅被其他的共用程序库 (Library) 调出，没有用户直接呼叫的情况。
- 直接使用时，请在理解了程序内容的基础上加以使用。

pltLetCnt（共用程序库 (Library)）

功　能

码垛堆积总计数器的设定

格　式

pltLetCnt （< 码垛堆积编号 >、 < 总计数器设定值 >）

说　明

可以设定被指定的码垛堆积编号的码垛堆积总计数器。

通过操作码垛堆积计数器，可以做成锯齿型及拔牙型的码垛堆积。

相关项目

pltDecCnt、 pltGetCnt、 pltIncCnt、 pltKernel

应用示例

```
CALL pltLetCnt(1,12)          ' 将码垛堆积编号为 1 号的码垛堆积总计数器
                              ' 设定为 12。
```

pltLetK1（共用程序库 (Library)）

功　能

码垛堆积计数器 K1 的设定

格　式

pltLetK1 （< 码垛堆积编号 >、 <K1 设定值 >）

说　明

可以设定被指定的码垛堆积编号的计数器 K1。

通过操作码垛堆积计数器，可以做成锯齿型及拔牙型的码垛堆积。

相关项目

pltGetK1、 pltKernel

应用示例

```
CALL pltLetK1(1,1)            ' 将码垛堆积编号为 1 号的码垛堆积计数器 K1
                              ' 设定为 1。
```

pltLetM1（共用程序库 (Library)）

功　能

码垛堆积计数器 M1 的设定

格　式

pltLetM1 （< 码垛堆积编号 >、 <M1 设定值 >）

说　明

可以设定被指定的码垛堆积编号的计数器 M1。

通过操作码垛堆积计数器，可以做成锯齿型及拔牙型的码垛堆积。

相关项目

pltGetM1、 pltKernel

应用示例

```
CALL pltLetM1(1,4)          ' 将码垛堆积编号为 1 号的码垛堆积计数器 M1
                            ' 设定为 4。
```

pltLetN1（共用程序库 (Library)）

功　能

码垛堆积计数器 N1 的设定

格　式

pltLetN1 （< 码垛堆积编号 >、 <N1 设定值 >）

说　明

可以设定被指定的码垛堆积编号的计数器 N1。

通过操作码垛堆积计数器，可以做成锯齿型及拔牙型的码垛堆积。

相关项目

pltGetN1、 pltKernel

应用示例

```
CALL pltLetN1(1,3)          ' 将码垛堆积编号为 1 号的码垛堆积计数器 N1
                            ' 设定为 3。
```

pltMain1 （共用程序库 (Library)）

功　　能

码垛堆积 模板 1

格　　式

pltMain1

说　　明

是从台架取出并向组装位置移动的基本作业。

宏定义

pltIndex : 码垛堆积编号

相关项目

pltMain2

备　　注

请根据需要修正后使用。

pltMain2 （共用程序库 (Library)）

功　　能

码垛堆积 模板 2

格　　式

pltMain2

说　　明

是从台架取出并向组装位置移动的基本作业。
在取出之前检查上一次已进行的取出是否正常。

宏定义

pltIndex : 码垛堆积编号
ChuckNG : 取出 NG 信号 （DIO 编号）

相关项目

pltMain1

备　　注

请根据需要修正后使用。

pltMove（共用程序库 (Library)）

功　能

标准码垛堆积 模板 1

格　式

pltMove （< 码垛堆积编号 >）

说　明

执行 pltInitialize 中被定义的标准码垛堆积动作。

按照备注可以插入回避点等。

控制器的软件版本为 Ver.1.2* 之前版本时，请使用 Ver.1.2 互换的类别中的版本。如果使用码垛堆积类中的程序库，则会发生编译异常。

控制器软件的版本确认方法请参照操作指南 "5.7［设定（主）］视窗的显示、各个模块的版本信息的显示、［F6 设定］—［F6 维护］—［F2 版本］"。

宏定义

mcNextPos : 存放下一位置的 P 型变量编号

mcApprLen : 存放接近长度的 F 型变量编号

mcDepLen : 存放脱离长度的 F 型变量编号

注意

当这些定义与用户使用的变量编号相冲突时，请重新进行宏定义。

相关项目

pltInitialize、 pltMove0、 pltKernel

备　注

在获取了下一位置时码垛堆积计数器被递加。因此，只获取而不动作等情况，需要通过 pltDecCnt 等递减。

应用示例

```
CALL pltMove(1)                    ' 执行码垛堆积编号为 1 号的码垛堆积动作。
```

pltMove0（共用程序库 (Library)）

功　能

标准码垛堆积动作 1

格　式

pltMove0

说　明

是特定 pltMove (Index) 的 Index = 0 后的程序。

想对每个台架变更动作时，请追加该程序，将 #DEFINE pltIndex 0 行变更为其台架编号，并变更程序名称。

控制器的软件版本为 Ver.1.2* 之前版本时，请使用 Ver.1.2 互换的类别中的版本。如果使用码垛堆积类中的程序库，则会发生编译异常。

控制器软件的版本确认方法请参照操作指南 "5.7［设定（主）］视窗的显示、各个模块的版本信息的显示、［F6 设定］—［F6 维护］—［F2 版本］"。

宏定义

mcNextPos : 存放下一位置的 P 型变量编号

mcApprLen : 存放接近长度的 F 型变量编号

mcDepLen : 存放脱离长度的 F 型变量编号

注意

当这些定义与用户使用的编号相冲突时，请重新进行宏定义。

pltIndex : 台架编号

相关项目

pltInitialize、 pltMove、 pltKernel

备　注

在获取了下一位置时码垛堆积计数器被递加。因此，只获取而不动作等情况，需要通过 pltDecCnt 等递减。

应用示例

```
CALL pltMove0                    ' 执行码垛堆积编号为 0 号的码垛堆积动作。
```

pltResetAll（共用程序库 (Library)）

功　能

所有码垛堆积计数器的清零

格　式

pltResetAll （< 码垛堆积编号 >）

说　明

清零所有码垛堆积计数器。

与 pltLetCnt（< 码垛堆积编号 >、0）具有几乎相同的含义，但该共用程序库 (Library) 还会把段的结束标志 (Flag) 清零。

无需进行码垛堆积动作的再定义，且只要把计数器清零的话，则可以使用该共用程序库 (Library) 而不是 pltInitialize。

相关项目

pltInitialize、 pltResetPLTEND、 pltKernel

应用示例

```
CALL pltResetA11(1)          ' 将码垛堆积编号为 1 号的码垛堆积计数器
                             ' 全部清零。
```

pltResetPLT1END（共用程序库 (Library)）

功　能

码垛堆积 1 段结束标志 (Flag) 的清零

格　式

pltResetPLT1END （< 码垛堆积编号 >）

说　明

把码垛堆积 1 段结束标志 (Flag) 清零。

相关项目

pltGetPLT1END、 pltResetPLTEND、 pltKernel

应用示例

```
CALL pltResetPLT1END(1)      ' 码垛堆积编号为 1 号的 1 段结束标志 (Flag) 清零。
```

pltResetPLTEND（共用程序库 (Library)）

功　　能

码垛堆积全段结束标志 (Flag) 的清零

格　　式

pltResetPLTEND （< 码垛堆积编号 >）

说　　明

把码垛堆积全段结束标志 (Flag) 清零。

相关项目

pltGetPLTEND、 pltResetPLT1END、 pltKernel

应用示例

CALL pltResetPLTEND (1)　　’码垛堆积编号为 1 号的全段结束标志 (Flag) 清零。

第 5 章　工具操作的共用程序库 (Library)

对工具操作相关功能的共用程序库 (Library) 进行如下说明。

tolChange（共用程序库 (Library)）

功　能

工具更换

格　式

tolChange （< 工具安装编号 >、 < 工具编号 >）

说　明

该共用程序库 (Library) 执行 tolInitialize 中被定义的标准的工具切换动作。

相关项目

tolInitialize、 tolKernel

备　注

PAC 的工具坐标系切换命令 (ChangeTool) 只是切换工具坐标系，并不是装换实际的工具的命令。

应用示例

```
CALL tolChange(1,2)                    ' 将 2 号的工具分配给工具安装编号 1。
```

tolInit1（共用程序库 (Library)）

功　能

工具更换初始化模板 1

格　式

tolInit1

说　明

这是 tolInitialize 的使用示例。

自变量 (argument) 的含义请参照 tolInitialize 的自变量 (argument) 信息。

相关项目

tolInitialize、 tolKernel

tolInitialize（共用程序库 (Library)）

功　能

工具更换初始化

格　式

tolInitialize （< 工具安装编号 >、 < 工具编号 >、 < 卡盘点 P 型变量编号 >、
< 接近长度 >、 < 卡盘 DIO 编号 >、 < 松开卡盘 DIO 编号 >）

说　明

定义标准的工具更换动作。

相关项目

tolInit1、 tolKernel

应用示例

```
CALL tolInitialize(0,1,50,50,40,41)        ' 工具安装编号 : 0、工具编号 : 1
                                           ' 卡盘点 : P [50]、接近长度 : 50mm
                                           ' 卡盘点 : IO [40]、松开卡盘 : IO [41]
```

tolKernel（共用程序库 (Library)）

功　能

工具更换动作 （内核）

格　式

tolKernel （< 工具安装编号 >、 < 工具编号 >、 < 行动编号 >、 < 行动自变量 (argument)>）

说　明

是工具更换的核心程序。在使用其他所有的工具更换共用程序库 (Library) 时，需要该共用程序库 (Library)。

宏定义

mcToolMax : 工具的最大数量

mcTools : 多肢夹治具的情况下最大可安装数量

相关项目

tolInitialize

tolMain1（共用程序库 (Library)）

功　　能

工具更换 模板 1

格　　式

tolMain1

说　　明

是工具更换的使用示例。

工具编号由 ioINB 用 8 比特被指定后，会切换到被指定的工具。

宏定义

ioINB: 工具编号指定 IO 的开头编号

相关项目

tolChange、 tolInitialize、 tolKernel

第 6 章　输入输出的共用程序库 (Library)

是 DIO、 RS232C 输入输出相关功能的共用程序库 (Library)。

dioSync（共用程序库 (Library)）

功　能

与被 DIO 连接的外部设备 （序列器 (sequencer) 等）的同步

格　式

dioSync （< 可以接收数据的信号编号 >、 < 数据接收完成信号编号 >）

说　明

是与被 DIO 连接的外部设备 （定序器等）取得同步的标准手续。

应用示例

```
CALL dioSync(220.221)          ' 将可以接收数据的信号分配给 IO [220]，
                               ' 将数据接收完成信号分配给 IO [221]，获取同步。
```

dioConstantDistanceIoOut （共用程序库 （Library））【Ver.3.0 以上版本】

功　能

当轴处于一定角度 （Z 轴、直动轴等则是距离）动作时，反转 I/O 的输出状态。

格　式

dioConstantDistanceIoOut （< 检测周期 >）

说　明

事先将各轴的角度 （距离）与 I/O 输出端口编号设定好，如果将 < 检测周期 > 设置得大于 1，并执行此共用程序库 （Library），则反转所设定的角度 （距离）、动作时所设定的 I/O 输出端口的输出状态。如果将 < 检测周期 > 设置成 0，并执行此共用程序库 （Library），则不反转 I/O 的输出状态。

角度 （距离）与 I/O 输出端口编号的设定，可在各轴上进行。

利用多功能教导器或 WINCAPS Ⅲ进行。

- 操作路径：多功能教导器
  [F4 I/O] - [F6 辅助功能] - [F1 夹治具设定]
- 操作路径：WINCAPS Ⅲ
  在 " 项目 " 菜单选择 " 参数 "，点击 "I/O" 图标。

项目	No. 注 1	参数名称	说明
1 轴至 8 轴的 I/O 输出端口编号	85	位置输出 输出 IO 编号 (J1)	请输入输出 I/O 端口编号。
	86	位置输出 输出 IO 编号 (J2)	
	:	:	
	91	位置输出 输出 IO 编号 (J7)	
	92	位置输出 输出 IO 编号 (J8)	
1 轴至 8 轴的角度（距离）	93	位置输出 检测角度 (J1 deg*10^5)	请输入用指定角度乘以 100000 得出的数值。如果设置为 0，其轴的输出 I/O 端口不会翻转。
	94	位置输出 检测角度 (J2 deg*10^5)	
	:	:	
	99	位置输出 检测角度 (J7 deg*10^5)	
	100	位置输出 检测角度 (J8 deg*10^5)	

注 1：在 WINCAPS Ⅲ中，则在编号上加 1。

利用以下算式将 < 检测周期 > 处设定的数值换算成实际周期，并在此周期内检测处各轴的角度 （距离）。

此外，< 检测周期 > 的设定范围为 0~32。

周期 =250μ sec × < 检测周期 >

宏定义

需要 <pacman.h> 文件夹。

注意事项

如果将 < 检测周期 > 设置得大于 1，并执行此共用程序库 （Library），将设定的 I/O 输出端口置于 OFF 之后，再开始检测。

此功能是一个可选功能。请购买功能许可证。

应用示例

```
TAKEARM
DRIVEA @E (4,-180)                  ' 将 4 轴归至初始位置。
CALL dioConstantDistanceIoOut(1)    ' 开始检测 （250usec 周期）
DRIVEA @E (4,180)                   ' 让 4 轴进行 360 度动作。
CALL dioConstantDistanceIoOut(0)    ' 停止检测
```

第 7 章　臂动作的共用程序库 (Library)

对臂动作相关功能的共用程序库 (Library) 进行如下说明。

mvResetPulseWidth（共用程序库 (Library)）

功　能

将停止时的允许脉冲宽度设为默认值。

格　式

mvResetPulseWidth

说　明

将停止时允许脉冲宽度的设定值全轴设定为默认值的 20。

所谓停止时允许脉冲宽度是指在进行编码器值确认动作 （@E 指定）时被视为停止的误差脉冲。

宏定义

需要 <pacman.h>。

应用示例

```
CALL mvResetPulseWidth                    ' 将停止时的允许脉冲宽度设为默认值。
```

mvResetPulseWidthJnt（共用程序库 (Library)）【Ver.1.7 以上版本】

功　　能

将指定的附加轴的停止时允许脉冲宽度设定回默认值。

格　　式

mvResetPulseWidthJnt （< 轴编号 >）

说　　明

将用 < 轴编号 > 指定的附加轴的停止时允许脉冲宽度设定回默认值的 20。所谓停止时允许脉冲宽度是指在进行编码器值确认动作 （@E 指定）时被视为停止的误差脉冲。

宏定义

需要 <pacman.h>。

相关项目

mvResetPulseWidth、 mvSetPulseWidthJnt

注意事项

mvResetPulseWidthJnt 是附加轴专用的。请对机械手轴使用 mvResetPulseWidth。

应用示例

```
CALL mvResetPulseWidthJnt(7)          ' 将 7 轴的停止时允许脉冲宽度设定回默认值。
```

mvResetTimeOut（共用程序库 (Library)）

功　能

将动作结束超时值设为默认值。

格　式

mvResetTimeOut

说　明

将动作结束超时的设定值设定为默认值的 5600 (ms)（5.6 秒）。

所谓动作结束超时是指在编码器值确认动作（@E 指定）过程中，直到进入停止时允许脉冲宽度为止的时间的极限值，即使超出了极限值也没有进入脉冲宽度时，则为错误 "6651 检查命令超时 "。

宏定义

需要 <pacman.h>。

应用示例

```
CALL mvResetTimeOut                ' 将动作结束超时值设为默认值。
```

mvReverseFlip（共用程序库 (Library)）

功　能

4 轴形态反转

格　式

mvReverseFlip （< 基准位置 >、 < 非检查位置 >、 < 判断基准值 >）

说　明

比较 < 基准位置 > 与 < 非检查位置 > 的 4 轴值，如果超出了 < 判断基准值 > 则反转 FLIP / NONFLIP。

应用示例

```
CALL mvReverseFlip(P0,P1,90)       ' 比较基准位置 P [0] 与非检查位置 P [1],
                                   ' 如果绝对值超出了 90 则反转形态。
```

mvSetPulseWidth（共用程序库 (Library)）

功　　能

设定停止时允许脉冲宽度

格　　式

mvSetPulseWidth（< 停止时允许脉冲宽度 J1>、< 停止时允许脉冲宽度 J2>、< 停止时允许脉冲宽度 J3>、< 停止时允许脉冲宽度 J4>、< 停止时允许脉冲宽度 J5>、< 停止时允许脉冲宽度 J6>）

说　　明

设定停止时允许脉冲宽度。

所谓停止时允许脉冲宽度是指在进行编码器值确认动作 （@E 指定）时被视为停止的误差脉冲。

宏定义

需要 <pacman.h>。

注意事项

请将脉冲宽度设定为 1 以上。指定为 0 时，会成为默认值 (20)，指定为负数时，会发生错误 "6003 超出有效的数值范围 "。此外，如果设定为较小值，则有时会发生错误 "6651 检查命令超时 "。

应用示例

```
CALL mvSetPulseWidth (10， 10， 10， 10， 10， 10)
                                        ' 将停止时允许脉冲宽度全轴设为 10 脉冲。
```

mvSetPulseWidthJnt（共用程序库 (Library)）【Ver.1.7 以上版本】

功　　能

设定指定的附加轴的停止时允许脉冲宽度。

格　　式

mvSetPulseWidthJnt （< 轴编号 >、 < 停止时允许脉冲宽度 >）

说　　明

将用 < 轴编号 > 指定的附加轴的停止时允许脉冲宽度设定 < 停止时允许脉冲宽度 >。所谓停止时允许脉冲宽度是指在进行编码器值确认动作 （@E 指定）时被视为停止的误差脉冲。

宏定义

需要 <pacman.h>。

相关项目

mvResetPulseWidthJnt、 mvSetPulseWidth

注意事项

mvSetPulseWidthJnt 是附加轴专用的。请对机械手轴使用 mvSetPulseWidth。

应用示例

```
CALL mvSetPulseWidthJnt(7,10)                    ' 将 7 轴的停止时允许脉冲宽度设定为值 10。
```

mvSetTimeOut（共用程序库 (Library)）

功　能

设定动作结束超时值

格　式

mvSetTimeOut （< 动作结束超时 >）

说　明

设定动作结束超时值。（单位 : ms）

所谓动作结束超时是指在编码器值确认动作 （@E 指定）过程中，直到进入停止时允许脉冲宽度为止的时间的极限值，即使超出了极限值也没有进入脉冲宽度时，则为错误 "6651 检查命令超时 "。

宏定义

需要 <pacman.h>。

注意事项

请将超时值设定为 1 以上。指定为 0 时，则成为默认值 (5600)，指定为负数时，则会发生错误 "6003 超出有效的数值范围 "。此外，如果设定为较小值，则有时会发生错误 "6651 检查命令超时 "。

应用示例

```
CALL mvSetTimeOut (3000)                ' 将超时时间设定为 3 秒。
```

SetGravity（共用程序库 (Library)）【Ver.1.2 以上版本】

功　　能

修正各个关节的静负荷 （重力转矩），设定重力平衡。

格　　式

SetGravity

说　　明

机械手的各个关节，由于重力承受向下的静荷载重量 （重力扭矩）。重力扭矩根据工具及工件的质量、重心位置、机械手的姿势而变化。设定电流限制，限制电机的扭矩时，如果限制扭矩在重力扭矩以下，会导致各关节在重力方向上动作。为此，此程序修正从限制扭矩到重力扭矩的部分，并且具有即使在设定电流限制时也不会在重力方向上动作的功能，起到维持重力平衡的作用。

宏定义

必须有 <pacman.h> 文件。

相关项目

SetCurLmt、 ResetGravity、 SetGrvOffset

注意事项

- 请通过获取 (TAKEARM) 机械手控制权的任务来执行。尚未获取机械手控制权的情况下，会发生 " 不能获取 21F7 臂信号 (semaphore)" 错误。
- 请正确设定前端负荷质量和负荷重心位置。在没被正确设定的状态下，如果将电流限制设定值进行减小（30 以下左右）设定，则有时会发生重力下降请予以注意。前端负荷质量和负荷重心位置设定，请参照程序 1 的 p. 4 - 15 "4.7 ' 使用条件 ' 中最佳可搬运质量设定功能 "。
- 不知道前端部负荷重量和负荷重心位置是否正确，即使正确进行了设定在电流限制设定时依然重力下降时，使用重力补偿修正功能 (SetGrvOffset)，可以进行重力补偿值的修正。请参照重力补偿修正共用程序库 (Library)。
- 在教导器上将使用条件的重力补偿有效无效设定为 1 时，重力补偿为有效。用教导器设定为有效时，自启动控制器电源，执行校准之后开始，重力补偿变为有效。

应用示例

```
CALL SetGravity                 ' 使重力补偿有效。
Delay 100                       ' 等待重力补偿的反映。
CALL SetCurLmt (2,30)           ' 将 2 轴的电流限制值设定为 30%。
```

ResetGravity（共用程序库 (Library)）【Ver.1.2 以上版本】

功　　能

将重力平衡设为无效。

格　　式

ResetGravity

说　　明

将重力平衡设为无效。

宏定义

必须有 <pacman.h> 文件。

相关项目

ResetCurLmt、 SetEralw

注意事项

1) 请通过获取 (TAKEARM) 机械手控制权的任务来执行。尚未获取机械手控制权的情况下，会发生 " 不能获取 21F7 臂信号 (semaphore)" 错误。
2) 如果在电流限制过程中执行命令，会发生 " 不能将 665b 重力补偿设为无效 " 错误。请在解除了电流限制之后再次执行。
3) 在教导器上将使用条件的重力补偿有效无效设定即使设定为 0 时，重力补偿也为无效。但是，与 2) 相同，电流限制过程中不能进行设定。

应用示例

CALL ResetGravity

SetGrvOffset（共用程序库 (Library)）【Ver.1.2 以上版本】

功　　能

通过各个关节的重力转矩修正重力补偿值。

格　　式

SetGrvOffset

说　　明

机械手的各个关节，由于重力承受向下的静荷载重量（重力扭矩）。虽然可以通过重力补偿共用程序库 (Library) (SetGravity) 设定重力平衡，但是由于前端负荷质量设定值与实际的负荷质量的偏差等，在重力平衡上有时会产生偏差。本功能在机械手停止的状态下推测重力扭矩并修正重力补偿值。

宏定义

必须有 <pacman.h> 文件。

相关项目

SetCurLmt、 SetGravity、 ResetGrvOffset

注意事项

- 请通过获取 (TAKEARM) 机械手控制权的任务来执行。尚未获取机械手控制权的情况下，会发生 " 不能获取 21F7 臂信号 (semaphore)" 错误。
- 请在电机电源为 ON 机械手停止状态下执行。在电机电源 OFF 状态下执行时，会发生 "6006 电机电源关闭 " 错误。在机械手动作中执行时，会发生 "600B 机械手正在 动作 " 错误。
- 执行重力补偿修正的姿势有较大的变化时，在修正上会发生偏离。此时，请再次实施修正。
- 使用条件的电流限制清零设定值为 1、 3、 5、 7 以外的情况下，接通电机电源时修正值被清零为 0。

应用示例

CALL SetGrvOffset

ResetGrvOffset（共用程序库 (Library)）【Ver.1.2 以上版本】

功　　能

将重力补偿值的修正设为无效。

格　　式

ResetGrvOffset

说　　明

将重力补偿值的修正设为无效。

宏定义

必须有 <pacman.h> 文件。

相关项目

SetGrvOffset

注意事项

- 请通过获取 (TAKEARM) 机械手控制权的任务来执行。尚未获取机械手控制权的情况下，会发生 " 不能获取 21F7 臂信号 (semaphore)" 错误。
- 请在机械手停止状态下执行。在机械手动作中执行时，会发生 "600B 机械手正在 动作 " 错误。即使电机电源 OFF 当中也可以执行。

应用示例

CALL ResetGrvOffset

SetCurLmt（共用程序库 (Library)）【Ver.1.2 以上版本】

功　能

限制指定轴的电机电流值。

格　式

SetCurLmt （< 轴编号 >、 < 设定值 >）

说　明

将 < 轴编号 > 所指定轴的电机电流值 （力矩）限制设定值。在插入操作、对接操作时，需要限制施加在工件上的力时使用。

< 设定值 > 中电机额定电流值为 100。设定超过各个轴的允许值时，被允许值限制。

请设定 1 以上的数值。设定 0 以下的数值时，会发生错误 6003 " 超出有效数值范围 "。

宏定义

必须有 <pacman.h> 文件。

相关项目

ResetCurLmt、 SetGravity、 SetGrvOffset、 SetEralw

注意事项

- 正在进行电流限制设定时，因电机电流被限制，所以不能在最高速度、最高加速度下动作。请仅在必要的步骤使用电流限制。同时，使用电流限制时，请降低加速度。
- 使用电流限制即使限制推力，但在高速下工件碰撞时，也会因工件和夹治具和轴的惯性产生冲击力。电流限制请在工件接触之前设定，并降低速度。
- 电流限制请在机械手停止状态下设定。如果在通过 (Pass) 动作过程中设定，则会发生错误。
- 请通过获取 (TAKEARM) 机械手控制权的任务来执行。尚未获取机械手控制权的情况下，会发生 " 不能获取 21F7 臂信号 (semaphore)" 错误。
- 使用条件的电流限制清零设定值在 1、 3、 5、 7 以外的情况下，接通电机电源时电流限制被清零。
  电机电源接通之后，置电流限制为有效时，请将电流限制清零设定值的最下位比特设定为 1。

6 轴

- 设定电流限制时，请务必将重力补偿功能设为有效。在重力补偿功能无效的状态下设定电流限制时，会发生 " 不能设定 665a 电流限制 " 错误。重力补偿请参照 SetGravity。
- 请正确设定前端负荷质量和负荷重心位置。在没被正确设定的状态下，如果将设定值进行减小 （30 以下左右）设定，则有时会发生重力下降请予以注意。前端负荷质量和负荷重心位置设定，请参照 p. 4-15 "4.7 ' 使用条件 ' 中最佳可搬运质量设定功能 "。
- 电流限制清零设定值的最下位比特被设定为 1 时，接通电机电源之后，有时会发生重力下降。请在电机电源 OFF 状态下复位电流限制 （执行 ResetCurLmt），接通电机电源。
- 不知道前端部负荷重量和负荷重心位置是否正确，即使正确地进行设定依然重力下降时，请使用重力补偿修正功能 (SetGrvOffset)，修正重力补偿值。

4 轴、 HS-E

在没有气体平衡的 4 轴机械手的 Z 轴、T 轴上设定电流限制时，如果减小限制值，则会发生 Z 轴下降或 T 轴旋转。请在运行 Z 轴、 T 轴重力补偿值设定功能 (st_SetZBalance) 之后，实行电流限制。

应用示例

6 轴

```
CALL SetGravity                 ' 使重力补偿有效。
CALL SetGrvOffet                ' 修正重力补偿值。
CALL SetEralw(2, 20)            ' 将 2 轴的偏差允许值设定为 20 度。
CALL SetCurLmt(2, 30)           ' 将 2 轴的电流限制值设定为 30%。
```

4 轴

```
CALL SetEralw(2, 20)            ' 将 2 轴的偏差允许值设定为 20 度。
CALL SetCurLmt(2, 30)           ' 将 2 轴的电流限制值设定为 30%。
```

4 轴、 HS-E

```
st_SetZBalance                  ' 设定 Z 轴重力补偿值。
CALL SetEralw(3, 100)           ' 将 Z 轴的偏差允许值设定为 100mm。
CALL SetEralw(4, 30)            ' 将 T 轴的偏差允许值设定为 30 度。
CALL SetCurLmt(3, 10)           ' 将 Z 轴的电流限制值设定为 10%。
CALL SetCurLmt(4, 10)           ' 将 T 轴的电流限制值设定为 10%。
```

ResetCurLmt（共用程序库 (Library)）【Ver.1.2 以上版本】

功　能

解除指定轴的电机电流限制。

格　式

ResetCurLmt （< 轴编号 >）

说　明

解除 < 轴编号 > 所指定轴的电机电流限制。解除之后，电流限制值和偏差允许值返回到默认值。

［Ver.1.4 以前版本］

如果将 0 指定为轴编号，则解除所有轴的电流限制。

［Ver.1.5 以上版本］

如果在轴编号上指定 0，则解除 ResetCurLmt 执行任务的正在获取信号 (semaphore) 的轴的全部的电流限制。

宏定义

必须有 <pacman.h> 文件。

相关项目

SetCurLmt、 ResetEralw

注意事项

- 在解除电流限制时，实施去除偏差处理。当由于外力有角度偏差时，根据设定的速度、加速度，去除偏差处理时间会有变化。要缩短去除偏差处理时间，请提高速度、加速度的设定。
- 即使在电机电源 OFF 状态也可以执行。在电机电源 OFF 状态下解除电流限制时，请在结束正在获取臂信号 (semaphore) 的任务之后，执行以下的程序。

```
PRO999
TAKEARM
CALL ResetCurLmt(0)
END
```

［Ver.1.4 以前版本］

请通过获取 (TAKEARM) 机械手控制权的任务来执行。尚未获取机械手控制权的情况下，会发生 " 不能获取 21F7 臂信号 (semaphore)" 错误。

［Ver.1.5 以上版本］

请通过获取 (TAKEARM) 控制权的任务执行。指定尚未获取控制权的轴时，会发生 " 不能获取 27D* * 轴信号 (semaphore)" 错误。

应用示例

```
CALL ResetCurLmt(0)          ’解除所有轴的电流限制
CALL ResetGrvOffset          ’将重力补偿修正值清零
```

SetEralw（共用程序库 (Library)）【Ver.1.2 以上版本】

功　　能

变更指定轴的偏差允许值。

格　　式

SetEralw （< 轴编号 >、< 设定值 >）

说　　明

设定在 < 轴编号 > 上所指定的轴的偏差允许值。在限制电流时由于外力产生角度偏差时使用。

< 设定值 > 在旋转关节时，用臂关节角度 （度）指定；在直动关节时以臂关节位移 (mm) 指定。

所谓偏差允许值，是为了保证安全，当伺服偏差达到一定值以上时发生的错误 "611* J* 偏差过大 " 的允许范围。当使用电流限制进行装配时，有时会发生由于外力产生伺服偏差的错误。为了避免该现象，在暂时将偏差允许值放大时使用。此外，为了在电流限制调整中尽早查出重力掉落，在偏差允许值设定变小时也使用。

宏定义

必须有 <pacman.h> 文件。

相关项目

SetCurLmt、 ResetEralw

注意事项

请通过获取 (TAKEARM) 机械手控制权的任务来执行。尚未获取机械手控制权的情况下，会发生 "不能获取21F7 臂信号 (semaphore)" 错误。

应用示例

```
CALL SetEralw(2, 20)                    ' 将 2 轴的偏差允许值设定为 20 度。
```

ResetEralw（共用程序库 (Library)）【Ver.1.2 以上版本】

功　能

将指定轴的偏差允许值返回到默认值。

格　式

ResetEralw （< 轴编号 >）

说　明

将 < 轴编号 > 所指定轴的偏差允许值返回到默认值。

［Ver.1.4 以前版本］

如果将 0 指定为轴编号，则将所有轴的偏差允许值返回到默认值。

［Ver.1.5 以上版本］

如果在轴编号上指定 0，则将 ResetEralw 执行任务的正在获取信号 (semaphore) 的轴的全部偏差允许值返回默认值。

宏定义

必须有 <pacman.h> 文件。

相关项目

ResetCurLmt、 SetEralw

注意事项

在执行电流限制解除命令 (ResetCurLmt) 时，偏差允许值也成为默认值。

［Ver.1.4 以前版本］

请通过获取 (TAKEARM) 机械手控制权的任务来执行。尚未获取机械手控制权的情况下，会发生 " 不能获取 21F7 臂信号 (semaphore)" 错误。

［Ver.1.5 以上版本］

请通过获取 (TAKEARM) 控制权的任务执行。指定尚未获取控制权的轴时，会发生 " 不能获取 27D* * 轴信号 (semaphore)" 错误。

应用示例

```
CALL ResetEralw(0)                    ' 将所有轴的偏差允许值返回到默认值。
```

OnSrvLock（共用程序库 (Library)）

功　能

将指定轴置为伺服锁定状态。（4 轴机械手专用）

格　式

OnSrvLock （< 指定轴 >）

说　明

提供与原来语言的 ON SVLOCK 命令同等的功能。

所谓伺服锁定是指机械手的臂被控制，其位置被锁定的状态。

宏定义

必须有 <pacman.h> 文件。

相关项目

OffSrvLock

注意事项

- 伺服锁定请在机械手停止状态下设定。如果在通过 (Pass) 动作过程中设定，则会发生错误。
- 请通过获取 (TAKEARM) 机械手控制权的任务来执行。尚未获取机械手控制权的情况下，会发生 " 不能获取 21F7 臂信号 (semaphore)" 错误。

应用示例

```
CALL  OnSrvLock(1)          '1 轴的伺服锁定
CALL  OnSrvLock(0)          ' 所有轴的伺服锁定
```

OffSrvLock（共用程序库 (Library)）

功　能

解除指定轴的伺服锁定。（4 轴机械手专用）

格　式

OffSrvLock

说　明

提供与原来语言的 OFF SVLOCK 命令同等的功能。

所谓伺服锁定是指机械手的臂被控制，其位置被锁定的状态。如果解除伺服锁定则机械手的臂的位置不能被定位，一旦施加外力则位置发生偏移。

宏定义

必须有 <pacman.h> 文件。

相关项目

OnSrvLock

注意事项

- 伺服锁定处于解除状态的轴，不能执行动作指令。
- 请在机械手停止状态下设定伺服锁定解除。如果在通过 (Pass) 动作过程中设定，则会发生错误。
- 请通过获取 (TAKEARM) 机械手控制权的任务来执行。尚未获取机械手控制权的情况下，会发生 " 不能获取 21F7 臂信号 (semaphore)" 错误。
- 使用条件的 "25: 电流限制清零 " 设定值的第 2 比特为 "0" （初始值）时，在电机电源接通时，伺服锁定解除被清零（伺服锁定）。电机电源接通之后紧接着将伺服锁定解除设为有效时，请将 "+2" 设定为电流限制清零设定值。
  使用条件 "25: 电流限制清零 " 设定示例

	PWM	SVLock	Curlmt	
	*	*	*	
仅 SVLock 有效时	0	1	0	＝ 2
全都有效时	1	1	1	＝ 7

应用示例

```
CALL  OnSrvLock(1)          '1 轴的伺服锁定
```

OnPWM（共用程序库 (Library)）

功　　能

将指定的轴进行 PWM 切换控制。（4 轴机械手专用）

格　　式

OnPWM （< 指定轴 >）

说　　明

所谓 PWM 切换状态是指机械手的臂被控制，其位置被保持的状态。

宏定义

必须有 <pacman.h> 文件。

相关项目

OffPWM

注意事项

- PWM 切换请在机械手停止状态下设定。如果在通过 (Pass) 动作过程中设定，则会发生错误。
- 请通过获取 (TAKEARM) 机械手控制权的任务来执行。尚未获取机械手控制权的情况下，会发生 " 不能获取 21F7 臂信号 (semaphore)" 错误。

应用示例

```
CALL  OnPWM(1)          '1 轴的 PWM 切换
CALL  OnPWM(0)          ' 全轴的 PWM 切换
```

OffPWM（共用程序库 (Library)）

功　能

解除指定轴的 PWM 切换控制。（4 轴机械手专用）

格　式

OffPWM （< 指定轴 >）

说　明

如果解除 PWM 转换则机械手的臂的位置不能被保持，一旦施加外力则位置发生偏移。机械手为完全自由的状态。

宏定义

必须有 <pacman.h> 文件。

相关项目

OnSrvLock

注意事项

- PWM 转换处于解除状态的轴不能执行动作指令。
- PWM 切换解除请在机械手停止状态下设定。如果在通过 (Pass) 动作过程中设定，则会发生错误。
- 请通过获取 (TAKEARM) 机械手控制权的任务来执行。尚未获取机械手控制权的情况下，会发生 " 不能获取 21F7 臂信号 (semaphore)" 错误。
- 使用条件的 "25: 电流限制清零 " 设定值的第 3 比特为 "0"（初始值）时，在电机电源接通时，PWM 切换解除被清零（PWM 切换）。自电机电源接通之后，为了使 PWM 转换解除有效，请将电流限制清零设定值设定为 "+4"。
  使用条件 "25: 电流限制清零 " 设定示例

	PWM	SVLock	Curlmt	
	*	*	*	
仅 PWM 有效时	1	0	0	＝ 4
全都有效时	1	1	1	＝ 7

应用示例

```
CALL  OffPWM(1)                ’1 轴的 PWM 转换解除
```

Setcycloid（共用程序库 (Library)）

功　能

转移至抑制 PTP 动作结束动作时的过冲量及残留振动的摆线动作模式。

格　式

Setcycloid

说　明

- 在摆线动作模式中，可以使减速时的速度变化更平顺。
  为此，可以降低停止时的超程量及残留振动。
- 在摆线动作模式中，动作时间多少有些增加。请确认循环时间。
- 在执行了 Setcycloid 之后执行了其他的程序时，本模式为有效的。解除时，请执行 Resetcycloid。

宏定义

必须有 <pacman.h> 文件。

相关项目

Resetcycloid

注意事项

- 摆线动作模式仅在最佳可搬运质量设定功能被设定为模式 0 或模式 2 时有效。
- 当最佳可搬运质量设定功能被设定为模式 1 或模式 3 时，则优先最佳可搬运质量设定功能。

应用示例

CALL Setcycloid ，转移到摆线动作模式

SetCycloidJnt（共用程序库 (Library)）【Ver.1.7 以上版本】

功　　能

转移至抑制附加轴结束动作时的超程量及残留振动的摆线动作模式。

格　　式

SetCycloidJnt （< 轴编号 >）

说　　明

- 在摆线动作模式中，可以使减速时的速度变化更平顺。为此，可以降低停止时的超程量及残留振动。
- 在使用摆线动作模式时，动作时间会增加。请在确认循环时间有所剩余后再使用。
- 执行 SetCycloidJnt 之后，其他的程序指令已设定的轴动作时，本模式仍然有效。解除时，请执行 ResetCycloidJnt。
- 在与机械手的同步动作中，该设定不被参照。按照机械手的设定动作。
- 当多个附加轴同步动作时，如果动作轴中的 1 个被设定为摆线动作模式，则同步动作的所有的轴都被适用该设定。不能按照每个轴改变设定。

宏定义

必须有 <pacman.h> 文件。

相关项目

ResetCycloidJnt

注意事项

如果没有获取指定的轴的控制权 (TAKEARM)，则不能执行。

SetCycloidJnt 是附加轴专用的。请对机械手轴使用 SetCycloid。

应用示例

```
CALL  SetCycloidJnt(8)                    ’将 8 轴转移到摆线动作模式
```

Resetcycloid（共用程序库 (Library)）

功　　能

从摆线动作模式转移到通常模式。

格　　式

Resetcycloid

说　　明

从摆线动作模式转移到通常动作模式。再次想要置为摆线动作模式时，请执行 Setcycloid。

宏定义

必须有 <pacman.h> 文件。

相关项目

Setcycloid

注意事项

请通过获取 (TAKEARM) 机械手控制权的任务来执行。尚未获取机械手控制权的情况下，会发生 "不能获取21F7臂信号 (semaphore)" 错误。

应用示例

```
CALL  Resetcycloid                    ' 摆线动作模式的解除
```

ResetCycloidJnt（共用程序库 (Library)）【Ver.1.7 以上版本】

功　　能

解除摆线动作模式，转移到通常模式。

格　　式

ResetCycloidJnt （< 轴编号 >）

说　　明

从摆线动作模式转移到通常动作模式。若再次想要置为摆线动作模式，请执行 SetCycloidJnt。

宏定义

必须有 <pacman.h> 文件。

相关项目

SetCycloidJnt

注意事项

如果没有获取指定的轴的控制权 (TAKEARM)，则不能执行。

ResetCycloidJnt 是附加轴专用的。请对机械手轴使用 ResetCycloid。

应用示例

```
CALL  ResetCycloidJnt(8)                    ' 解除 8 轴的摆线动作模式
```

SetCPSpdMode（共用程序库 (Library)）【Ver.1.8 以上版本】

功　　能

变更 CP 动作时的 TCP 速度设定。

格　　式

SetCPSpdMode （< 设定值 >）

说　　明

默认设定中，当进行包括手指的旋转动作的 CO 动作时，根据旋转动作量 TCP 速度会被减速。通过将设定设为 1，旋转动作量如果处于一定量以下， TCP 速度就可以保持稳定。

注意事项

当设定为 1 时，为了保持 TCP 速度稳定，根据动作内容旋转速度会变化。当欲执行超出被设定的旋转速度的限界值的动作时，在被显示警告的同时 TCP 速度减速、动作执行。

应用示例

```
CALL SetCPSpdMode(0)          ’设为默认的速度设定
CALL SetCPSpdMode(1)          ’设定 TCP 速度为 1
```

SetForce_HM（共用程序库 (Library)）

功　　能

HM/HS 机械手指定 Z 轴的推力 （单位 : N）的电流限制共用程序库 (Library)。

格　　式

SetForce_HM （< 推力 （单位 : N） >）

说　　明

通过限制 Z 轴的电机电流来限制推力。单位是 N （牛顿）。

虽然根据与周围设备相接触时产生的冲击力的不同最大推力有很大的不同，但是其可以指定的推力是在碰撞时的冲击力几乎为 "0" 时的推力。（极低速状态下碰撞时）此外由于机械的摩擦 Z 轴的位置变化也会使推力不同，所以请始终视其为标准。（不是保证值）此外与电流限制共用程序库 (Library) (SetCurlmt) 相同，当设定了超出允许值时，被允许值限制。

宏定义

必须有 <pacman.h> 文件。

相关项目

SetForce_HC、 SetCurLmt、 ResetCurLmt

注意事项

- 执行电流限制时，因为电机电流被限制，所以不能在最高速度、最高加速度下动作。请仅在必要的步骤使用电流限制。同时，使用电流限制时，请降低加速度。
- 使用电流限制即使限制推力，但在高速下工件碰撞时，也会因工件和夹治具的惯性产生冲击力。请在工件接触前设定电流限制，并降低速度。
- 请在机械手停止状态下设定电流限制。如果在通过 (Pass) 动作时设定，则有时会发生错误。
- 请通过获取了机械手控制权 (TAKEARM) 的任务执行。尚未获取机械手控制权的情况下，会发生 " 不能获取 21F7 臂信号 (semaphore)" 错误。
- 使用条件的 "25: 电流限制清零 " 设定值的第 1 比特为 "0" （初始值）时，在电机电源接通时，伺服锁定解除被清零 （伺服锁定）。自电机电源接通之后，为了使伺服锁定解除有效，请将电流限制清零设定值设定为 "+1"。设定方法请参照 "OffSrvlock" 的注意事项。
- 当解除该指令时，请执行 "ResetCurlmt (3) "。
- 即使进行详细设定，有时会因摩擦等关系指定的推力不被输出。

应用示例

```
CALL  SetForce_HM(10.0)                ’指定 10.0 (N) 的推力
```

SetForce_HC（共用程序库 (Library)）

功　能

是通过 HC 机械手指定 Z 轴推力 （单位 : N）的电流限制共用程序库 (Library)。

格　式

SetForce_HC （< 推力 （单位 : N） >）

说　明

通过限制 Z 轴的电机电流来限制推力。单位是 N （牛顿）。

虽然根据与周围设备相接触时产生的冲击力的不同最大推力有很大的不同，但是其可以指定的推力是在碰撞时的冲击力几乎为 "0" 时的推力。（极低速状态下碰撞时）此外由于机械的摩擦 Z 轴的位置变化也会使推力不同，所以请始终视其为标准。（不是保证值）此外与电流限制共用程序库 (Library) (SetCurlmt) 相同，当设定了超出允许值时，被允许值限制。

宏定义

必须有 <pacman.h> 文件。

相关项目

SetForce_HM、 SetCurLmt、 ResetCurLmt

注意事项

- 执行电流限制时，因为电机电流被限制，所以不能在最高速度、最高加速度下动作。请仅在必要的步骤使用电流限制。同时，使用电流限制时，请降低加速度。
- 使用电流限制即使限制推力，但在高速下工件碰撞时，也会因工件和夹治具的惯性产生冲击力。请在工件接触前设定电流限制，并降低速度。
- 请在机械手停止状态下设定电流限制。如果在通过 (Pass) 动作时设定，则有时会发生错误。
- 请通过获取了机械手控制权 (TAKEARM) 的任务执行。尚未获取机械手控制权的情况下，会发生 " 不能获取 21F7 臂信号 (semaphore)" 错误。
- 使用条件的 "25: 电流限制清零 " 设定值的第 1 比特为 "0" （初始值）时，在电机电源接通时，伺服锁定解除被清零 （伺服锁定）。自电机电源接通之后，为了使伺服锁定解除有效，请将电流限制清零设定值设定为 "+1"。设定方法请参照 "DffSrvlock" 的注意事项。
- 当解除该指令时，请执行 "ResetCurlmt (3) "。
- 即使进行详细设定，有时会因摩擦等关系指定的推力不被输出。

应用示例

```
CALL  SetForce_HC(10.0)                ’指定 10 (N) 的推力
```

SetCompControl（共用程序库 (Library)）【Ver.1.4 以上版本】

功　　能

将力限制功能设为有效。（6 轴专用命令）

格　　式

SetCompControl

说　　明

将力限制功能设为有效。将通过 SetFrcLimit、 SetCompRate、 SetFrcCoord 等被设定的力限制条件置为有效。

宏定义

必须有 <pacman.h> 文件。

相关项目

SetFrcLimit、 SetCompRate、 SetFrcCoord、 ResetCompControl、 SetCompFControl

注意事项

- 在重力补偿无效、电流限制有效的情况下，执行本共用程序库 (Library) 时，则会发生错误 "60f5 不能执行力限制 "。请将电流限制置为无效，重力补偿置为有效后再次执行。电流限制无效化请参照 ResetCurLmt、重力补偿有效化请参照 SetGravity。
- 如在电机电源 OFF 状态下执行，力限制为无效。此外，在力限制过程中将电机电源 OFF 时，力限制也为无效。
- 请通过获取 (TAKEARM) 机械手控制权的任务来执行。尚未获取机械手控制权的情况下，会发生 " 不能获取 21F7 臂信号 (semaphore)" 错误。
- 请在机械手停止状态下执行。当在通过 (Pass) 动作中执行本程序库时，会结束动作。此外，在机械手动作中执行本程序库，发生错误 "600b 机械手正在动作 " 时，请通过 Delay 命令等等待机械手停止后再执行。
- 在力限制过程中机械手因外力等作用动作时，有时会发生 "611* 偏差过大 " 错误。当发生错误时，请变更偏差允许值。偏差允许值变更请参照 SetEralw。
- 请不要在机械手接触状态等对机械手施加力的状态下执行本共用程序库 (Library)。在施加力的状态下将力限制功能置为有效时，请使用 SetCompFControl。
- 执行 SetCompControl 之后，有时由于重力补偿的修正值产生误差，机械手的姿势会发生在机械手重力方向上动作等很大变化。在力限制中姿势有很大变化时，请中途通过 ResetCompControl 将力限制置为无效后再次通过 SetCompControl 将力限制置为有效。

应用示例

```
CALL SetFrcCoord(1)                    ’设定力限制坐标系。
CALL SetFrcLimit(100,0,100,100,100,100)
                                       ’设定力限制比例。
CALL SetCompControl                    ’将力限制功能设为有效。
CALL SetEralw(1,90)                    ’设定偏差允许值。
```

SetCompFControl（共用程序库 (Library)）【Ver.1.4 以上版本】

功　能

将力限制功能设为有效。（6 轴专用命令）

格　式

SetCompFControl

说　明

与 SetCompControl 相同，将力限制功能置为有效。但是，通过 SetCompControl 被执行的重力补偿修正处理不被执行。

宏定义

必须有 <pacman.h> 文件。

相关项目

SetCompControl

注意事项

- 在重力补偿无效、电流限制有效的情况下，执行本共用程序库 (Library) 时，则会发生错误 "60f5 不能执行力限制 "。请将电流限制置为无效，重力补偿置为有效后再次执行。
- 如在电机电源 OFF 状态下执行，力限制为无效。此外，在力限制过程中将电机电源 OFF 时，力限制也为无效。
- 请通过获取 (TAKEARM) 机械手控制权的任务来执行。尚未获取机械手控制权的情况下，会发生 " 不能获取 21F7 臂信号 (semaphore)" 错误。
- 请在机械手停止状态下执行。当在通过 (Pass) 动作中执行本程序库时，会结束动作。此外，在机械手动作中执行本程序库，发生错误 "600b 机械手正在动作 " 时，请通过 Delay 命令等等待机械手停止后再执行。
- 请正确设定前端负荷设定。当前端负荷设定值与实际的前端负荷不同时，有时会向重力方向掉落。此外，请通过 SetGrvOffset 执行重力补偿修正，能够防止重力掉落。

应用示例

```
CALL SetGrvOffset                    ' 计算重力补偿修正值。
CALL SetFrcCoord(1)                  ' 设定力限制坐标系。
CALL SetFrcLimit(100,0,100,100,100,100)
                                     ' 设定力限制比例。
CALL SetCompFControl                 ' 将力限制有效化。
```

ResetCompControl（共用程序库 (Library)）【Ver.1.4 以上版本】

功　　能

将力限制功能设为无效。（6 轴专用命令）

格　　式

ResetCompControl

说　　明

将力限制功能设为无效。

宏定义

必须有 <pacman.h> 文件。

相关项目

SetCompControl

注意事项

- 请通过获取 (TAKEARM) 机械手控制权的任务来执行。尚未获取机械手控制权的情况下，会发生 " 不能获取 21F7 臂信号 (semaphore)" 错误。
- 请在机械手停止状态下执行。当在通过 (Pass) 动作中执行本程序库时，会结束动作。此外，在机械手动作过程中执行本共用程序库，机械手紧急停止或发生错误 "612* 过电流 " 时，请将 < 设定值 > 设定为 1，或用 Delay 命令等等待机械手停止之后再执行。
- 在本共用程序库 (Library) 执行中瞬时停止，进行步骤回退或程序清零操作时，会发生错误 "60f9 力限制无效操作异常 "。
- 在力限制有效时，通过 SetEralw 变更了偏差允许值时，偏差允许值返回初始值。但是，在执行 ResetCompControl 中发生错误时，有时会不返回初始值。所以在力限制无效后，请通过 ResetEralw 将偏差允许值返回到初始值。
- 有时会发生错误 "608* J* 指令速度限界超程 "。在此种情况下，在执行 ResetCompControl 前请通过 aspChange 将最佳可搬运质量模式变更为 2 或 3，执行之后将最佳可搬运质量设定模式返回到原来值。

应用示例

```
CALL ResetCompControl          ' 将力限制功能无效化。
CALL ResetEralw(0)             ' 将偏差允许值初始化。
```

SetFrcCoord （共用程序库 (Library)）【Ver.1.4 以上版本】

功　能

选择力限制设定坐标系。（6 轴专用命令）

格　式

SetFrcCoord （〈设定值〉）

说　明

选择通过 SetFrcLimit、 SetCompRate 设定的力限制值的坐标系。

设定值为 0 时，选择机械手的底座坐标；设定值为 1 时，选择工具坐标；设定值为 2 时，选择工件坐标。

宏定义

必须有 <pacman.h> 文件。

相关项目

SetFrcLimit、 SetCompRate、 SetFrcCoord、 ResetCompControl

注意事项

- 请通过获取 (TAKEARM) 机械手控制权的任务来执行。尚未获取机械手控制权的情况下，会发生 " 不能获取 21F7 臂信号 (semaphore)" 错误。
- 在力限制过程中，不能执行。如果在力限制中执行本共用程序库 (Library)，则会发生错误 "60fa 正在力限制 "。
- 当 < 设定值 > 为 1，选择了工具坐标时，工具坐标成为力限制有效设定 （执行 SetCompControl）时的工具坐标。在力限制过程中，即使通过 changetool 命令变更工具坐标，力限制坐标也不会变化。
- 将 < 设定值 > 设为 2，选择工件坐标时，工件坐标成为力限制有效设定 （执行 SetCompControl）时的工件坐标。在力限制过程中，即使通过 changework 命令变更工件坐标，力限制坐标也不会变化。
- 控制器电源启动之后，设定值立即被初始化，成为 0 （底座坐标）。

应用示例

```
CALL SetFrcCoord(1)                    ' 将力限制坐标系设定为工具坐标。
Changetool 2                           ' 将工具坐标设定为工具 2。
CALL SetFrcLimit(100,0,100,100,100,100)
                                       ' 设定力限制比例。
CALL SetCompControl                    ' 将工具 2 坐标系的 Y 方向的力限制比例设定为 0%，
                                       ' 将力功能设为有效。
```

SetFrcLimit（共用程序库 (Library)）【Ver.1.4 以上版本】

功　　能

设定力限制比例。（6 轴专用命令）

格　　式

SetFrcLimit（<X 方向限制比例 >、<Y 方向限制比例 >、<Z 方向限制比例 >、<X 转动限制比例 >、<Y 转动限制比例 >、 <Z 转动限制比例 >）

说　　明

设定通过 SetFrcCoord 被设定的坐标系的 X 轴方向、Y 轴方向、Z 轴方向、X 轴转动、Y 轴转动、Z 轴转动的力限制比例。

设定值在 0 ～ 100 范围进行设定。到小数点后 2 位有效。

宏定义

必须有 <pacman.h> 文件。

相关项目

ResetFrcLimit、 SetFrcCoord、 SetCompControl

注意事项

- 请通过获取 (TAKEARM) 机械手控制权的任务来执行。尚未获取机械手控制权的情况下，会发生 " 不能获取 21F7 臂信号 (semaphore)" 错误。
- 在力限制过程中，不能执行。如果在力限制中执行本共用程序库 (Library)，则会发生错误 "60fa 正在力限制 "。
- 控制器电源启动之后，设定值立即被初始化，X 轴方向、Y 轴方向、Z 轴方向、X 轴转动、Y 轴转动、Z 轴转动均为 100。

应用示例

```
CALL SetFrcCoord(1)                    ' 将力限制坐标系设定为工具坐标。
CALL SetFrcLimit(100,0,100,100,100,100)
                                       ' 设定力限制比例。
CALL SetCompControl                    ' 将工具坐标系的 Y 方向的力限制比例设定为 0%，
                                       ' 将力功能设为有效。
```

ResetFrcLimit（共用程序库 (Library)）【Ver.1.4 以上版本】

功　　能

将力限制比例初始化。（6 轴专用命令）

格　　式

ResetFrcLimit

说　　明

将力限制比例初始化。 X 轴方向、 Y 轴方向、 Z 轴方向、 X 轴转动、 Y 轴转动、 Z 轴转动均为 100%。

宏定义

必须有 <pacman.h> 文件。

相关项目

SetFrcLimit

注意事项

- 请通过获取 (TAKEARM) 机械手控制权的任务来执行。尚未获取机械手控制权的情况下，会发生 " 不能获取 21F7 臂信号 (semaphore)" 错误。
- 在力限制过程中，不能执行。如果在力限制中执行本共用程序库 (Library)，则会发生错误 "60fa 正在力限制 "。

应用示例

```
CALL ResetCompControl          ' 将力限制设为无效。
CALL ResetFrcLimit             ' 将力限制比例初始化。
```

SetCompRate（共用程序库 (Library)）【Ver.1.4 以上版本】

功　　能

设定力限制时的柔度的比例。（6 轴专用命令）

格　　式

SetCompRate（<X 方向柔度 >、<Y 方向柔度 >、<Z 方向柔度 >、<X 转动柔度 >、<Y 转动柔度 >、<Z 转动柔度 >）

说　　明

设定通过 SetFrcCoord 被设定的坐标系的 X 轴方向、Y 轴方向、Z 轴方向、X 轴转动、Y 轴转动、Z 轴转动的柔度的比例。

设定值在 0 ～ 100 范围内设定。 0 时为最柔软。到小数点后 2 位有效。

宏定义

必须有 <pacman.h> 文件。

相关项目

ResetCompRate、 SetFrcCoord、 SetCompControl

注意事项

- 请通过获取 (TAKEARM) 机械手控制权的任务来执行。尚未获取机械手控制权的情况下，会发生 " 不能获取 21F7 臂信号 (semaphore)" 错误。
- 在力限制过程中，不能执行。如果在力限制中执行本共用程序库 (Library)，则会发生错误 "60fa 正在力限制 "。
- 控制器电源启动之后，设定值立即被初始化，X 轴方向、Y 轴方向、Z 轴方向、X 轴转动、Y 轴转动、Z 轴转动均为 100。

应用示例

```
CALL SetFrcCoord(1)                 ' 将力限制坐标系设定为工具坐标。
CALL SetCompRate(100,0,100,100,100,100)
                                    ' 设定柔度比例。
CALL SetCompControl                 ' 将工具坐标系的 Y 方向的柔度比例设定为 0%，
                                    ' 将力功能设为有效。
```

ResetCompRate（共用程序库 (Library)）【Ver.1.4 以上版本】

功　　能

将力限制时的柔度的比例初始化。（6 轴专用命令）

格　　式

ResetCompRate

说　　明

将 X 轴方向、 Y 轴方向、 Z 轴方向、 X 轴转动、 Y 轴转动、 Z 轴转动的柔度比例初始化，全部设为 100。

宏定义

必须有 <pacman.h> 文件。

相关项目

SetCompRate

注意事项

- 请通过获取 (TAKEARM) 机械手控制权的任务来执行。尚未获取机械手控制权的情况下，会发生 " 不能获取 21F7 臂信号 (semaphore)" 错误。
- 在力限制过程中，不能执行。如果在力限制中执行本共用程序库 (Library)，则会发生错误 "60fa 正在力限制 "。

应用示例

```
CALL ResetCompControl        ’将力功能设为无效。
CALL ResetCompRate           ’将柔度比例初始化。
```

SetFrcAssist（共用程序库 (Library)）【Ver.1.4 以上版本】

功　　能

设定力限制时的补偿力。（力限制特殊功能共用程序库 (Library)）（6 轴专用命令）

格　　式

SetFrcAssist（<X 方向补偿力 >、<Y 方向补偿力 >、<Z 方向补偿力 >、<X 转动补偿惯性力矩 >、<Y 转动补偿惯性力矩 >、 <Z 转动补偿惯性力矩 >）

说　　明

对 SetFrcCoord 所设定的坐标系的 X 轴方向、Y 轴方向、Z 轴方向、X 轴转动、Y 轴转动、Z 轴转动的补偿力、力矩进行设定。力限制最大值的 10% 为设定最大值。

力设定值的单位是 [N]。力矩设定值的单位是 [Nm]。到小数点后 1 位有效。

宏定义

必须有 <pacman.h> 文件。

相关项目

ResetFrcAssist、 SetFrcCoord、 SetCompControl

注意事项

- 请通过获取 (TAKEARM) 机械手控制权的任务来执行。尚未获取机械手控制权的情况下，会发生 " 不能获取 21F7 臂信号 (semaphore)" 错误。
- 有时机械手会沿施加补偿力、力矩的方向运转。当发生运转时，请降低设定值。
- 控制器电源启动之后，设定值立即被初始化，X 轴方向、Y 轴方向、Z 轴方向、X 轴转动、Y 轴转动、Z 轴转动均为 0。

应用示例

```
CALL SetFrcCoord(1)                    ' 将力限制坐标系设定为工具坐标。
CALL SetFrcAssist(-30,0,0,0,0,0)
                                       ' 在 -X 方向设定 30 [N] 的补偿力。
CALL SetFrcLimit(0,100,100,100,100,100)
                                       ' 设定力限制比例。
CALL SetCompControl                    ' 将力功能设为有效。 X 方向被 0% 力限制，
                                       ' 在 -X 方向施加 30 [N] 的力。
```

ResetFrcAssist（共用程序库 (Library)）【Ver.1.4 以上版本】

功　能

将力限制时的补偿力初始化。（力限制特殊功能共用程序库 (Library)）
（6 轴专用命令）

格　式

ResetFrcAssist

说　明

将通过 SetFrcCoord 被设定的坐标系的 X 轴方向、Y 轴方向、Z 轴方向、X 轴转动、Y 轴转动、Z 轴转动的补偿力、力矩设定为 0。

宏定义

必须有 <pacman.h> 文件。

相关项目

SetFrcAssist

注意事项

请通过获取 (TAKEARM) 机械手控制权的任务来执行。尚未获取机械手控制权的情况下，会发生 "不能获取21F7臂信号 (semaphore)" 错误。

应用示例

```
CALL ResetCompControl          ' 将力限制设为无效。
CALL ResetFrcAssist            ' 将补偿力、力矩设定为 0。
```

SetCompJLimit（共用程序库 (Library)）【Ver.1.4 以上版本】

功　　能

设定力限制时的电流限制值。（力限制特殊功能共用程序库 (Library)）（6 轴专用命令）

格　　式

SetCompJLimit（<J1 电流限制值 >、<J2 电流限制值 >、<J3 电流限制值 >、<J4 电流限制值 >、<J5 电流限制值 >、 <J6 电流限制值 >）

说　　明

设定力限制时的电流限制值。电机额定电流值为 100。通过 SetFrcLimit 将全方向设定为 0 时，电机电流被限制在设定值以下。

设定值在 0 ～ 100 范围内设定。

宏定义

必须有 <pacman.h> 文件。

相关项目

ResetCompJLimit、 SetCompControl

注意事项

- 请通过获取 (TAKEARM) 机械手控制权的任务来执行。尚未获取机械手控制权的情况下，会发生 " 不能获取 21F7 臂信号 (semaphore)" 错误。
- 控制器电源启动之后，设定值立即被初始化，电流限制值全部变为 0。
- 在 SetCompJLimit 上设定比较大的值时，有时机械手会产生振动而发生错误。此时，请通过 SetCompRate、SetDumpRate 调整力限制的柔度。

应用示例

```
CALL SetFrcCoord(1)                    ’将力限制坐标系设定为工具坐标。
CALL SetCompJLimit(30,0,0,0,0,0)
                                       ’将 J1 轴的电流限制值设定为 30%。
CALL SetFrcLimit(100,0,100,100,100,100)
                                       ’设定力限制比例。
CALL SetCompControl                    ’将力功能设为有效。
```

ResetCompJLimit（共用程序库 (Library)）【Ver.1.4 以上版本】

功　　能

将力限制时的电流限制值初始化。（力限制特殊功能共用程序库 (Library)）
（6 轴专用命令）

格　　式

ResetCompJLimit

说　　明

将力限制时的电流限制值进行初始化，所有轴设定为 0。

宏定义

必须有 <pacman.h> 文件。

相关项目

SetCompJLimit、 SetCompControl

注意事项

请通过获取 (TAKEARM) 机械手控制权的任务来执行。尚未获取机械手控制权的情况下，会发生 "不能获取21F7臂信号 (semaphore)" 错误。

应用示例

```
CALL ResetCompControl          ’将力功能设为无效。
CALL ResetCompJLimit           ’将电流限制值初始化。
```

SetCompVMode（共用程序库 (Library)）【Ver.1.4 以上版本】

功　　能

设定力限制时的速度控制模式。（力限制特殊功能共用程序库 (Library)）
（6 轴专用命令）

格　　式

SetCompVMode

说　　明

将执行 SetCompControl 时的力限制速度控制模式置为有效。

宏定义

必须有 <pacman.h> 文件。

相关项目

ResetCompVMode

注意事项

请通过获取 (TAKEARM) 机械手控制权的任务来执行。尚未获取机械手控制权的情况下，会发生 "不能获取21F7臂信号 (semaphore)" 错误。

应用示例

```
CALL SetCompVMode          ' 将力限制速度控制模式设为有效。
CALL SetCompControl        ' 将力限制设为有效。
```

ResetCompVMode（共用程序库 (Library)）【Ver.1.4 以上版本】

功　能

将力限制时的速度控制模式设为无效。（力限制特殊功能共用程序库 (Library)）
（6 轴专用命令）

格　式

ResetCompVMode

说　明

将执行 SetCompControl 时的力限制速度控制模式置为无效。

宏定义

必须有 <pacman.h> 文件。

相关项目

SetCompVMode

注意事项

请通过获取 (TAKEARM) 机械手控制权的任务来执行。尚未获取机械手控制权的情况下，会发生 "不能获取21F7臂信号 (semaphore)" 错误。

应用示例

```
CALL ResetCompVMode          ’将力限制速度控制模式设为无效。
CALL SetCompControl          ’将力限制设为有效。
```

SetCompEralw（共用程序库 (Library)）【Ver.1.4 以上版本】

功　　能

设定力限制时的工具端的位置、姿势偏差允许值。（6 轴专用命令）

格　　式

SetCompEralw（<X 方向偏差允许值 >、<Y 方向偏差允许值 >、<Z 方向偏差允许值 >、<X 转动偏差允许值 >、<Y 转动偏差允许值 >、 <Z 转动偏差允许值 >）

说　　明

设定力限制时的工具端的位置、姿势偏差允许值。X、Y、Z 方向的偏差允许值的单位是 (mm)，X、Y、Z 转动的偏差允许值的单位是 （度）。到小数点后 1 位有效。

宏定义

必须有 <pacman.h> 文件。

相关项目

ResetCompEralw、 SetFrcCoord

注意事项

- 请通过获取 (TAKEARM) 机械手控制权的任务来执行。尚未获取机械手控制权的情况下，会发生 " 不能获取 21F7 臂信号 (semaphore)" 错误。
- 控制器电源起动之后的偏差允许值的初始值是 X、Y、Z 方向为 100 (mm) X、Y、Z 转动为 30（度）。X、Y、 Z 周围是最大值为 175 （度）。进行超过最大值的设定时，会发生 "6003 超过有效数值范围 " 错误。
- 超过工具的位置、姿势偏差所设定的允许值时，会发生 "60f8 力限制时位置偏差过大异常 " 错误。
- 偏差设定坐标系成为通过 SetFrcCoord 被设定的坐标系。例如当设定了 SetFrcCoord(1) 时，SetCompEralw 的设定坐标系成为工具坐标系。

应用示例

```
CALL SetFrcCoord(2)                    ' 将力限制坐标系设为工件坐标。
CALL SetFrcLimit(100,0,100,100,100,100)
                                       ' 将工件坐标的 Y 方向的力限制比例设为 0%
CALL SetCompEralw(10,150,10,5,5,5)
                                       ' 将工件坐标 X、 Z 方向的偏差允许值设为 10 (mm)，
                                       'Y 方向的偏差允许值设为 150 (mm)，
                                       'X、 Y、 Z 转动的偏差允许值设为 5 （度）。
```

ResetCompEralw （共用程序库 (Library)）【Ver.1.4 以上版本】

功　　能

将力限制时的工具端的位置、姿势偏差允许值初始化。（6 轴专用命令）

格　　式

ResetCompEralw

说　　明

将力限制时的工具端的位置、姿势偏差允许值初始化。X、Y、Z 方向的偏差允许值的初始值是 100 (mm)，X、Y、 Z 转动的偏差允许值的初始值是 30 （度）。

宏定义

必须有 <pacman.h> 文件。

相关项目

SetCompEralw

注意事项

请通过获取 (TAKEARM) 机械手控制权的任务来执行。尚未获取机械手控制权的情况下，会发生 "不能获取21F7 臂信号 (semaphore)" 错误。

应用示例

CALL ResetCompEralw

SetDampRate（共用程序库 (Library)）【Ver.1.4 以上版本】

功　　能

设定力限制时的粘性比例。（6 轴专用命令）

格　　式

SetDampRate（<X 方向粘性比例 >、<Y 方向粘性比例 >、<Z 方向粘性比例 >、<X 转动粘性比例 >、<Y 转动粘性比例 >、 <Z 转动粘性比例 >）

说　　明

设定通过 SetFrcCoord 被设定的坐标系的 X 轴方向、Y 轴方向、Z 轴方向、X 轴转动、Y 轴转动、Z 轴转动的粘性比例。

设定值在 0 ～ 100 范围内设定。 0 时的粘性最小。到小数点后 2 位有效。

宏定义

必须有 <pacman.h> 文件。

相关项目

ResetDampRate、 SetFrcCoord、 SetCompRate

注意事项

- 请通过获取 (TAKEARM) 机械手控制权的任务来执行。尚未获取机械手控制权的情况下，会发生 " 不能获取 21F7 臂信号 (semaphore)" 错误。
- 在力限制过程中，不能执行。如果在力限制中执行本共用程序库 (Library)，则会发生错误 "60fa 正在力限制 "。
- 控制器电源启动之后，设定值立即被初始化，X 轴方向、Y 轴方向、Z 轴方向、X 轴转动、Y 轴转动、Z 轴转动均为 100。
- 请不要设定比通过 SetCompRate 设定的柔度比例更小的值。当设定了比柔度比例更小的值时，有时机械手会因振动而停止。
- 执行 SetDampRate 之后，如果设定 SetCompRate，则粘性比例成为 SetCompRate 的设定值。

应用示例

```
CALL SetFrcCoord(1)                    ' 将力限制坐标系设定为工具坐标。
CALL SetCompRate(100,0,100,100,100,100)
                                       ' 设定柔度比例。
CALL SetDampRate(100,20,100,100,100,100)
                                       ' 设定粘性比例。
CALL SetCompControl                    ' 将工具坐标系的 Y 方向的柔度比例设定为 0%，
                                       ' 粘性比例设定为 20%，并将力功能置为有效。
```

ResetDampRate（共用程序库 (Library)）【Ver.1.4 以上版本】

功　　能

将力限制时的粘性比例初始化。（6 轴专用命令）

格　　式

ResetDampRate

说　　明

将 X 轴方向、Y 轴方向、Z 轴方向、X 轴转动、Y 轴转动、Z 轴转动的粘性比例进行初始化，全部设定为 100。

宏定义

必须有 <pacman.h> 文件。

相关项目

SetDampRate

注意事项

- 请通过获取 (TAKEARM) 机械手控制权的任务来执行。尚未获取机械手控制权的情况下，会发生 " 不能获取 21F7 臂信号 (semaphore)" 错误。
- 在力限制过程中，不能执行。如果在力限制中执行本共用程序库 (Library)，则会发生错误 "60fa 正在力限制 "。

应用示例

```
CALL ResetDampRate                    ' 将粘性比例初始化。
```

SetVibControl（共用程序库 (Library)）【Ver.1.4 以上版本】【4 轴 : Ver.1.9 以上版本】

功　　能

置为降低残留振动控制模式。

格　　式

SetVibControl

说　　明

切换至降低残留振动控制模式，降低机械手动作停止时的残留振动。

宏定义

必须有 <pacman.h> 文件。

相关项目

ResetVibControl

注意事项

- 请通过获取 (TAKEARM) 机械手控制权的任务来执行。尚未获取机械手控制权的情况下，会发生 " 不能获取 21F7 臂信号 (semaphore)" 错误。
- 当通过 SetEralw 在变更角度偏差允许值当中执行了 SetVibControl 时，角度偏差允许值被初始化。请再次通过 SetEralw 再次设定角度偏差允许值。
- 降低残留振动控制模式中，动作跟踪迟延增加，轨迹精度降低。请不要在需要轨迹精度的运行中使用。
- 当在通过 (Pass) 动作中执行了 SetVibControl 时，等待动作停止后执行。为此变为结束动作。

应用示例

CALL SetVibControl　　' 设定为降低残留振动控制模式。

ResetVibControl（共用程序库 (Library)）【Ver.1.4 以上版本】【4 轴 : Ver.1.9 以上版本】

功　　能

从降低残留振动控制模式返回到通常控制模式。

格　　式

ResetVibControl

说　　明

从降低残留振动控制模式返回到通常控制模式。

宏定义

必须有 <pacman.h> 文件。

相关项目

ResetVibControl

注意事项

- 请通过获取 (TAKEARM) 机械手控制权的任务来执行。尚未获取机械手控制权的情况下，会发生 " 不能获取 21F7 臂信号 (semaphore)" 错误。
- 当通过 SetEralw 在变更角度偏差允许值当中执行了 ResetVibControl 时，角度偏差允许值被初始化。请再次通过 SetEralw 再次设定角度偏差允许值。
- 当在通过 (Pass) 动作中执行了 ResetVibControl 时，等待动作停止后执行。为此变为结束动作。

应用示例

CALL ResetVibControl　　’设定为通常控制模式。

MotionSkip（共用程序库 (Library)）【Ver.1.5 以上版本】

功　　能

中断执行中的运行命令。（对应 Ver. 1.5 以上版本）

格　　式

MotionSkip

说　　明

执行了 MotionSkip 命令的任务仅中断执行中的动作命令。其他任务的动作不被中断。

宏定义

相关项目

GetJntData、 GetSrvData

注意事项

- 请通过获取 (TAKEARM) 控制权的任务执行。尚未获取控制权的情况下，会发生 " 不能执行 " 的错误。
- 在通过机械手动作任务执行了 MotionSkip 命令时，中断机械手动作命令。在通过附加轴动作任务执行了 MotionSkip 命令时，中断附加轴动作。
  在通过包含机械手和附加轴的臂组的动作任务执行了 MotionSkip 命令时，机械手、附加轴均中断动作。

应用示例

```
defjnt lj1
defsng lf1
move p,P1,next
lj1=GetSrvData(2)                    ’输入各轴的偏差
lf1=ABS(JOINT(2,lj1))                ’抽取 2 轴的偏差
if lf1 > 10000 then
  CALL MotionSkip                    ’如果 2 轴的偏差超出了 10000 (Pulse)，则中断动作命令
endif
```

MotionComp （共用程序库 (Library)）【Ver.1.5 以上版本】

功　能

判断执行动作命令是否完成了。（对应 Ver. 1.5 以上版本）

格　式

MotionComp （< 动作命令完成状态 >）

说　明

当动作命令执行完成时，通过 < 动作指令完成状态 > 返回 1。

用执行了 MotionComp 命令的任务判断执行中的动作命令是否完成执行。不判断其他任务的动作。

当命令是编码器值确认动作时，通过编码器值收敛判断动作完成。其他情况下，当动作指令值是停止时，判断动作完成。

宏定义

相关项目

GetSrvData、 GetJntData、 MotionSkip

注意事项

- 请通过获取 (TAKEARM) 控制权的任务执行。尚未获取控制权的情况下，会发生 " 不能执行 " 的错误。
- 通过机械手动作任务执行了 MotionComp 命令时，判断机械手动作命令的完成。通过附加轴动作任务执行了 MotionComp 命令时，判断附加轴动作的完成。
  通过包含机械手和附加轴的臂组的动作任务执行了 MotionComp 命令时，判断机械手、附加轴同步动作完成。
- 瞬时停止中判断为正在动作。
- 对 < 动作命令完成状态 > 使用局部 (Local) 变量时，请事先将局部 (Local) 变量初始化为 0。

应用示例

```
defint comp=0                    ' 动作命令完成状态的初始化
defjnt lj1
defsng lf1
move p,P1,next
DO
  lj1=GetSrvData(2)              ' 输入各轴的偏差
  lf1=ABS(JOINT(2,lj1))          ' 抽取 2 轴的偏差
  if lf1 > 10000 then
    CALL MotionSkip              ' 如果 2 轴的偏差超出了 10000 (Pulse)，则中断动作命令，
                                 ' 结束循环。
    EXIT DO
  endif
  CALL MotionComp(comp)
LOOP UNTIL comp=1                ' 动作命令循环到完成为止。
```

ArchMove（共用程序库 (Library)）【仅限 Ver.1.9 以上版本的 4 轴机械手】

功　能

实施弧形运行。

格　式

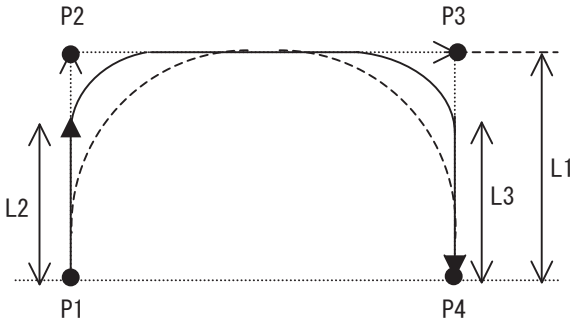
ArchMove （< 目标位置 （P 型变量） >、 <Z 轴动作量 >）

说　明

通过当前位置 P1 和目标位置 P4 以及经由点 P2、 P3 的 4 点进行 P&P 动作。

经由点 P3 是使相对目标位置指定的 <Z 轴动作量 > (L1) 上升后的位置。经由点 P2 是在当前位置的 Z 轴延长线上与经由点 P3 的 Z 轴值相同的位置。

图中的弧形起始位置 (L2)、弧形完成位置 (L3) 通过 SetArchParam 设定。



宏定义

必须有 <pacman.h> 文件。

相关项目

SetArchParam

注意事项

- 请通过获取了机械手控制权 (TAKEARM) 的任务执行。
- 该动作并非是进行位置控制。因为动作速度的不同动作路径会有变化，所以请在动作时注意与周围设备的相互干扰。
- 当前位置与目标位置的 Z 值不同时，不能正常发挥功能。请在与 Z 方向的位置相同的状态下使用。
- 不支持工件坐标系。请在底座坐标系上使用。

应用示例

```
CALL  ArchMove (P10,100)        ' 在目标位置 P10、 Z 轴动作量 100mm 下动作。
                                ' 另外，弧形起始位置、弧形完成位置适用当前的设定值。
                                ' 适用当前的设定值。
```

ArchMoveV （共用程序库 （Library））【仅限 Ver.3.2 以上版本的 6 轴机械手】

功　能

实施弧形运行。

格　式

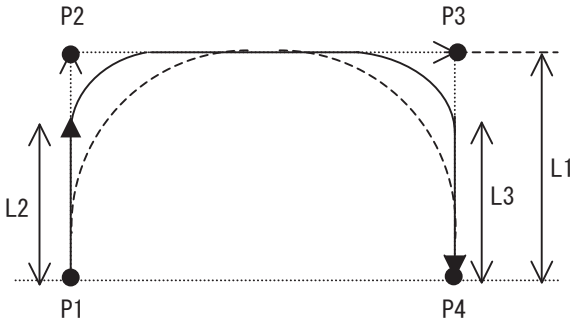
ArchMoveV （< 目标位置 （P 型变量） >、< 接近长度 >）

说　明

在当前位置 P1 和目标位置 P4 以及经由点 P2、 P3 的 4 点进行 P&P 动作。

经由点 P3 是使相对目标位置指定的 < 接近长度 >（L1）上升后的位置。经由点 P2 则是从当前位置仅上升了接近长度 （L1）后的位置。

图中的弧形起始位置 （L2），弧形完成位置 （L3）通过 SetArchParam 设定。



宏定义

需要 <pacman.h> 文件。

相关项目

SetArchParam

注意事项

- 请执行获取了机械手控制权 （TAKEARM）的任务。
- 该动作并非是进行位置控制。因为动作速度的不同动作路径会有变化，所以请在动作时注意与周围设备的相互干涉。
- 不支持工件坐标系。请在底座坐标系上使用。
- 根据动作条件情况，机械手有可能会出现速度异常、加速度异常，以及无法工作的情况。此时，请采取降低速度等方法，重新设置动作条件。

应用示例

```
CALL  ArchMoveV (P10,100)          ' 在目标位置 P10 、接近长度 100mm 处动作。
```

SetArchParam（共用程序库 (Library)）【仅限 Ver.1.9 以上版本的 4 轴机械手】

功　能

弧形运行的在上升动作中开始横向动作的位置（弧形起始位置）及在下降动作中结束横向动作的位置（弧形完成位置）的设定

格　式

SetArchParam （< 弧形起始位置 >、 < 弧形完成位置 >）

说　明

- 可以设定执行 ArchMove 时的弧形的形状。
- 弧形起始位置、弧形完成位置分别以当前位置、目标位置距 Z 轴方向的距离 [mm] 来指定。
- 设定值可以以 0 ～（Z 轴方向的最大可动作距离）的范围，以 1mm 单位来设定。
- 机械手控制器的电源接通后的初始值，弧形起始位置、弧形完成位置均为 0mm。此外，用 SetArchParam 设定的值直到切断机械手控制器的电源为止均有效。

宏定义

必须有 <pacman.h> 文件。

相关项目

ArchMove

注意事项

无效的数值被设定后，不会立即发生错误。在执行 ArchMove 时发生错误。

应用示例

```
CALL  SetArchParam (10,20)          ' 设定使其上升 10mm 之后开始横方向动作，
                                    ' 并在下降完成 20mm 跟前使横方向动作完成。
   .
   .
   .
CALL  ArchMove (P10,100)            ' 在目标位置 P10、 Z 轴动作量 100mm 下动作。
```

SetMonitorCond（共用程序库 (Library)）【Ver.1.5 以上版本】

功　能

设定伺服单轴数据监视功能的监视条件。（对应 Ver. 1.5 以上版本）

格　式

SetMonitorCond （< 轴编号 >、 < 监视器数据 1 选择 >、 < 监视器数据 2 选择 >、
< 抽样间隔 >）

说　明

作为伺服单轴数据监视功能的监视条件，设定监视的轴编号、监视的数据 （2 种）、抽样间隔 (ms)。能够监视的数据有以下 5 种，通过 < 监视数据 1 选择 >、 < 监视数据 2 选择 > 的指定可以一次选择 2 种类型的数据。

< 监视器数据 1 选择 > < 监视器数据 2 选择 >	监视的数据
0	电机速度指令值 [r / min]
1	电机速度当前值 （实际速度） [r / min]
2	电机扭矩指令值 （去除转矩偏移的值）［额定比 %］
3	电机角度偏差 （电机角度指令值 - 电机实际角度） [Pulse]
4	电机电流绝对值 （电机 3 相的电流查出值的绝对值中，最大的值）［额定比 %］

< 抽样间隔 > 以 1 ～ 8ms 的 8 个单位进行设定。

宏定义

相关项目

ClearSrvMonitor、 StartSrvMonitor、 StopSrvMonitor

注意事项

- (1) 执行监视器开始命令 (StartSrvMonitor) 后，若执行本命令，则发生 " 不能执行 6001" 的错误。请务必在监视开始之前设定条件。
- 当对轴编号、数据类型、抽样间隔设定了错误数值时，则会发生错误 " 范围外 "。在发生错误时请确认指定值。

应用示例

```
CALL SetMonitorCond(7,0,3,4)        ' 将 7 轴的速度指令值、角度偏差进行每 4ms 输入一次的设定。

CALL StartSrvMonitor                ' 开始数据监视。
```

StartSrvMonitor（共用程序库 (Library)）【Ver.1.5 以上版本】

功　　能

开始伺服单轴数据监视。（对应 Ver. 1.5 以上版本）

格　　式

StartSrvMonitor

说　　明

开始伺服单轴数据监视。执行 StartSrvMonitor 后到执行 StopSrvMonitor 为止，最多获取 1250 个的伺服数据。

宏定义

相关项目

ClearSrvMonitor、 SetMonitorCond、 StopSrvMonitor

注意事项

- 当从执行 StartSrvMonitor （监视开始）到执行 StopSrvMonitor （监视结束）为止的数据数不足 1250 个时，监视从开始到结束的数据。当超出 1250 个时，监视结束之前的 1250 个数据。
- 当在从监视开始到结束之间发生错误等且电机电源 OFF 时，监视电机 OFF 后的 400 个和电机 OFF 前，最多 850 个的数据。
- 电机 OFF 状态时不能监视数据。请在电机 ON 状态下执行。

应用示例

```
CALL SetMonitorCond(7,0,3,4)        ' 将 7 轴的速度指令值、角度偏差进行每 4ms 输入一次的设定。

CALL StartSrvMonitor                ' 开始数据监视。
```

StopSrvMonitor （共用程序库 (Library)）【Ver.1.5 以上版本】

功　　能

结束伺服单轴数据监视。（对应 Ver. 1.5 以上版本）

格　　式

StopSrvMonitor

说　　明

结束伺服单轴数据监视。在 StartSrvMonitor 执行后到执行 StopSrvMonitor 为止，最多获取 1250 个的伺服数据。

宏定义

相关项目

ClearSrvMonitor、 SetMonitorCond、 StartSrvMonitor

注意事项

- 当从执行 StartSrvMonitor （监视开始）到执行 StopSrvMonitor （监视结束）为止的数据数不足 1250 个时，监视从开始到结束的数据。当超出 1250 个时，监视结束之前的 1250 个数据。
- 当在从监视开始到结束之间发生错误等情况且电机电源 OFF 终了时，最多监视电机 OFF 后的 400 个和电机 OFF 前的 850 个的数据。

应用示例

CALL StopSrvMonitor ’结束数据监视。

ClearSrvMonitor（共用程序库 (Library)）【Ver.1.5 以上版本】

功　　能

初始化伺服单轴数据监视的获取数据的指针 (Pointer)。（对应 Ver. 1.5 以上版本）

格　　式

ClearSrvMonitor

说　　明

初始化伺服单轴数据监视的获取数据的指针 (Pointer)，并开始新的 1250 个数据的获取。

宏定义

相关项目

ClearSrvMonitor、 SetMonitorCond、 StartSrvMonitor

注意事项

当从执行 ClearSrvMonitor （清空）到执行 StopSrvMonitor （监视结束）为止的数据数不足 1250 个时，监视从清空到结束的数据。当超出 1250 个时，监视结束之前的 1250 个数据。

应用示例

```
CALL StartSrvMonitor            ' 开始数据监视。
    .
    .
    .
CALL ClearSrvMonitor            ' 清空 StartSrvMonitor 之后的数据。
    .
    .
    .
CALL StopSrvMonitor             ' 结束数据监视。
                                ' 监视从 ClearSrvMonitor 至 StopSrvMonitor 之间的数据。
```

xdWAITSPLINE （共用程序库 (Library)）【Ver.2.3 以上版本】

功　能

等待通过自由曲线所指定的通过点。

格　式

xdWAITSPLINE （< 通过点编号 >、< 等待条件 >）

说　明

针对执行中的自由曲线动作，等待直到通过由 < 通过点编号 > 所指定的点。

在 < 等待条件 > 上指定了 0 时，等待直到通过由指令值所指定的通过点。

指定了 0 以外时，等待直到通过由编码器值所指定的通过点。

相关项目

MOVE、 SETSPLINEPOINT

注意事项

当在未执行自由曲线状态下 CALL 了 xdWAITSPLINE 时，不成为等待状态。

此外，当在通过由 < 通过点编号 > 所指定的通过点之后 CALL 了 xdWAITSPLINE 时，不成为等待状态。

应用示例

```
PROGRAM PRO1
TAKEARM
CLRSPLINEPOINT 5              ' 在轨道编号 5 号的自由曲线上将通过点全部清空。
SETSPLINEPOINT 5, P4          ' 在轨道编号 5 号的自由曲线上将 P4 作为第 1 号的通过。
SETSPLINEPOINT 5, P1          ' 在轨道编号 5 号的自由曲线上将 P1 作为第 2 号的通过。
SETSPLINEPOINT 5, J5          ' 在轨道编号 5 号的自由曲线上将 J5 作为第 3 号的通过。
MOVE S, 5, NEXT               ' 执行通过 P4、 P1 向 J5 移动的自由曲线动作。
CALL xdWAITSPLINE(1,1)        ' 机械手等待通过第 1 个通过点 (P4)。
SET IO[240]                   ' 打开端口 240。
CALL xdWAITSPLINE(2,1)        ' 机械手等待通过第 2 个通过点 (P1)。
RESET IO[240]                 ' 关闭端口 240。
```

xdSPLPASSNUM（共用程序库 (Library)）【Ver.2.3 以上版本】

功　　能

获取自由曲线已经通过的通过点编号。

格　　式

xdSPLPASSNUM （< 通过点编号 >）

说　　明

针对执行中的自由曲线动作，指令值将已经通过的通过点编号设定在 < 通过点编号 > 上。

相关项目

MOVE、 SETSPLINEPOINT

应用示例

```
PROGRAM PRO1
TAKEARM
CLRSPLINEPOINT 5              ' 在轨道编号 5 号的自由曲线上将通过点全部清空。
SETSPLINEPOINT 5, P4          ' 在轨道编号 5 号的自由曲线上将 P4 作为第 1 号的通过。
SETSPLINEPOINT 5, P1          ' 在轨道编号 5 号的自由曲线上将 P1 作为第 2 号的通过。
SETSPLINEPOINT 5, J5          ' 在轨道编号 5 号的自由曲线上将 J5 作为第 3 号的通过。
MOVE S, 5, NEXT               ' 执行通过 P4、 P1 向 J5 移动的自由曲线动作。
DELAY 500
CALL xdSPLPASSNUM(I1)         ' 将已经通过的通过点编号设定在 I1。
```

xdSPLClrTakeArm（共用程序库 (Library)）【Ver.2.3 以上版本】

功　　能

变更在 TakeArm 时执行的自由曲线的通过点删除处理的有效、无效。

格　　式

xdSPLClrTakeArm （< 设定值 >）

说　　明

当 < 设定值 > 为 0 时，在 TakeArm 时不删除自由曲线的通过点。

当 < 设定值 > 为 0 以外时，在 TakeArm 时删除自由曲线的通过点。

相关项目

CLRSPLINEPOINT、 SETSPLINEPOINT

注意事项

控制器电源接通后的初始状态是在 TakeArm 时删除自由曲线的通过点。因此，当在用与自由曲线动作程序不同的其他的程序登录自由曲线的通过点（用初始化程序登录）时，请通过 CALL xdSPLClrTakeArm (0) 进行设定，使其在 TakeArm 时不删除自由曲线的通过点。

应用示例

```
PROGRAM INITIAL
CLRSPLINEPOINT 5              ' 在轨道编号 5 号的自由曲线上将通过点全部清空。
SETSPLINEPOINT 5, P4          ' 在轨道编号 5 号的自由曲线上将 P4 作为第 1 号的通过。
SETSPLINEPOINT 5, P1          ' 在轨道编号 5 号的自由曲线上将 P1 作为第 2 号的通过。
SETSPLINEPOINT 5, J5          ' 在轨道编号 5 号的自由曲线上将 J5 作为第 3 号的通过。
CALL xdSPLClrTaleArm(0)       ' 在 TakeArm 时不删除自由曲线的通过点。
```

SetHighPathAccuracy（共用程序库 (Library)）【Ver.2.61 以上版本】

功　　能

向提高 CP 动作 （直线、圆弧、自由曲线）时的动作轨迹的高轨迹控制功能过渡。

格　　式

SetHighPathAccuracy

说　　明

（1）高轨迹控制功能可以使机械手在提高动作轨迹的基础上动作。

（2）用紧接着 SetHighPathAccuracy 的动作指令，可使高轨迹控制功能变为有效，
用 ResetHighPathAccuracy 则可解除高轨迹控制功能。

宏定义

必须有 <pacman.h> 文件。

相关项目

ResetHighPathAccuracy

注意事项

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

应用示例

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

ResetHighPathAccuracy （共用程序库 （Library））【Ver.2.61 以上版本】

功　　能

使提高 CP 动作 （直线、圆弧、自由曲线）时的动作轨迹的高轨迹控制功能无效。

格　　式

ResetHighPathAccuracy

说　　明

（1）高轨迹控制功能可以使机械手在提高动作轨迹的基础上动作。

（2）用紧接着 SetHighPathAccuracy 的动作指令，可使高轨迹控制功能变为有效，
用 ResetHighPathAccuracy 则可解除高轨迹控制功能。

宏定义

必须有 <pacman.h> 文件。

相关项目

SetHighPathAccuracy

注意事项

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

应用示例

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

SetSingularAvoid （共用程序库 (Library)）【Ver.2.61 以上版本】

功　能

使特异点回避功能有效或无效。（6 轴专用）

格　式

SetSingularAvoid （〈有效或无效设定〉）

说　明

在 〈有效或无效设定〉指定 1 时，特异点回避功能将有效，指定 0 时将无效。

如果使特异点回避功能有效，在直线、圆弧、自由曲线等的 CP 插补动作中， 5 轴通过 0 度附近时，可以抑制 4 轴大幅度旋转的举动。

宏定义

必须有 <pacman.h> 文件。

注意事项

- 4 轴机械手和 5 轴机械手不能使用。
- PTP 插补动作时不起作用。
- 功能有效时，动作中的轨道可能会因动作条件而大幅度偏差。
- 特异点回避功能无法在旧版卡上发挥功能。

应用示例

```
PROGRAM PRO1
TakeArm                          ’获得控制权
MOVE P,P1                        ’通过 PTP 插补向 P1 移动。
CALL  SetSingularAvoid (1)       ’使特异点回避功能有效。
MOVE L,P2                        ’通过直线插补向 P2 移动。
CALL  SetSingularAvoid (0)       ’使特异点回避功能无效。
END
```

SetCollisionJnt（共用程序库 (Library)）【Ver.2.61 以上版本】

功　能

使指定轴的碰撞检测有效。

格　式

SetCollisionJnt （〈轴编号〉）

说　明

使指定轴的碰撞检测有效。

宏定义

必须有 <pacman.h> 文件。

相关项目

ResetCollisionJnt

注意事项

- 请指定机械手轴的轴编号。指定机械手轴以外时，会发生摘指定轴超出范围攡错误。
- 在力限制功能有效时，不能执行。
- 碰撞检测功能可在下述控制器版本中发挥功能。
  当您使用的是低于以下控制器版本的旧版项目数据时，请参照编程手册 I 中的 " 碰撞检测功能 "。
  - VS-G 系列：Ver.2.61 以后
  - VS-*** 系列：Ver.3.20 以后
  - VM-G、 VP-G 系列：Ver.2.70 以后
  - HM-G、 HS-G 系列：Ver.2.80 以后
- 碰撞检测功能无法在旧版卡上发挥功能。

应用示例

```
TAKEARM                    ' 获得机械手控制权。
CALL SetCollisionJnt(2)    ' 使 2 轴的碰撞检测有效。
MOVE P, P1,P2              ' 在 2 轴的碰撞检测有效的状态下，
                           ' 从 P1 点向 P2 点进行 PTP 移动。
CALL ResetCollisionJnt(2)  ' 使 2 轴的碰撞检测无效。
MOVE P, P2,P3              ' 在 2 轴的碰撞检测无效的状态下，
                           ' 从 P2 点向 P3 点进行 PTP 移动。
```

ResetCollisionJnt（共用程序库 (Library)）【Ver.2.61 以上版本】

功　　能

使指定轴的碰撞检测无效。

格　　式

ResetCollisionJnt （〈轴编号〉）

说　　明

使指定轴的碰撞检测无效。

宏定义

必须有 <pacman.h> 文件。

相关项目

SetCollisionJnt

注意事项

- 请指定机械手轴的轴编号。指定机械手轴以外时，会发生摄指定轴超出范围攡错误。
- 碰撞检测功能可在下述控制器版本中发挥功能。
  当您使用的是低于以下控制器版本的旧版项目数据时，请参照编程手册Ⅰ中的 " 碰撞检测功能 "。
  - VS-G 系列：Ver.2.61 以后
  - VS-*** 系列：Ver.3.20 以后
  - VM-G、 VP-G 系列：Ver.2.70 以后
  - HM-G、 HS-G 系列：Ver.2.80 以后
- 碰撞检测功能无法在旧版卡上发挥功能。

应用示例

```
TAKEARM                     ' 获得机械手控制权。
CALL SetCollisionJnt(2)     ' 使 2 轴的碰撞检测有效。
MOVE P, P1,P2               ' 在 2 轴的碰撞检测有效的状态下，
                            ' 从 P1 点向 P2 点进行 PTP 移动。
CALL ResetCollisionJnt(2)   ' 使 2 轴的碰撞检测无效。
MOVE P, P2,P3               ' 在 2 轴的碰撞检测无效的状态下，
                            ' 从 P2 点向 P3 点进行 PTP 移动。
```

GetCollisionForce（共用程序库 (Library)）【Ver.2.61 以上版本】

功　能

获得外力最大值。

格　式

GetCollisionForce （〈J 型变量〉）

说　明

将外力最大值设定为 〈J 型变量〉。

相关项目

ClearCollisionForce

注意事项

- 只有电机为 ON 状态的碰撞检测有效设定轴，才按手动模式和自动模式计算外力最大值。
- 自动模式时，设定自动模式时的外力最大值；手动模式时、教导检查模式时，设定手动模式时的外力最大值。
- 碰撞检测功能可在下述控制器版本中发挥功能。
  当您使用的是低于以下控制器版本的旧版项目数据时，请参照编程手册 I 中的 " 碰撞检测功能 "。
  - VS-G 系列：Ver.2.61 以后
  - VS-*** 系列：Ver.3.20 以后
  - VM-G、 VP-G 系列：Ver.2.70 以后
  - HM-G、 HS-G 系列：Ver.2.80 以后
- 碰撞检测功能无法在旧版卡上发挥功能。

应用示例

```
TAKEARM                      ' 获得机械手控制权。
CALL ClearCollisionForce     ' 将外力最大值初始化。
CALL SetCollisionJnt(2)      ' 使 2 轴的碰撞检测有效。
MOVE P, P1,P2                ' 从 P1 点向 P2 点进行 PTP 移动。
CALL GetCollisionForce(J1)   ' 将从 P1 点向 P2 点移动时的外力最大值设定为 J1。

CALL ClearCollisionForce     ' 将外力最大值初始化。
MOVE P, P2,P3                ' 从 P2 点向 P3 点进行 PTP 移动。
CALL GetCollisionForce(J2)   ' 将从 P2 点向 P3 点移动时的外力最大值设定为 J2。
```

ClearCollisionForce（共用程序库 (Library)）【Ver.2.61 以上版本】

功　　能

将外力最大值初始化。

格　　式

ClearCollisionForce

说　　明

使外力最大值返回到 0 （初始值）。

相关项目

GetCollisionForce

注意事项

- 本指令根据全部对象轴的手动模式时和自动模式时的外力最大值进行初始化。
- 碰撞检测功能可在下述控制器版本中发挥功能。
  当您使用的是低于以下控制器版本的旧版项目数据时，请参照编程手册Ⅰ中的 " 碰撞检测功能 "。
  - VS-G 系列：Ver.2.61 以后
  - VS-*** 系列：Ver.3.20 以后
  - VM-G、 VP-G 系列：Ver.2.70 以后
  - HM-G、 HS-G 系列：Ver.2.80 以后
- 碰撞检测功能无法在旧版卡上发挥功能。

应用示例

```
TAKEARM                     ' 获得机械手控制权。
CALL SetCollisionJnt(2)     ' 使 2 轴的碰撞检测有效。
MOVE P, P1,P2               ' 从 P1 点向 P2 点进行 PTP 移动。
CALL ClearCollisionForce    ' 将外力最大值初始化。
MOVE P, P2,P3               ' 从 P2 点向 P3 点进行 PTP 移动。
CALL GetCollisionForce(J2)  ' 将从 P2 点向 P3 点移动时的外力最大值设定为 J2。
```

SetCollisionLevel（共用程序库 (Library)）【Ver.2.61 以上版本】

功　能

设定指定轴的碰撞检测等级。

格　式

SetCollisionLevel （〈轴编号、等级〉）

说　明

设定指定轴的碰撞检测等级。标准等级为 100。将等级设定得较小时，碰撞检测灵敏度将会变高。将等级设定得较大时，碰撞检测灵敏度将会变低。

宏定义

必须有 <pacman.h> 文件。

相关项目

SetCollisionJnt、 ResetCollisionJnt

注意事项

- 请指定机械手轴的轴编号。指定机械手轴以外时，会发生撾指定轴超出范围攡错误。
- 标准等级是正常作业时不会发生碰撞误检的等级。正确设定机械手的负荷设定值，可以降低等级，提高碰撞检测灵敏度。
- 误检碰撞时，请提高降低等级，降低碰撞检测灵敏度。
- 碰撞检测功能可在下述控制器版本中发挥功能。
  当您使用的是低于以下控制器版本的旧版项目数据时，请参照编程手册 I 中的 " 碰撞检测功能 "。
  - VS-G 系列：Ver.2.61 以后
  - VS-*** 系列：Ver.3.20 以后
  - VM-G、 VP-G 系列：Ver.2.70 以后
  - HM-G、 HS-G 系列：Ver.2.80 以后
- 碰撞检测功能无法在旧版卡上发挥功能。

应用示例

```
TAKEARM                      ' 获得机械手控制权。
CALL SetCollisionJnt(3)      ' 使 3 轴的碰撞检测有效。
CALL  SetCollisionLevel(3,80)
                             ' 将 3 轴的碰撞检测等级设定为 80。
MOVE P, P1,P2                ' 在 3 轴的碰撞检测敏感的状态下，
                             ' 从 P1 点向 P2 点进行 PTP 移动。
CALL  SetCollisionLevel(3,120)
                             ' 将 3 轴的碰撞检测等级设定为 120。
MOVE P, P2,P3                ' 在 3 轴的碰撞检测钝感的状态下，
                             ' 从 P2 点向 P3 点进行 PTP 移动。
```

SetExtForceDetect（共用程序库 (Library)）【Ver.2.61 以上版本】

功　能

设定外力检测的有效或无效。

格　式

SetExtForceDetect （〈设定值〉）

说　明

〈设定值〉为 1 时，外力检测有效。为 0 时无效。

宏定义

必须有 <pacman.h> 文件。

相关项目

SetCollisionJnt、 GetCollisionForce

注意事项

- 请将设定值设定为 0 或 1。设定其他值时会发生错误。
- 默认值为 1 （有效）。即使变更为无效，在切换手动和自动模式时，或重新接通电源之后，将返回到 1。
- 本指令的目的是指定是否更新机械手存储的外力最大值，而非设定碰撞检测的有效或无效。设定为无效时，即使机械手检测出超过存储的外力最大值的外力，也不会更新外力最大值。请在确认特定区间的外力最大值等情况下使用。
- 碰撞检测功能可在下述控制器版本中发挥功能。
  当您使用的是低于以下控制器版本的旧版项目数据时，请参照编程手册 I 中的 " 碰撞检测功能 "。
  - VS-G 系列：Ver.2.61 以后
  - VS-*** 系列：Ver.3.20 以后
  - VM-G、 VP-G 系列：Ver.2.70 以后
  - HM-G、 HS-G 系列：Ver.2.80 以后
- 碰撞检测功能无法在旧版卡上发挥功能。

应用示例

```
TAKEARM                      ' 获得机械手控制权。
CALL SetExtForceDetect(0)    ' 使外力检测无效。
CALL ClearCollisionForce     ' 将外力最大值初始化。
CALL SetCollisionJnt(3)      ' 使 3 轴的碰撞检测有效。
MOVE P, P1,P2                ' 从 P1 点向 P2 点进行 PTP 移动。
CALL SetExtForceDetect(1)    ' 使外力检测有效。
MOVE P, P2,P3                ' 从 P2 点向 P3 点进行 PTP 移动。
CALL GetCollisionForce(J2)   ' 将从 P2 点向 P3 点移动时的外力最大值设定为 J2。
```

第 8 章　创建 TP 简易操作盘画面

8.1　操作盘画面创建示例程序的使用方法

在创建操作盘画面时，作为雏形的示例程序收录于 WINCAPS Ⅲ中安装盘内的 Samples 文件夹内。
要将其引入时，请通过程序 (P) → 程序引入 (I) 选择需要的示例程序。

8.2　操作盘画面创建示例程序

将被 WINCAPS Ⅲ准备的示例程序进行如下说明。

Select_Screen（示例程序）【Ver.1.7 以上版本】

功　能

是通用操作盘画面切换处理的使用示例。

说　明

该程序对通用操作盘功能的画面的切换进行控制。
监视通过各个操作盘画面被定义的画面切换用按钮的 I / O 编号，并在 I / O 打开时，显示该操作盘画面。
另外，该程序因为是示例程序，所以在实际使用时需要将程序进行变更。

宏定义

需要 <button.h>。

相关项目

Build_Lamp_Screen、 Build_Man_Screen、 Build_Plan_Screen

Build_Lamp_Screen（示例程序）【Ver.1.7 以上版本】

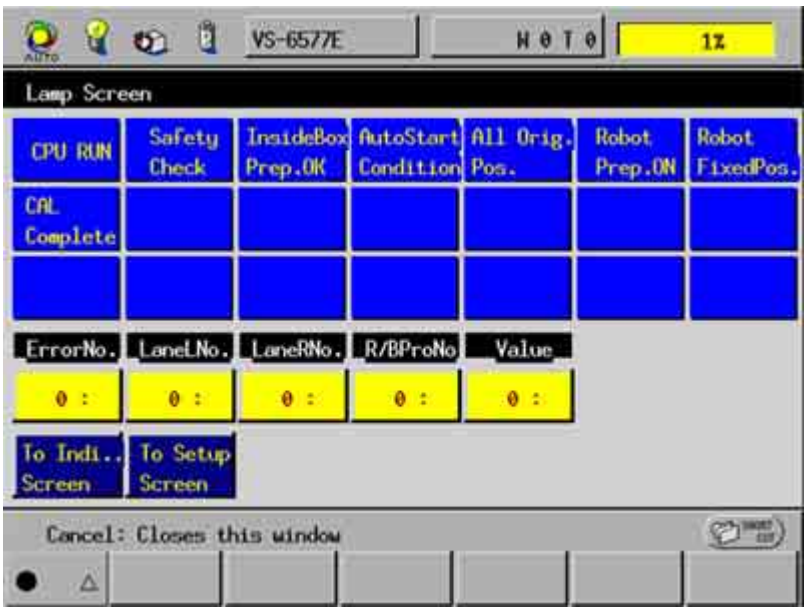
功　　能

创建如下的指示灯画面。（作为自变量 (argument) 需要画面编号）

说　　明

该程序调用各种按钮创建程序，创建如下的指示灯画面。

另外，该程序因为是示例程序，所以在实际使用时需要将程序进行变更。



宏定义

需要 <button.h>。

相关项目

Lamp_pl_btns、 Lamp_variable_btns、 Lamp_pb_btns

Build_Man_Screen（示例程序）【Ver.1.7 以上版本】

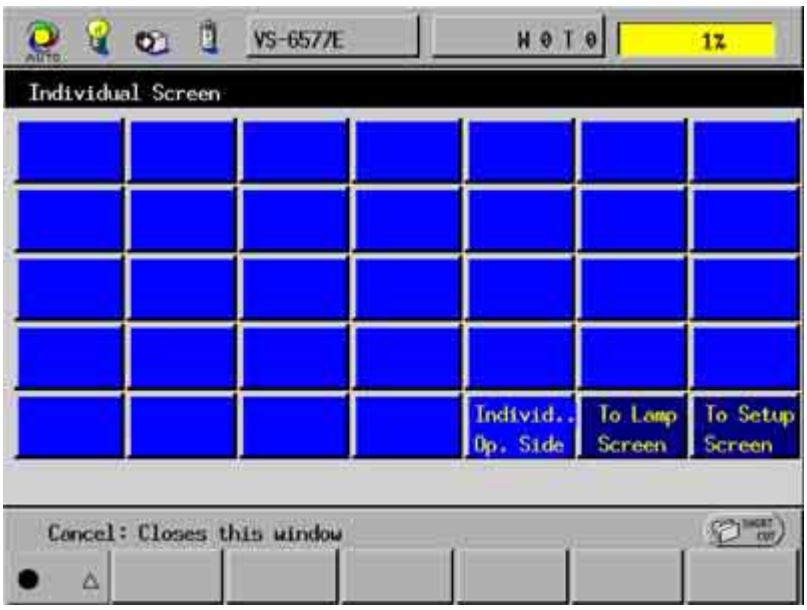
功　　能

创建如下的各个画面。（作为自变量 (argument) 需要画面编号）

说　　明

该程序调用 PB 按钮创建程序，创建如下的各个画面。

另外，该程序因为是示例程序，所以在实际使用时需要将程序进行变更。



宏定义

需要 <button.h>。

相关项目

Man_pb_btns

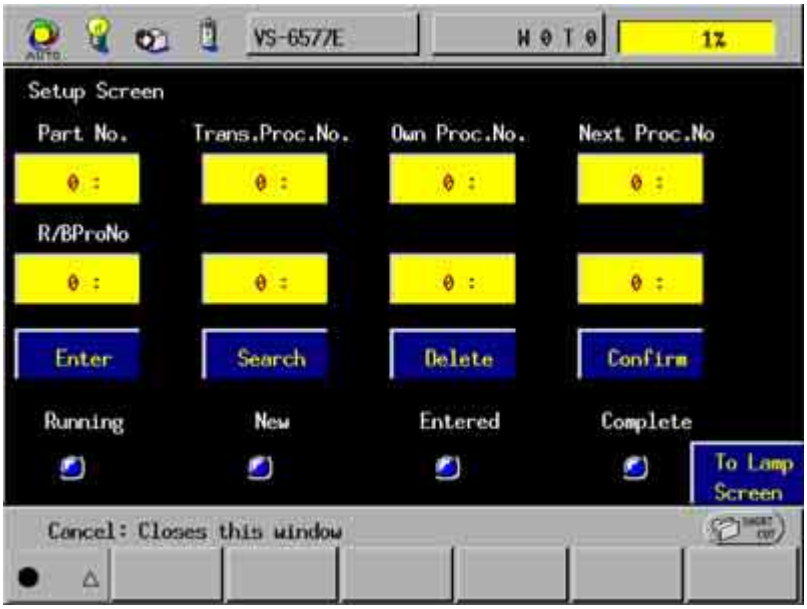
Build_Plan_Screen（示例程序）【Ver.1.7 以上版本】

功　能

创建如下的配置画面。（作为自变量 (argument) 需要画面编号）

说　明

该程序调用各种按钮创建程序，创建如下的配置画面。另外，该程序因为是示例程序，所以在实际使用时需要将程序进行变更。



宏定义

需要 <button.h>。

相关项目

Plan_variable_btns、 Plan_pb_btns、 Plan_pl_btns、 Plan_pb_singlebtn

Lamp_pl_btns（示例程序）【Ver.1.7 以上版本】

功　　能

在指定画面上创建指示灯 (LED)。（作为自变量 (argument) 需要画面编号）

说　　明

通过输入按钮创建所需要的各个参数生成指示灯按钮。

另外，该程序因为是示例程序，所以在实际使用时需要将程序进行变更。

宏定义

需要 <button.h>。

相关项目

make_LED

Lamp_variable_btns（示例程序）【Ver.1.7 以上版本】

功　　能

在指定画面上创建变量按钮 (数据显示 BOX)。（作为自变量 (argument) 需要画面编号）

说　　明

通过输入按钮创建所需要的各个参数生成变量按钮。

另外，该程序因为是示例程序，所以在实际使用时需要将程序进行变更。

宏定义

需要 <button.h>。

相关项目

make_PARAM_BOX、 make_LABEL

Lamp_pb_btns（示例程序）【Ver.1.7 以上版本】

功　能

在指定画面上创建 PB 按钮 （画面切换按钮）。（作为自变量 (argument) 需要画面编号）

说　明

通过输入按钮创建所需要的各个参数生成 PB 按钮。

另外，该程序因为是示例程序，所以在实际使用时需要将程序进行变更。

宏定义

需要 <button.h>。

相关项目

make_PB

Man_pb_btns（示例程序）【Ver.1.7 以上版本】

功　能

在指定画面上创建 PB 按钮。（作为自变量 (argument) 需要画面编号）

说　明

通过输入按钮创建所需要的各个参数生成 PB 按钮。

另外，该程序因为是示例程序，所以在实际使用时需要将程序进行变更。

宏定义

需要 <button.h>。

相关项目

make_PB

Plan_variable_btns（示例程序）【Ver.1.7 以上版本】

功　　能

在指定画面上创建变量输入按钮 （配置参数输入 BOX）。（作为自变量 (argument) 需要画面编号）

说　　明

通过输入按钮创建所需要的各个参数生成变量按钮。

另外，该程序因为是示例程序，所以在实际使用时需要将程序进行变更。

宏定义

需要 <button.h>。

相关项目

make_PARAM_BOX、 make_LABEL

Plan_pb_btns（示例程序）【Ver.1.7 以上版本】

功　　能

在指定画面上创建 PB 按钮 （数据登录处理按钮）。（作为自变量 (argument) 需要画面编号）

说　　明

通过输入按钮创建所需要的各个参数生成 PB 按钮。

另外，该程序因为是示例程序，所以在实际使用时需要将程序进行变更。

宏定义

需要 <button.h>。

相关项目

make_PB

Plan_pl_btns（示例程序）【Ver.1.7 以上版本】

功　　能

在指定画面上创建指示灯按钮。（作为自变量 (argument) 需要画面编号）

说　　明

通过输入按钮创建所需要的各个参数生成指示灯按钮。

另外，该程序因为是示例程序，所以在实际使用时需要将程序进行变更。

宏定义

需要 <button.h>。

相关项目

make_LED、 make_LABEL

Plan_pb_singlebtn（示例程序）【Ver.1.7 以上版本】

功　　能

在指定画面上仅创建一个按钮。（作为自变量 (argument) 需要画面编号）

说　　明

通过输入按钮创建所需要的各个参数生成 PB 按钮。

另外，该程序因为是示例程序，所以在实际使用时需要将程序中的参数进行变更。

宏定义

需要 <button.h>。

相关项目

single_button_set

8.3 用于创建操作盘画面的共用程序库 (Library)

对用于在多功能教导器的屏幕上创建操作盘画面的共用程序库 (Library) 进行如下说明。

make_PB（共用程序库 (Library)）【Ver.1.7 以上版本】

功　能

创建 PB 按钮。

格　式

make_PB

说　明

根据被指定的参数、显示字符、按钮数，在画面上设定 PB 按钮。

宏定义

需要 <button.h>。

应用示例

call make_PB(sysprm(),pb_title(),PB_NUM)

make_LED（共用程序库 (Library)）【Ver.1.7 以上版本】

功　能

创建 LED 按钮。

格　式

make_LED

说　明

根据被指定的参数、显示字符、按钮数，在画面上设定 LED 按钮。

宏定义

需要 <button.h>。

应用示例

call make_LED(sysprm(),pl_title(),MAX_PL)

make_LABEL（共用程序库 (Library)）【Ver.1.7 以上版本】

功　能

创建标题 （标签）。

格　式

make_LABEL

说　明

根据被指定的参数、显示字符、按钮数，在画面上设定标题。

宏定义

需要 <button.h>。

应用示例

call make_LABEL(sysprm(),pl_title(),PL_NUM)

make_PARAM_BOX（共用程序库 (Library)）【Ver.1.7 以上版本】

功　能

创建变量按钮 （输入& 显示 BOX）。

格　式

make_PARAM_BOX

说　明

根据被指定的参数、按钮数，在画面上设定变量按钮。

宏定义

需要 <button.h>。

应用示例

call make_PARAM_BOX(sysprm(),VBTN_NUM)

single_button_set（共用程序库 (Library)）【Ver.1.7 以上版本】

功　能

仅创建一个按钮。

格　式

single_button_set

说　明

通过在被指定的按钮编号上设定参数 , 设定一个按钮。

宏定义

需要 <button.h>。

应用示例

call single_button_set(btn_no,prm())

set_button_param（共用程序库 (Library)）【Ver.1.7 以上版本】

功　能

指定按钮属性 （类型、颜色、形状等）。

格　式

set_button_param（< 显示按钮数 >、< 前头按钮数 >、< 显示按钮类型 >、< 按钮文字颜色 >、< 按钮背景颜色 >、< 按钮使用标志 (Flag)>、< 按钮可见标志 (Flag)>、< 按钮对应 I 变量编号 >、< 按钮对应 I / O 编号 >、< 按钮显示面板编号 >、 < 按钮文字位置 >）

说　明

该程序是指定按钮显示所需的参数的程序。这里总计需要指定 11 个参数。

宏定义

需要 <button.h>。

应用示例

call set_button_param(BUTTON_NUM,B_START_NUM,BUTTON_TYPE,
CHAR_COLOR,BKGRND_COLOR,BUTTON_USE,BUTTON_VIS,B_VARIABLE_NUM,
B_ASSIGN_IO, PANEL_NO, CAP_POS)

arrange_button_size（共用程序库 (Library)）【Ver.1.7 以上版本】

功　能

指定画面上的按钮配置 （位置、大小等）。

格　式

arrange_button_size（< 显示按钮数 >、< 按钮显示开始 X 坐标 >、< 按钮显示开始 Y 坐标 >、< 按钮宽度 >、< 按钮高度 >、< 前头按钮编号 >、< 按钮横间隙 >、< 按钮纵间隙 >、< 按钮显示面板编号 >）

说　明

该程序是将按钮配置所需的参数作为自变量 (argument)，并以此决定画面上的按钮位置的程序。

虽然与 arrange_button_pos 几乎是相同的处理，但在用于按钮创建的自变量 (argument) 中，

是指定按钮大小的类型的处理。

宏定义

需要 <button.h>。

应用示例

```
call arrange_button_size(BUTTON_NUM,ORIGIN_PX,ORIGIN_PY,BUTTON_W,
    BUTTON_H,B_START_NUM,H_GAP,V_GAP,PANEL_NO)
```

arrange_button_pos（共用程序库 (Library)）【Ver.1.7 以上版本】

功　能

指定画面上的按钮配置 （位置、大小等）。

格　式

arrange_button_pos（< 显示按钮数 >、< 按钮显示开始 X 坐标 >、< 按钮显示开始 Y 坐标 >、< 画面横方向的按钮数 >、< 画面纵方向的按钮数 >、< 前头按钮编号 >、< 按钮横间隙 >、< 按钮纵间隙 >、< 按钮显示面板编号 >）

说　明

该程序是将按钮配置所需的参数作为自变量 (argument)，并以此决定画面上的按钮位置的程序。

虽然与 set_button_size 几乎是相同的处理，但在用于按钮创建的自变量 (argument) 中，是指定画面纵横方向上按钮个数的类型的处理。

宏定义

需要 <button.h>。

应用示例

```
call arrange_button_size(BUTTON_NUM,ORIGIN_PX,ORIGIN_PY,PANEL_M,
    PANEL_N,,B_START_NUM,H_GAP,V_GAP,PANEL_NO)
```

第 9 章　视觉的共用程序库 (Library)

将视觉功能的共用程序库 (Library) 进行如下说明。

viTran6（共用程序库 (Library)）【用于 μ Vision 板】

功　能

将视觉坐标转换为机械手坐标 （6 轴）

格　式

viTran6 （<CAL 编号 >、 < 视觉坐标 X>、 < 视觉坐标 Y>、 < 机械手坐标 （P 型） >）

说　明

将视觉的坐标 (X, Y) 转换为机械手的绝对坐标 (X, Y, Z)。

为了执行该共用程序库 (Library)，需要事先执行视觉 CAL，并将视觉 CAL 数据转发到控制器。

在 CAL 编号中指定执行了视觉 CAL 的编号。

因为在该坐标转换中只为求取绝对坐标 (X, Y, Z)，要想执行 MOVE 命令，需要另外设定姿势数据。

应用示例

```
'F1 中存放有视觉坐标 X
'F2 中存放有视觉坐标 Y
TAKEARM
CHANGETOOL 1                          ' 设定为进行了视觉 CAL 的工具
MOVE P,P10                            ' 转移到待机位置
P11 = P10                             ' 复制待机位置的姿势数据
CALL viTran6 （0， F1， F2， P11）     ' 将视觉坐标转换为机械手坐标 （使用视觉 CAL 数据的
                                      "0"）

APPROACH P， P11， 100
MOVE L， P11                          ' 转移到视觉所测量的位置
DEPART L,100
CHANGETOOL 0
```

viTran6S（共用程序库 (Library)）【用于 μ Vision -21】

功　能

将视觉坐标转换为机械手坐标 （6 轴）

格　式

viTran6S （<CAL 编号 >、< 视觉坐标 X>、< 视觉坐标 Y>、< 机械手坐标 X>、< 机械手坐标 Y>、< 机械手坐标 Z>）

说　明

将视觉的坐标 (X, Y) 转换为机械手的绝对坐标 (X, Y, Z)。

为了执行该共用程序库 (Library)，需要事先执行视觉 CAL，并将视觉 CAL 数据转发到控制器。

在 CAL 编号中指定执行了视觉 CAL 的编号。

因为在该坐标转换中只为求取绝对坐标 (X, Y, Z)，要想执行 MOVE 命令，需要另外设定姿势数据。

DENSO机械手

编程手册 II
PAC 共用程序库（Library）

初　版　2008年　1月
第 2 版　2009年　4月
第 3 版　2011年　9月

DENSO WAVE INCORPORATED

9N**C