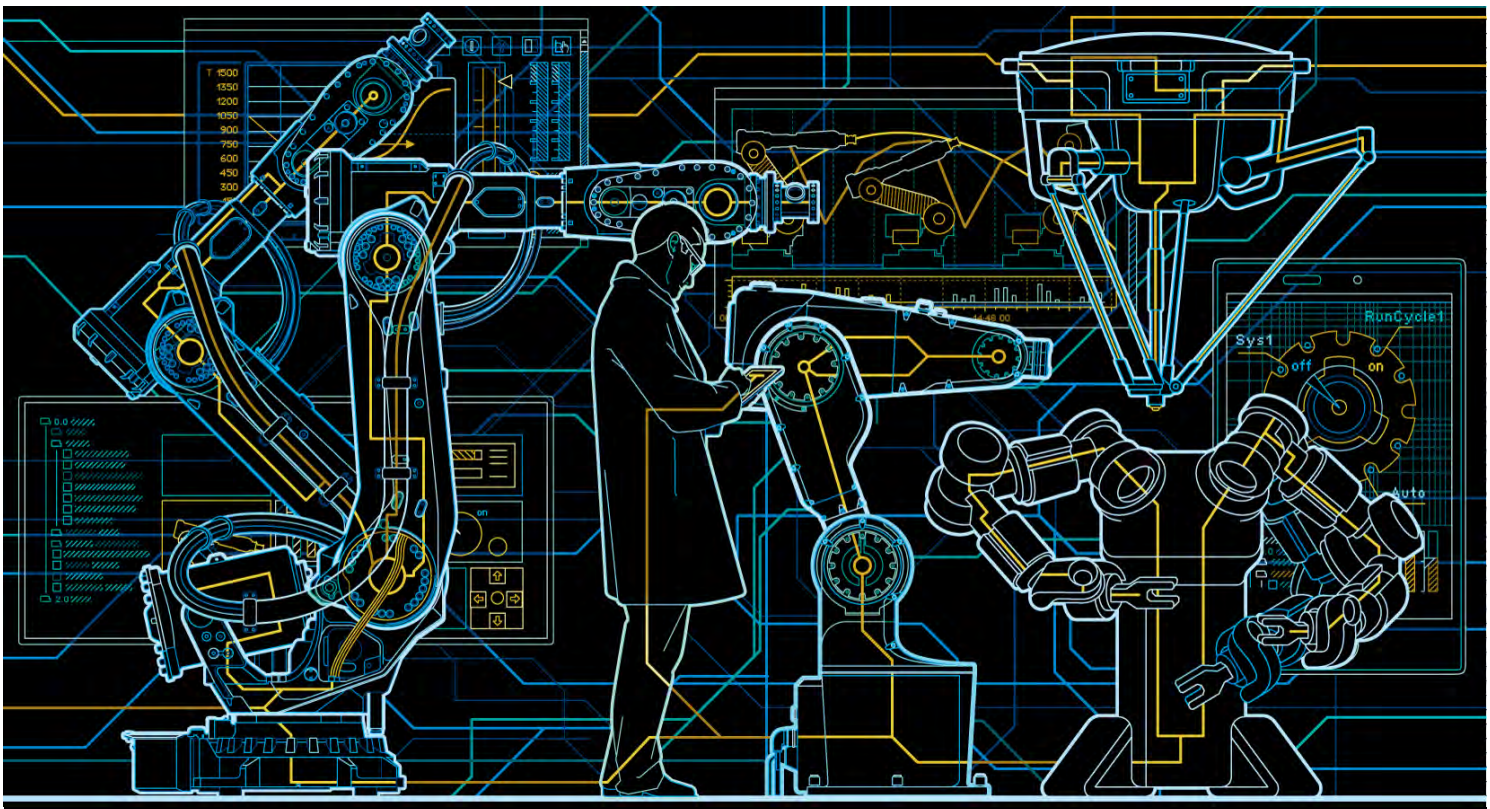


技术参考手册
RAPID指令、函数和数据类型

Power and productivity
for a better world™

ABB

Trace back information:
Workspace R16-1 version a6
Checked in 2016-03-01
Skribenta version 4.6.209

技术参考手册

RAPID指令、函数和数据类型

RobotWare 6.03

文档编号: 3HAC050917-010

修订: C

ABB AB
Robotics Products
Se-721 68 Västerås
瑞典

目表

手册概述

关于本手册

本文是供RAPID程序员使用的技术参考手册。本手册详细描述了RAPID基本指令、函数和数据类型。

手册用法

编程期间以及需要有关RAPID指令、函数或者数据类型的特定信息时，应阅读本手册。

本手册的阅读对象

本手册适用于有一些编程经验的人员，例如，机械臂程序员。

操作前提

阅读本手册的人员应具备一些编程经验，并且已经研究了

- 操作员手册 - *Introduction to RAPID*
- 技术参考手册 - *RAPID*语言概览

各章结构

本手册由以下各章组成：

章节	目录
1指令	所有RAPID基本指令的详细说明，包括如何使用此类指令的实例。
2函数	所有RAPID基本函数的详细说明，包括如何使用此类功能的实例。
3数据类型	所有RAPID基本数据类型的详细说明，包括如何使用此类数据类型的实例。
4编程类型实例	概述如何编写包含不同指令/函数/数据类型的程序代码。本章同时包含编程技巧和解释。

参考信息

参考文档	文档编号
技术参考手册 – *RAPID*指令、函数和数据类型带RobotWare 5的IRC5。	3HAC16581-001
操作员手册 - *Introduction to RAPID*	*3HAC029364-010*
技术参考手册 - *RAPID*语言概览	*3HAC050947-010*
技术参考手册 - *RAPID*语言内核	*3HAC050946-010*
应用手册 - 控制器软件*IRC5*	*3HAC050798-010*

修订版

版本号	描述
-	随 RobotWare 6.0 发布。 **注意** 关于带RobotWare 5的IRC5，请参见手册3HAC16581-001。

下一页继续

版本号	描述
A	随 RobotWare 6.01 发布。 • 增加了以下指令、函数和数据类型： 第*30*页的*AliasIOReset*-重置*I/O*信号以及别名， 第*792*页的*TriggJIOs* - 接头机械臂移动以及*I/O*事件 • 将关于7轴机械臂的信息增加到数据类型第*1375*页的*confdata* - 机械臂配置数据。
B	随 RobotWare 6.02 发布。 • 增加了以下普通指令、函数和数据类型： 第*545*页的*SaveCfgData* - 将系统参数保存至文件，第*1373*页的*cfgdomain* - 配置域， 第*759*页的*TriggDataCopy* - 复制触发数据变量中的内容， 第*761*页的*TriggDataReset* - 重置触发数据变量中的内容， 第*1284*页的*TriggDataValid* - 检查触发数据变量中的内容是否有效 第*960*页的*AInput* - 读取模拟信号输入信号值，第*1056*页的*DInput* - 读取数字信号输入信号值， 第*1109*页的*GInput* - 读取组输入信号的值 • 针对RobotWare选项*Integrated Vision*，增加了所有指令、函数和数据类型。 • 将关于制动距离的警告，增加至第*643*页的*SoftAct* - 启用软伺服。 • 为数据类型dnum增加了三角函数： ACosDnum, ASinDnum, ATanDnum, ATan2Dnum, CosDnum, TanDnum, SinDnum • 增加了EGM Path Correction功能的RAPID指令： 第*149*页的*EGMActJoint* – 为一个关节目标点编写一次*EGM*移动, 第*159*页的*EGMMoveC* – 经过路径校正的圆形*EGM*移动, 第*162*页的*EGMMoveL* – 经过路径校正的直线*EGM*移动, 第*180*页的*EGMSetupLTAPP* – 为*EGM*设置相应的*LTAPP*协议 • 细微纠正。
C	随 RobotWare 6.03 发布。 • 在指令中加入新功能第*331*页的*MotionProcessModeSet* - 设置运动过程模式。 • 添加*CAP*指令、函数和数据类型。 • 添加*Cyclic bool*指令和函数 • 添加与Functional Safety相关的指令和函数。 • signalxx此刻为能够进行数值运算的半值数据类型，请参见第*1476*页的*signalxx* - 数字信号和模拟信号。 • 细微纠正。

1 指令：

1.1 AccSet – 降低加速度

手册用法

处理脆弱负载时，使用了AccSet，可允许更低的加速度和减速度，使得机械臂的移动更加顺畅。

本指令仅可用于主任务T_ROB1，或者如果在MultiMove系统中，则可用于运动任务中。

基本示例

以下实例介绍了指令AccSet：

例 1

```
AccSet 50, 100;
```

将加速度限制在正常值的50%。

例 2

```
AccSet 100, 50;
```

将加速斜面限制在正常值的50%。

例 3

```
AccSet 100, 100 \FinePointRamp:=50;
```

当朝精点减速时，将减速斜面限制在正常值的50%。

变元

```
AccSet Acc Ramp [\FinePointRamp]
```

Acc

数据类型：num

加速度和减速度占正常值的百分比。100%相当于最大加速度。最大值：100%。输入值<20%，得出最大加速度的20%。

Ramp

数据类型：num

加速度和减速度增加的速率占正常值的百分比。通过降低该值，可限制顿挫。100%相当于最大速率。最大值：100%。输入值<10%，得出最大速率的10%。

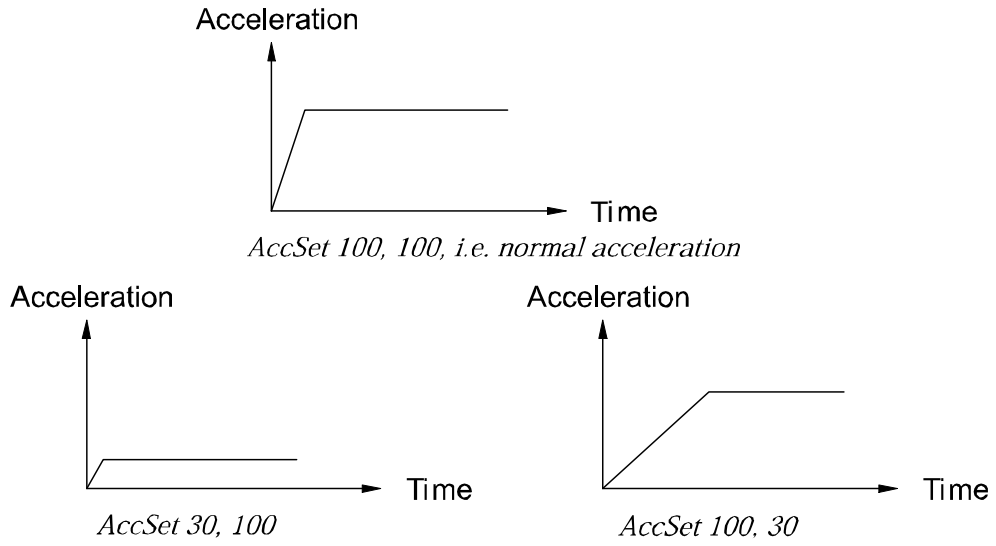
下一页继续

1.1 AccSet – 降低加速度

RobotWare - OS

续前页

数字表明，降低加速度会使移动更加顺畅。



xx0500002146

[\FinePointRamp]

数据类型：num

减速度降低的速率占正常值的百分比。当机械臂朝精点减速时，本参数仅会对斜面造成影响。在精点处，减速斜面值是本参数与Ramp值的组合，Ramp*FinePointRamp。本参数必须大于0，且介于0到100%之间。

如果未使用此可选参数，则将FinePointRamp值设置为默认值，100%。

程序执行

加速度适用于机械臂和外轴，直至执行一个新的AccSet指令。

默认值（100%）是自动设置的

- 当使用重启模式重置RAPID时
- 当加载一则新程序或一个新模块时
- 当从起点开始执行程序时
- 当将程序指针移动到main时
- 当将程序指针移动到子程序时
- 移动程序指针造成执行顺序丢失时

语法

```
AccSet
  [ Acc ':=' ] < expression (IN) of num > ','
  [ Ramp ':=' ] < expression (IN) of num >
  [ '\'FinePointRamp ':=' < expression (IN) of num > ] ';'
```

相关信息

信息，关于	请参阅
控制世界坐标系中的加速度。	第908页的*WorldAccLim - 控制世界坐标系中的加速度*

下一页继续

信息，关于	请参阅
降低路径沿线的工具中心接触点（TCP）加速度	第*428*页的*PathAccLim - 降低路径沿线的TCP加速度*
定位器指令	技术参考手册 - *RAPID*语言概览
运动设置数据	第*1430*页的*motsetdata - 运动设置数据*

1.2 ActEventBuffer - 事件缓冲启用

描述

ActEventBuffer用于在当前运动程序任务中启用事件缓冲。

当结合使用精点的应用以及持续应用（因缓慢的工艺设备而需要提前设置信号）时，应当使用指令ActEventBuffer和DeactEventBuffer。

本指令仅可用于主任务T_ROB1，或者如果在MultiMove系统中，则可用于运动任务中。

基本示例

以下实例介绍了指令ActEventBuffer：

例 1

```
...
DeactEventBuffer;
! Use an application that uses finepoints, such as SpotWelding
...
! Activate the event buffer again
ActEventBuffer;
! Now it is possible to use an application that needs
! to set signals in advance, such as Dispense
...
```

DeactEventBuffer停用配置的事件缓冲。当使用有精点的应用时，机械臂将以更快的速度从精点启动。当通过ActEventBuffer来启用事件缓冲时，针对采用缓慢工艺设备的应用而提前设置信号是可能的。

程序执行

事件缓冲的使用适用于下一步执行的机械臂的任一类型运动，并始终有效，直至执行DeactEventBuffer指令。

在启用事件缓冲之前，本指令将等待，直至机械臂和外轴已经达到停止点（当前运动指令的ToPoint）。因此，建议将运动指令编程在ActEventBuffer以及精点之前。

自动设置默认值（使用事件缓冲 = TRUE）

- 当使用重启模式重置RAPID时
- 当加载一则新程序或一个新模块时
- 当从起点开始执行程序时
- 当将程序指针移动到main时
- 当将程序指针移动到子程序时
- 移动程序指针造成执行顺序丢失时

限制

无法在与任意下列特殊系统事件关联的RAPID程序中执行ActEventBuffer：PowerOn、Stop、QStop、Restart或者Step。

语法

```
ActEventBuffer ';'
```

下一页继续

相关信息

信息，关于	请参阅
事件缓冲停用	第*138*页的*DeactEventBuffer - 事件缓冲启用*
Event preset time的配置	技术参考手册 - 系统参数
运动设置数据	第*1430*页的*motsetdata - 运动设置数据*

1.3 ActUnit - 启用机械单元

手册用法

ActUnit用于启用机械单元。

其可以用于确定，例如，当使用公共驱动单元时，应启用哪一个单元。

本指令仅可用于主任务T_ROB1，或者如果在MultiMove系统中，则可用于运动任务中。

基本示例

以下实例介绍了指令ActUnit：

例 1

```
ActUnit orbit_a;
```

orbit_a机械单元启用。

变元

```
ActUnit MechUnit
```

MechUnit

Mechanical Unit

数据类型：mecunit

待启用机械单元的名称。

程序执行

当机械臂与外轴实际路径就绪时，清除当前路径等级上的路径，并启用指定机械单元。这意味着通过机械臂进行控制和监测。

如果若干机械单元共享一个公共驱动单元，则启用其中一个机械单元，亦将把该单元同公共驱动单元相连。

限制

如果该指令位于移动指令之后，则必须使用停止点（zonedatafine）而非飞越点来编程移动指令，否则，将无法在电源故障后重启。

无法在与任意下列特殊系统事件关联的RAPID程序中执行ActUnit：PowerOn、Stop、QStop、Restart、Reset或者Step。

有可能将ActUnit-DeactUnit用于StorePath等级，但是，当正在进行RestoPath且已经完成StorePath时，必须启用相同的机械单元。关于路径记录器的操作以及基准等级的路径均将十分完整，但是，将清除StorePath等级的路径。

语法

```
ActUnit
  [ MechUnit ':=' ] < variable (VAR) of mecunit> ';'
```

相关信息

信息，关于	请参阅
停用机械单元	第*140*页的*DeactUnit - 停用机械单元*
机械单元	第*1428*页的*mecunit - 机械单元*

下一页继续

续前页

信息，关于	请参阅
更多示例	第*140*页的*DeactUnit - 停用机械单元*
检查机械单元是否启用。	第*1130*页的*IsMechUnitActive - 机械单元是否有效*
路径记录器	第*431*页的*PathRecMoveBwd - 将路径记录器向后移动*

1.4 Add-增加数值

手册用法

Add用于从数值变量或者永久数据对象增减一个数值。

基本示例

以下实例介绍了指令Add：

例 1

```
Add reg1, 3;
```

将3增加到reg1，即reg1:=reg1+3。

例 2

```
Add reg1, -reg2;
```

reg2的值得以从reg1中减去，即，reg1:=reg1-reg2。

例 3

```
VAR dnum mydnum:=5;
Add mydnum, 500000000;
```

将500000000增加到mydnum，即，mynum:=mynum+500000000。

例 4

```
VAR dnum mydnum:=5000;
VAR num mynum:=6000;
Add mynum, DnumToNum(mydnum \Integer);
```

将5000增加到mynum，即，mynum:=mynum+5000。必须使用DnumToNum以获得num数值，该数值可以同num变量mynum一同使用。

变元

```
Add Name | Dname AddValue | AddDvalue
```

Name

数据类型：num

待改变变量或者永久数据对象的名称。

Dname

数据类型：dnum

待改变变量或者永久数据对象的名称。

AddValue

数据类型：num

有待增加的值。

AddDvalue

数据类型：dnum

有待增加的值。

下一页继续

限制

如果有待增加的值是dnum型，且应当更改的变量/永久数据对象是num，则将产生一个运行时错误。不可能结合参数（关于如何解决此问题，请参见上文实例4）。

语法

```
Add
  [ Name ':=' ] < var or pers (INOUT) of num >
  | [ Dname ':=' ] < var or pers (INOUT) of dnum > ','
  [ AddValue ':=' ] < expression (IN) of num >
  | [ AddDvalue ':=' ] < expression (IN) of dnum > ';'
```

相关信息

信息，关于	请参阅
将变量增加1	第*239*页的*Incr - 增量为1*
将变量减少1	第*142*页的*Decr - 减量为1*
运用任意表达式（例如，乘法）来更改数据	第*32*页的*":=" - 分配一个数值*

1.5 AliasIO - 确定I/O信号以及别名

手册用法

AliasIO用于确定任意类型的信号以及别名，或者使用内置任务模块中的信号。

信号以及别名可用于预先确定通用程序，在不同的机械臂装置中运行前，不会对程序进行任何修改。

在使用实际信号之前，必须运行指令AliasIO 。关于已加载模块，请参见第*28*页的基本示例，关于已安装模块，第*29*页的更多示例。

基本示例

以下实例介绍了指令AliasIO：

另请参阅第*29*页的更多示例

例 1

```
VAR signaldo alias_do;
PROC prog_start()
  AliasIO config_do, alias_do;
ENDPROC
```

程序prog_start与系统参数中的START事件相关联。确定数字信号输出信号alias_do的程序与程序启动时的配置数字信号输出信号config_do相关联。

变元

```
AliasIO FromSignal ToSignal
```

FromSignal

数据类型：signalxx 或string

已加载的模块：

根据配置（数据类型signalxx）命名信号标识符，由此复制信号描述符。必须在I/O配置中确定信号。

已安装的模块或者已加载的模块

参考（CONST,VAR或者其中的参数）包含信号名称（数据类型string），由此在搜索后复制系统中的信号描述符。必须在I/O配置中确定信号。

ToSignal

数据类型：signalxx

信号标识符符合程序（数据类型signalxx），由此复制信号描述符。必须在RAPID程序中声明信号。

必须针对参数FromSignal和ToSignal 使用（或发现）相同的数据类型，且必须为一类signalxx（signalai、signalao、signaldi、signaldo、signalgi或signalgo）。

程序执行

从参数FromSignal中给出的信号到参数ToSignal中给出的信号，复制信号描述符的值。

下一页继续

错误处理

系统会生成下列可恢复错误，并在错误处理器中处理这些错误。系统变量ERRNO将被设置成：

名称	错误原因
ERR_ALIASIO_DEF	未在I/O配置中确定FromSignal，或未在RAPID程序中声明ToSignal，或未在I/O配置中确定ToSignal。
ERR_ALIASIO_TYPE	自变量FromSignal和ToSignal的数据类型并不相同。
ERR_NO_ALIASIO_DEF	该信号变量是在RAPID中声明的一个变量，与用指令AliasIO在I / O配置中定义的I / O信号无关。

更多示例

有关于指令AliasIO的更多例子阐述如下。

例 1

```
VAR signaldi alias_di;
PROC prog_start()
  CONST string config_string := "config_di";
  AliasIO config_string, alias_di;
ENDPROC
```

程序prog_start 与系统参数中的START事件相关联。本程序确定的数字信号输入信号alias_di与程序启动时的配置数字信号输入信号config_di（通过常量config_string）相关联。

限制

当启动本程序时，无法使用别名信号，直至执行AliasIO指令。

指令AliasIO 必须得以发出。

- 或者以程序启动（事件START）时执行的事件程序
- 或者以每次程序启动后（使用信号之前）执行的程序部分

为防止出现错误，建议使用动态重新连接，将AliasIO信号与不同的物理信号相连。

语法

```
AliasIO
  [ FromSignal ':=' ] < reference (REF) of anytype > ','
  [ ToSignal ':=' ] < variable (VAR) of anytype > ';'
```

相关信息

信息，关于	请参阅
输入/输出指令	技术参考手册 - *RAPID*语言概览
输入/输出功能性概述	技术参考手册 - *RAPID*语言概览
I/O配置	技术参考手册 - 系统参数
确定事件程序	技术参考手册 - 系统参数
已加载/已安装的任务模块	技术参考手册 - 系统参数
Advanced RAPID	应用手册 - 控制器软件*IRC5*

1.6 AliasIOReset-重置I/O信号以及别名

手册用法

AliasIOReset用于重置信号，该信号已经在先前的要求中用于AliasIO。

基本示例

以下实例介绍了指令AliasIOReset：

例 1

```
VAR signaldo alias_do;
PROC myproc()
  AliasIO config_do, alias_do;
  SetDO alias_do, 1;
  ..
  AliasIOReset alias_do;
ENDPROC
```

程序确定的数字信号输出信号alias_do与配置的数字信号输出信号config_do在无返回值程序myproc开始时相连。在I/O配置中确定信号config_do。随后，当不得再使用alias_do时，去除别名耦合。

变元

```
AliasIOReset Signal
```

Signal

数据类型：signalxx

信号标识符符合应当予以重置的程序（数据类型signalxx）。必须在RAPID程序中声明该信号。

程序执行

去除整个别名耦合。无法使用信号，直至完成与AliasIO的新别名耦合。

限制

无法重置I/O配置中确定的信号。仅可使用AliasIO指令中用到以及RAPID程序中声明的信号。

语法

```
AliasIOReset
  [ Signal ':=' ] < variable (VAR) of anytype > ';'
```

相关信息

信息，关于	请参阅
确定I/O信号以及别名	第28页的*AliasIO - 确定I/O信号以及别名*
输入/输出指令	技术参考手册 - *RAPID*语言概览
输入/输出功能性概述	技术参考手册 - *RAPID*语言概览
I/O配置	技术参考手册 - 系统参数
确定事件程序	技术参考手册 - 系统参数

下一页继续

信息，关于	请参阅
已加载/已安装的任务模块	技术参考手册 - 系统参数
Advanced RAPID	应用手册 - 控制器软件*IRC5*

1.7 ":=" - 分配一个数值

手册用法

“:=”指令用于向数据分配新值。该值可以是一个恒定值，亦可以是一个算术表达式，例如，reg1+5*reg3。

基本示例

以下实例介绍了指令“:=”：

另请参阅第32页的更多示例

例 1

```
reg1 := 5;
```

将reg1指定为值5。

例 2

```
reg1 := reg2 - reg3;
```

将reg1的值指定为reg2-reg3的计算结果。

例 3

```
counter := counter + 1;
```

将counter增加一。

变元

```
Data := Value
```

Data

数据类型：All

将被分配新值的数据。

Value

数据类型：Same as Data

期望值。

更多示例

有关于指令“:=”的更多例子阐述如下。

例 1

```
tool1.tframe.trans.x := tool1.tframe.trans.x + 20;
```

将tool1的TCP在X方向移动20mm。

例 2

```
pallet{5,8} := Abs(value);
```

向pallet矩阵中的元素分配等于value变量绝对值的值。

限制

数值有待改变的数据不得为

- 常量
- 非值数据类型。

数据和数值必须具有类似（相同或者别名）的数据类型。

下一页继续

语法

```
<assignment target> ':=' <expression> ';'
```

相关信息

信息，关于	请参阅
表达式	技术参考手册 - *RAPID*语言概览
非值数据类型	技术参考手册 - *RAPID*语言概览
向数据分配一个初始值	操作员手册 - 带 *FlexPendant* 的 *IRC5*

1.8 BitClear - 在一个字节或双数值数据中清除一个特定位

手册用法

BitClear用于在确定的byte数据或者dnum数据中清除（设置为0）一个特定位。

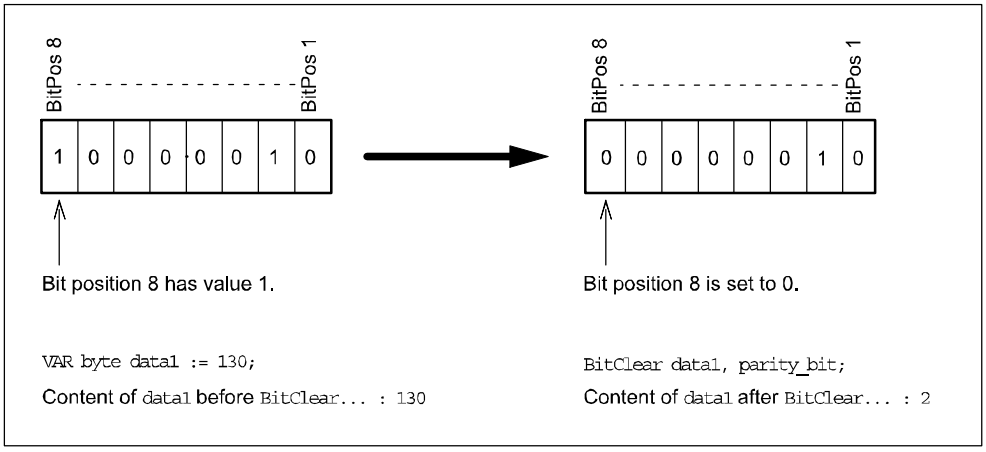
基本示例

以下实例介绍了指令BitClear：

例 1

```
CONST num parity_bit := 8;
VAR byte data1 := 130;
BitClear data1, parity_bit;
```

将变量data1中的比特编号8（parity_bit）设置为0，例如，将变量data1的容量从130改变为2（整数表示）。使用BitClear时，下图阐明了数据类型byte的位操作。



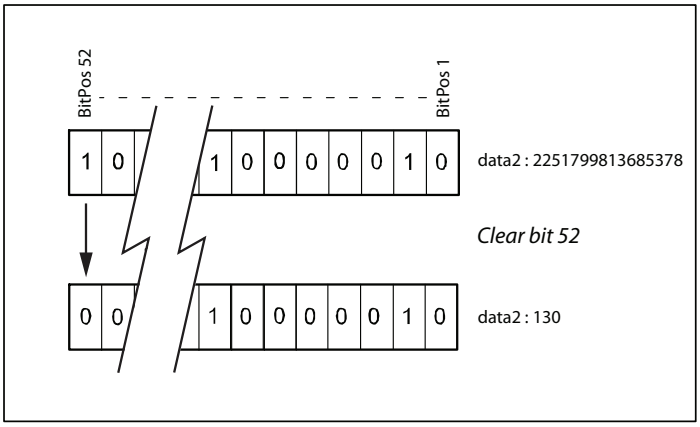
xx0500002147

例 2

```
CONST num parity_bit := 52;
VAR dnum data2 := 2251799813685378;
BitClear data2, parity_bit;
```

下一页继续

将变量data2中的比特编号52（parity_bit）设置为0，例如，将变量data2的容量从2251799813685378改变为130（整数表示）。使用BitClear时，下文阐明了数据类型dnum的位操作。



xx1200000014

变元

```
BitClear BitData | DnumData BitPos
```

BitData

数据类型：byte

以整数表示且有待改变的位数据。

DnumData

数据类型：dnum

以整数表示有待改变的双数值位数据。

BitPos

Bit Position

数据类型：num

将BitData中的数位位置（1-8），或者DnumData中的数位位置（1-52）设置为0。

限制

数据类型byte的范围为0 - 255位小数。

数据类型byte的数位位置1-8有效。

数据类型dnum的范围为0 - 4503599627370495位小数。

数据类型dnum的数位位置1-52有效。

语法

```
BitClear
  [ BitData ':=' ] < var or pers (INOUT) of byte >
  | [ DnumData ':=' ] < var or pers (INOUT) of dnum > ','
  [ BitPos ':=' ] < expression (IN) of num > ';'
```

下一页继续

1.8 BitClear - 在一个字节或双数值数据中清除一个特定位
RobotWare - OS
续前页

相关信息

信息，关于	请参阅
在一个字节或双数值数据中设置一个特定位	第*37*页的*BitSet* - 在一个字节或者双数值数据中设置一个特定位
检查字节数据中的特定位是否设置完毕	第*978*页的*BitCheck* - 检查字节数据中的特定位是否设置完毕
检查双数值数据中的特定位是否设置完毕	第*980*页的*BitCheckDnum* - 检查双数值数据中的特定位是否设置完毕
其他位函数	技术参考手册 - *RAPID*语言概览
Advanced RAPID	应用手册 - 控制器软件*IRC5*

1.9 BitSet - 在一个字节或者双数值数据中设置一个特定位

手册用法

BitSet用于将byte数据或dnum数据中确定的特定位设置为1。

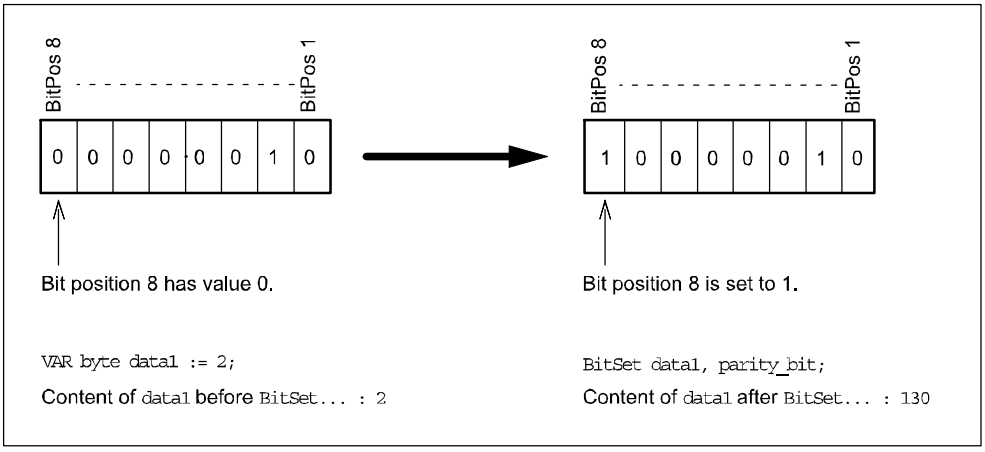
基本示例

以下实例介绍了指令BitSet：

例 1

```
CONST num parity_bit := 8;
VAR byte data1 := 2;
BitSet data1, parity_bit;
```

将变量data1中的比特编号8（parity_bit）设置为1，例如，将变量data1的容量从2改变为130（整数表示）。使用BitSet时，下图阐明了数据类型byte的位操作。



xx0500002148

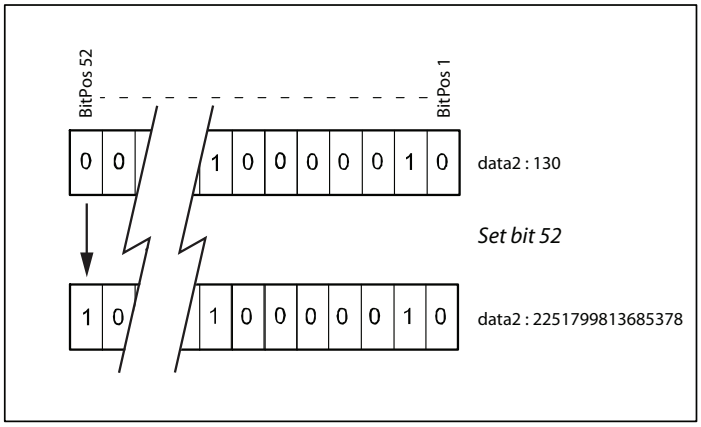
例 2

```
CONST num parity_bit := 52;
VAR dnum data2 := 130;
BitSet data2, parity_bit;
```

下一页继续

1.9 BitSet - 在一个字节或者双数值数据中设置一个特定位
RobotWare - OS
续前页

将变量data2中的比特编号52（parity_bit）设置为1，例如，将变量data2的容量从130改变为2251799813685378（整数表示）。使用BitSet时，下图阐明了数据类型dnum的位操作。



xx1200000015

变元

```
BitSet BitData | DnumData BitPos
```

BitData

数据类型：byte

以整数表示且有待改变的位数据。

DnumData

数据类型：dnum

以整数表示且有待改变的位数据。

BitPos

Bit Position

数据类型：num

将BitData中的数位位置（1-8），或者DnumData中的数位位置（1-52）设置为1。

限制

数据类型byte的范围为整数0 - 255。

数据类型byte的数位位置1-8有效。

数据类型dnum的范围为整数0 - 4503599627370495。

数据类型dnum的数位位置1-52有效。

语法

```
BitSet
  [ BitData':=' ] < var or pers (INOUT) of byte >
  | [ DnumData':=' ] < var or pers (INOUT) of dnum > ','
  [ BitPos':=' ] < expression (IN) of num > ';'
```

下一页继续

相关信息

信息，关于	请参阅
在一个字节或双数值数据中清除一个特定位	第*34*页的*BitClear - 在一个字节或双数值数据中清除一个特定位*
检查字节数据中的特定位是否设置完毕	第*978*页的*BitCheck - 检查字节数据中的特定位是否设置完毕*
检查双数值数据中的特定位是否设置完毕	第*980*页的*BitCheckDnum - 检查双数值数据中的特定位是否设置完毕*
其他位函数	技术参考手册 - *RAPID*语言概览
Advanced RAPID	应用手册 - 控制器软件*IRC5*

1.10 BookErrNo - 登记RAPID系统错误编号

手册用法

BookErrNo用于登记新的RAPID系统错误编号。

基本示例

以下实例介绍了指令BookErrNo：

例 1

```
! Introduce a new error number in a glue system
! Note: The new error variable must be declared with the initial
    value -1
VAR errnum ERR_GLUEFLOW := -1;

! Book the new RAPID system error number
BookErrNo ERR_GLUEFLOW;
```

将变量ERR_GLUEFLOW分配给自由系统错误编号，以便在在RAPID代码中使用。

```
! Use the new error number
IF di1 = 0 THEN
  RAISE ERR_GLUEFLOW;
ELSE
...
ENDIF

! Error handling
ERROR
  IF ERRNO = ERR_GLUEFLOW THEN
    ...
  ELSE
    ...
  ENDIF
```

如果数字信号输入di1为 0，则新登记的错误编号将会提高，并将系统错误变量ERRNO设置为新登记的错误编号。随后，照常用错误处理器来处理此类由用户产生的错误。

变元

```
BookErrNo ErrorName
```

ErrorName

数据类型：errnum

新RAPID系统错误变量名称。

限制

不得将新错误变量声明为程序变量。

须声明新错误变量的初始值为-1，表明该错误应当为RAPID系统错误。

语法

```
BookErrNo
  [ ErrorName ':='] < variable (VAR) of errnum > ';'
```

下一页继续

相关信息

信息，关于	请参阅
错误处理	技术参考手册 - *RAPID*语言概览
错误编号	第*1395*页的*errnum - 错误编号*
调用错误处理器	第*481*页的*RAISE - 调用错误处理器*
Advanced RAPID	应用手册 - 控制器软件*IRC5*

1.11 Break - 中断程序执行

手册用法

出于RAPID程序代码调试目的，将Break用于立即中断程序执行。机械臂立即停止运动。

基本示例

以下实例介绍了指令Break：

例 1

```
...
Break;
...
```

停止程序执行，且有可能出于调试目的来分析变量、数值等。

程序执行

指令立即停止程序执行，且无需等待机械臂和外轴达到当时其程序规定的运动目的点。随后，可以从下一个指令重新开始程序执行。

如果某些程序事件中存在Break 指令，则将中断程序的执行，且将不会执行任何停止程序事件。将从相同事件下一次出现时开始执行程序事件。

语法

```
Break';'
```

相关信息

信息，关于	请参阅
停止程序动作	第*685*页的*Stop* - 停止程序执行
在出现致命错误后停止	第*202*页的*EXIT* - 终止程序执行
终止程序执行	第*202*页的*EXIT* - 终止程序执行
仅停止机械臂的移动	第*690*页的*StopMove* - 停止机械臂的移动

1.12 CallByVar - 通过变量，调用无返回值程序

手册用法

CallByVar（*Call By Variable*）可用于调用具有特定名称的无返回值程序，例如，经由变量的proc_name1, proc_name2, proc_name3 ... proc_namex。

基本示例

以下实例介绍了指令CallByVar：

另请参阅第*43*页的更多示例

例 1

```
reg1 := 2;
CallByVar "proc", reg1;
```

调用无返回值程序proc2。

变元

```
CallByVar Name Number
```

Name

数据类型：string

无返回值程序名称的第一部分，例如，proc_name。

Number

数据类型：num

无返回值程序编号的数值。将该值转换成字符串，并作为无返回值程序名称的第二部分，例如，1。该值必须为正整数。

更多示例

关于如何进行过程调用静态和动态选择的的更多实例。

实例1 - 过程调用的静态选择

```
TEST reg1
  CASE 1:
    lf_door door_loc;
  CASE 2:
    rf_door door_loc;
  CASE 3:
    lr_door door_loc;
  CASE 4:
    rr_door door_loc;
  DEFAULT:
    EXIT;
ENDTEST
```

根据登记值reg1为1、2、3或4，调用不同的无返回值程序，以针对选定的门实施适当类型的工作。位于参数door_loc中的门位置。

实例2 - 过程调用以及RAPID语法的动态选择

```
reg1 := 2;
%"proc"+NumToStr(reg1,0)% door_loc;
```

下一页继续

1.12 CallByVar - 通过变量，调用无返回值程序
RobotWare - OS
续前页

使用参数door_loc，调用无返回值程序proc2。

限制：所有无返回值程序必须拥有特定的名称，例如，proc1，proc2，proc3.

实例3 - 过程调用以及CallByVar的动态选择

```
reg1 := 2;
CallByVar "proc",reg1;
```

调用无返回值程序proc2。

限制：所有无返回值程序必须拥有特定的名称，例如，proc1、proc2、proc3，且无法使用任何参数。

限制

仅可用于调用不带参数的无返回值程序。

无法用于调用局部无返回值程序。

执行CallByVar，将花费比执行普通过程调用略长的时间。

错误处理

可能会产生下列可恢复错误。错误可以由错误处理程序处理。系统变量ERRNO将设置为：

名称	错误原因
ERR_ARGVALERR	参数Number为<0或者非整数。
ERR_REFUNKPRC	参考适用于未知无返回值程序。
ERR_CALLPROC	过程调用错误（非无返回值程序）。

语法

```
CallByVar
  [Name ':='] <expression (IN) of string>','
  [Number ':='] <expression (IN) of num>';'
```

相关信息

信息，关于	请参阅
调用无返回值程序	技术参考手册 - *RAPID*语言概览 操作员手册 - 带 *FlexPendant* 的 *IRC5*

1.13 CamFlush - 从摄像头删除集合数据

手册用法

CamFlush 用于清空（删除）摄像头的 cameratarget 集合数据。

基本示例

以下示例介绍了 CamFlush 指令。

例 1

```
CamFlush mycamera;
```

摄像头的 mycamera 集合数据被删除。

变元

```
CamFlush Camera
```

Camera

数据类型：cameradev

摄像头名称。

语法

```
CamFlush
  [ Camera ':=' ] < variable (VAR) of cameradev > ';'
```

1.14 CamGetParameter - 获取不同名称的摄像头参数

手册用法

CamGetParameter 用于获取摄像头可能提供的命名的参数与。用户必须知道参数的名称及其返回类型才能获取其值。

基本示例

以下示例介绍了 CamGetParameter 指令。

例 1

```
VAR bool mybool:=FALSE;
...
CamGetParameter mycamera, "Pattern_1.Tool_Enabled_Status"
     \BoolVar:=mybool;
TPWite "The current value of Pattern_1.Tool_Enabled_Status is: "
     \Bool:=mybool;
```

获得命名的布尔参数 Pattern_1.Tool_Enabled_Status 并将值写在 FlexPendant 上。

变元

```
CamGetParameter Camera ParName [\Num] | [\Bool] | [\Str]
```

Camera

数据类型：cameradev

摄像头名称。

ParName

Parameter Name

数据类型：string

摄像头中参数的名称。

[\NumVar]

数据类型：num

变量 (**VAR**) 用于存储所获取的数据对象的数字值。

[\BoolVar]

数据类型：bool

变量 (**VAR**) 用于存储所获取的数据对象的布尔值（真假值）。

[\StrVar]

数据类型：string

变量 (**VAR**) 用于存储所获取的数据对象的布尔值（真假值）。

程序执行

在此指令被执行时会直接读取指定参数并返回值。

如果指令被用于从图像分析读取结果，请在获取数据前确保摄像头已经完成图像处理。

下一页继续

错误处理

可能会产生下列可恢复错误。错误可以由错误处理程序处理。系统变量 ERRNO 将设置为：

名称	错误原因
ERR_CAM_BUSY	摄像头正忙于处理其他请求，无法执行当前命令。
ERR_CAM_COM_TIMEOUT	与摄像头通信错误。摄像头可能已断开。
ERR_CAM_GET_MISMATCH	用 CamGetParameter 指令从摄像头获取的参数的数据类型错误。

语法

```
CamGetParameter
  [ Camera ':=' ] < variable (VAR) of cameradev > ','
  [ ParName ':=' ] < expression (IN) of string >
  [ '\'NumVar ':=' < variable (VAR) of num > ]
  | [ '\'BoolVar ':=' < variable (VAR) of bool > ]
  | [ '\'StrVar ':=' < variable (VAR) of string > ] ';'
```

1.15 CamGetResult - 从集合获取摄像头目标

手册用法

CamGetResult （摄像头获取结果）用于从图像结果集合获取摄像头目标。

基本示例

以下示例介绍了 CamGetResult 指令。

例 1

```
VAR num mysceneid;
VAR cameratarget mycamtarget;
...
CamReqImage mycamera \SceneId:= mysceneid;
CamGetResult mycamera, mycamtarget \SceneId:= mysceneid;
```

命令摄像头 mycamera 采集图像。使用 SceneId 获取从图像生成的图像结果。

变元

```
CamGetResult Camera CamTarget [\SceneId] [\MaxTime]
```

Camera

数据类型：cameradev

摄像头名称。

CamTarget

摄像头目标

数据类型：cameratarget

作为图像结果保存位置的变量。

[\SceneId]

场景识别

数据类型：num

SceneId 是一个识别程序，指定 cameratarget 是从哪个图像生成的。

[\MaxTime]

最大时间

数据类型：num

程序执行可以等待的最大时间（以秒为单位）。允许的最大值是 120 秒。

程序执行

CamGetResult从图像结果集合获取摄像头目标。如果没有使用 SceneId 或 MaxTime，则不会获取结果，指令将永远停止。如果在 CamGetResult 中使用了 SceneId，则结果将在 CamReqImage 指令后生成。

SceneId 仅在已经从指令 CamReqImage 请求了图像时可用。如果图像是由外部 I/O 信号生成的，则SceneId 不能在指令 CamGetResult 中使用。

下一页继续

错误处理

可能会产生下列可恢复错误。错误可以由错误处理程序处理。系统变量 ERRNO 将设置为：

名称	错误原因
ERR_CAM_BUSY	摄像头正忙于处理其他请求，无法执行当前命令。
ERR_CAM_MAXTIME	在超时时间不能获取任何结果。
ERR_CAM_NO_MORE_DATA	不能为已经使用的 SceneId 获取更多图像结果，否则在超时时间无法获取结果。

语法

```
CamGetResult
  [ Camera ':=' ] < variable (VAR) of cameradev > ','
  [ CamTarget ':=' ] < variable (VAR) of CameraTarget >
  [ '\'SceneId ':=' < expression (IN) of num > ]
  [ '\'MaxTime ':=' < expression (IN) of num > ] ';'
```

1.16 CamLoadJob -加载摄像头任务到摄像头

手册用法

CamLoadJob （摄像头加载作业） 加载摄像头任务，*job*，说明了曝光参数、校准以及要应用的图像工具。

基本示例

以下示例介绍了 CamLoadJob 指令。

例 1

```
CamSetProgramMode mycamera;
CamLoadJob mycamera, "myjob.job";
CamSetRunMode mycamera;
```

作业 myjob 加载到名为 mycamera 的摄像头。

变元

```
CamLoadJob Camera JobName [\KeepTargets] [\MaxTime]
```

Camera

数据类型：cameradev

摄像头名称。

Name

数据类型：string

加载到摄像头的作业名称。

[\KeepTargets]

数据类型：switch

此参数用于指定是否保留摄像头产生的任何现有摄像头目标。

[\MaxTime]

数据类型：num

程序执行可以等待的最大时间（以秒为单位）。允许的最大值是 120 秒。

程序执行

CamLoadJob 的执行将会等到作业加载完毕或经过超时错误失败。如果使用可选参数 KeepTargets，则保留指定摄像头的集合数据。默认的操作是删除（清空）就集合数据。

错误处理

可能会产生下列可恢复错误。错误可以由错误处理程序处理。系统变量 ERRNO 将设置为：

名称	错误原因
ERR_CAM_BUSY	摄像头正忙于处理其他请求，无法执行当前命令。
ERR_CAM_COM_TIMEOUT	与摄像头通信错误。摄像头可能已断开。
ERR_CAM_MAXTIME	摄像头作业不会在超时时间加载。
ERR_CAM_NO_PROGMODE	摄像头未处于编程模式

下一页继续

限制

当摄像头设置为编程模式时，才可以执行 CamLoadJob。使用指令 CamSetProgramMode 可将摄像头设置为编程模式。

为了能加载作业，作业文件必须存储在摄像头的闪存盘。

语法

```
CamLoadJob
  [ Camera ':=' ] < variable (VAR) of cameradev > ','
  [ JobName ':=' ] <expression (IN) of string >
  [ '\'KeepTargets ]
  [ '\'MaxTime ':=' <expression (IN) of num>]';'
```

1.17 CamReqImage - 命令摄像头采集图像

手册用法

CamReqImage (摄像头请求图像)命令摄像头采集图像。

基本示例

以下示例介绍了 CamReqImage 指令。

例 1

```
CamReqImage mycamera;
```

命令摄像头 mycamera 采集图像。

变元

```
CamReqImage Camera [\SceneId] [\KeepTargets] [\AwaitComplete]
```

Camera

数据类型：cameradev

摄像头名称。

[\SceneId]

场景识别

数据类型：num

可选参数 SceneId 是所采集图像的一个标识符。这是由 CamReqImage 加上可选变量 SceneId 执行生成的。标识符是一个 1 到 8388608 之间的整数。如果没有使用 SceneId，则标识符值设置为 0。

[\KeepTargets]

数据类型：switch

此参数用于指定是否保留指定摄像头的旧集合数据。

[\AwaitComplete]

数据类型：switch

如果指定可选参数\AwaitComplete，则指令等待，直至已经收到来自图像的结果。如果未产生任何结果，例如，因为图像中没有一个部分，则会产生错误 ERR_CAM_REQ_IMAGE。

当使用\AwaitComplete时，必须将相机触发类型设置为外部。

程序执行

CamReqImage 用于命令指定摄像头采集图像。如果使用了可选参数 SceneId，则所采集图像的可用图像结果使用该指令生成的唯一数字标记。

如果使用可选参数 KeepTargets，则保留指定摄像头的旧集合数据。默认的操作是删除（清空）所有旧集合数据。

下一页继续

错误处理

可能会产生下列可恢复错误。错误可以由错误处理程序处理。系统变量 ERRNO 将设置为：

名称	错误原因
ERR_CAM_BUSY	摄像头正忙于处理其他请求，无法执行当前命令。
ERR_CAM_COM_TIMEOUT	与摄像头通信错误。摄像头可能已断开。
ERR_CAM_NO_RUNMODE	摄像头未处于运行模式
ERR_CAM_REQ_IMAGE	相机无法生成任何图像结果。

限制

当摄像头设置为运行模式时，才可以执行 CamReqImage。使用指令 CamSetRunMode 可将摄像头设置为运行模式。

语法

```
CamReqImage
  [ Camera ':=' ] < variable (VAR) of cameradev > ','
  [ '\'SceneId ':=' < variable (VAR) of num > ]
  [ '\'KeepTargets ]
  [ '\'AwaitComplete ]';'
```

1.18 CamSetExposure - 设置具体摄像头的数据

手册用法

CamSetExposure (摄像头设置曝光) 设置具体摄像头的数据并可以实现根据环境照明条件调整图像参数。

基本示例

以下示例介绍了 CamSetExposure 指令。

例 1

```
CamSetExposure mycamera \ExposureTime:=10;
```

命令摄像头 mycamera 将曝光时间修改为 10 ms。

变元

```
CamSetExposure Camera [\ExposureTime] [\Brightness] [\Contrast]
```

Camera

数据类型：cameradev

摄像头名称。

[\ExposureTime]

数据类型：num

如果使用了本可选参数，则摄像头的曝光时间会更新。该值以毫秒 (ms) 为单位。

[\Brightness]

数据类型：num

如果使用了本可选参数，则将更新摄像头的亮度设置。其值通常以 0 到 1 之间的刻度表示。

[\Contrast]

数据类型：num

如果使用了本可选参数，则将更新摄像头的对比度设置。其值通常以 0 到 1 之间的刻度表示。

程序执行

如果具体摄像头的对应参数可能更新，则此指令更新曝光时间、亮度和对比度。如果摄像头不支持某个设置，则会向用户显示错误消息，程序停止执行。

错误处理

可能会产生下列可恢复错误。错误可以由错误处理程序处理。系统变量 ERRNO 将设置为：

名称	错误原因
ERR_CAM_COM_TIMEOUT	与摄像头通信错误。摄像头可能已断开。

语法

```
CamSetExposure
  [ Camera ':=' ] < variable (VAR) of cameradev > ','
```

下一页继续

```
[ '\'ExposureTime ':=' < variable (IN) of num > ]
[ '\'Brightness ':=' < variable (IN) of num > ]
[ '\'Contrast ':=' < variable (IN) of num > ] ';'
```

1.19 CamSetParameter - 设置不同名称的摄像头参数

手册用法

CamSetParameter 用于设置摄像头可能接触的不同名称的摄像头参数。在此命令中可以在运行时修改摄像头中的不同参数。用户i须知道参数名称及其类型才能设置其值。

基本示例

以下示例介绍了 CamSetParameter 指令。

例 1

```
CamSetParameter mycamera, "Pattern_1.Tool_Enabled" \BoolVal:=FALSE;
CamSetRunMode mycamera;
```

在本例中名为 "Pattern_1.Tool_Enabled" 的参数被设为假，这表示在采集到图像时不应执行指定的图像工具。

这将使图像工具的执行更快。但是，工具仍然用最后一次有效执行得到的值产生结果。为了不使用这些目标，应将它们从 RAPID 程序中剔除出去。

变元

```
CamSetParameter Camera ParName [\Num] | [\Bool] | [\Str]
```

Camera

数据类型：cameradev

摄像头名称。

ParName

数据类型：string

摄像头中参数的名称。

[\NumVal]

数据类型：num

摄像头的数值在参数 ParName 设置名称时设置。

[\BoolVal]

数据类型：bool

摄像头的布尔值在参数 ParName 设置名称时设置。

[\StrVal]

数据类型：string

摄像头的字符串值在参数 ParName 设置名称时设置。

错误处理

可能会产生下列可恢复错误。错误可以由错误处理程序处理。系统变量 ERRNO 将设置为：

名称	错误原因
ERR_CAM_BUSY	摄像头正忙于处理其他请求，无法执行当前命令。
ERR_CAM_COM_TIMEOUT	与摄像头通信错误。摄像头可能已断开。

下一页继续

名称	错误原因
ERR_CAM_SET_MISMATCH	使用命令 CamSetParameter 写入摄像头的参数数据类型错误，或者其值超出范围。

语法

```
CamSetParameter
  [ Camera ':=' ] < variable (VAR) of cameradev > ','
  [ ParName ':=' ] < expression (IN) of string >
  [ '\'NumVal ':=' < expression (IN) of num > ]
  | [ '\'BoolVal ':=' < expression (IN) of bool > ]
  | [ '\'StrVal ':=' < expression (IN) of string > ] ';'
```

1.20 CamSetProgramMode - 命令摄像头进入编程模式

手册用法

CamSetProgramMode *(*摄像头设置编程模式*)* 命令摄像头进入编程模式（离线）。

基本示例

以下示例介绍了 CamSetProgramMode 指令。

例 1

```
CamSetProgramMode mycamera;
CamLoadJob mycamera, "myjob.job";
CamSetRunMode mycamera;
...
```

首先，将摄像头改为编程模式。然后加载 myjob 到摄像头。然后命令摄像头进入运行模式。

变元

```
CamSetProgramMode Camera
```

Camera

数据类型：cameradev

摄像头名称。

程序执行

当使用 CamSetProgramMode 指令命令摄像头进入编程模式时，可以修改设置并加载作业到摄像头。

错误处理

可能会产生下列可恢复错误。错误可以由错误处理程序处理。系统变量 ERRNO 将设置为：

名称	错误原因
ERR_CAM_BUSY	摄像头正忙于处理其他请求，无法执行当前命令。
ERR_CAM_COM_TIMEOUT	与摄像头通信错误。摄像头可能已断开。

语法

```
CamSetProgramMode
  [ Camera ':=' ] < variable (VAR) of cameradev > ';'
```

1.21 CamSetRunMode - 命令摄像头进入运行模式

手册用法

CamSetRunMode （摄像头设置的运行模式）命令摄像头进入运行模式（在线），并将控制器更新为“输出到 RAPID”的当前配置。

基本示例

以下示例介绍了 CamSetRunMode 指令。

例 1

```
CamSetProgramMode mycamera;
CamLoadJob mycamera, "myjob.job";
...
CamSetRunMode mycamera;
```

首先，将摄像头改为编程模式。然后加载 myjob 到摄像头。然后使用 CamSetRunMode 指令命令摄像头进入运行模式。

变元

```
CamSetRunMode Camera
```

Camera

数据类型：cameradev

摄像头名称。

程序执行

在使用 CamSetRunMode 命令摄像头进入运行模式时，可以开始采集图像。

错误处理

可能会产生下列可恢复错误。错误可以由错误处理程序处理。系统变量 ERRNO 将设置为：

名称	错误原因
ERR_CAM_BUSY	摄像头正忙于处理其他请求，无法执行当前命令。
ERR_CAM_COM_TIMEOUT	与摄像头通信错误。摄像头可能已断开。

语法

```
CamSetRunMode
  [ Camera ':=' ] < variable (VAR) of cameradev > ';'
```

1.22 CamStartLoadJob - 开始加载摄像头任务到摄像头

手册用法

CamStartLoadJob将开始加载作业到摄像头，然后执行将在下一个指令继续进行。当加载进行时，其他指令可以并行执行。

基本示例

以下示例介绍了 CamStartLoadJob 指令。

例 1

```
...
CamStartLoadJob mycamera, "myjob.job";
MoveL p1, v1000, fine, tool2;
CamWaitLoadJob mycamera;
CamSetRunMode mycamera;
CamReqImage mycamera;
...
```

首先开始加载作业到摄像头，在加载进行时，执行了一个移动到位置 p1 的操作。当移动就绪后，加载也完成了，图像也采集好了。

变元

```
CamStartLoadJob Camera Name [\KeepTargets]
```

Camera

数据类型：cameradev

摄像头名称。

Name

数据类型：string

加载到摄像头的作业名称。

[\KeepTargets]

数据类型：switch

此参数用于指定是否保留指定摄像头的旧集合数据。

程序执行

CamStartLoadJob的执行将会命令开始加载，然后无需等待加载完成直接继续下一个指令，如果使用了可选参数 \KeepTargets，则不会删除指定摄像头的旧集合数据。默认操作是删除（清空）指定摄像头的旧集合数据。

错误处理

可能会产生下列可恢复错误。错误可以由错误处理程序处理。系统变量ERRNO将设置为：

名称	错误原因
ERR_CAM_BUSY	摄像头正忙于处理其他请求，无法执行当前命令。

下一页继续

限制

当摄像头设置为编程模式时，才可以执行 CamStartLoadJob。使用指令 CamSetProgramMode 可将摄像头设置为编程模式。

当作业的加载在执行中时，无法使用任何其他指令或函数访问对应的摄像头。后续的摄像头指令或函数必须是一个 CamWaitLoadJob 指令。

为了能加载作业，作业文件必须存储在摄像头的闪存盘。

语法

```
CamStartLoadJob
  [ Camera ':=' ] < variable (VAR) of cameradev > ','
  [ Name ':='] <expression (IN) of string >
  [ '\'KeepTargets ]';'
```

1.23 CamWaitLoadJob – 等待摄像头任务加载完毕

手册用法

CamWaitLoadJob *(*摄像头等待加载作业*)* 将会等待作业加载到摄像头完成。

基本示例

以下示例介绍了 CamWaitLoadJob 指令。

例 1

```
...
CamStartLoadJob mycamera, "myjob.job";
MoveL p1, v1000, fine, tool2;
CamWaitLoadJob mycamera;
CamSetRunMode mycamera;
CamReqImage mycamera;
...
```

首先开始加载作业到摄像头，在加载进行时，执行了一个移动到位置 p1 的操作。当移动就绪后，加载也完成了，图像也采集好了。

变元

```
CamWaitLoadJob Camera
```

Camera

数据类型：cameradev

摄像头名称。

错误处理

可能会产生下列可恢复错误。错误可以由错误处理程序处理。系统变量 ERRNO 将设置为：

名称	错误原因
ERR_CAM_COM_TIMEOUT	与摄像头通信错误。摄像头可能已断开。

限制

当摄像头设置为编程模式时，才可以执行 CamWaitLoadJob。使用指令 CamSetProgramMode 可将摄像头设置为编程模式。

当作业的加载在执行中时，无法使用任何其他指令或函数访问对应的摄像头。后续的摄像头指令或函数必须是一个 CamWaitLoadJob 指令。

语法

```
CamWaitLoadJob
  [ Camera ':=' ] < variable (VAR) of cameradev > ';'
```

1.24 CancelLoad - 取消模块加载

手册用法

CancelLoad可用于取消指令StartLoad产生的加载操作。

CancelLoad仅可用于指令StartLoad和WaitLoad之间。

基本示例

以下实例介绍了指令CancelLoad：

另请参阅第*63*页的更多示例

Example1

```
CancelLoad load1;
```

取消加载会话load1。

变元

```
CancelLoad LoadNo
```

LoadNo

数据类型：loadsession

加载会话的引用，其由指令StartLoad创建。

更多示例

有关于如何使用指令CancelLoad的更多例子阐述如下。

例 1

```
VAR loadsession load1;

StartLoad "HOME:"\File:="PART_B.MOD",load1;
...
IF ...
  CancelLoad load1;
  StartLoad "HOME:"\File:="PART_C.MOD",load1;
ENDIF
...
WaitLoad load1;
```

指令CancelLoad将取消加载中的模块PART_B.MOD，而非使其能够加载PART_C.MOD。

错误处理

可能会产生下列可恢复错误。错误可以由错误处理程序处理。系统变量ERRNO将设置为：

名称	错误原因
ERR_LOADNO_NOUSE	当前未使用参数LoadNo中规定的变量，表明当前未使用加载会话。

限制

CancelLoad仅可用于指令StartLoad就绪之后且指令WaitLoad启用之前的序列中。

下一页继续

1.24 CancelLoad - 取消模块加载

RobotWare - OS

续前页

语法

```
CancelLoad
  [ LoadNo ':=' ] < variable (VAR) of loadsession >';'
```

相关信息

信息，关于	请参阅
执行期间，加载普通程序模块	第*660*页的*StartLoad* - 执行期间，加载普通程序模块
将加载的模块与任务相连	第*881*页的*WaitLoad* - 将加载的模块与任务相连
加载会话	第*1427*页的*loadsession* - 程序加载会话
加载普通程序模块	第*312*页的*Load* - 执行期间，加载普通程序模块
卸载普通程序模块	第*843*页的*UnLoad* - 执行期间，卸载普通程序模块
检查程序参考	第*97*页的*CheckProgRef* - 检查程序参考

1.25 CapAPTrSetup - 设置At-Point-Tracker

手册用法

CapAPTrSetup（设置*At-Point-Tracker*）将用于设置传感器的At-Point-Tracker类型，比如，*WeldGuide*或AWC。

传感器接口利用RTP1传输协议，通过串行通道，最多与一个传感器通信。

基本示例

SIO.cfg:

```
COM_PHY_CHANNEL:
-name sio1:" -Connector "COM1"
COM_TRP:
-Name "swg:" -Type "RTP1" -PhyChannel "sio1"
```

RAPID代码：

```
! Define variable numbers
CONST num SensorOn := 6;
CONST num XCoord := 8;
CONST num YCoord := 9;
CONST num ZCoord := 10;
VAR pos SensorPos;

! Setup a Weldguide
CapAPTrSetup "swg:", do_left, 80, do_right, 80;
```

变元

```
CapAPTrSetup device DoLeft LevelLeft DoRight LevelRight [\LogFile]
     [\LogSize]
```

device

数据类型：string

为采用的传感器，在sio.cfg中配置I/O装置名称。

DoLeft

数据类型：signaldo

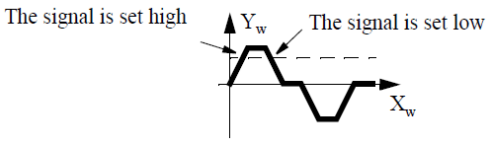
左摆动循环摆动同步数字输出信号

LevelLeft

数据类型：num

摆动运动时的左侧端部位置。指定值为在摆动中心左侧的宽度百分比。摆动超出该点时，数字输出信号将被自动设置到高位（前提是信号已被定义）。

此类坐标可用于凭电弧跟踪进行焊缝跟踪。



xx1200000178

下一页继续

1.25 CapAPTrSetup - 设置At-Point-Tracker

Continuous Application Platform (CAP)

续前页

DoRight

数据类型：signaldo

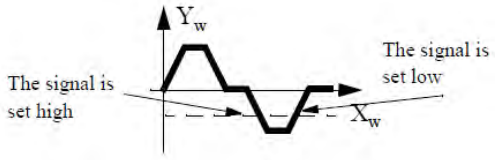
右摆动循环摆动同步数字输出信号

LevelRight

数据类型：num

摆动运动时的右侧端部位置。指定值为在摆动中心右侧的宽度百分比。摆动超出该点时，数字输出信号将被自动设置到高位（前提是信号已被定义）。

此类坐标可用于凭电弧跟踪进行焊缝跟踪。



xx1200000179

[\LogFile]

数据类型：string

跟踪日志文件名称

[\LogSize]

数据类型：num

跟踪日志环形缓冲区大小，即，跟踪期间可缓冲的传感器测量数。

默认值：1000。

语法

```
CapAPTrSetup
  [device ':='] < expression (IN) of string> ','
  [DoLeft ':='] < expression (IN) of signaldo > ','
  [LevelLeft ':='] < expression (IN) of num > ','
  [DoRight ':='] < expression (IN) of signaldo > ','
  [LevelRight ':='] < expression (IN) of num >
  ['\'LogFile ':='] < expression (IN) of string >
  ['\'LogSize ':='] < expression (IN) of num > ';'
```

相关信息

	参见：
Continuous Application Platform	应用手册 - *Continuous Application Platform*
Sensor Interface	应用手册 - 控制器软件*IRC5*

1.26 CapC - CAP圆周运动指令

手册用法

CapC将用于沿圆弧路径将工具中心接触点（TCP）移动到给定终点，同时控制一个连续进程。此外，还可以最多将八个事件与CapC关联起来。将利用指令TriggRampAO、TriggIO、TriggEquip、TriggInt、TriggCheckIO或TriggSpeed来定义各个事件。

基本示例

例 1

凭CapC进行圆周运动。

```
CapC cirp, p1, v100, cdata, weavestart, weave, fine, gun1;
```

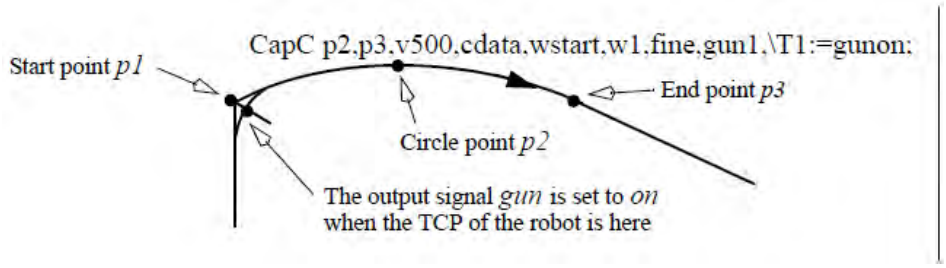
工具的TCP（gun1）以cdata中规定的速度，沿圆弧路径移动到精确点p1。

例 2

凭用户事件和CAP事件进行圆周运动。

```
VAR intnum start_intno;
...
PROC main()
  VAR triggdata gunon;

  IDelete start_intno;
  CONNECT start_intno WITH start_trap;
  ICap start_intno, CAP_START;
  TriggIO gunon, 0 \Start \DOp:=gun, on;

  MoveJ p1, v500, z50, gun1;
  CapC p2,p3,v500,cdata,wstart,w1,fine,gun1,\T1:=gunon;
ENDPROC

TRAP start_trap
  ! This routine will be executed when the event CAP_START is
       reported
ENDTRAP
```

当机器人的TCP通过p1点圆角路径中点时，将设置数字输出信号枪。当CAP进程启动时，将执行软中断子程序start_trap。



xx1200000174

下一页继续

1.26 CapC - CAP圆周运动指令

Continuous Application Platform (CAP)

续前页

变元

```
CapC Cirpoint ToPoint [\Id] Speed Cdata [\MoveStartTimer] Weavestart
     Weave Zone [\Inpos] Tool [\WObj] [\Track] | [\Corr] [\Time]
     [\T1] [\T2] [\T3] [\T4] [\T5] [\T6] [\T7] [\T8] [\TLoad]
```

`Cirpoint`

数据类型：robtarget

相关机器人的圆弧点。圆弧点是指相关起点与终点间的圆弧上的某个位置。若要获得最好的准确度，则宜把该点放在相关起点与终点的中间附近。如果该点太靠近起点或终点，那么相关机器人就可能发出一条警告。将圆弧点定义为一个已命名的位置，或将其直接保存在相关指令（在指令中用一个*标注）中。

`ToPoint`

数据类型：robtarget

机器人和附加轴的终点。机器人和附加轴的终点被定义为一个已命名位置或直接保存在指令（在指令中用一个*标注）中。

`[\Id]`

（同步标识）

数据类型：identno

同步模式下，MultiMove系统中RAPID运动指令的同步标识。

`Speed`

数据类型：speeddata

无需活跃CAP进程便应用到运动中的速度数据。速度数据定义了工具中心接触点、附加轴和工具姿态调整的速度。如果CAP进程处于活跃状态（未受阻），那么，Cdata参数将定义TCP速度。

`Cdata`

（*CAP*进程数据）

数据类型：capdata

CAP进程数据，有关详尽说明，请参见第*1355*页的*capdata - CAP*数据。

`[\Movestart_timer]`

（时间按秒计）

数据类型：num

同步模式下MultiMove系统中机器人TCP移动的实际开始与进程命令开始之间的时间差上限。

`Weavestart`

（摆动启动数据）

数据类型：weavestartdata

CAP进程的摆动启动数据，有关详尽说明，请参见第*1522*页的*weavestartdata - 摆动启动数据*。

`Weave`

（摆动数据）

数据类型：capweavedata

下一页继续

CAP进程的摆动数据，有关详尽说明，请参见第*1366*页的*capweavedata - CAP*摆动数据。

`Zone`

数据类型：zonedata

相关移动的区域数据。区域数据描述了所生成拐角路径的大小。

`[\Inpos]`

（就位）

数据类型：stoppointdata

该参数用于规定停止点中机械臂TCP位置的收敛准则。停止点数据取代`Zone`参数中的指定区域。

`Tool`

数据类型：tooldata

相关机器人移动时使用的工具。工具中心接触点（TCP）是被移到指定目标位置的一点。

`[\WObj]`

数据类型：wobjdata

指令中机器人位置关联的工件（坐标系）。

用户可以忽略该参数，且如果忽略了该参数，那么相关位置就会与全局坐标系关联起来。另一方面，如果使用了一个固定TCP或若干协同附加轴，那么就必须为一个线性运动（相对于待执行工件）指定该参数。

`[\Track]`

（跟踪传感器数据）

数据类型：captrackdata

该数据结构包含使用路径校正发生传感器和`CapC`所需的数据，请参见第*1363*页的*captrackdata - CAP*跟踪数据。该参数不得与参数`\Corr`一起使用。

`[\Corr]`

（使用校正发生器）

数据类型：switch

该参数让`CapC`从校正发生器读取路径校正，请参见第*131*页的*CorrCon* - 与修正发电机相连。该参数不得与参数`\Track`一起使用。

`[\Time]`

数据类型：num

该参数用于指定机器人和附加轴移动的总时间（按秒计）。然后，该参数将替代相应的速度数据。

`[\T1] [\T2] [\T3] [\T4] [\T5] [\T6] [\T7] [\T8]`

（触发器*x*）

数据类型：triggdata

利用指令`TriggRampAO`、`TriggIO`、`TriggEquip`或`TriggInt`较早期地在程序中定义涉及触发条件和触发活动的变量。

下一页继续

1.26 CapC - CAP圆周运动指令

Continuous Application Platform (CAP)

续前页

[\TLoad]

数据类型：loaddata

参数\TLoad描述了运动中采用的总负载。总负载为工具负载以及工具承载的有效负载。如果采用参数\TLoad，那么不考虑当前loaddata中的tooldata。

如果参数\TLoad被设为load0，那么不考虑该参数，相反，将采用当前tooldata中的loaddata。有关参数TLoad的完整说明，请参见MoveL，第388页的*MoveL - 使机械臂沿直线移动*。

错误处理

有若干不同类型错误可在CapC/CapL指令的错误处理器中进行处理：

- 监控错误
- 传感器特定错误
- MultiMove系统的特定错误
- 从TriggX功能传承的错误
- 其他CAP错误

如果假设接受监控的一个信号没有正确值，或者如果该信号在监控期间改变了值，那么，将设置系统变量ERRNO。

如果不可从跟踪传感器读取任何值，那么将设置系统变量ERRNO。

对于以同步模式运行的MultiMove系统，错误处理器必须关注两个其他错误。一个错误用于报告一些其他应用程序已检测到一项可恢复错误。这能促使在同步RAPID任务中进行可恢复错误处理。如果进程命令开始与同步模式下MultiMove系统中的机器人TCP移动实际开始之间的时间到期，那么将报告另一个错误CAP_MOV_WATCHDOG。将在CapC指令的可选参数Movestart_timer中指定使用的时间。

如果检测到任何异常，将停止执行程序。但是，如果某个错误处理器被编程，那么下列错误可被纠正，不用停止生产。然而，建议对于一些错误（带CAP_XX的错误），这些错误不得呈现给终端用户。将这些错误映射到某个应用程序的特定错误。对于监控错误，可利用指令CapGetFailSigs来获知哪些特定信号失败。

监控错误

系统会生成下列可恢复错误，并在错误处理器中处理这些错误。系统变量ERRNO将被设置成：

CAP_PRE_ERR	当PRE监控列表中出错时，即，当指定时间期（在pre_cond超时中指定）内未满足列表中的条件时，出现此错误。
CAP_PRESTART_ERR	在监控PRE阶段的过程中出错时，出现此错误。
CAP_END_PRE_ERR	当END_PRE监控列表中出错时，即，当指定时间期（在start_cond超时中指定）内未满足列表中的条件时，发生此事件。
CAP_START_ERR	当START监控列表中出错时，即，当指定时间期（在start_cond超时中指定）内未满足列表中的条件时，发生此事件。
CAP_MAIN_ERR	在主阶段监控过程中出错时，将出现此错误。
CAP_ENDMAIN_ERR	当END_MAIN监控列表中出错时，即，当指定时间期（在end_main_cond超时中指定）内未满足列表中的条件时，出现此错误。

下一页继续

CAP_START_POST1_ERR	当START_POST1监控列表中出错时，即，当指定时间期（在end_main_cond超时中指定）内未满足列表中的条件时，发生此事件。
CAP_POST1_ERR	在监控POST1阶段的过程中出错时，出现此错误。
CAP_POST1END_ERR	当END_POST1监控列表中出错时，即，当指定时间期（在end_main_cond超时中指定）内未满足列表中的条件时，出现此错误。
CAP_START_POST2_ERR	当START_POST1监控列表中出错时，即，当指定时间期（在end_main_cond超时中指定）内未满足列表中的条件时，发生此事件。
CAP_POST2_ERR	在监控POST2阶段的过程中出错时，出现此错误。
CAP_POST2END_ERR	当END_POST2监控列表中出错时，即，当指定时间期（在end_main_cond超时中指定）内未满足列表中的条件时，出现此错误。 如果在同一阶段靠两个不同信号完成监控，两个信号均失败，那么设置SetupSuperv的第一个信号是产生错误的信号。

传感器相关错误

系统会生成下列可恢复错误，并在错误处理器中处理这些错误。系统变量ERRNO将被设置成：

CAP_TRACK_ERR	从传感器读取数据并且随后未收到有效数据时，出现跟踪错误。出现这种情况的原因之一在于传感器不能指示焊缝。
CAP_TRACKSTA_ERR	未能从激光跟踪传感器读取有效数据时，出现跟踪启动错误。
CAP_TRACKCOR_ERR	偏移量计算出错时，出现跟踪校正错误。
CAP_TRACKCOM_ERR	机器人控制柜和传感器设备之间的通信被破坏。
CAP_TRACKPFR_ERR	如果跟踪期间发生电源故障，那么无法继续跟踪。
CAP_SEN_NO_MEAS	控制柜未从传感器取得有效测量值。
CAP_SEN_NOREADY	传感器还未就绪。
CAP_SEN_GENERRO	出现一般传感器错误。
CAP_SEN_BUSY	传感器忙，无法回复请求。
CAP_SEN_UNKNOWN	发送给传感器的指令对传感器来说是未知的。
CAP_SEN_ILLEGAL	发送给传感器的变量或块编号是非法的。
CAP_SEN_EXALARM	传感器中出现外部报警。
CAP_SEN_CAALARM	传感器中出现摄像机报警。
CAP_SEN_TEMP	传感器温度超出范围。
CAP_SEN_VALUE	发送给传感器的值超出范围。
CAP_SEN_CAMCHECK	摄像机检查失败。
CAP_SEN_TIMEOUT	传感器未在超时范围内做出响应。

下一页继续

1.26 CapC - CAP圆周运动指令

Continuous Application Platform (CAP)

续前页

MultiMove系统中可能出现的错误

系统会生成下列可恢复错误，并在错误处理器中处理这些错误。系统变量ERRNO将被设置成：

ERR_PATH_STOP	采用同步运动时，当控制一个机械单元的应用程序检测到一个可恢复错误，并通知其他应用程序此项错误，那么，此错误被报告。如果从指令CapC收到此错误代码，那么，此错误是对另一个错误的反应。在MultiMove系统以同步模式采用运动指令的所有任务应让错误处理器定义此ERRNO值。

从TriggX传承的错误

指令CapC基于指令TriggC。因此，如在TriggC中一样，您可以获取并处理错误ERR_AO_LIM和ERR_DIPLAG_LIM。

系统变量ERRNO将发送至：

ERR_AO_LIM	如果指定模拟输出信号AOp/AOutput的编程ScaleValue/SetValue参数处于一些关联TriggSpeed/TriggRampAO指令中，那么，对于模拟信号以及此指令的编程Speed，结果超出限制。系统变量ERRNO设为ERR_AO_LIM。
ERR_DIPLAG_LIM	如果一些关联TriggSpeed指令中的编程DipLag参数相对于采用的系统参数*Event Preset Time*来说过大，那么，系统变量ERRNO设为ERR_DIPLAG_LIM。

其他CAP错误

系统会生成下列可恢复错误，并在错误处理器中处理这些错误。系统变量ERRNO将被设置成：

CAP_NOPROC_END	当不用应用工艺，而是采用指令CapNoProcess来运行特定距离，并到达此距离的终点时，出现此项错误。此错误并不是真正的错误，但采用了错误恢复机制。
CAP_MOV_WATCHDOG	当指定了开关\Movestart_timer，并且进程开始（MAIN_STARTED）和机器人运动开始之间的时间超出用开关指定的时间时，出现此项错误。

程序执行

有关线性运动信息，请参见第*388*页的*MoveL - 使机械臂沿直线移动*。

有关发生触发事件的线性运动信息，请参见第*785*页的*TriggL - 关于事件的机械臂线性运动*。

CAP进程

同时以自动模式和手动模式连续执行期间，CAP进程在运行，除非CAP进程受阻。这意味着，控制CAP进程的所有数据（即，Cdata、Weavestart、Weave和Movestart_timer）均被采用。在这些模式下，所有CAP触发活动被执行，请参见第*227*页的*ICap - 将CAP事件与软中断子程序关联起来*。

在所有其他执行模式下，CAP进程未运行，并且CapC指令像MoveC指令一样发挥作用。

触发条件[\T1]至[\T8]

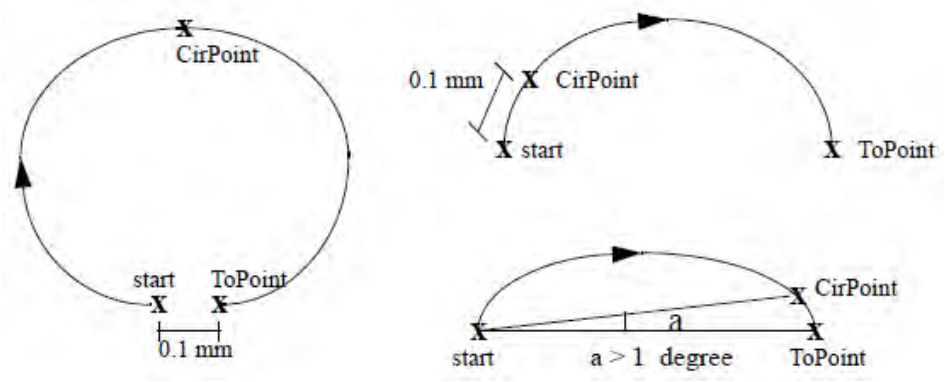
因为当机械臂定位越来越接近终点时，触发条件得以满足，因此，实施指定触发活动。在指令终点前的一定距离处、或在指令起点后一定距离处，或在指令终点前的特定时间点（限于短时间），触发条件得以满足。

下一页继续

步进执行期间，I/O活动被执行，但中断子程序未运行。步退执行期间，根本未执行任何触发活动。

限制

在可以如何放置*CirPoint*和*ToPoint*方面存在一些限制，如下图所示。



xx1200000175

- 起点到ToPoint之间的最短距离为0.1 mm。
- 起点到CirPoint之间的最短距离为0.1 mm。
- 从起点到CirPoint和ToPoint之间形成的最小夹角为1度。

接近极限时，准确度可能不好，比如，如果在圆弧上起点和ToPoint接近彼此，那么倾斜圆弧引起的错误可能远超出被编程的各点的准确度。

机器人停止在圆弧路径上时，禁止执行模式在前进和后退之间相互切换，否则将导致发出错误消息。

在TCP处于圆弧点和终点之间的情况下，决不能从一开始就启动指令CapC（或包含圆周运动的任何其他指令）。否则，机器人不会遵循编程路径（相较编程方向，沿圆弧路径处于另一方向）。

请确保机器人在程序执行期间可到达圆弧点，在必要时分割圆弧段。

如果当前起点偏离常规起点，造成指令CapC的总定位长度短于常规，那么可能出现这样的情况：在同样的位置，若干或所有触发条件即刻被满足。在此类情况下，触发活动执行序列将不明。对于一项“未完成的运动”，用户程序中的程序逻辑不可基于触发活动的常规序列。

语法

```
CapC
  [CirPoint ':='] < expression (IN) of robtarget >
  [ToPoint ':='] < expression (IN) of robtarget >
  ['\' Id ':=' < expression (IN) of identno > ] ','
  [Speed ':='] < expression (IN) of speeddata >
  [Cdata ':='] < persistent (PERS) of capdata >
  ['\' Movestart_timer ':=' < expression (IN) of num > ] ','
  [Weavestart ':='] <persistent (PERS) of weavestartdata >
  [Weave ':='] < persistent (PERS) of capweavedata >
  [Zone ':='] < expression (IN) of zonedata >
  ['\' Inpos ':=' < expression (IN) of stoppointdata >] ','
  [Tool ':='] < persistent (PERS) of tooldata >
```

下一页继续

1.26 CapC - CAP圆周运动指令

Continuous Application Platform (CAP)

续前页

```
['\' WObj ':=' < persistent (PERS) of wobjdata > ]
['\' Track ':=' < persistent (PERS) of captrackdata > ]
|[ '\' Corr]
['\' Time ':=' < expression (IN) of num > ]
['\' T1 ':=' < variable (VAR) of triggdata > ]
['\' T2 ':=' < variable (VAR) of triggdata > ]
['\' T3 ':=' < variable (VAR) of triggdata > ]
['\' T4 ':=' < variable (VAR) of triggdata > ]
['\' T5 ':=' < variable (VAR) of triggdata > ]
['\' T6 ':=' < variable (VAR) of triggdata > ]
['\' T7 ':=' < variable (VAR) of triggdata > ]
['\' T8 ':=' < variable (VAR) of triggdata > ]
['\' TLoad ':=' < persistent (PERS) of loaddata > ] ';'
```

相关信息

信息，关于	请参阅
Continuous Application Platform	应用手册 - *Continuous Application Platform*
圆形运动	第*341*页的*MoveC - 使机械臂沿圆周移动*
触发时的圆周移动	第*747*页的*TriggC - 关于事件的机械臂圆周移动*
CAP数据定义	第*1355*页的*capdata - CAP数据*
摆动启动数据定义	第*1522*页的*weavestartdata - 摆动启动数据*
摆动数据定义	第*1366*页的*capweavedata - CAP摆动数据*
跟踪数据定义	第*1363*页的*captrackdata - CAP跟踪数据*

1.27 CapCondSetDO - 设置TCP停止时的数字输出信号

手册用法

CapCondSetDO用于定义数据输出信号及其值。在CAP序列完成前，当CAP指令（CapL或CapC）执行期间，运行CAP的机器人TCP停止移动时，将设置数字输出信号及其值。

利用CAP指令InitSuperv来清除这类信号的已有定义。

基本示例

```
CapCondSetDO do15, 1;
```

当TCP停止时，信号do15设为1。

```
CapCondSetDO weld, off;
```

当TCP停止时，信号weld设为off。

变元

```
CapCondSetDO Signal Value
```

Signal

数据类型：signaldo

待改变信号的名称。

Value

数据类型：dionum

信号的期望值，0或1。

指定Value	将数字信号输出设置为
0	0
除0以外的任意值	1

限制

信号的最终值取决于信号配置。如果在系统参数中，信号被颠倒，那么，实体通道的值将相反。

每个RAPID任务最多可以设置10个信号。

语法

```
CapCondSetDO
  [Signal ':='] < variable (VAR) of signaldo > ','
  [Value ':='] < expression (IN) of dionum > ';'
```

相关信息

信息，关于	请参阅
Continuous Application Platform	应用手册 - *Continuous Application Platform*
InitSuperv指令	第*258*页的*InitSuperv - 重置CAP*的所有监控
SetupSuperv指令	第*599*页的*SetupSuperv - 设置CAP*信号监控条件

下一页继续

1.27 CapCondSetDO - 设置TCP停止时的数字输出信号

Continuous Application Platform (CAP)

续前页

信息，关于	请参阅
RemoveSuperv指令	第505页的*RemoveSuperv - 撤除一个信号的条件*

1.28 CapEquiDist - 产生等距事件

手册用法

CapEquiDist用于让CAP在CAP路径上产生等距RAPID事件（EQUIDIST）。第一个事件产生在CAP指令序列中的第一个CAP指令的起点处，凭借RAPID，可以利用ICap来预定此事件。

基本示例

```
VAR intnum intno_equi;

PROC main()
  ......
  IDelete intno_equi;
  Connect intno_equi equi_trp;
  ICap intno_equi, EQUIDIST
  ......
  CapEquiDist\Distance:=5.0;
  MoveL p60, v1000, fine, tWeldGun;
  CapL p_fig3_l_1, v500, cd, wsd, cwd, z10, tWeldGun;
  CapL p_fig3_l_2, v500, cd, wsd, cwd, fine, tWeldGun;
  ......
  CapEquiDist\Reset;
  MoveL p70, v1000, fine, tWeldGun;
  CapL p_fig3_l_3, v500, cd, wsd, cwd, fine, tWeldGun;
  ......

  ERROR
    Retry;
ENDPROC

TRAP equi_trp
  ! do whatever you want, but it must not take too long time
ENDTRAP
```

在此示例中，在第一条CAP路径上将产生事件EQUIDIST。将随着各区域，通过若干CAP指令在此路径上每隔5 mm发送一次该事件。

变元

```
CapEquiDist [\Distance] [\Reset]
```

[\Distance]

距离按毫米计。

数据类型：num

此可选参数配备的数据定义了两个连续等距事件之间的距离，单位按毫米计。

[\Reset]

对事件产生进行重置。

数据类型：switch

下一页继续

1.28 CapEquiDist - 产生等距事件

Continuous Application Platform (CAP)

续前页

如果存在此开关，那么，事件产生将被重置，即，在CapL/CapC路径上不再产生等距事件。此开关优先于\Distance开关。

限制

如果CAP路径相较事件距离而言较长，那么系统可能用完事件资源，并发送错误消息**50368等距事件之间距离过短。**

语法

```
CapEquiDist
  ['\' Distance ':=' < expression (IN) of num >]
  ['\' Reset] ';'
```

相关信息

信息，关于	请参阅
Continuous Application Platform	应用手册 - *Continuous Application Platform*

1.29 CapL - CAP线性运动指令

手册用法

CapL将用于沿线性路径将工具中心接触点（TCP）移动到给定终点，同时控制一个连续进程。此外，还可以最多将八个事件与CapL关联起来。将利用指令TriggRampAO、TriggIO、TriggEquip、TriggInt、TriggCheckIO或TriggSpeed来定义各个事件。

基本示例

Example1

利用CapL进行线性运动。

```
CapL p1, v100, cdata, weavestart, weave, z50, gun1;
```

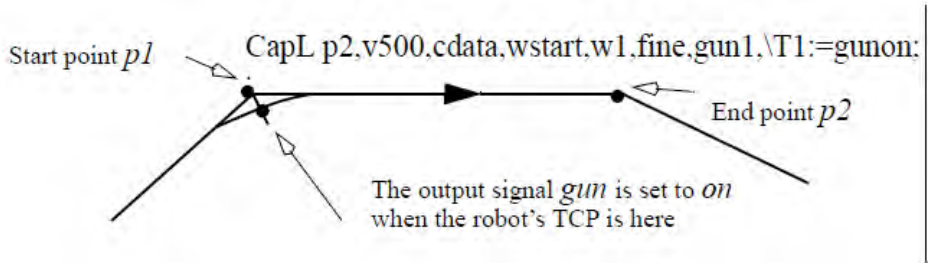
工具的TCP（gun1）以区域数据z50以及cdata中规定的速度，沿线性路径移动到位置p1。

例 2

凭用户事件和CAP事件进行圆周运动。

```
VAR intnum start_intno;
...
PROC main()
  VAR triggdata gunon;
  IDelete start_intno;
  CONNECT start_intno WITH start_trap;
  ICap start_intno, CAP_START;
  TriggIO gunon, 0 \Start \DOp:=gun, on;
  MoveJ p1, v500, z50, gun1;
  CapL p2, v500, cdata, wstart, w1, fine, gun1 \T1:=gunon;
ENDPROC

TRAP start_trap
  !This routine is executed when event CAP_START arrives
ENDTRAP
```

当机器人TCP通过p1点圆角路径中点时，将设置数字输出信号枪。当CAP进程启动时，将执行软中断子程序start_trap。



xx1200000173

下一页继续

1.29 CapL - CAP线性运动指令
Continuous Application Platform (CAP)
续前页

变元

```
CapLToPoint [\Id] Speed Cdata [\MoveStartTimer] Weavestart Weave
     Zone [\Inpos] Tool [\WObj] [\Track] | [\Corr] [\Time] [\T1]
     [\T2] [\T3] [\T4] [\T5] [\T6] [\T7] [\T8] [\TLoad]
```

`ToPoint`

数据类型：robtarget

机器人和附加轴的终点。机器人和附加轴的终点被定义为一个已命名位置或直接保存在指令（在指令中用一个*标注）中。

`[ \Id ]`

（同步标识）

数据类型：identno

同步模式下，MultiMove系统中RAPID运动指令的同步标识。

`Speed`

数据类型：speeddata

无需活跃CAP进程便应用到运动中的速度数据。速度数据定义了工具中心接触点、附加轴和工具姿态调整的速度。如果CAP进程处于活跃状态（未受阻），那么，Cdata参数将定义TCP速度。

`Cdata`

（*CAP*进程数据）

数据类型：capdata

CAP进程数据，有关详尽说明，请参见第*1355*页的*capdata - CAP*数据。

`[ \Movestart_timer ]`

（时间按秒计）

数据类型：num

同步模式下MultiMove系统中机器人TCP移动的实际开始与进程命令开始之间的时间差上限。

`Weavestart`

（摆动启动数据）

数据类型：weavestartdata

CAP进程的摆动启动数据，有关详尽说明，请参见第*1522*页的*weavestartdata - 摆动启动数据*。

`Weave`

（摆动数据）

数据类型：capweavedata

CAP进程的摆动数据，有关详尽说明，请参见第*1366*页的*capweavedata - CAP摆动数据*。

`Zone`

数据类型：zonedata

相关移动的区域数据。区域数据描述了所生成拐角路径的大小。

下一页继续

[\Inpos]

（就位）

数据类型：stoppointdata

该参数用于规定停止点中机械臂TCP位置的收敛准则。停止点数据取代Zone参数中的指定区域。

Tool

数据类型：tooldata

相关机器人移动时使用的工具。工具中心接触点（TCP）是被移到指定目标位置的一点。

[\WObj]

数据类型：wobjdata

指令中机器人位置关联的工件（坐标系）。

用户可以忽略该参数，且如果忽略了该参数，那么相关位置就会与全局坐标系关联起来。另一方面，如果使用了一个固定TCP或若干协同附加轴，那么就必须为一个线性运动（相对于待执行工件）指定该参数。

[\Track]

（跟踪传感器数据）

数据类型：captrackdata

该数据结构包含使用路径校正发生传感器和CapL所需的数据，请参见第*1363*页的 *captrackdata - CAP*跟踪数据。该参数不得与参数\Corr一起使用。

[\Corr]

（使用校正发生器）

数据类型：switch

该参数让CapL从校正发生器读取路径校正，请参见第*131*页的*CorrCon - 与修正发电机相连*。该参数不得与参数\Track一起使用。

[\Time]

数据类型：num

该参数用于指定机器人和附加轴移动的总时间（按秒计）。然后，该参数将替代相应的速度数据。

[\T1]至[\T8]

（触发器*x*）

数据类型：triggdata

利用指令TriggRampAO、 TriggIO、TriggEquip或TriggInt较早期地在程序中定义涉及触发条件和触发活动的变量。

[\TLoad]

数据类型：loaddata

参数\TLoad描述了运动中采用的总负载。总负载为工具负载以及工具承载的有效负载。如果采用参数\TLoad，那么不考虑当前loaddata中的tooldata。

下一页继续

如果参数\TLoad被设为load0，那么不考虑该参数，相反，将采用当前tooldata中的loaddata。有关参数TLoad的完整说明，请参见MoveL，第388页的*MoveL - 使机械臂沿直线移动*。

错误处理

有若干不同类型错误可在CapC/CapL指令的错误处理器中进行处理：

- 监控错误
- 传感器特定错误
- MultiMove系统的特定错误
- 从TriggX功能传承的错误
- 其他CAP错误

如果假设接受监控的一个信号没有正确值，或者如果该信号在监控期间改变了值，那么，将设置系统变量ERRNO。

如果不可从跟踪传感器读取任何值，那么将设置系统变量ERRNO。

对于以同步模式运行的MultiMove系统，错误处理器必须关注两个其他错误。一个错误用于报告一些其他应用程序已检测到一项可恢复错误。这能促使在同步RAPID任务中进行可恢复错误处理。如果进程命令开始与同步模式下MultiMove系统中的机器人TCP移动实际开始之间的时间到期，那么将报告另一个错误CAP_MOV_WATCHDOG。将在CapL指令的可选参数Movestart_timer中指定使用的时间。

如果检测到任何异常，将停止执行程序。但是，如果某个错误处理器被编程，那么下列错误可被纠正，不用停止生产。然而，建议对于一些错误（带CAP_XX的错误），这些错误不得呈现给终端用户。将这些错误映射到某个应用程序的特定错误。对于监控错误，可利用指令CapGetFailSigs来获知哪些特定信号失败。

监控错误

系统会生成下列可恢复错误，并在错误处理器中处理这些错误。系统变量ERRNO将被设置成：

CAP_PRE_ERR	当PRE监控列表中出错时，即，当指定时间期（在pre_cond超时中指定）内未满足列表中的条件时，出现此错误。
CAP_PRESTART_ERR	在监控PRE阶段的过程中出错时，出现此错误。
CAP_END_PRE_ERR	当END_PRE监控列表中出错时，即，当指定时间期（在start_cond超时中指定）内未满足列表中的条件时，发生此事件。
CAP_START_ERR	当START监控列表中出错时，即，当指定时间期（在start_cond超时中指定）内未满足列表中的条件时，发生此事件。
CAP_MAIN_ERR	在主阶段监控过程中出错时，将出现此错误。
CAP_ENDMAIN_ERR	当END_MAIN监控列表中出错时，即，当指定时间期（在end_main_cond超时中指定）内未满足列表中的条件时，出现此错误。
CAP_START_POST1_ERR	当START_POST1监控列表中出错时，即，当指定时间期（在end_main_cond超时中指定）内未满足列表中的条件时，发生此事件。
CAP_POST1_ERR	在监控POST1阶段的过程中出错时，出现此错误。

下一页继续

CAP_POST1END_ERR	当END_POST1监控列表中出错时，即，当指定时间期（在end_main_cond超时中指定）内未满足列表中的条件时，出现此错误。
CAP_START_POST2_ERR	当START_POST1监控列表中出错时，即，当指定时间期（在end_main_cond超时中指定）内未满足列表中的条件时，发生此事件。
CAP_POST2_ERR	在监控POST2阶段的过程中出错时，出现此错误。
CAP_POST2END_ERR	当END_POST2监控列表中出错时，即，当指定时间期（在end_main_cond超时中指定）内未满足列表中的条件时，出现此错误。 如果在同一阶段靠两个不同信号完成监控，两个信号均失败，那么设置SetupSuperv的第一个信号是产生错误的信号。

传感器相关错误

系统会生成下列可恢复错误，并在错误处理器中处理这些错误。系统变量ERRNO将被设置成：

CAP_TRACK_ERR	从传感器读取数据并且随后未收到有效数据时，出现跟踪错误。出现这种情况的原因之一在于传感器不能指示焊缝。
CAP_TRACKSTA_ERR	未能从激光跟踪传感器读取有效数据时，出现跟踪启动错误。
CAP_TRACKCOR_ERR	偏移量计算出错时，出现跟踪校正错误。
CAP_TRACKCOM_ERR	机器人控制柜和传感器设备之间的通信被破坏。
CAP_TRACKPFR_ERR	如果跟踪期间发生电源故障，那么无法继续跟踪。
CAP_SEN_NO_MEAS	控制柜未从传感器取得有效测量值。
CAP_SEN_NOREADY	传感器还未就绪。
CAP_SEN_GENERRO	出现一般传感器错误。
CAP_SEN_BUSY	传感器忙，无法回复请求。
CAP_SEN_UNKNOWN	发送给传感器的指令对传感器来说是未知的。
CAP_SEN_ILLEGAL	发送给传感器的变量或块编号是非法的。
CAP_SEN_EXALARM	传感器中出现外部报警。
CAP_SEN_CAALARM	传感器中出现摄像机报警。
CAP_SEN_TEMP	传感器温度超出范围。
CAP_SEN_VALUE	发送给传感器的值超出范围。
CAP_SEN_CAMCHECK	摄像机检查失败。
CAP_SEN_TIMEOUT	传感器未在超时范围内做出响应。

MultiMove系统中可能出现的错误

系统会生成下列可恢复错误，并在错误处理器中处理这些错误。系统变量ERRNO将被设置成：

ERR_PATH_STOP	采用同步运动时，当控制一个机械单元的应用程序检测到一个可恢复错误，并通知其他应用程序此项错误，那么，此错误被报告。如果从指令CapL收到此错误代码，那么，此错误是对另一个错误的反应。在MultiMove系统以同步模式采用运动指令的所有任务应让错误处理器定义此ERRNO值。

下一页继续

1.29 CapL - CAP线性运动指令

Continuous Application Platform (CAP)

续前页

从TriggX传承的错误

指令CapL基于指令TriggL。因此，如在TriggL中一样，您可以获取并处理错误ERR_AO_LIM和ERR_DIPLAG_LIM。

系统变量ERRNO将发送至：

ERR_AO_LIM	如果指定模拟输出信号AOp/AOutput的编程ScaleValue/SetValue参数处于一些关联TriggSpeed/TriggRampAO指令中，那么，对于模拟信号以及此指令的编程Speed，结果超出限制。系统变量ERRNO设为ERR_AO_LIM。
ERR_DIPLAG_LIM	如果一些关联TriggSpeed指令中的编程DipLag参数相对于采用的系统参数*Event Preset Time*来说过大，那么，系统变量ERRNO设为ERR_DIPLAG_LIM。

其他CAP错误

系统会生成下列可恢复错误，并在错误处理器中处理这些错误。系统变量ERRNO将被设置成：

CAP_NOPROC_END	当不用应用工艺，而是采用指令CapNoProcess来运行特定距离，并到达此距离的终点时，出现此项错误。此错误并不是真正的错误，但采用了错误恢复机制。
CAP_MOV_WATCHDOG	当指定了开关\Movestart_timer，并且进程开始（MAIN_STARTED）和机器人运动开始之间的时间超出用开关指定的时间时，出现此项错误。

程序执行

有关线性运动信息，请参见第*388*页的*MoveL - 使机械臂沿直线移动*。

有关发生触发事件的线性运动信息，请参见第*785*页的*TriggL - 关于事件的机械臂线性运动*。

CAP进程

同时以自动模式和手动模式连续执行期间，CAP进程在运行，除非CAP进程受阻。这意味着，控制CAP进程的所有数据（即，Cdata、Weavestart、Weave和Movestart_timer）均被采用。在这些模式下，所有CAP触发活动被执行，请参见第*227*页的*ICap - 将CAP事件与软中断子程序关联起来*。

在所有其他执行模式下，CAP进程未运行，并且CapL指令像MoveL指令一样发挥作用。

触发条件[\T1]至[\T8]

因为当机械臂定位越来越接近终点时，触发条件得以满足，因此，实施指定触发活动。在指令终点前的一定距离处、或在指令起点后一定距离处，或在指令终点前的特定时间点（限于短时间），触发条件得以满足。

步进执行期间，I/O活动被执行，但中断子程序未运行。步退执行期间，根本未执行任何触发活动。

限制

如果当前起点偏离常规起点，造成指令CapL的总定位长度短于常规（比如，在机器人位置处于终点的情况下，处于CapL起点处），那么可能出现这样的情况：在同样的位置，若干或所有触发条件即刻被满足。在此类情况下，触发活动执行序列将不明。对于一项“未完成的运动”，用户程序中的程序逻辑不可基于触发活动的常规序列。

下一页继续

如果随极短的TCP移动（<1 mm），CapL或CapC指令执行期间出现错误，那么，CAP进程的行为可能未定义。

语法

```
CapL
  [ToPoint ':='] < expression (IN) of robtarget >
  ['\' Id ':=' < expression (IN) of identno >] ','
  [Speed ':='] < expression (IN) of speeddata > ','
  [Cdata ':='] < persistent (PERS) of capdata >
  ['\' Movestart_timer ':=' < expression (IN) of num >] ','
  [Weavestart ':='] <persistent (PERS) of weavestartdata > ','
  [Weave ':='] < persistent (PERS) of capweavedata > ','
  [Zone ':='] < expression (IN) of zonedata >
  ['\' Inpos ':=' < expression (IN) of stoppointdata >] ','
  [Tool ':='] < persistent (PERS) of tooldata >
  ['\' WObj ':=' < persistent (PERS) of wobjdata >]
  ['\' Track ':=' < persistent (PERS) of captrackdata >]
  |[ '\' Corr]
  ['\' Time ':=' < expression (IN) of num >]
  ['\' T1 ':=' < variable (VAR) of triggdata >]
  ['\' T2 ':=' < variable (VAR) of triggdata >]
  ['\' T3 ':=' < variable (VAR) of triggdata >]
  ['\' T4 ':=' < variable (VAR) of triggdata >]
  ['\' T5 ':=' < variable (VAR) of triggdata >]
  ['\' T6 ':=' < variable (VAR) of triggdata >]
  ['\' T7 ':=' < variable (VAR) of triggdata >]
  ['\' T8 ':=' < variable (VAR) of triggdata >]
  ['\' TLoad':=' < persistent (PERS) of loaddata >] ';'
```

相关信息

信息，关于	请参阅
Continuous Application Platform	应用手册 - *Continuous Application Platform*
线性移动	第*388*页的*MoveL - 使机械臂沿直线移动*
触发时的线性移动	第*785*页的*TriggL - 关于事件的机械臂线性运动*
CAP数据定义	第*1355*页的*capdata - CAP数据*
摆动启动数据定义	第*1522*页的*weavestartdata - 摆动启动数据*
摆动数据定义	第*1366*页的*capweavedata - CAP摆动数据*
跟踪数据定义	第*1363*页的*captrackdata - CAP跟踪数据*

1.30 CapLATrSetup - 设置Look-Ahead-Tracker

手册用法

CapLATrSetup（设置*Look-Ahead-Tracker*）将用于设置传感器的Look-Ahead-Tracker类型，比如，激光跟踪器。

传感器接口利用RTP1传输协议，通过串行通道，最多与两个传感器通信。两个通道必须被命名为*laser1:*以及*swg:*。

基本示例

SIO.cfg:

```
COM_TRP:
-Name "SCOUT:" -Type "RTP1"
-Name "digi-ip:" -Type "SOCKDEV" -PhyChannel "LAN1" -RemoteAdress
     "192.168.125.5"
```

RAPID代码：

```
! Define variable numbers
CONST num SensorOn := 6;
CONST num XCoord := 8;
CONST num YCoord := 9;
CONST num ZCoord := 10;
! Sensor calibration frame
PERS pose calibFrame := [[236.4,0.3,96.3],[1,0,0,0]];
! Trackdata
PERS captrackdata captrack1 := ["digi-ip:", [1,10,1,0,0,0,0,0]];

! Set up a Laser Tracker
CapLATrSetup “digi-ip:”,
     calibFrame\SensorFreq:=20\CorrFilter:=5\MaxBlind:=100\MaxIncCorr:=2;

! Request start of sensor measurements
WriteVar "digi-ip:", SensorOn, 1;

! Track using Cap
CapL p_fig1_l_1, v200, cd_event1, wsd_event, cwd_event, z20,
     tWeldGun\Track:=captrack1;

! Stop sensor
WriteVar "digi-ip:", SensorOn, 0;
```

变元

```
CapLATrSetup device CalibFrame CalibPos [\WarnMaxCorr] [\LogFile]
     [\LogSize] [\SensorFreq] [\IpolServoDelay] [\IpolCorrGain]
     [\ServoSensFactor] [\CorrFilter] [\IpolCorrFilter]
     [\ServoCorrFilter] [\ErrRampIn] [\ErrRampOut] [\CBAngle]
     [\MaxBlind] [\MaxIncCorr] [\CalibFrame2] [\CalibFrame3]
```

device

数据类型：string

sio.cfg中定义的装置名称。

下一页继续

calibframe

数据类型：pose

LATR校准坐标系（相对于预定义工具tool0的位置和姿态）。

CalibPos

数据类型：pose

LATR校准偏移量。传感器坐标系调整，其中将路径校正坐标系的起点放置在校准期间采用的工具坐标系水平附近。

[\WarnMaxCorr]

数据类型：switch

如果存在此开关，那么当超出跟踪数据中指定的最大校正限值时，程序执行不会被中断，只会发出警告。

[\Logfile]

数据类型：string

跟踪日志文件名称

[\LogSize]

数据类型：num

跟踪日志环形缓冲区大小，即，跟踪期间可缓冲的传感器测量数。

默认：1000。

[\SensorFreq]

数据类型：num

定义采用的传感器的取样频率（比如，M-Spot-90的取样频率为5Hz）。

最高可取值取决于通信链路及其速度。我们建议不要采用高于20Hz的值。

默认：5 Hz。

[\IpolServoDelay]

数据类型：num

定义ipol任务和servo任务之间的机器人控制柜内部延迟时间间隔。

默认：74 ms。

注意

不要改变默认值！

[\IpolCorrGain]

数据类型：num

定义对ipol施加的校正增益系数。

默认：0.0。

注意

不要改变默认值！

下一页继续

1.30 CapLATrSetup - 设置Look-Ahead-Tracker
Continuous Application Platform (CAP)
续前页

[\ServoSensFactor]

数据类型：num

定义每个传感器读数的servo校正数。

默认：0。

> **注意**
>
> 不要改变默认值！

[\CorrFilter]

数据类型：num

采用校正滤波器值的平均值来定义计算的校正滤波。

默认：1。

> **注意**
>
> 不要改变默认值！

[\IpolCorrFilter]

数据类型：num

采用路径滤波器值的平均值来定义ipol校正滤波。

默认：1。

> **注意**
>
> 不要改变默认值！

[\ServoCorrFilter]

数据类型：num

采用路径servo滤波器值的平均值来定义servo校正滤波。

默认：1。

> **注意**
>
> 不要改变默认值！

[\ErrRampIn]

数据类型：num

定义在传感器读数引起错误后，在多少个传感器读数期间完成逐渐逼近。

默认：1。

[\ErrorRampOut]

数据类型：num

定义在传感器读数引起错误后，在多少个传感器读数期间完成逐渐远离。

默认：1。

下一页继续

[\CBAngle]

数据类型：num

定义3D传感器光束与传感器Z轴之间的角度。

默认：0.0。

[\MaxBlind]

数据类型：num

TCP可移动的最长距离假设，最新校正仍有效。

开始跟踪后，从传感器前方看，MaxBlind距离自动增加。

默认：无限制。

[\MaxIncCorr]

数据类型：num

允许的最大递增校正。

如果TCP递增校正大于\MaxIncCorr，并且指定\WarnMaxCorr，那么机器人将继续沿路径移动，但采用的递增校正不会超出\MaxIncCorr。如果未指定\WarnMaxCorr，那么将报告跟踪错误，程序执行将停止。

默认：5 mm。

[\CalibFrame2]

数据类型：pose

备选LATR校准坐标系编号2（相对于预定义工具tool0的位置和姿态）。

[\CalibFrame3]

数据类型：pose

备选LATR校准坐标系编号3（相对于预定义工具tool0的位置和姿态）。

语法

```
CapLATrSetup
  [device ':='] < expression (IN) of string> ','
  [CalibFrame ':='] < persistent (PERS) of pose > ','
  [CalibPos ':='] < persistent (PERS) of pos >
  [\WarnMaxCorr]
  [\LogFile ':=' < expression (IN) of string >]
  [\LogSize ':=' < expression (IN) of num >]
  [\SensorFreq ':=' < expression (IN) of num >]
  [\IpolServoDelay ':=' < expression (IN) of num >]
  [\IpolCorrGain ':=' < expression (IN) of num >]
  [\ServoSensFactor ':=' < expression (IN) of num >]
  [\CorrFilter ':=' < expression (IN) of num >]
  [\IpolCorrFilter ':=' < expression (IN) of num >]
  [\ServoCorrFilter ':=' < expression (IN) of num >]
  [\ErrRampIn ':=' < expression (IN) of num >]
  [\ErrRampOut ':=' < expression (IN) of num >]
  [\CBAngle ':=' < expression (IN) of num >]
  [\MaxBlind ':=' < expression (IN) of num >]
  [\MaxIncCorr ':=' < expression (IN) of num >]
  [\CalibFrame2 ':=' < persistent (PERS) of pose >]
```

下一页继续

1.30 CapLATrSetup - 设置Look-Ahead-Tracker

Continuous Application Platform (CAP)

续前页

```
[\CalibFrame3 ':=' < persistent (PERS) of pose >] ';'
```

相关信息

信息，关于	请参阅
Sensor Interface	应用手册 - 控制器软件*IRC5*
Continuous Application Platform	应用手册 - *Continuous Application Platform*

1.31 CapNoProcess - 无需进程的情况下，运行CAP

手册用法

CapNoProcess将用于在无需进程的情况下，让CAP运行一定距离。

利用CapNoProcess，可以无需进程，让CAP执行一定距离（单位毫米）。在发生可恢复进程错误的情况下，这很有用。可恢复进程错误可能在定位错误后，在某种程度上，无法重启进程。

在移动距离的起点和终点，路径支持（capdata中的restart_dist分量）将受抑制。

在移动距离的终点，将产生错误，错误编号为CAP_NOPROC_END。

基本示例

```
VAR num skip_dist := 0.0;
VAR bool cap_skip := FALSE;

PROC main()
  ......
  skip_dist := 25.0;
  CapL p_fig3_l_1, v500, cd, wsd, cwd, fine, tWeldGun;
  ......
  skip_dist := 15.0;
  CapL p_fig3_l_3, v500, cd, wsd, cwd, fine, tWeldGun;
  ......

  ERROR
    StorePath;
    TEST ERRNO
    CASE CAP_NOPROC_END:
      IF cap_skip THEN
        ! This is the end of the skip distance
        cap_skip := FALSE;
      ENDIF
    CASE CAP_MAIN_ERR:
      IF skip_dist > 0.0 THEN
        ! This is the start of the skip distance
        CapNoProcess skip_dist;
        cap_skip := TRUE;
      ENDIF
    DEFAULT:
    ENDTEST
    RestoPath;
    StartMoveRetry;
ENDPROC
ENDMODULE
```

在此示例中，在出现可恢复错误CAP_MAIN_ERR后，将针对首个CapL指令，在无需进程的情况下，进行25 mm移动（速度：10 mm/s），并且将针对第二个指令，进行15 mm移动。在该距离的终点处，将产生CAP_NOPROC_END，将重启进程。

下一页继续

1.31 CapNoProcess - 无需进程的情况下，运行CAP

Continuous Application Platform (CAP)

续前页

变元

```
CapNoProcess skip_distance
```

skip_distance

距离按毫米计。

数据类型：num

CapNoProcess具备num变量来作为输入参数，该输入参数定义了移动距离（单位：毫米）。

限制

TCP的移动速度为10 mm/s。最短的移动距离为10 mm。

在同步MultiMove系统中，为不同的同步工艺机器人定义的所有移动距离中的最短距离将为实际距离。

如果移动距离长于当前TCP位置与当前CAP指令序列终点之间的距离，那么不会发生特殊情况：RAPID执行将继续如常进行，不会让机器人停止。

语法

```
CapNoProcess
  [skip_dist ':='] < variable (IN) of num >';'
```

相关信息

信息，关于	请参阅
Continuous Application Platform	应用手册 - *Continuous Application Platform*
InitSuperv指令	第*258*页的*InitSuperv - 重置CAP*的所有监控
SetupSuperv指令	第*599*页的*SetupSuperv - 设置CAP*信号监控条件
RemoveSuperv指令	第*505*页的*RemoveSuperv - 撤除一个信号的条件*

1.32 CapRefresh - 更新CAP数据

手册用法

CapRefresh将用于让CAP进程更新自身的进程数据。比如，可以用于在程序执行期间，调整进程参数。

基本示例

```
PROC PulseSpeed()
  ! Setup a 1 Hz timer interrupt
  CONNECT intno1 WITH TuneTrp;
  ITimer 1, intno1;
  CapL p1, v100, cdata, wstartdata, wdata, fine, gun1;
  IDelete intno1;
ENDPROC

TRAP TuneTrp
  ! Modify the main speed component of active cdata
  IF HighValueFlag = TRUE THEN
    cdata.speed_data.start := 10;
    HighValueFlag := FALSE;
  ELSE
    cdata.speed_data.start := 15;
    HighValueFlag := TRUE;
  ENDIF
  ! Order the process control to refresh process parameters
  CapRefresh;
ENDTRAP
```

在此示例中，速度将按1 Hz频率，在10mm/s和15 mm/s之间切换。

变元

```
CapRefresh [\MainSpeed] [MainWeave] [\StartWeave] [\RestartDist]
```

在无需可选参数的情况下，从当前活跃的CAP指令指定的PERSISTENTRAPID变量，重新读取CAP数据capdata、capweavedata、weavestartdata、captrackdata和movestarttimer（如有）。

[\MainSpeed]

数据类型：switch

如果存在此开关，CAP将重新读取当前活跃的CAP指令的分量capdata.speed_data.main。

[\MainWeave]

数据类型：switch

如果存在此开关，CAP将重新读取当前活跃的CAP指令的分量capweavedata.width、capweavedata. length、capweavedata.bias和capweavedata.active。

[\StartWeave]

数据类型：bool

下一页继续

1.32 CapRefresh - 更新CAP数据

Continuous Application Platform (CAP)

续前页

如果存在此开关，CAP将使用自己的值来替代当前活跃的CAP指令的 `weavestartdata.active`。当前活跃的CAP指令的数据将保持不受影响。

[\RestartDist]

数据类型：`num`

如果存在此开关，CAP将使用自己的值来替代当前活跃的CAP指令的 `weavestartdata.active`。当前活跃的CAP指令的数据将保持不受影响。

语法

```
CapRefresh
  ['\' MainSpeed]
  ['\' MainWeave]
  ['\' Startweave ':=' < expression (IN) of bool >]
  ['\' RestartDist ':=' < expression (IN) of num >] ';'
```

相关信息

信息，关于	请参阅
Continuous Application Platform	应用手册 - *Continuous Application Platform*

1.33 CapWeaveSync - 设置摆动同步信号和电平

手册用法

CapWeaveSync将用于设置摆动同步信号，无需传感器。必须在EIO.cfg中定义I/O信号。

基本示例

RAPID程序

```
PROC main()
  ...
  CapWeaveSync \DoLeft:=do_sync_left \LevelLeft:=80
       \DoRight:=do_sync_right \LevelRight:=80;
  ...
ENDPROC
```

在此示例中，将按80%摆动度来设置信号do_sync_left and do_sync_right。

CapWeaveSync指令只能运行一次，例如从最开始的位置。

变元

```
CapWeaveSync [\Reset] [\DoLeft] [\LevelLeft] [\DoRight]
     [\LevelRight]
```

[\Reset]

数据类型：switch

清除摆动同步数据。

[\DoLeft]

数据类型：signaldo

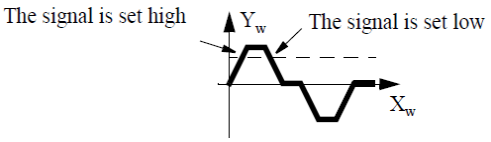
左摆动循环摆动同步数字输出信号

[\LevelLeft]

数据类型：num

摆动运动时的左侧端部位置。指定值为在摆动中心左侧的宽度百分比。摆动超出该点时，数字输出信号将被自动设置到高位（前提是信号已被定义）。

此类坐标可用于凭电弧跟踪进行焊缝跟踪。



xx1200000176

[\LevelLeft]

数据类型：num

摆动运动时的左侧端部位置。指定值为在摆动中心左侧的宽度百分比。摆动超出该点时，数字输出信号将被自动设置到高位（前提是信号已被定义）。

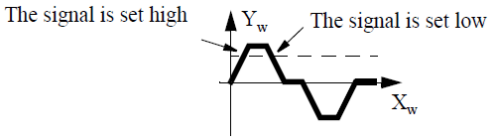
下一页继续

1.33 CapWeaveSync - 设置摆动同步信号和电平

Continuous Application Platform (CAP)

续前页

此类坐标可用于凭电弧跟踪进行焊缝跟踪。



xx1200000176

[\DoRight]

数据类型：signaldo

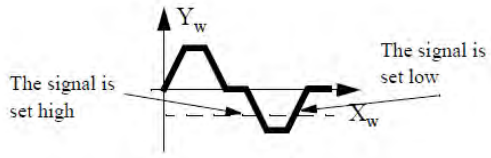
右摆动循环摆动同步数字输出信号

[\LevelRight]

数据类型：num

摆动运动时的右侧端部位置。指定值为在摆动中心右侧的宽度百分比。摆动超出该点时，数字输出信号将被自动设置到高位（前提是信号已被定义）。

此类坐标可用于凭电弧跟踪进行焊缝跟踪。



xx1200000177

程序执行

在无传感器的情况下运行时，将检查和设置定义的信号。

限制

必须在EIO.cfg中定义信号。

无法只利用电平或相应信号。加载RAPID文件时不会导致错误，但会导致CapWeaveSynch指令的RAPID运行时间错误。

语法

```
CapWeaveSync
  ['\' Reset]
  [DoLeft ':=' < expression (IN) of signaldo >]
  [LevelLeft ':=' < expression (IN) of num >]
  [DoRight ':=' < expression (IN) of signaldo >]
  [LevelRight ':=' < expression (IN) of num >] ';'
```

相关信息

信息，关于	请参阅
Continuous Application Platform	应用手册 - *Continuous Application Platform*
capweavedata数据类型	*第1366页的capweavedata - CAP摆动数据*

1.34 CheckProgRef - 检查程序参考

手册用法

CheckProgRef用于在执行期间的任意时间，检查未解决的参考。

基本示例

以下实例介绍了指令CheckProgRef：

例 1

```
Load \Dynamic, diskhome \File:="PART_B.MOD" \CheckRef;
Unload "PART_A.MOD";
CheckProgRef;
```

在这种情况下，包含模块的程序称之为PART_A.MOD。加载新模块PART_B.MOD，其检查所有参考是否就绪。随后，卸载PART_A.MOD。为检查卸载后未解决的参考，完成对CheckProgRef的调用。

程序执行

程序执行推动新的程序任务链接，并检查未解决的参考。

如果在CheckProgRef期间出现错误，则程序未受影响，其仅仅表明程序任务中存在未解决的参考。因此，在更改程序任务（加载或卸载）中模块数后，立即使用CheckProgRef，能够知晓引起链接错误的模块。

该指令亦可用于替代指令Load或WaitLoad中的可选参数\CheckRef。

错误处理

可能会产生下列可恢复错误。错误可以由错误处理程序处理。系统变量ERRNO将设置为：

名称	错误原因
ERR_LINKREF	程序任务包含未解决的参考。

语法

```
CheckProgRef';'
```

相关信息

信息，关于	请参阅
普通程序模块的加载	第*312*页的*Load* - 执行期间，加载普通程序模块
普通程序模块的卸载	第*843*页的*UnLoad* - 执行期间，卸载普通程序模块
开始普通程序模块的加载	第*660*页的*StartLoad* - 执行期间，加载普通程序模块
完成普通程序模块的加载	第*881*页的*WaitLoad* - 将加载的模块与任务相连

1.35 CirPathMode - 圆周路径期间的工具方位调整

手册用法

CirPathMode（*Circle Path Mode*）使其有可能选择不同的模式，从在圆周运动期间，对工具进行再定位。

本指令仅可用于主任务T_ROB1，或者如果在MultiMove系统中，则可用于运动任务中。

基本示例

以下实例介绍了指令CirPathMode：

例 1

```
CirPathMode \PathFrame;
```

有关于所有成功圆周运动期间，从起点至ToPoint的实际路径坐标系中的工具方位调整的标准模式。此为系统中的默认模式。

例 2

```
CirPathMode \ObjectFrame;
```

有关于所有成功圆周运动期间，从起点至ToPoint的实际工件坐标系中的工具方位调整的改进模式。

例 3

```
CirPathMode \CirPointOri;
```

有关于所有成功圆周运动期间，从起点经由编程的CirPoint方位至ToPoint的工具方位调整的改进模式。

例 4

```
CirPathMode \Wrist45;
```

在改进模式下，工具z轴在剖面上的投影将遵循编程的圆周运动顺序。仅使用腕轴4和5。此模式仅用于薄的工件。

例 5

```
CirPathMode \Wrist46;
```

在改进模式下，工具z轴在剖面上的投影将遵循编程的圆周运动顺序。仅使用腕轴4和6。此模式仅用于薄的工件。

例 6

```
CirPathMode \Wrist56;
```

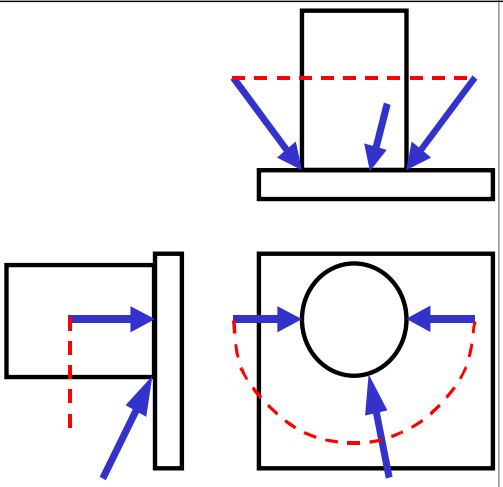
在改进模式下，工具z轴在剖面上的投影将遵循编程的圆周运动顺序。仅使用腕轴5和6。此模式仅用于薄的工件。

下一页继续

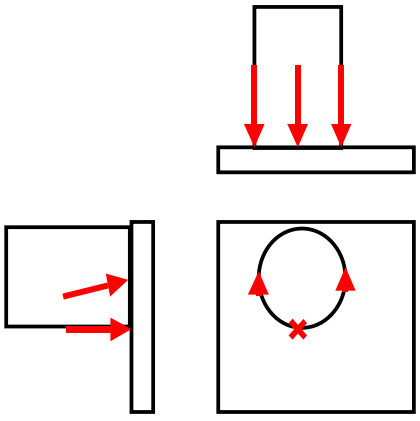
描述

路径坐标系

表格中的图表明了有关于标准模式\PathFrame的工具方位调整。

图示	描述
 xx0500002152	箭头表明了适用于编程点、由腕中心点到工具中心点的工具。图中，腕中心点的路径以虚线表示。 \PathFrame模式使其易于获得与圆柱体周围的工具相同的角。机械臂腕将不会经过CirPoint中的编程方位。

表格中的数字表明，使用标准模式\PathFrame以及固定的工具方位。

图示	描述
 xx0500002153	图片表明通过使用倾斜工具和\PathFrame模式而获得的周期中段的工具的方位。 当使用 \ObjectFrame模式时，与下图进行对比。

下一页继续

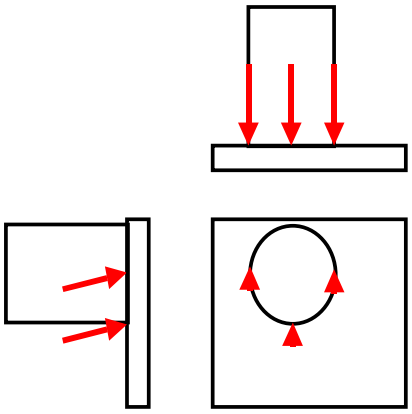
1.35 CirPathMode - 圆周路径期间的工具方位调整

RobotWare - OS

续前页

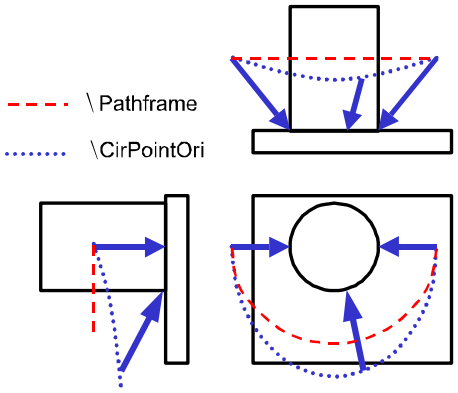
工件坐标系

表格中的数字表明，使用改进模式\ObjectFrame以及固定的工具方位。

图示	描述
 xx0500002151	本图表明通过使用倾斜工具和\ObjectFrame模式而获得的周期中段的工具的方位。 本模式将以与MoveL相同的方式，进行工具的线性再定位。机械臂腕将不会经过CirPoint中的编程方位。 当使用\PathFrame模式时，与前一张图进行对比。

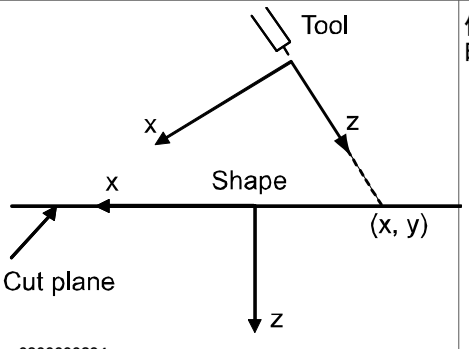
CirPointOri

表格中的图表明了标准模式\PathFrame与改进模式\CirPointOri之间的不同工具重新定位。

图示	描述
 xx0500002150	箭头表明了适用于编程点、由腕中心点到工具中心点的工具。图中，腕中心点的不同路径以虚线表示。 \CirPointOri模式将使机械臂腕经过CirPoint中的编程方位。

腕45 / 腕46 / 腕56

表格中的图表明了使用轴4和5切割模型时所涉及的坐标系。。

图示	描述
 xx0800000294	假定切割束与工具的z轴对准。当执行MoveC指令时，切割平面的坐标系由机械臂的起动位置决定。

下一页继续

变元

```
CirPathMode [\PathFrame] | [\ObjectFrame] | [\CirPointOri] |
     [\Wrist45] | [\Wrist46] | [\Wrist56]
```

`[ \PathFrame ]`

数据类型：`switch`

圆周运动期间，从实际路径坐标系中的起点方位到`ToPoint`方位，持续对工具实施再定位。这便是系统中的标准模式。

`[ \ObjectFrame ]`

数据类型：`switch`

圆周运动期间，从实际工件坐标系中的起点方位到`ToPoint`方位，持续对工具实施再定位。

`[ \CirPointOri ]`

数据类型：`switch`

圆周运动期间，从起点方位到编程`CirPoint`方位，以及进一步到`ToPoint`方位，持续对工具实施再定位。

`[ \Wrist45 ]`

数据类型：`switch`

机械臂将移动轴4和5，以便工具的z轴在切割平面上的投影符合编程周期运动顺序。应当仅将此模式用于薄的工件，因为仅使用了2个腕轴，并因此增加准确性，同时减少控制。

注意

此开关需要选项腕移动。

`[ \Wrist46 ]`

数据类型：`switch`

机械臂将移动轴4和6，以便工具的z轴在切割平面上的投影符合编程周期运动顺序。应当仅将此模式用于薄的工件，因为仅使用了2个腕轴，并因此增加准确性，同时减少控制。

注意

此开关需要选项腕移动。

`[ \Wrist56 ]`

数据类型：`switch`

机械臂将移动轴5和6，以便工具的z轴在切割平面上的投影符合编程周期运动顺序。应当仅将此模式用于薄的工件，因为仅使用了2个腕轴，并因此增加准确性，同时减少控制。

下一页继续

1.35 CirPathMode - 圆周路径期间的工具方位调整

RobotWare - OS

续前页

如果使用不含任何开关的CirPathMode，则结果与CirPointMode \PathFrame相同

> **注意**
>
> 此开关需要选项腕移动。

程序执行

特定圆形工具方位调整模式适用于下次执行的任意类型的机械臂圆周移动（MoveC，SearchC，TriggC，MoveCDO，MoveCSync，ArcC，PaintC等），且始终有效，直至执行新的CirPathMode（或废除的CirPathReori）指令。

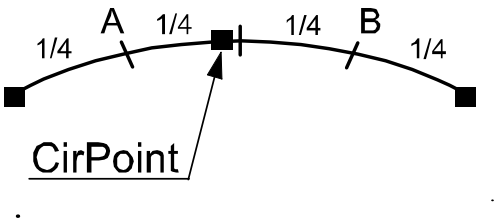
自动设置标准圆形再定位模式（CirPathMode \PathFrame）。

- 当使用重启模式重置RAPID时
- 当加载一则新程序或一个新模块时
- 当从起点开始执行程序时
- 当将程序指针移动到main时
- 当将程序指针移动到子程序时
- 移动程序指针造成执行顺序丢失时

限制

本指令仅影响圆周运动。

根据下图，当运用\CirPointOri模式时，CirPoint必须介于点 A和点B之间，以便使圆周运动经过CirPoint中的编程方位。



xx0500002149

\Wrist45、\Wrist46和\Wrist56模式应当仅用于切割薄的工件，因为当仅使用两个腕轴时，会失去控制工具角的能力。由于主轴锁定，因此不可能实现协调运动。

如果在腕奇点作业，且已经执行了指令SingArea\Wrist，则指令CirPathMode无任何影响，因为系统随后会选择适用于圆周运动的另一个工具方位调整模式（接头插补）。

该指令替代旧指令CirPathReori（即使在未来，亦将奏效，但是将不再进行记录）。

语法

```
CirPathMode
  ['\'PathFrame]
  | ['\'ObjectFrame]
  | ['\'CirPointOri]
  | ['\'Wrist45]
  | ['\'Wrist46]
  | ['\'Wrist56] ';'
```

下一页继续

相关信息

信息，关于	请参阅
插补	技术参考手册 - *RAPID*语言概览
运动设置数据	第*1430*页的*motsetdata - 运动设置数据*
圆形移动指令	第*341*页的*MoveC - 使机械臂沿圆周移动*
腕运动	应用手册 - 控制器软件*IRC5*，腕移动一节

1.36 Clear - 清除数值

手册用法

Clear用于清除数值变量或永久数据对象，即，将数值设置为0。

基本示例

以下实例介绍了指令Clear：

例 1

```
Clear reg1;
```

Reg1得以清除，即，reg1:=0。

例 2

```
CVAR dnum mydnum:=5;
Clear mydnum;
```

mydnum得以清除，即，mydnum:=0。

变元

```
Clear Name | Dname
```

Name

数据类型：num

待清除变量或者永久数据对象的名称。

Dname

数据类型：dnum

待清除变量或者永久数据对象的名称。

语法

```
Clear
  [ Name ':=' ] < var or pers (INOUT) of num >
  | [ Dname ':=' ] < var or pers (INOUT) of dnum > ';'
```

相关信息

信息，关于	请参阅
将变量增加1	第*239*页的*Incr - 增量为1*
将变量减少1	第*142*页的*Decr - 减量为1*
向变量增加任意数值。	第*26*页的*Add*-增加数值
改变任意数据	第*32*页的*":="* - 分配一个数值

1.37 ClearIOBuff - 清除串行通道的输入缓存

手册用法

ClearIOBuff（*Clear I/O Buffer*）用于清除串行通道的输入缓存。删除输入串行通道的所有缓存字符。

基本示例

以下实例介绍了指令ClearIOBuff：

例 1

```
VAR iodev channel1;
...
Open "com1:", channel1 \Bin;
ClearIOBuff channel1;
WaitTime 0.1;
```

清除channel1参考的串行通道输入缓存。等待时间应确保有足够的时间来完成清除操作。

变元

```
ClearIOBuff IODevice
```

IODevice

数据类型：iodev

输入缓存已被清除的串行通道的名称（引用）。

程序执行

删除输入串行通道的所有缓存字符。下一个读取指令将等待来自通道的新输入。

上电失败重启时，将关闭系统中所有打开的文件或串行通道，并将重置iodev型变量中的I/O描述符。

限制

本指令仅可用于串行通道。切勿等待确认完成操作。推荐指令后，允许等待0.1，从而为各个应用留下充足的操作时间。

错误处理

可能会产生下列可恢复错误。错误可以由错误处理程序处理。系统变量ERRNO将设置为：

名称	错误原因
ERR_FILEACC	将本指令用于文件。

语法

```
ClearIOBuff
  [IODevice ':='] <variable (VAR) of iodev>';'
```

下一页继续

1.37 ClearIOBuff - 清除串行通道的输入缓存
RobotWare - OS
续前页

相关信息

信息，关于	请参阅
打开串行通道	技术参考手册 - *RAPID*语言概览
文件和串行通道处理	应用手册 - 控制器软件*IRC5*

1.38 ClearPath - 清除当前路径

手册用法

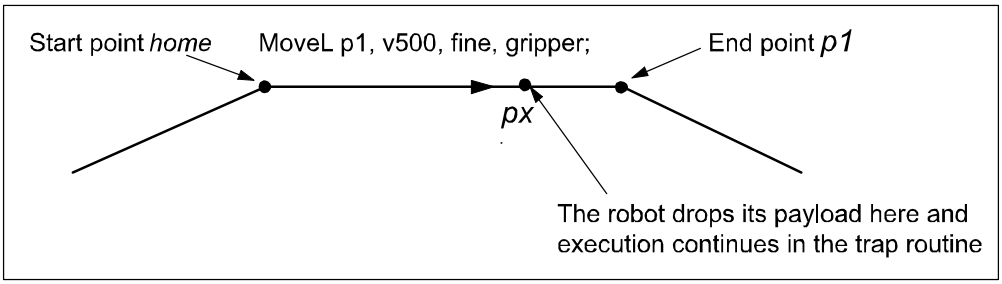
ClearPath（*Clear Path*）清除当前运动路径等级（基础等级或StorePath等级）上的整个运动路径。

使用运动路径，意味着所有运动指令的运动顺序，均已在RAPID中执行，但是并非由机械臂在ClearPath执行期间予以实施。

在能够执行指令ClearPath之前，机械臂必须位于停止点位置，或者必须通过StopMove来停止。

基本示例

以下实例介绍了指令ClearPath：



xx0500002154

在以下程序实例中，机械臂从位置home移动至位置p1。在点px处，信号di1将表明有效负载已经下降。在软中断程序gohome中持续执行。在px处，机械臂将停止移动（开始制动），路径将得以清除，机械臂将移动至位置home。错误将得以提高至调用程序minicycle，且整个用户确定程序周期proc1 ... proc2将再次从开始执行一次。

例 1

```
VAR intnum drop_payload;
VAR errnum ERR_DROP_LOAD := -1;

PROC minicycle()
  BookErrNo ERR_DROP_LOAD;
  proc1;
  ...
  ERROR (ERR_DROP_LOAD)
    ! Restart the interupted movement on motion base path level
    StartMove;
    RETRY;
ENDPROC

PROC proc1()
  ...
  proc2;
  ...
ENDPROC
```

下一页继续

1.38 ClearPath - 清除当前路径

Robot Ware - OS

续前页

```
PROC proc2()
  CONNECT drop_payload WITH gohome;
  ISignalDI \Single, di1, 1, drop_payload;
  MoveL p1, v500, fine, gripper;
  ...........
  IDelete drop_payload;
ENDPROC

TRAP gohome
  StopMove \Quick;
  ClearPath;
  IDelete drop_payload;
  StorePath;
  MoveL home, v500, fine, gripper;
  RestoPath;
  RAISE ERR_DROP_LOAD;
  ERROR
    RAISE;
ENDTRAP
```

如果正在运行相同的程序，但是在软中断程序gohome中不含StopMove和ClearPath，在返回位置home之前，机械臂将继续位于p1。

限制

有关于指令ClearPath 的限制实例阐述如下。

实例1 - 限制

```
VAR intnum int_move_stop;
...
PROC test_move_stop()
  CONNECT int_move_stop WITH trap_move_stop;
  ISignalDI di1, 1, int_move_stop;
  MoveJ p10, v200, z20, gripper;
  MoveL p20, v200, z20, gripper;
ENDPROC

TRAP trap_move_stop
  StopMove;
  ClearPath;
  StorePath;
  MoveJ p10, v200, z20, gripper;
  RestoPath;
  StartMove;
ENDTRAP
```

此为ClearPath限制的实例。在机械臂移动至 p10和p20期间，停止进行中的移动，清除运动路径，但是并未采取任何行动，以中断PROC test_move_stop中的有效指令MoveJ p10或MoveL p20。因此，将中断进行中的移动，机械臂将转到TRAP trap_move_stop中的p10，但是并未进一步移动到PROC test_move_stop中的p10或p20。将暂停程序执行。

下一页继续

可通过下文实例2中描述的长跳转来恢复错误，或者通过与指令ProcerrRecovery异步的错误，以解决此问题。

实例2 - 无限制

```
VAR intnum int_move_stop;
VAR errnum err_move_stop := -1;
...
PROC test_move_stop()
  BookErrNo err_move_stop;
  CONNECT int_move_stop WITH trap_move_stop;
  ISignalDI di1, 1, int_move_stop;
  MoveJ p10, v200, z20, gripper;
  MoveL p20, v200, z20, gripper;
  ERROR (err_move_stop)
    StopMove;
    ClearPath;
    StorePath;
    MoveJ p10, v200, z20, gripper;
    RestoPath;
    ! Restart the interupted movement on motion base path level
    StartMove;
    RETRY;
ENDPROC

TRAP trap_move_stop
  RAISE err_move_stop;
  ERROR
    RAISE;
ENDTRAP
```

此为如何使用错误恢复及长跳转连同不含任何限制的ClearPath 的实例。在机械臂移动至p10 和p20期间，停止正在进行中的移动。清除运动路径，且由于通过执行等级边界来恢复错误，中断有效指令MoveJ p10或MoveL p20。因此，将中断正在进行中的移动，且机械臂将转到ERROR handler中的p10，并再次执行PROC test_move_stop中的中断指令MoveJ p10或MoveL p20。

语法

```
ClearPath ';'
```

相关信息

信息，关于	请参阅
停止机械臂的移动	第*690*页的*StopMove* - 停止机械臂的移动
错误恢复	技术参考手册 - *RAPID*语言概览 技术参考手册 - *RAPID*语言内核
异步引起的错误	第*456*页的*ProcerrRecovery* - 由过程运动错误产生和恢复

1.39 ClearRawBytes - 清除原始数据字节数据的内容

手册用法

ClearRawBytes用于设置rawbytes变量至0的所有内容。

基本示例

以下实例介绍了指令ClearRawBytes：

例 1

```
VAR rawbytes raw_data;
VAR num integer := 8
VAR num float := 13.4;

PackRawBytes integer, raw_data, 1 \IntX := DINT;
PackRawBytes float, raw_data, (RawBytesLen(raw_data)+1) \Float4;

ClearRawBytes raw_data \FromIndex := 5;
```

在前4个字节，放置integer的数值（自索引1开始），在自索引5开始的下4个字节，放置float的数值。

例子中的最后指令清除raw_data的内容，自索引5开始，即float将得以清除，但是，integer得以保持在raw_data中。

将raw_data中有效字节的当前长度设置为4。

变元

```
ClearRawBytes RawData [ \FromIndex ]
```

RawData

数据类型：rawbytes

RawData是将被清除的数据容器。

[\FromIndex]

数据类型：num

通过\FromIndex，规定从何处开始清除RawData的内容。将所有内容清除彻底。

如果未指定\FromIndex，则清除始于索引1的所有数据。

程序执行

将指定变量中来自索引1（默认）或者\FromIndex的数据重置为0。

如果\FromIndex得以编程，则将指定变量中有效字节的当前长度设置为0（默认）或（FromIndex - 1）。

语法

```
ClearRawBytes
  [RawData ':=' ] < variable (VAR) of rawbytes>
  ['\'FromIndex ':=' <expression (IN) of num>]';'
```

下一页继续

相关信息

信息，关于	请参阅
rawbytes数据	第*1455*页的原始数据字节 - 原始数据
获取rawbytes数据的长度	第*1193*页的*RawBytesLen* - 获取原始数据字节数据的长度
复制rawbytes数据的内容	第*128*页的*CopyRawBytes* - 复制原始数据字节数据的内容
将DeviceNet标题装入rawbytes数据	第*421*页的*PackDNHeader* - 将*DeviceNet*标题装入原始数据字节数据中。
将数据装入rawbytes数据	第*424*页的*PackRawBytes* - 将数据装入原始数据字节数据
写入rawbytes数据	第*923*页的*WriteRawBytes* - 写入原始数据字节数据
读取rawbytes数据	第*499*页的*ReadRawBytes* - 读取原始数据字节数据
打开来自rawbytes数据的数据	第*845*页的*UnpackRawBytes* - 打开来自原始数据字节数据的数据
文件和串行通道处理	应用手册 - 控制器软件*IRC5*

1.40 ClkReset - 重置用于定时的时钟

手册用法

ClkReset用于重置作为定时用秒表的时钟。

使用时钟之前，可以使用此指令，以确保设置为0。

基本示例

以下实例介绍了指令ClkReset：

例 1

```
ClkReset clock1;
```

重置时钟clock1。

变元

```
ClkReset Clock
```

Clock

数据类型：clock

用于重置的时钟的名称。

程序执行

重置时钟时，将其设置为0。

如果时钟正在运行中，则将使其停止，然后进行重置。

语法

```
ClkReset
  [ Clock ':=' ] < variable (VAR) of clock > ';'
```

相关信息

信息，关于	请参阅
其他时钟指令	技术参考手册 - *RAPID*语言概览

1.41 ClkStart - 启动用于定时的时钟

手册用法

ClkStart用于启动作为定时用秒表的时钟。

基本示例

以下实例介绍了指令ClkStart：

例 1

```
ClkStart clock1;
```

启动时钟clock1。

变元

```
ClkStart Clock
```

Clock

数据类型：clock

用于启动的时钟的名称。

程序执行

启动时钟时，其将运行并持续读秒，直至停止。

当启动程序停止时，时钟继续运行。但是，针对时间的事件可能不再有效。例如，如果程序正在测量输入的等待时间，则程序停止时，可能已经收到了输入。在这种情况下，程序将无法“发现”程序停止时出现的事件。

只要电池为保留包含时钟变量的程序供电，则当机械臂掉电时，时钟会继续运行。

如果时钟正在运行中，则可以进行读数、停止或重置。

更多示例

有关于指令ClkStart的更多例子阐述如下。

例 1

```
VAR clock clock2;
VAR num time;

ClkReset clock2;
ClkStart clock2;
WaitUntil di1 = 1;
ClkStop clock2;
time:=ClkRead(clock2);
```

测量di1成为1所等待的时间。

错误处理

可能会产生下列可恢复错误。错误可以由错误处理程序处理。系统变量 ERRNO 将设置为：

名称	错误原因
ERR_OVERFLOW.	时钟运行4,294,967秒（49天17小时2分钟47秒），随后溢出。

下一页继续

1.41 ClkStart - 启动用于定时的时钟

RobotWare - OS

续前页

语法

```
ClkStart
  [ Clock ':=' ] < variable (VAR) of clock >';'
```

相关信息

信息，关于	请参阅
其他时钟指令	技术参考手册 - *RAPID*语言概览

1.42 ClkStop - 停止用于定时的时钟

手册用法

ClkStop用于停止作为定时用秒表的时钟。

基本示例

以下实例介绍了指令ClkStop：

```
ClkStop clock1;
```

停止时钟clock1。

变元

```
ClkStop Clock
```

Clock

数据类型：clock

用于停止的时钟的名称。

程序执行

当时钟停止时，其将停止运行。

如果时钟停止，则可以进行读数、重启或重置。

错误处理

如果时钟运行4,294,967秒（49天17小时2分钟47秒），则其会溢出，且系统变量ERRNO得以设置为ERR_OVERFLOW。

可以用错误处理器来处理错误。

语法

```
ClkStop
[ Clock ':=' ] < variable (VAR) of clock >';'
```

相关信息

信息，关于	请参阅
其他时钟指令	技术参考手册 - *RAPID*语言概览
更多示例	第*113*页的*ClkStart* - 启动用于定时的时钟

1.43 Close - 关闭文件或者串行通道

手册用法

Close用于关闭文件或串行通道。

基本示例

以下实例介绍了指令Close：

例 1

```
Close channel2;
```

关闭channel2参考的串行通道。

变元

```
Close IODevice
```

IODevice

数据类型：iodev

有待关闭的文件或串行通道的名称（引用）。

程序执行

关闭指定文件或串行通道，且必须在读取或写入之前重新打开。如果定文件或串行通道已经关闭，则忽略该指令。

语法

```
Close
  [IODevice ':='] <variable (VAR) of iodev>';'
```

相关信息

信息，关于	请参阅
打开文件或串行通道	技术参考手册 - *RAPID*语言概览
文件和串行通道处理	应用手册 - 控制器软件*IRC5*

1.44 CloseDir - 关闭路径

手册用法

CloseDir用于关闭同OpenDir平衡的路径。

基本示例

以下实例介绍了指令CloseDir：

例 1

```
PROC lsdir(string dirname)
  VAR dir directory;
  VAR string filename;
  OpenDir directory, dirname;
  WHILE ReadDir(directory, filename) DO
    TPWrite filename;
  ENDWHILE
CloseDir directory;
ENDPROC
```

此例子打印出指定路径下所有文件或子路径的名称。

变元

```
CloseDir Dev
```

Dev

数据类型：dir

通过指令OpenDir，取得有关于路径的变量。

语法

```
CloseDir
  [ Dev ':=' ] < variable (VAR) of dir>';'
```

相关信息

信息，关于	请参阅
目录	第*1384*页的*dir - 路径结构*
建立路径	第*322*页的*MakeDir - 创建新路径*
打开路径	第*419*页的*OpenDir - 打开路径*
读取路径	第*1197*页的*ReadDir - 读取路径中的下个条目*
删除路径	第*503*页的*RemoveDir - 删除路径*
删除文件	第*504*页的*RemoveFile - 删除文件*
重命名文件	第*507*页的*RenameFile - 重命名文件*
文件和串行通道处理	应用手册 - 控制器软件*IRC5*

1.45 Comment - 备注

手册用法

Comment仅用于使程序更易于理解。其不会对程序的执行产生影响。

基本示例

以下实例介绍了指令Comment：

例 1

```
! Goto the position above pallet
MoveL p100, v500, z20, tool1;
```

将备注插入到程序中，使其更易于理解。

变元

```
! Comment
```

Comment

文本串

任意文本。

程序执行

执行该指令时，未出现任何情况。

语法

```
'!' {<character>} <newline>
```

相关信息

信息，关于	请参阅
备注中许可的字符	技术参考手册 - *RAPID*语言概览
数据和程序声明中的备注	技术参考手册 - *RAPID*语言概览

1.46 Compact IF - 如果满足条件，那么...（一个指令）

手册用法

当单个指令仅在满足给定条件的情况下执行时，使用Compact IF 。
如果将执行不同的指令，则根据是否满足特定条件，使用IF指令。

基本示例

以下实例介绍了指令CompactIF：

例 1

```
IF reg1 > 5 GOTO next;
```

如果reg1 大于5，在next标签处继续程序执行。

例 2

```
IF counter > 10 Set do1;
```

如果counter > 10，则设置do1信号。

变元

```
IF Condition ...
```

Condition

数据类型：bool
必须满足与执行指令相关的条件。

语法

```
IF <conditional expression> ( <instruction> | <SMT>) ';'
```

相关信息

信息，关于	请参阅
条件（逻辑表达式	技术参考手册 - *RAPID*语言概览
如果使用若干指令	第*237*页的*IF* - 如果满足条件，那么...；否则...

1.47 ConfJ - 接头移动期间,控制配置

手册用法

ConfJ(*Configuration Joint*)用于确定在接头运动期间,机械臂的配置是否得到控制。如果未受到控制，则机械臂有时可以使用不同于编程配置的配置。

使用ConfJ \Off，机械臂无法切换主轴配置 – 其将寻找一种同当前配置相同的主轴配置的解决方案，但是其运动至最接近轴4和6的腕配置。

本指令仅可用于主任务T_ROB1，或者如果在MultiMove系统中，则可用于运动任务中。

基本示例

以下实例介绍了指令ConfJ：

例 1

```
ConfJ \Off;
MoveJ *, v1000, fine, tool1;
```

机械臂运动至编程位置和方位。如果可通过不同的轴配置，以若干种不同的方式达到该位置，则可能选择最接近的位置。

例 2

```
ConfJ \On;
MoveJ *, v1000, fine, tool1;
```

机器人移动到编程设定的位置、姿态和轴配置。

变元

```
ConfJ [\On] | [\Off]
```

[\On]

数据类型：switch

机器人移动到编程设定的位置，同时配置参数与confdata中给定的配置参数相等或接近。

如果正在执行程序位移或路径纠正，长距离移动的风险会增加，因为编程配置数据的基础是原始位置。

IRB5400机器人将移动到编程设定的轴配置或到接近编程设定的轴配置。

[\Off]

数据类型：switch

机器人使用最近的轴配置移动到编程设定的位置。

程序执行

如果选择了参数\On（或不带参数）机器人会使用等于或接近给定配置参数的将机器人移动到编程设定的位置。

如果正在执行程序位移或路径纠正，长距离移动的风险会增加，因为编程配置数据的基础是原始位置。

如果选定参数\Off，则机械臂始终移动至最接近的轴配置。如果已经以手动形式错误地指定配置，或者如果已经实施了程序位移，则可能与已编程的轴配置有所不同。

下一页继续

对配置进行控制（ConfJ \On）默认有效。其自动设置：

- 当使用重启模式重置RAPID时
- 当加载一则新程序或一个新模块时
- 当从起点开始执行程序时
- 当将程序指针移动到main时
- 当将程序指针移动到子程序时
- 移动程序指针造成执行顺序丢失时

语法

```
ConfJ
  [ '\' On] | [ '\' Off]';'
```

相关信息

信息，关于	请参阅
处理不同的配置	技术参考手册 - *RAPID*语言概览
线性运动期间的机械臂配置	第*122*页的*ConfL - 线性运动期间,监测配置*
运动设置数据	第*1430*页的*motsetdata - 运动设置数据*

1.48 ConfL - 线性运动期间,监测配置

手册用法

ConfL（*Configuration Linear*）用于确定是否在线性或圆周运动期间监测机械臂的配置。如果未对其进行监测，则执行期间的配置可能与编程期间的配置有所不同。当模式变更为接头移动时，亦可能导致意外清除机械臂运动。

本指令仅可用于主任务T_ROB1，或者如果在MultiMove系统中，则可用于运动任务中。

> **注意**
>
> 针对IRB 5400，机械臂监测始终独立于ConfL中的指定内容。

基本示例

以下实例介绍了指令ConfL：

例 1

```
ConfL \On;
MoveL *, v1000, fine, tool1;
```

当不可能从当前位置达到编程配置时，停止程序执行。

例 2

```
SingArea \Wrist;
ConfL \On;
MoveL *, v1000, fine, tool1;
```

机械臂运动至编程位置、方位和腕轴配置。如果不可能，则停止程序执行。

例 3

```
ConfL \Off;
MoveL *, v1000, fine, tool1;
```

机械臂运动至编程位置和方位，但是与可能的轴配置最为接近，其可能与编程情形有所不同。

变元

```
ConfL [\On]|[\Off]
```

`[ \On ]`

数据类型：switch

监测机械臂配置。

`[ \Off ]`

数据类型：switch

未监测机械臂配置。

程序执行

线性或者圆周运动期间，机械臂始终运动至具有与可能的轴配置最接近的轴配置的编程位置和方位。如果使用参数\On（或无参数），那么一旦存在无法从当前位置达到编程位置配置的风险，随即停止程序执行。确定的方式随机械臂类型而有所不同，参见第*1375*页的*confdata - 机械臂配置数据*。

下一页继续

在开始进行指定运动之前，进行验证，以查看是否可能实现编程配置。如果不可能，则停止程序。完成运动时（在区域或精点中），同时验证机械臂已经达到编程配置。

如果使用SingArea\Wrist，则机械臂始终运动至编程腕轴配置。

如果使用参数\Off，则未进行监测。

在因配置错误而导致停止后，可能以手动模式重启RAPID程序。注意，在这种情况下，由于出现报告中的错误，机械臂将不可能运动至正确配置。

针对ConfL\On和\Off，避免出现问题的简单法则是插入中间点，从而使各个轴在点与点之间的运动小于180度。

如果ConfL\Off与大幅度运动一同使用，其可以导致直接停止，或者随后在程序出现50050 Position outside reach或50080 Position not compatible错误时停止。在带有ConfL\Off的程序中，推荐运动至已知配置点，并将“ConfJ\On+MoveJ”或“ConfL\On+SingArea\Wrist+MoveL”作为不同程序部分的起始点。

监测默认有效。其自动设置：

- 当使用重启模式重置RAPID时
- 当加载一则新程序或一个新模块时
- 当从起点开始执行程序时
- 当将程序指针移动到main时
- 当将程序指针移动到子程序时
- 移动程序指针造成执行顺序丢失时

语法

```
ConfL
  [ '\' On] | [ '\' Off]';'
```

相关信息

信息，关于	请参阅
处理不同的配置	技术参考手册 - *RAPID*语言概览
接头移动期间的机械臂配置	第*120*页的*ConfJ - 接头移动期间,控制配置*
确定奇点周围的插补	第*608*页的*SingArea - 确定奇点周围的插补*
运动设置数据	第*1430*页的*motsetdata - 运动设置数据*
机械臂配置数据	第*1375*页的*confdata - 机械臂配置数据*

1.49 CONNECT - 将中断与软中断程序相连

手册用法

CONNECT用于发现中断识别号，并将其与软中断程序相连。

通过下达中断事件指令并规定其识别号，确定中断。因此，当出现该事件时，自动执行软中断程序。

基本示例

以下实例介绍了指令CONNECT：

例 1

```
VAR intnum feeder_low;
PROC main()
  CONNECT feeder_low WITH feeder_empty;
  ISignalDI di1, 1 , feeder_low;
  ...
```

创建中断识别号feeder_low，并将其与软中断程序feeder_empty相连。当输入di1变高时，将会出现中断。换句话说，当信号变高时，执行feeder_empty软中断程序。

变元

```
CONNECT Interrupt WITH Trap routine
```

`Interrupt`

数据类型：intnum

被分配以中断识别号的变量。不得在程序（程序数据）内进行声明。

`Trap routine`

Identifier

软中断程序的名称。

程序执行

向变量分配在下达中断指令或停用中断时所应使用的中断识别号。该识别号同时与特定软中断程序相连。

> **注意**
>
> 当针对该任务，将程序指针设置为主要时，取消任务中的所有中断，且必须重新相连。中断将不会受到上电失败或重启的影响。

限制

无法将中断（中断识别号）与多个软中断程序相连。但是，可将不同的中断与同一软中断程序相连。

当已经将中断与软中断程序相连时，无法重新连接或者转移到另一个程序；其必须首先通过运用指令IDelete来删除。

停止程序执行时，将忽略开始或未处理的中断。当停止程序时，不考虑中断。程序停止时，将不再考虑已经设置为安全的中断。当程序再次启动时，将处理上述中断。

下一页继续

错误处理

可能会产生下列可恢复错误。错误可以由错误处理程序处理。系统变量 ERRNO 将设置为：

名称	错误原因
ERR_ALRDYCNT	已经将中断变量与TRAP程序相连。
ERR_CNTNOTVAR	中断变量并非变量参考。
ERR_INOMAX	无法获得更多的中断编号。

语法

```
CONNECT <connect target> WITH <trap>';'
<connect target> ::= <variable>
                   | <parameter>
                   | <VAR>
<trap> ::= <identifier>
```

相关信息

信息，关于	请参阅
中断概述	技术参考手册 - *RAPID*语言概览
更多关于中断管理的信息	技术参考手册 - *RAPID*语言概览
中断的数据类型	第*1414*页的*intnum - 中断识别号*
取消中断	第*231*页的*IDelete - 取消中断*

1.50 CopyFile - 复制文件

手册用法

CopyFile用于复制现有文件。

基本示例

以下实例介绍了指令CopyFile：

例 1

CopyFile“HOME:/myfile”、“HOME:/yourfile”；

将文件myfile复制到yourfile。随后，两种文件完全相同。

```
CopyFile "HOME:/myfile", "HOME:/mydir/yourfile";
```

将文件myfile复制到路径mydir.中的yourfile。

变元

```
CopyFile OldPath NewPath
```

OldPath

数据类型：string

复制文件的完整路径。

NewPath

数据类型：string

将文件复制到指定位置的完整路径。

程序执行

将OldPath中指定的文件复制到NewPath中指定的文件。

错误处理

可能会产生下列可恢复错误。错误可以由错误处理程序处理。系统变量 ERRNO 将设置为：

名称	错误原因
ERR_FILEEXIST	NewPath中指定的文件已经存在。

语法

```
CopyFile
[ OldPath ':=' ] < expression (IN) of string > ','
[ NewPath ':=' ] < expression (IN) of string >';'
```

相关信息

信息，关于	请参阅
建立路径	第*322*页的*MakeDir - 创建新路径*
删除路径	第*503*页的*RemoveDir - 删除路径*
重命名文件	第*507*页的*RenameFile - 重命名文件*
删除文件	第*504*页的*RemoveFile - 删除文件*

下一页继续

信息，关于	请参阅
检查文件类型	第*1126*页的*IsFile - 检查文件的类型*
检查文件大小	第*1077*页的*FileSize - 检索文件的大小*
检查文件系统大小	第*1082*页的*FSSize - 检索文件系统的大小*
文件和串行通道处理	应用手册 - 控制器软件*IRC5*

1.51 CopyRawBytes - 复制原始数据字节数据的内容

手册用法

CopyRawBytes用于将所有或部分内容从一个rawbytes变量复制到另一个。

基本示例

以下实例介绍了指令CopyRawBytes：

例 1

```
VAR rawbytes from_raw_data;
VAR rawbytes to_raw_data;
VAR num integer := 8
VAR num float := 13.4;

ClearRawBytes from_raw_data;
PackRawBytes integer, from_raw_data, 1 \IntX := DINT;
PackRawBytes float, from_raw_data, (RawBytesLen(from_raw_data)+1)
     \Float4;
CopyRawBytes from_raw_data, 1, to_raw_data, 3,
     RawBytesLen(from_raw_data);
```

在此例子中，首先清除rawbytes类from_raw_data变量，即将所有字节设置为0。随后，在前4个字节中放置integer的值，并在下4个字节中放置float的值。

在向from_raw_data填充数据之后，将内容（8个字节）复制到to_raw_data，开始于位置3。

变元

```
CopyRawBytes FromRawData FromIndex ToRawData ToIndex[\NoOfBytes]
```

FromRawData

数据类型：rawbytes

FromRawData是数据容器，应当由此复制rawbytes数据。

FromIndex

数据类型：num

FromIndex是FromRawData中的位置，由此开始复制数据。索引始于1。

ToRawData

数据类型：rawbytes

ToRawData是数据容器，应当将rawbytes数据复制于此。

ToIndex

数据类型：num

ToIndex是ToRawData中的位置，将待复制的数据放置于此。将所有内容复制彻底。索引始于1。

[\NoOfBytes]

数据类型：num

通过\NoOfBytes指定的数值是从FromRawData复制到ToRawData的字节数。

下一页继续

如果未指定\NoOfBytes，则复制从FromIndex到FromRawData中有效字节当前长度末端的所有字节。

程序执行

程序执行期间，复制从一个rawbytes变量到另一个的数据。

将ToRawData变量中的有效字节当前长度设置为：

- （ToIndex + copied_number_of_bytes - 1）
- 如果在ToRawData变量中的最初有效字节当前长度内完成完整的复制操作，则不会改变ToRawData变量中最初的有效字节当前长度。

限制

CopyRawBytes无法用于将一些数据从一个rawbytes变量复制到同一rawbytes变量的其他部分。

语法

```
CopyRawBytes
  [FromRawData ':=' ] < variable (VAR) of rawbytes> ','
  [FromIndex ':=' ] < expression (IN) of num> ','
  [ToRawData ':=' ] < variable (VAR) of rawbytes> ','
  [ToIndex ':=' ] < expression (IN) of num>
  ['\'NoOfBytes ':=' < expression (IN) of num> ]';'
```

相关信息

信息，关于	请参阅
rawbytes数据	第*1455*页的原始数据字节 - 原始数据
获取rawbytes数据的长度	第*1193*页的*RawBytesLen* - 获取原始数据字节数据的长度
清除rawbytes数据的内容	第*110*页的*ClearRawBytes* - 清除原始数据字节数据的内容
将DeviceNet标题装入rawbytes数据	第*421*页的*PackDNHeader* - 将*DeviceNet*标题装入原始数据字节数据中。
将数据装入rawbytes数据	第*424*页的*PackRawBytes* - 将数据装入原始数据字节数据
写入rawbytes数据	第*923*页的*WriteRawBytes* - 写入原始数据字节数据
读取rawbytes数据	第*499*页的*ReadRawBytes* - 读取原始数据字节数据
打开来自rawbytes数据的数据	第*845*页的*UnpackRawBytes* - 打开来自原始数据字节数据的数据
文件和串行通道处理	应用手册 - 控制器软件*IRC5*

1.52 CorrClear - 移除所有修正发电机

描述

CorrClear用于移除所有连接的修正发电机。本指令可用于移除由修正发电机早先提供的偏移量。

基本示例

以下实例介绍了指令CorrClear：

例 1

CorrClear ;

本指令移除所有相连的修正发电机。

> **注意**
>
> 用于确保所有修正发电机（带修正）均在程序启动时移除的一种简单方式是在START事件程序中运行CorrClear。
>
> 参见技术参考手册 - 系统参数，题目*Controller*。

语法

```
CorrClear ';'
```

相关信息

信息，关于	请参阅
与修正发电机相连	第*131*页的*CorrCon - 与修正发电机相连*
与修正发电机断开	第*136*页的*CorrDiscon - 与修正发电机断开*
写入修正发电机	第*137*页的*CorrWrite - 写入修正发电机*
读取当前的总偏移量	第*1030*页的*CorrRead - 读取当前的总偏移量*
修正描述符	第*1381*页的*corrdescr - 修正发电机描述符*

1.53 CorrCon - 与修正发电机相连

手册用法

CorrCon用于同修正发电机相连。

基本示例

以下实例介绍了指令CorrCon：

另请参阅第*131*页的更多示例

Example1

```
VAR corrdescr id;
...
CorrCon id;
```

修正发电机参考相当于变量id保留。

变元

```
CorrCon Descr
```

Descr

数据类型：corrdescr

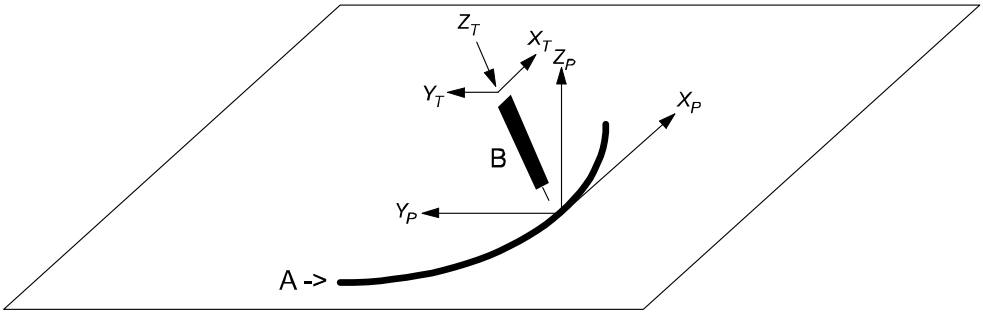
修正发电机的描述符

更多示例

有关于指令CorrCon的更多例子阐述如下。

路径坐标系

在路径坐标系中增加所有路径修正（关于路径的偏移量）。路径坐标系定义如下：



A	路径方向
B	工具
X	路径坐标系
X	工具坐标系

- 将路径坐标X轴作为路径的切线。
- 根据路径坐标X轴和工具坐标Z轴的叉积，推导出路径坐标Y轴。
- 根据路径坐标X轴和路径坐标Y轴的叉积，推导出路径坐标Z轴。

下一页继续

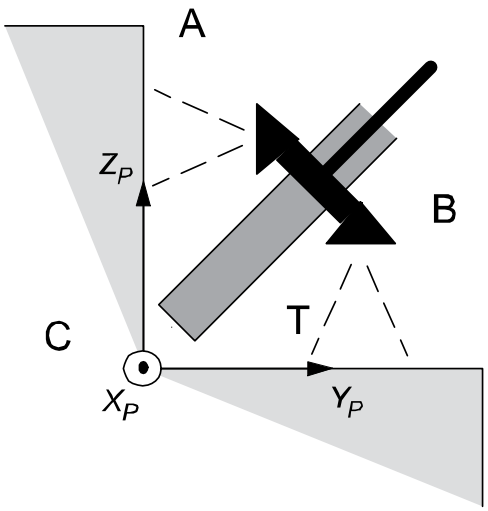
1.53 CorrCon - 与修正发电机相连

Path Offset

续前页

应用实例

运用路径修正的应用实例是一个握住工具以及安装在工具上的两个传感器的机械臂，以探测与工件的垂直和水平距离。下图介绍了一种路径修正装置。



xx0500002155

A	水平修正传感器
B	垂直修正传感器
C	路径坐标系
T	工具

程序示例

注意

hori_sig和vert_sig是系统参数中确定的模拟信号。

```
CONST num TARGET_DIST := 5;
CONST num SCALE_FACTOR := 0.5;
VAR intnum intno1;
VAR corrdescr hori_id;
VAR corrdescr vert_id;
VAR pos total_offset;
VAR pos write_offset;
VAR bool conFlag;
PROC PathRoutine()
  ! Connect to the correction generators for horizontal and vertical
       correction.
  CorrCon hori_id;
  CorrCon vert_id;
  conFlag := TRUE;

  ! Setup a 5 Hz timer interrupt. The trap routine will read the
       sensor values and
  ! compute the path corrections.
  CONNECT intno1 WITH ReadSensors;
```

下一页继续

```
  ITimer\Single, 0.2, intno1;

  ! Position for start of contour tracking
  MoveJ p10, v100, z10, tool1;
  ! Run MoveL with both vertical and horizontal correction.
  MoveL p20, v100, z10, tool1 \Corr;

  ! Read the total corrections added by all connected correction
       generators.
  total_offset := CorrRead();
  ! Write the total vertical correction on the FlexPendant.
  TPWrite "The total vertical correction is:" \Num:=total_offset.z;

  ! Disconnect the correction generator for vertical correction.
  ! Horizontal corrections will be unaffected.
  CorrDiscon vert_id;
  conFlag := FALSE;

  ! Run MoveL with only horizontal interrupt correction.
  MoveL p30, v100, fine, tool1 \Corr;
  ! Remove all outstanding connected correction generators.
  ! In this case, the only connected correction generator is the
       one for horizontal
  ! correction.
  CorrClear;
  ! Remove the timer interrupt.
  IDelete intno1;
ENDPROC
TRAP ReadSensors
  VAR num horiSig;
  VAR num vertSig;

  ! Compute the horizontal correction values and execute the
       correction.
  horiSig := hori_sig;
  write_offset.x := 0;
  write_offset.y := (hori_sig - TARGET_DIST)*SCALE_FACTOR;
  write_offset.z := 0;
  CorrWrite hori_id, write_offset;

  IF conFlag THEN
    ! Compute the vertical correction values and execute the
         correction.
    write_offset.x := 0;
    write_offset.y := 0;
    write_offset.z := (vert_sig - TARGET_DIST)*SCALE_FACTOR;
    CorrWrite vert_id, write_offset;
  ENDIF
  !Setup interrupt again
  IDelete intnol;
  CONNECT intno1 WITH ReadSensors;
```

下一页继续

1.53 CorrCon - 与修正发电机相连

Path Offset

续前页

```
    ITimer\single, 0.2, intno1;
ENDTRAP
```

程序说明

将两个修正发电机与指令CorrCon相连。各修正发电机被独特的corrdescr类描述符（hori_id和vert_id）所参考。两个传感器将各使用一台修正发电机。

设置定时器中断，以调用频率为5 Hz的软中断程序ReadSensors。通过指令CorrWrite，将路径修正所需要的偏移量在软中断程序中进行计算，并写入相应的修正发电机（为描述符hori_id和vert_id所参考）。所有修正均会立即对路径产生影响。

当使用路径修正时，必须通过切换参数Corr，来对MoveL指令进行编程。否则，将不会执行任何修正。

当第一个MoveL指令就绪时，功能CorrRead用于读取由所有相连修正发电机所给出的所有修正的总和（总路径修正）。通过指令TPWrite，将总垂直路径修正的结果写入FlexPendant。

随后，CorrDiscon将断开垂直修正的修正发电机（为描述符vert_id所参考）。将从总路径修正中除去由此修正发电机所增加的所有修正。仍将保留修正发电机为水平修正所增加的修正。

最后，功能CorrClear将移除所有剩余的相连修正发电机及其先前增加的修正。在这种情况下，将仅移除水平修正的修正发电机。亦将通过指令IDelete移除定时器中断。

修正发电机

下表介绍了修正发电机。

X	Y	Z	路径坐标轴
0	0	3	垂直修正发电机及其路径修正的总和
0	1	0	水平修正发电机及其路径修正的总和
-	-	-	未连接的修正发电机
-	-	-	未连接的修正发电机
-	-	-	未连接的修正发电机
0	1	3	由所有相连修正发电机完成的所有修正的总和

限制

可以同时连接最多5台修正发电机。

连接的修正发电机未能通过控制器重启。

语法

```
CorrCon
  [ Descr ':=' ] < variable (VAR) of corrdescr > ';'
```

相关信息

信息，关于	请参阅
与修正发电机断开	第136页的*CorrDiscon - 与修正发电机断开*
写入修正发电机	第137页的*CorrWrite - 写入修正发电机*

下一页继续

续前页

信息，关于	请参阅
读取当前的总偏移量	第*1030*页的*CorrRead - 读取当前的总偏移量*
移除所有修正发电机	第*130*页的*CorrClear - 移除所有修正发电机*
修正发电机描述符	第*1381*页的*corrdescr - 修正发电机描述符*

1.54 CorrDiscon - 与修正发电机断开

描述

CorrDiscon用于与先前连接的修正发电机断开。本指令可用于移除先前进行的修正。

基本示例

以下实例介绍了指令CorrDiscon：

另请参阅第*136*页的更多示例

例 1

```
VAR corrdescr id;
...
CorrCon id;
...
CorrDiscon id;
```

CorrDiscon与描述符id所参考的先前连接的修正发电机断开。

变元

```
CorrDiscon Descr
```

Descr

数据类型：corrdescr

修正发电机的描述符

更多示例

关于指令CorrDiscon的更多实例，请参见第*131*页的*CorrCon - 与修正发电机相连*。

语法

```
CorrDiscon
  [ Descr ':=' ] < variable (VAR) of corrdescr > ';'
```

相关信息

信息，关于	请参阅
与修正发电机相连	第*131*页的*CorrCon - 与修正发电机相连*
写入修正发电机	第*137*页的*CorrWrite - 写入修正发电机*
读取当前的总偏移量	第*1030*页的*CorrRead - 读取当前的总偏移量*
移除所有修正发电机	第*130*页的*CorrClear - 移除所有修正发电机*
修正描述符	第*1381*页的*corrdescr - 修正发电机描述符*

1.55 CorrWrite - 写入修正发电机

描述

CorrWrite用于将偏移量写入修正发电机的路径坐标系统中。

基本示例

以下实例介绍了指令CorrWrite：

例 1

```
VAR corrdescr id;
VAR pos offset;
...
CorrWrite id, offset;
```

将变量偏移量中储存的当前偏移量写入到描述符id所参考的修正发电机中。

变元

```
CorrWrite Descr Data
```

Descr

数据类型：corrdescr

修正发电机的描述符

数据

数据类型：pos

待写入的偏移量。

更多示例

关于指令CorrWrite的更多实例，请参见第131页的*CorrCon - 与修正发电机相连*。

限制

在直线路径上实现了最佳性能。随着连续线性路径之间的速度和角度的增加，同预期路径的偏差亦将有所增加。相同原理适用于半径不断减少的圆。

语法

```
CorrWrite
  [ Descr ':=' ] < variable (VAR) of corrdescr > ','
  [ Data ':=' ] < expression (IN) of pos > ';'
```

相关信息

信息，关于	请参阅
与修正发电机相连	第131页的*CorrCon - 与修正发电机相连*
与修正发电机断开	第136页的*CorrDiscon - 与修正发电机断开*
读取当前的总偏移量	第1030页的*CorrRead - 读取当前的总偏移量*
移除所有修正发电机	第130页的*CorrClear - 移除所有修正发电机*
修正发电机描述符	第1381页的*corrdescr - 修正发电机描述符*

1.56 DeactEventBuffer - 事件缓冲启用

描述

DeactEventBuffer用于在当前运动程序任务中停用事件缓冲。

当结合使用精点的应用以及持续应用（因缓慢的工艺设备而需要提前设置信号）时，应当使用指令DeactEventBuffer和ActEventBuffer。

本指令仅可用于主任务T_ROB1，或者如果在MultiMove系统中，则可用于运动任务中。

基本示例

以下实例介绍了指令DeactEventBuffer：

例 1

```
..
DeactEventBuffer;
! Use an application that use finepoints, such as SpotWelding
..
! Activate the event buffer again
ActEventBuffer;
! Now it is possible to use an application that needs
! to set signals in advance, such as Dispense
..
```

DeactEventBuffer使配置的事件缓冲停用。当使用有关于精点的应用时，机械臂将以更快的速度从精点启动。当通过ActEventBuffer来启用事件缓冲时，有可能针对采用缓慢工艺设备的应用提前设置信号。

程序执行

事件缓冲的停用适用于下一次执行的任意类型的机械臂运动，其始终有效，直至执行ActEventBuffer指令。

在停用事件缓冲之前，本指令将等待，直至机械臂和外轴已经达到停止点（当前运动指令的ToPoint）。因此，建议将运动指令编程在DeactEventBuffer以及精点之前。

自动设置默认值（使用事件缓冲 = TRUE）

- 当使用重启模式重置RAPID时
- 当加载一则新程序或一个新模块时
- 当从起点开始执行程序时
- 当将程序指针移动到main时
- 当将程序指针移动到子程序时
- 移动程序指针造成执行顺序丢失时

限制

无法在与任意下列特殊系统事件关联的RAPID程序中执行DeactEventBuffer：PowerOn、Stop、QStop、Restart或者Step。

语法

```
DeactEventBuffer ';'
```

下一页继续

相关信息

信息，关于	请参阅
事件缓冲停用	第22页的*ActEventBuffer - 事件缓冲启用*
Event preset time的配置	技术参考手册 - 系统参数
运动设置数据	第1430页的*motsetdata - 运动设置数据*

1.57 DeactUnit - 停用机械单元

手册用法

DeactUnit用于停用机械单元。

其可以用于确定，例如，当使用公共驱动单元时，应启用哪一个单元。

本指令仅可用于主任务T_ROB1，或者如果在MultiMove系统中，则可用于运动任务中。

示例

以下实例介绍了指令DeactUnit：

例 1

```
DeactUnit orbit_a;
```

orbit_a机械单元停用。

例 2

```
MoveL p10, v100, fine, tool1;
DeactUnit track_motion;
MoveL p20, v100, z10, tool1;
MoveL p30, v100, fine, tool1;
ActUnit track_motion;
MoveL p40, v100, z10, tool1;
```

当机械臂运动至p20和p30时，单元track_motion将固定。此后，机械臂与track_motion均将移动至p40。

例 3

```
MoveL p10, v100, fine, tool1;
DeactUnit orbit1;
ActUnit orbit2;
MoveL p20, v100, z10, tool1;
```

停用单元orbit1，并启用orbit2。

变元

```
DeactUnit MechUnit
```

MechUnit

Mechanical Unit

数据类型：mecunit

待停用机械单元的名称。

程序执行

当机械臂和外轴的实际路径就绪时，清除当前路径等级上的路径，并停用指定的机械单元。这意味着既不对其进行控制，亦不对其进行监测，直至重新启用为止。

如果若干机械单元共享一个公共驱动单元，则停用其中一个机械单元，亦将把该单元同公共驱动单元断开。

限制

当其中一个机械单元处于独立模式时，无法使用指令DeactUnit。

下一页继续

如果该指令位于移动指令之后，则必须使用停止点（zonedatafine）而非飞越点来编程移动指令，否则，将无法在电源故障后重启。

无法在与任意下列特殊系统事件相连的RAPID程序中执行DeactUnit：PowerOn、Stop、QStop、Restart或者Step。

有可能将ActUnit - DeactUnit用于StorePath等级，但是，当正在进行RestoPath且已经完成StorePath时，必须启用相同的机械单元。如果关于路径记录器的操作以及基准等级的路径均将十分完整，但是，将清除StorePath等级的路径。

语法

```
DeactUnit
  [MechUnit ':='] < variable (VAR) of mecunit> ';'
```

相关信息

信息，关于	请参阅
启动机械单元	第*24*页的*ActUnit - 启用机械单元*
机械单元	第*1428*页的*mecunit - 机械单元*
检查机械单元是否启用。	第*1130*页的*IsMechUnitActive - 机械单元是否有效*
路径记录器	第*431*页的*PathRecMoveBwd - 将路径记录器向后移动*第*1428*页的*mecunit - 机械单元*

1.58 Decr - 减量为1

手册用法

Decr用于从数值变量或者永久数据对象减去1。

基本示例

以下实例介绍了指令Decr：

另请参阅第*142*页的更多示例

例 1

```
Decr reg1;
```

从reg1中减去1，即reg1:=reg1-1。

变元

```
Decr Name | Dname
```

Name

数据类型：num

待缩减变量或者永久数据对象的名称。

Dname

数据类型：dnum

待缩减变量或者永久数据对象的名称。

更多示例

有关于指令Decr的更多例子阐述如下。

例 1

```
VAR num no_of_parts:=0;
...
TPReadNum no_of_parts, "How many parts should be produced? ";
WHILE no_of_parts>0 DO
  produce_part;
  Decr no_of_parts;
ENDWHILE
```

要求操作员输入待生产零件的数量。变量no_of_parts用于统计必须继续生产的数量。

例 2

```
VAR dnum no_of_parts:=0;
...
TPReadDnum no_of_parts, "How many parts should be produced? ";
WHILE no_of_parts>0 DO
  produce_part;
  Decr no_of_parts;
ENDWHILE
```

要求操作员输入待生产零件的数量。变量no_of_parts用于统计必须继续生产的数量。

下一页继续

语法

```
Decr
  [ Name ':=' ] < var or pers (INOUT) of num >
  | [ Dname ':=' ] < var or pers (INOUT) of dnum >' ;'
```

相关信息

信息，关于	请参阅
将变量增加1	第*239*页的*Incr - 增量为1*
从变量中减去任意数值。	第*26*页的*Add*-增加数值
运用任意表达式（例如，乘法）来更改数据	第*32*页的*":="* - 分配一个数值

1.59 DitherAct - 促使软伺服抖动

手册用法

DitherAct用于启用抖动功能，其将减少IRB 7600软伺服的摩擦。

本指令仅可用于主任务T_ROB1，或者如果在MultiMove系统中，则可用于运动任务中。

基本示例

以下实例介绍了指令DitherAct：

例 1

```
SoftAct \MechUnit:=ROB_1, 2, 100;
WaitTime 2;
DitherAct \MechUnit:=ROB_1, 2;
WaitTime 1;
DitherDeact;
SoftDeact;
```

软伺服期间，仅能抖动一秒钟。

例 2

```
DitherAct \MechUnit:=ROB_1, 2;
SoftAct \MechUnit:=ROB_1, 2, 100;
WaitTime 1;
MoveL p1, v50, z20, tool1;
SoftDeact;
DitherDeact;
```

促使轴2抖动。显示长达一秒钟的运动，从而为SoftAct斜面留下充足的转换时间。如果在SoftAct之前调用DitherAct，则在针对该轴执行SoftAct时，将开始抖动。如果未调用DitherDeact，抖动将保持激活，以满足所有后续的SoftAct调用。

变元

```
DitherAct [\MechUnit] Axis [\Level]
```

[\MechUnit]

Mechanical Unit

数据类型：mecunit

机械单元的名称。如果省略参数，则意味着启用指定机械臂轴的软伺服。

Axis

数据类型：num

轴编号（1-6）

[\Level]

数据类型：num

抖动振幅（50-150%）。在50%下，振动得以降低（增加摩擦）。在150%下，振幅达到最大（可能导致末端执行器振动）。默认值为100%。

下一页继续

程序执行

在SoftAct前后，可以调用DitherAct。在SoftAct之后调用DitherAct更为迅速，但是具有其他限制。

IRB 7600的轴1通常不需要抖动。对于轴2和3减小摩擦的影响最大。

抖动参数自我调整。在过程位置执行三次或四次SoftAct后，实现完整的抖动性能。

限制

在SoftAct之后调用DitherAct，可能导致机械臂出现不必要的运动。消除这种行为的唯一方式是在SoftAct之前调用DitherAct。如果仍然存在运动，则应当增加SoftAct斜面时间。

转换时间为斜面时间，其将在机械臂之间变动，并乘以SoftAct指令的斜面系数。

抖动不适用于轴6。

存在电源故障时，抖动始终停用。

本指令仅用于IRB 7600。

警告

当在SoftAct之前调用DitherAct时，机械臂必须位于精点。与此同时，使精点不被允许，直至斜面的转换时间结束。这可能损坏齿轮箱。

语法

```
DitherAct
  [ '\' MechUnit ':=' < variable (VAR) of mecunit > ]
  [Axis ':=' ] < expression (IN) of num >
  [ '\' Level ':=' < expression (IN) of num > ] ';'
```

相关信息

信息，关于	请参阅
启用软伺服	第*643*页的*SoftAct - 启用软伺服*
有关软伺服启用时的行为	技术参考手册 - *RAPID*语言概览
抖动停止	第*146*页的*DitherDeact - 促使软伺服停止抖动*

1.60 DitherDeact - 促使软伺服停止抖动

手册用法

DitherDeact用于禁用IRB 7600软伺服的抖动功能。

本指令仅可用于主任务T_ROB1，或者如果在MultiMove系统中，则可用于运动任务中。

基本示例

以下实例介绍了指令DitherDeact：

例 1

```
DitherDeact;
```

停用所有轴上的抖动

程序执行

随时可以使用DitherDeact。如果采用软伺服，则所有轴上的抖动立即停止。如果未采用软伺服，当执行下一个SoftAct时，抖动将不会启用。

自动禁用抖动

- 重启时。
- 当载入一段新程序时。
- 当从起点开始执行程序时。

语法

```
DitherDeact';'
```

相关信息

信息，关于	请参阅
启用抖动	第144页的*DitherAct - 促使软伺服抖动*

1.61 DropSensor - 使物体落于传感器上

手册用法

DropSensor用于与当前工件断开，且针对下一个工件的程序已经就绪。

DropSensor用于传感器同步，但是不用于模拟同步。

基本示例

```
MoveL *, v1000, z10, tool, \WObj:=wobj0;
SyncToSensor Ssync1\Off;
MoveL *, v1000, fine, tool, \WObj:=wobj0;
DropSensor Ssync1;
MoveL *, v1000, z10, tool, \WObj:=wobj0;
```

变元

```
DropSensor MechUnit
```

MechUnit

Mechanical Unit

数据类型：mecunit

运动中的机械单元与指令中的机械臂位置相关。

程序执行

使工件下落意味着编码器单元不再跟踪工件。从工件队列移除工件，且无法恢复。

限制

如果在机械臂主动地运用传感器工件时发出指令，则运动停止。必须在机械臂已经通过最后的同步robtarget后发出指令。

可能仅在非同步移动已用于先前的运动指令以及精点或若干（>1）角部之后，发出指令。

语法

```
DropSensor
  [ MechUnit ':='] < variable (VAR) of mecunit> ';'
```

相关信息

信息，关于	请参阅
等候传感器的连接	第887页的*WaitSensor - 等待传感器连接*
与传感器同步	第721页的*SyncToSensor - 同步至传感器*
Machine Synchronization	应用手册 - 控制器软件*IRC5*

1.62 DropWObj - 使工件落于传送带上

手册用法

DropWObj（*Drop Work Object*）用于与当前工件断开，且针对传送带上的下一个工件的程序已经就绪。

基本示例

以下实例介绍了指令DropWObj：

例 1

```
MoveL *, v1000, z10, tool, \WObj:=wobj_on_cnv1;
MoveL *, v1000, fine, tool, \WObj:=wobj0;
DropWObj wobj_on_cnv1;
MoveL *, v1000, z10, tool, \WObj:=wobj0;
```

变元

```
DropWObj WObj
```

WObj

Work Object

数据类型： wobjdata

运动中的工件（坐标系）与指令中的机械臂位置相关。通过工件中的ufmec，指定机械单元传送带。

程序执行

使工件下落意味着编码器单元不再跟踪工件。从工件队列移除工件，且无法恢复。

限制

如果在机械臂主动地运用传送带调整的工件时发出指令，则运动停止。

可能仅在固定工件已用于先前的运动指令以及精点或若干（>1）角部之后，发出指令。

语法

```
DropWObj
  [ WObj ':='] < persistent (PERS) of wobjdata>';'
```

相关信息

信息，关于	请参阅
等待工件	第*903*页的*WaitWObj - 等待传送带上的工件*
传送带跟踪	应用手册 - 传送带跟踪

1.63 EGMActJoint – 为一个关节目标点编写一次EGM移动

手册用法

EGMActJoint会激活一项特定的EGM进程，并为在传感器指引下前往某关节目标点的移动定义所需的静态数据（即那些不会因EGM移动的不同而频繁改变的数据）。

基本示例

```
VAR egmident egmID1;
PERS pose pose1:=[[0,0,0], [1,0,0,0]];
CONST egm_minmax egm_minmax1:=[-1,1];

EGMGetId egmID1;
EGMSetupAI ROB_1, egmID1, "default" \Pose \aiR1x:=ai_01
     \aiR2y:=ai_02 \aiR3z:=ai_03 \aiR4rx:=ai_04 \aiR5ry:=ai_05
     \aiR6rz:=ai_06;
EGMActJoint egmID1 \J1:=egm_minmax1 \J3:=egm_minmax1
     \J4:=egm_minmax1;
```

变元

```
EGMActJoint EGMid [\Tool] [\WObj] [\TLoad] [\J1] [\J2] [\J3] [\J4]
     [\J5] [\J6] [\LpFilter] [\SampleRate] [\MaxPosDeviation]
     [\MaxSpeedDeviation]
```

EGMid

数据类型：egmident

EGM标识。

[\Tool]

数据类型： tooldata

按指令EGMRunJoint来执行移动的工具。

自变数[\Tool]是一个可选项。当忽略该自变数时，其默认值为tool0。

[\Wobj]

数据类型： wobjdata

按指令EGMRunJoint来执行移动的工件。

自变数[\Wobj]是一个可选项。当忽略该自变数时，其默认值为wobj0。

[\TLoad]

Total load

数据类型：loaddata

按指令EGMRunJoint来执行移动的负载。

自变数[\TLoad]是一个可选项。当忽略该自变数时，其默认值为load0。

\TLoad主动轴描述了移动中使用的总负载。总负载就是相关的工具负载加上该工具正在处理的有效负载。如果使用了\TLoad自变数，那么就不考虑当前tooldata中的loaddata。

如果\TLoad自变数被设置成load0，那么就不考虑\TLoad自变数，而是以当前tooldata中的loaddata作为代替。

下一页继续

1.63 EGMActJoint - 为一个关节目标点编写一次EGM移动

Externally Guided Motion

续前页

想要使用\TLoad自变数，就必需将系统参数ModalPayLoadMode的数值设置成0。如果将ModalPayLoadMode设置成0，那么就再也无法使用指令GripLoad。

可用服务例程“负载标识”（LoadIdentify）来识别总负载。如果系统参数ModalPayLoadMode被设置成0，且系统正在运行该服务例程，那么操作员便可将相关工具的loaddata复制到一个现有的或新的loaddata永久变量中。

如果使用了关联到系统输入项SimMode（仿真模式）上的一个数字输入信号，那么便可在没有任何有效负载的情况下试运行该程序。如果该数字输入信号被设置成1，那么就不考虑可选自变数\TLoad中的loaddata，而是以当前tooldata中的loaddata作为代替。

> **注意**
>
> 处理有效负载的默认功能是使用指令GripLoad，因此系统参数ModalPayLoadMode的默认值为1。

[\J1] [\J2] [\J3] [\J4] [\J5] [\J6]

数据类型：egm_minmax

关节1到6的收敛标准（以度为单位）。默认值为±0.5度。

该收敛标准数据的作用是决定相关机器人是否已抵达了预定关节位置。如果预定关节位置与实际关节位置之间的差距在egm_minmax.min和egm_minmax.max的范围之内，那么就视为该关节已抵达了其预定位置。如果并未指定某一关节的收敛标准（在EGMRunJoint中选择），那么系统就会使用默认值。

只要EGMRunJoint中指定的所有关节都抵达了各自的预定位置，相关机器人就会抵达其本身的预定位置，而RAPID则会继续执行下一条RAPID指令。

[\LpFilter]

数据类型：num

低通滤波器带宽，以赫兹（Hz）为单位，用于过滤传感器噪声。

[\SampleRate]

数据类型：num

读取输入数据的取样率（4毫秒的倍数）。有效值为4、8、12和16等。

默认值为4毫秒。

[\MaxPosDeviation]

数据类型：num

与编程位置之间的最大关节偏差（以度为单位），即开始EGM移动的精确点。所有关节都采用同一数值。

默认值为1000度。

[\MaxSpeedDeviation]

数据类型：num

容许的最大关节速度变化（以度 / 秒为单位），即是说用该系数来调整加速度 / 减速度。

默认值为1.0度 / 秒。

下一页继续

限制

- 如果用同一EGMid执行了若干次EGMActJoint，那么各条EGMRunJoint指令就会使用最后的激活数据，直到系统执行一次新的EGMActJoint为止。
- EGMActJoint只能用在RAPID运动任务中。

语法

```
EGMActJoint
  [EGMid ':='] <variable (VAR) of egmident>
  ['\'Tool ':=' <persistent (PERS) of tooldata>]
  ['\'Wobj ':=' <persistent (PERS) of wobjdata>
  ['\'TLoad ':=' <persistent (PERS) of loaddata>]
  ['\'J1 ':=' <expression (IN) of egm_minmax>]
  ['\'J2 ':=' <expression (IN) of egm_minmax>]
  ['\'J3 ':=' <expression (IN) of egm_minmax>]
  ['\'J4 ':=' <expression (IN) of egm_minmax>]
  ['\'J5 ':=' <expression (IN) of egm_minmax>]
  ['\'J6 ':=' <expression (IN) of egm_minmax>]
  ['\'LpFilter ':=' <expression (IN) of num>]
  ['\'SampleRate ':=' <expression (IN) of num>]
  ['\'MaxPosDeviation ':=' <expression (IN) of num>]
  ['\'MaxSpeedDeviation ':=' <expression (IN) of num>] ';'
```

相关信息

信息，关于	请参阅
Externally Guided Motion	应用手册 - 控制器软件*IRC5*
数据类型egm_minmax	第*1390*页的*egm_minmax - EGM*的收敛标准
指令EGMRunJoint	第*166*页的*EGMRunJoint - 执行一次含一个关节目标点的EGM*移动

1.64 EGMActMove - 编写一次经过路径校正的EGM移动

手册用法

EGMActMove会激活一项特定的EGM进程，并为带路径校正的移动定义所需的静态数据（即不会因EGM路径校正移动的不同而频繁改变的数据）。

基本示例

下例说明了指令EGMActMove。

例 1

```
VAR egmident EGMid1;
PERS tooldata tLaser := [TRUE, [[148,50,326],
     [0.3902618,-0.589657,-0.589656,0.3902630]],
     [1,[-0.92,0,-0.39], [1,0,0,0], 0,0,0]];
EGMGetId EGMid1;
EGMSetupLTAPP ROB_1, EGMid1, "pathCorr", "OptSim", 1\LATR;
EGMActMove EGMid1, tLaser.tframe\SampleRate:=48;
```

该程序会登记一项EGM进程，然后设置一个使用通信协议LTAPP且类型为先行的传感器，并以该传感器作为数据来源（传感器）。该传感器应使用关节类型定义编号1来进行跟踪。此外也要设置控制器访问相关装置和该装置传感器框架的速率。

变元

```
EGMActMove EGMid, SensorFrame [\SampleRate]
```

EGMid

数据类型：egmident

EGM标识。

SensorFrame

数据类型：pose

传感器框架。

[\SampleRate]

数据类型：num

读取输入数据的取样率（24毫秒的倍数）。有效值为24、48和72等。

程序执行

该传感器框架和传感器取样率与一个EGM标识相关联，直到要么用EGMReset来重置它们、要么用另一条EGMActMove指令来更改它们为止。

语法

```
EGMActMove
  [EGMid ':='] <variable (VAR) of egmident> ','
  [SensorFrame ':='] < expression (IN) of pose>
  ['\'SampleRate ':=' <expression (IN) of num>] ';'
```

下一页继续

相关信息

信息，关于	请参阅
Externally Guided Motion	应用手册 - 控制器软件*IRC5*

1.65 EGMActPose – 为一个姿态目标点编写一次EGM移动

手册用法

EGMActPose会激活一项特定的EGM进程，并为在传感器指引下前往某姿态目标点的移动定义所需的静态数据（即那些不会因EGM移动的不同而频繁改变的数据）。

基本示例

```
VAR egmident egmID1;
PERS pose pose1:=[[0,0,0], [1,0,0,0]];
CONST egm_minmax egm_minmax_lin:=[-0.1,0.1];
CONST egm_minmax egm_minmax_rot:=[-0.1,0.2];
CONST pose posecor:=[[1200,400,900], [0,0,1,0]];
CONST pose posesens:=[[12.3313,-0.108707,416.142],
     [0.903899,-0.00320735,0.427666,0.00765917]];

EGMGetId egmID1;
EGMSetupAI ROB_1, egmID1, "default" \Pose \aiR1x:=ai_01
     \aiR2y:=ai_02 \aiR3z:=ai_03 \aiR4rx:=ai_04 \aiR5ry:=ai_05
     \aiR6rz:=ai_06;
EGMActPose egmID1 \Tool:=tool0 \WObj:=wobj0, posecor,
     EGM_FRAME_WOBJ, posesens, EGM_FRAME_TOOL \x:=egm_minmax_lin
     \y:=egm_minmax_lin \z:=egm_minmax_lin \rx:=egm_minmax_rot
     \ry:=egm_minmax_rot \rz:=egm_minmax_rot \LpFilter:=20;
```

变元

```
EGMActPose EGMid [\Tool] [\WObj] [\TLoad], CorrFrame, CorrFrType,
     SensorFrame, SensorFrType [\x] [\y] [\z] [\rx] [\ry] [\rz]
     [\LpFilter] [\SampleRate] [\MaxPosDeviation]
     [\MaxSpeedDeviation]
```

EGMid

数据类型：egmident

EGM标识。

[\Tool]

数据类型： tooldata

按指令EGMRunPose来执行移动的工具。

自变数[\Tool]是一个可选项。当忽略该自变数时，其默认值为tool0。

[\Wobj]

数据类型： wobjdata

按指令EGMRunPose来执行移动的工件。

自变数[\Wobj]是一个可选项。当忽略该自变数时，其默认值为wobj0。

[\TLoad]

Total load

数据类型：loaddata

按指令EGMRunPose来执行移动的负载。

自变数[\TLoad]是一个可选项。当忽略该自变数时，其默认值为load0。

下一页继续

\TLoad主动轴描述了移动中使用的总负载。总负载就是相关的工具负载加上该工具正在处理的有效负载。如果使用了\TLoad自变数，那么就不考虑当前tooldata中的loaddata。

如果\TLoad自变数被设置成load0，那么就不考虑\TLoad自变数，而是以当前tooldata中的loaddata作为代替。

想要使用\TLoad自变数，就必需将系统参数ModalPayLoadMode的数值设置成0。如果将ModalPayLoadMode设置成0，那么就再也无法使用指令GripLoad。

可用服务例程“负载标识”（LoadIdentify）来识别总负载。如果系统参数ModalPayLoadMode被设置成0，且系统正在运行该服务例程，那么操作员便可将相关工具的loaddata复制到一个现有的或新的loaddata永久变量中。

如果使用了关联到系统输入项SimMode（仿真模式）上的一个数字输入信号，那么便可在没有任何有效负载的情况下试运行该程序。如果该数字输入信号被设置成1，那么就不考虑可选自变数\TLoad中的loaddata，而是以当前tooldata中的loaddata作为代替。

注意

处理有效负载的默认功能是使用指令GripLoad，因此系统参数ModalPayLoadMode的默认值为1。

CorrFrame

数据类型：pose

校正框架。

CorrFrType

数据类型：egmframetype

校正框架的框架类型。

SensorFrame

数据类型：pose

传感器框架。

SensFrType

数据类型：egmframetype

传感器框架的框架类型。

[\x] [\y] [\z]

数据类型：egm_minmax

x、y和z的收敛标准（以毫米为单位）。默认值为±1.0毫米。

该收敛标准数据的作用是决定相关机器人是否已从指定的轴方向抵达了预定位置。如果预定位置与实际位置之间的差距在egm_minmax.min和egm_minmax.max的范围之内，那么就视为相关机器人已抵达了其预定位置。如果并未指定某一轴方向的收敛标准（在EGMRunPose中选择），那么系统就会使用默认值。

只要EGMRunPose中指定的所有轴都抵达了各自的预定位置，相关机器人就会抵达其本身的预定位置，而RAPID则会继续执行下一条RAPID指令。

下一页继续

1.65 EGMActPose – 为一个姿态目标点编写一次EGM移动

Externally Guided Motion

续前页

[\rx] [\ry] [\rz]

数据类型：egm_minmax

x、y和z的收敛标准（以度为单位）。默认值为±0.5度。

该收敛标准数据的作用是决定相关机器人是否已沿指定轴抵达了预定方位。如果预定方位与实际方位之间的差距在egm_minmax.min和egm_minmax.max的范围之内，那么就视为相关机器人已抵达了其预定方位。如果并未指定某一轴方位的收敛标准（在EGMRunPose中选择），那么系统就会使用默认值。

只要EGMRunPose中指定的所有轴都抵达了各自的预定方位，相关机器人就会抵达其本身的预定位置，而RAPID则会继续执行下一条RAPID指令。

[\LpFilter]

数据类型：num

低通滤波器带宽，以赫兹（Hz）为单位，用于过滤传感器噪声。

该默认值取自EGMSetupXX指令的相关配置。

[\SampleRate]

数据类型：num

读取输入数据的取样率（4毫秒的倍数）。有效值为4、8、12和16等。

默认值为4毫秒。

[\MaxPosDeviation]

数据类型：num

与编程位置之间的最大关节偏差（以度为单位），即开始EGM移动的精确点。所有关节都采用同一数值。

默认值为1000度。

[\MaxSpeedDeviation]

数据类型：num

容许的最大关节速度变化（以度 / 秒为单位），即是说用该系数来调整加速度 / 减速度。

默认值为1.0度 / 秒。

限制

- 如果用同一EGMid执行了若干次EGMActPose，那么各条EGMRunPose指令就会使用最后的激活数据，直到系统执行一次新的EGMActPose为止。
- EGMActPose只能用在RAPID运动任务中。

语法

```
EGMActPose
  [EGMid ':='] <variable (VAR) of egmident>
  ['\'Tool ':=' <persistent (PERS) of tooldata>]
  ['\'Wobj ':=' <persistent (PERS) of wobjdata>]
  ['\'TLoad ':=' <persistent (PERS) of loaddata>] ','
  [CorrFrame ':='] < expression (IN) of pose> ','
  [CorrFrType ':='] < expression (IN) of egmframetype> ','
  [SensorFrame ':='] < expression (IN) of pose> ','
  [SensorFrType ':='] < expression (IN) of egmframetype>
```

下一页继续

续前页

```
['\'x ':=' <expression (IN) of egm_minmax>]
['\'y ':=' <expression (IN) of egm_minmax>]
['\'z ':=' <expression (IN) of egm_minmax>]
['\'rx ':=' <expression (IN) of egm_minmax>]
['\'ry ':=' <expression (IN) of egm_minmax>]
['\'rz ':=' <expression (IN) of egm_minmax>]
['\'LpFilter ':=' <expression (IN) of num>]
['\'SampleRate ':=' <expression (IN) of num>]
['\'MaxPosDeviation ':=' <expression (IN) of num>]
['\'MaxSpeedDeviation ':=' <expression (IN) of num>] ';'
```

相关信息

信息，关于	请参阅
Externally Guided Motion	应用手册 - 控制器软件*IRC5*
数据类型egm_minmax	第*1390*页的*egm_minmax – EGM*的收敛标准
指令EGMRunPose	第*168*页的*EGMRunPose – 执行一次含一个姿态目标点的EGM*移动

1.66 EGMGetId - 获取一个EGM标识

手册用法

EGMGetId的作用是保留一个EGM标识（EGMid），之后便可在其它所有EGM RAPID指令和函数中使用该标识，从而识别关联到所属RAPID运动任务上的某一EGM进程。

egmident的标识就是其名称，即是说如果用相同的egmident来第二次或第三次调用EGMGetId，那么系统既不会保留一项新的EGM进程，也不会更改其内容。

若要释放其它EGM进程使用的egmident，则必须使用RAPID指令EGMReset。

同一时间最多能用4个不同的EGM标识。

基本示例

```
VAR egmident egmID1;
EGMGetId egmID1;
```

变元

```
EGMGetId EGMid
```

EGMid

数据类型：egmident

EGM标识。

限制

- EGMGetId只能用在RAPID运动任务中。

语法

```
EGMGetId
  [EGMid ':='] <variable (VAR) of egmident> ';'
```

相关信息

信息，关于	请参阅
Externally Guided Motion	应用手册 - 控制器软件*IRC5*
指令EGMReset	第*165*页的*EGMReset - 重置一项EGM进程*

1.67 EGMMoveC – 经过路径校正的圆形EGM移动

手册用法

EGMMoveC的作用是沿圆弧将工具中心点（TCP）移到经过路径校正的给定目标点处。在进行这种移动时，相关姿态通常会相对于圆圈保持不变。

基本示例

下例说明了指令EGMMoveC。

例 1

```
VAR egmident EGMid1;
PERS tooldata tReg := [TRUE, [[148,0,326],
     [0.8339007,0,0.551914,0]], [1,[0,0,100], [1,0,0,0], 0,0,0]];
PERS tooldata tLaser := [TRUE, [[148,50,326],
     [0.3902618,-0.589657,-0.589656,0.3902630]],
     [1,[-0.92,0,-0.39], [1,0,0,0], 0,0,0]];
EGMGetId EGMid1;
EGMSetupLTAPP ROB_1, EGMid1, "pathCorr", "OptSim", 1\LATR;
EGMActMove EGMid1, tLaser.tframe\SampleRate:=50;
MoveL p6, v10, fine, tReg\WObj:=wobj0;
EGMMoveL EGMid1, p12, v10, z5, tReg\WObj:=wobj0;
EGMMoveL EGMid1, p7, v10, z5, tReg\WObj:=wobj0;
EGMMoveC EGMid1, p13, p14, v10, z5, tReg\WObj:=wobj0;
EGMMoveL EGMid1, p15, v10, fine, tReg\WObj:=wobj0;
MoveL p8, v1000, z10, tReg\WObj:=wobj0;
EGMReset EGMid1;
```

该程序会登记一项EGM进程，然后设置一个使用通信协议LTAPP且类型为先行的传感器，并以该传感器作为数据来源（传感器）。该传感器应使用关节类型定义编号1来进行跟踪。此外也要设置控制器访问相关装置和该装置传感器框架的速率。

相关机器人会按一条MoveL指令而移到跟踪路径的起点处。EGMMove指令会按相关传感器的校正情况来执行相应的机器人移动。

最后相关机器人会移到一个出发位置，然后释放EGM标识。

变元

```
EGMMoveC EGMid, CirPoint, ToPoint, Speed, Zone, Tool, [\Wobj]
     [\TLoad] [\NoCorr]
```

EGMid

数据类型：egmident

EGM标识。

CirPoint

数据类型：robtarget

相关机器人的圆弧点。圆弧点是指相关起点与终点间的圆弧上的某个位置。若要获得最好的准确度，则宜把该点放在相关起点与终点的正中间处。如果该点太靠近起点或终点，那么相关机器人就可能发出一条警告。将圆弧点定义为一个已命名的位置，或将其直接保存在相关指令（在指令中用一个*标注）中。勿使用外轴的这一位置。

下一页继续

1.67 EGMMoveC – 经过路径校正的圆形EGM移动

Externally Guided Motion

续前页

`ToPoint`

数据类型：`robtarget`

机器人和外部轴的目标点。定义为已命名的位置或直接存储在指令中（在指令中加 * 标记）。

`Speed`

数据类型： `speeddata`

适用于移动的速度数据。速度数据定义了相关TCP的速度、工具的重定方位和外轴。

`Zone`

数据类型：`zonedata`

相关移动的区域数据。区域数据描述了所生成拐角路径的大小。

`Tool`

数据类型： `tooldata`

相关机器人移动时使用的工具。工具中心点是将移过去的指定目标点。

`[\WObj]`

Work Object

数据类型： `wobjdata`

该工件（对象坐标系）与相关指令中的机器人位置相关联。

用户可以忽略该自变数，且如果忽略了该自变数，那么相关位置就会与全局坐标系关联起来。另一方面，如果使用了一个固定TCP或若干协同外轴，那么就必须为一个圆圈（相对于待执行工件的圆圈）指定该自变数。

`[\TLoad]`

Total load

数据类型：`loaddata`

`\TLoad`主动轴描述了移动中使用的总负载。总负载就是相关的工具负载加上该工具正在处理的有效负载。如果使用了`\TLoad`自变数，那么就不考虑当前`tooldata`中的`loaddata`。

如果`\TLoad`自变数被设置成`load0`，那么就不考虑`\TLoad`自变数，而是以当前`tooldata`中的`loaddata`作为代替。

想要使用`\TLoad`自变数，就必需将系统参数`ModalPayLoadMode`的数值设置成0。如果将`ModalPayLoadMode`设置成0，那么就再也无法使用指令`GripLoad`。

可用服务例程“负载标识”（LoadIdentify）来识别总负载。如果系统参数`ModalPayLoadMode`被设置成0，且系统正在运行该服务例程，那么操作员便可将相关工具的`loaddata`复制到一个现有的或新的`loaddata`永久变量中。

如果使用了关联到系统输入项`SimMode`（仿真模式）上的一个数字输入信号，那么便可在没有任何有效负载的情况下试运行该程序。如果该数字输入信号被设置成1，那么

下一页继续

就不考虑可选自变数\TLoad中的loaddata，而是以当前tooldata中的loaddata作为代替。

> **注意**
>
> 处理有效负载的默认功能是使用指令GripLoad，因此系统参数ModalPayLoadMode的默认值为1。

[\NoCorr]

数据类型：switch

关闭路径校正。

程序执行

EGMMoveC会沿一条经某传感器双重校正的编程圆弧路径移动。在移动期间，相关指令会按EGMActMove设置的速率向相关传感器请求校正数据。如果存在可选自变数\NoCorr，那么就不会向该编程路径添加校正。

错误处理

系统会生成下列可恢复错误，并在错误处理器中处理这些错误。系统变量ERRNO将被设置成：

ERR_UDPUC_COMM	与相关UdpUc装置进行通信时发生的一项错误。

限制

- EGMMoveC只能用在RAPID运动任务中。

语法

```
EGMMoveC
  [GMid ':='] <variable (VAR) of egmident> ','
  [CirPoint ':='] < expression (IN) of robtarget> ','
  [ToPoint ':='] < expression (IN) of robtarget> ','
  [Speed ':='] < expression (IN) of speeddata> ','
  [Zone ':='] < expression (IN) of zonedata> ','
  [Tool ':='] < persistent (PERS) of tooldata>
  ['\'WObj ':=' < persistent (PERS) of wobjdata>]
  ['\'TLoad ':=' < persistent (PERS) of loaddata>]
  ['\'NoCorr] ';'
```

相关信息

信息，关于	请参阅
Externally Guided Motion	应用手册 - 控制器软件*IRC5*

1.68 EGMMoveL - 经过路径校正的直线EGM移动

手册用法

EGMMoveL的作用是沿直线将工具中心点（TCP）移到经过路径校正的给定目标点处。当该TCP保持固定时，用户也可用该指令来重定工具方位。

基本示例

下例说明了指令EGMMoveL。

例 1

```
VAR egmident EGMid1;
PERS tooldata tReg := [TRUE, [[148,0,326],
     [0.8339007,0,0.551914,0]], [1,[0,0,100], [1,0,0,0], 0,0,0]];
PERS tooldata tLaser := [TRUE, [[148,50,326],
     [0.3902618,-0.589657,-0.589656,0.3902630]],
     [1,[-0.92,0,-0.39], [1,0,0,0], 0,0,0]];
EGMGetId EGMid1;
EGMSetupLTAPP ROB_1, EGMid1, "pathCorr", "OptSim", 1\LATR;
EGMActMove EGMid1, tLaser.tframe\SampleRate:=50;
MoveL p6, v10, fine, tReg\WObj:=wobj0;
EGMMoveL EGMid1, p12, v10, z5, tReg\WObj:=wobj0;
EGMMoveL EGMid1, p7, v10, z5, tReg\WObj:=wobj0;
EGMMoveC EGMid1, p13, p14, v10, z5, tReg\WObj:=wobj0;
EGMMoveL EGMid1, p15, v10, fine, tReg\WObj:=wobj0;
MoveL p8, v1000, z10, tReg\WObj:=wobj0;
EGMReset EGMid1;
```

该程序会登记一项EGM进程，然后设置一个使用通信协议LTAPP且类型为先行的传感器，并以该传感器作为数据来源（传感器）。该传感器应使用关节类型定义编号1来进行跟踪。此外也要设置控制器访问相关装置和该装置传感器框架的速率。

相关机器人会按一条MoveL指令而移到跟踪路径的起点处。EGMMove指令会按相关传感器的校正情况来执行相应的机器人移动。

最后相关机器人会移到一个出发位置，然后释放EGM标识。

变元

```
EGMMoveL EGMid, ToPoint, Speed, Zone, Tool, [\Wobj] [\TLoad]
     [\NoCorr]
```

EGMid

数据类型：egmident

EGM标识。

ToPoint

数据类型：robtarget

机器人和外部轴的目标点。定义为已命名的位置或直接存储在指令中（在指令中加 * 标记）。

Speed

数据类型： speeddata

适用于移动的速度数据。速度数据定义了相关TCP的速度、工具的重定方位和外轴。

下一页继续

Zone

数据类型：zonedata

相关移动的区域数据。区域数据描述了所生成拐角路径的大小。

Tool

数据类型： tooldata

相关机器人移动时使用的工具。工具中心点是将移过去的指定目标点。

[\WObj]

Work Object

数据类型： wobjdata

该工件（对象坐标系）与相关指令中的机器人位置相关联。

用户可以忽略该自变数，且如果忽略了该自变数，那么相关位置就会与全局坐标系关联起来。另一方面，如果使用了一个固定TCP或若干协同外轴，那么就必须为一个圆圈（相对于待执行工件的圆圈）指定该自变数。

[\TLoad]

Total load

数据类型：loaddata

\TLoad主动轴描述了移动中使用的总负载。总负载就是相关的工具负载加上该工具正在处理的有效负载。如果使用了\TLoad自变数，那么就不考虑当前tooldata中的loaddata。

如果\TLoad自变数被设置成load0，那么就不考虑\TLoad自变数，而是以当前tooldata中的loaddata作为代替。

想要使用\TLoad自变数，就必需将系统参数ModalPayLoadMode的数值设置成0。如果将ModalPayLoadMode设置成0，那么就再也无法使用指令GripLoad。

可用服务例程“负载标识”（LoadIdentify）来识别总负载。如果系统参数ModalPayLoadMode被设置成0，且系统正在运行该服务例程，那么操作员便可将相关工具的loaddata复制到一个现有的或新的loaddata永久变量中。

如果使用了关联到系统输入项SimMode（仿真模式）上的一个数字输入信号，那么便可在没有任何有效负载的情况下试运行该程序。如果该数字输入信号被设置成1，那么就不考虑可选自变数\TLoad中的loaddata，而是以当前tooldata中的loaddata作为代替。

注意

处理有效负载的默认功能是使用指令GripLoad，因此系统参数ModalPayLoadMode的默认值为1。

[\NoCorr]

数据类型：switch

关闭路径校正。

下一页继续

1.68 EGMMoveL – 经过路径校正的直线EGM移动

Externally Guided Motion

续前页

程序执行

EGMMoveL会沿一条经某传感器双重校正的编程直线路径移动。在移动期间，相关指令会按EGMActMove设置的速率向相关传感器请求校正数据。如果存在可选自变数\NoCorr，那么就不会向该编程路径添加校正。

错误处理

系统会生成下列可恢复错误，并在错误处理器中处理这些错误。系统变量ERRNO将被设置成：

ERR_UDPUC_COMM	与相关UdpUc装置进行通信时发生的一项错误。

限制

- EGMMoveL只能用在RAPID运动任务中。

语法

```
EGMMoveL
  [EGMid ':='] <variable (VAR) of egmident> ','
  [ToPoint ':='] < expression (IN) of robtarget> ','
  [Speed ':='] < expression (IN) of speeddata> ','
  [Zone ':='] < expression (IN) of zonedata> ','
  [Tool ':='] < persistent (PERS) of tooldata>
  ['\'WObj ':=' < persistent (PERS) of wobjdata>]
  ['\'TLoad ':=' < persistent (PERS) of loaddata>]
  ['\'NoCorr] ';'
```

相关信息

信息，关于	请参阅
Externally Guided Motion	应用手册 - 控制器软件*IRC5*

1.69 EGMReset – 重置一项EGM进程

手册用法

EGMReset重置了一项特定的EGM进程（EGMid），即是说取消保留。

基本示例

```
VAR egmident egmID1;
PERS pose pose1:=[[0,0,0], [1,0,0,0]];
CONST egm_minmax egm_minmax_lin:=[-0.1,0.1];
CONST egm_minmax egm_minmax_rot:=[-0.1,0.2];
CONST pose posecor:=[[1200,400,900], [0,0,1,0]];
CONST pose posesens:=[[12.3313,-0.108707,416.142],
     [0.903899,-0.00320735,0.427666,0.00765917]];

EGMGetId egmID1;
EGMSetupAI ROB_1, egmID1, "default" \Pose \aiR1x:=ai_01
     \aiR2y:=ai_02 \aiR3z:=ai_03 \aiR4rx:=ai_04 \aiR5ry:=ai_05
     \aiR6rz:=ai_06;
EGMActPose egmID1 \Tool:=tool0 \WObj:=wobj0, posecor,
     EGM_FRAME_WOBJ, posesens, EGM_FRAME_TOOL \x:=egm_minmax_lin
     \y:=egm_minmax_lin \z:=egm_minmax_lin \rx:=egm_minmax_rot
     \ry:=egm_minmax_rot \rz:=egm_minmax_rot \LpFilter:=20;
EGMRunPose egmID1, EGM_STOP_HOLD \x \y \z \rx \ry \rz
     \RampInTime:=0.05;
EGMReset egmID1;
```

变元

```
EGMReset EGMid
```

EGMid

数据类型：egmident

EGM标识。

语法

```
EGMReset
  [EGMid ':='] <variable (VAR) of egmident>';'
```

相关信息

信息，关于	请参阅
Externally Guided Motion	应用手册 - 控制器软件*IRC5*

1.70 EGMRunJoint - 执行一次含一个关节目标点的EGM移动

手册用法

EGMRunJoint会从一项特定的EGM进程（EGMid）的一个精确点处执行一次受传感器指引而前往某个关节目标点的移动，并定义将要移动的关节。

基本示例

```
VAR egmident egmID1;
PERS pose pose1:=[[0,0,0],[1,0,0,0]];
CONST egm_minmax egm_minmax1:=[-1,1];

EGMGetId egmID1;
EGMSetupAI ROB_1, egmID1, "default" \Joint \aiR1x:=ai_01
     \aiR2y:=ai_02 \aiR3z:=ai_03 \aiR4rx:=ai_04 \aiR5ry:=ai_05
     \aiR6rz:=ai_06;
EGMActJoint egmID1, \J1:=egm_minmax1 \J3:=egm_minmax1
     \J4:=egm_minmax1;
EGMRunJoint egmID1, EGM_STOP_HOLD \J1 \J3 \RampInTime:=0.05;
```

变元

```
EGMRunJoint EGMid, Mode [\J1] [\J2] [\J3] [\J4] [\J5] [\J6]
     [\CondTime] [\RampInTime] [\RampOutTime] [\PosCorrGain]
```

EGMid

数据类型：egmident

EGM标识。

Mode

数据类型：egmstopmode

定义如何结束移动（EGM_STOP_HOLD, EGM_STOP_RAMP_DOWN）

[\J1] [\J2] [\J3] [\J4] [\J5] [\J6]

数据类型：switch

移动关节1到6。

[\CondTime]

数据类型：num

时间（以秒为单位），为EGMActJoint中定义的收敛标准。在视为抵达相关目标点、并由EGMRunJoint释放RAPID的执行过程来继续执行下一条指令前，必选先满足这项标准。

默认值为 1 s。

[\RampInTime]

数据类型：num

定义以多快的速率开始移动（以秒为单位）。

[\RampOutTime]

数据类型：num

定义以多快的速率来停止EGM。

下一页继续

如果参数Mode被设置成EGM_STOP_HOLD，那么该参数就毫无意义。

[\PosCorrGain]

数据类型：num

位置校正增益。为0到1之间的一个数值，默认为1。

错误处理

系统会生成下列可恢复错误，并在错误处理器中处理这些错误。系统变量ERRNO将被设置成：

ERR_UDPUC_COMM	与相关UdpUc装置进行通信时发生的一项错误。

限制

- 在首次使用EGMRunJoint前，由于已经启动过相关控制器，因此相关机器人肯定移动过（不论是点动过还是执行了RAPID的某条移动指令）。
- EGMRunJoint移动的起点必须是一个精确点。
- EGMRunJoint只能用在RAPID运动任务中。
- 如果执行了指令EGMActPose而非EGMActJoint，那么将会出现以下错误：*41826 EGM mode mismatch*。
- 如果并未指定从\J1到\J6中的任何一个开关，那么系统就不会执行移动，而RAPID则会继续执行下一条RAPID指令。

语法

```
EGMRunJoint
  [EGMid ':='] <variable (VAR) of egmident> ','
  [Mode ':='] < expression (IN) of egmstopmode>
  ['\'J1]
  ['\'J2]
  ['\'J3]
  ['\'J4]
  ['\'J5]
  ['\'J6]
  ['\'CondTime ':=' <expression (IN) of num>]
  ['\'RampInTime ':=' <expression (IN) of num>]
  ['\'RampOutTime ':=' <expression (IN) of num>]
  ['\'PosCorrGain ':=' <expression (IN) of num>] ';'
```

相关信息

信息，关于	请参阅
Externally Guided Motion	应用手册 - 控制器软件*IRC5*
数据类型egmstopmode	第*1392*页的*egmstopmode - 定义EGM*所需的停止模式

1.71 EGMRunPose – 执行一次含一个姿态目标点的EGM移动

手册用法

EGMRunPose会从一项特定的EGM进程（EGMid）的一个精确点处执行一次受传感器指引而前往某个姿态目标点的移动，并定义可能将要改动的方向和方位。

基本示例

```
VAR egmident egmID1;
PERS pose pose1:=[[0,0,0],[1,0,0,0]];
CONST egm_minmax egm_minmax_lin:=[-0.1,0.1];
CONST egm_minmax egm_minmax_rot:=[-0.1,0.2];
CONST pose posecor:=[[1200,400,900],[0,0,1,0]];
CONST pose posesens:=[[12.3313,-0.108707,416.142],
     [0.903899,-0.00320735,0.427666,0.00765917]];

EGMGetId egmID1;
EGMSetupAI ROB_1, egmID1, "default" \Pose \aiR1x:=ai_01
     \aiR2y:=ai_02 \aiR3z:=ai_03 \aiR4rx:=ai_04 \aiR5ry:=ai_05
     \aiR6rz:=ai_06;
EGMActPose egmID1 \Tool:=tool0 \WObj:=wobj0, posecor,
     EGM_FRAME_WOBJ, posesens, EGM_FRAME_TOOL \x:=egm_minmax_lin
     \y:=egm_minmax_lin \z:=egm_minmax_lin \rx:=egm_minmax_rot
     \ry:=egm_minmax_rot \rz:=egm_minmax_rot \LpFilter:=20;
EGMRunPose egmID1, EGM_STOP_HOLD \x \y \z \rx \ry \rz
     \RampInTime:=0.05;
```

变元

```
EGMRunPose EGMid, Mode [\x] [\y] [\z] [\rx] [\ry] [\rz] [\CondTime]
     [\RampInTime] [\RampOutTime] [\Offset] [\PosCorrGain]
```

EGMid

数据类型：egmident

EGM标识。

Mode

数据类型：egmstopmode

定义如何结束移动（EGM_STOP_HOLD, EGM_STOP_RAMP_DOWN）

[\x] [\y] [\z]

数据类型：switch

x、y和z方向上的移动。

[\rx] [\ry] [\rz]

数据类型：switch

围绕x、y和z轴的重定方位。

[\CondTime]

数据类型：num

下一页继续

时间（以秒为单位），为EGMActPose中定义的收敛标准。在视为抵达相关目标点、并由EGMRunPose释放RAPID的执行过程来继续执行下一条指令前，必选先满足这项标准。

默认值为 1 s。

[\RampInTime]

数据类型：num

定义以多快的速率开始移动（以秒为单位）。

[\RampOutTime]

数据类型：num

定义以多快的速率来停止EGM。

如果参数Mode被设置成EGM_STOP_HOLD，那么该参数就毫无意义。

[\Offset]

数据类型：pose

在相关传感器给出的数值最上方定义一项静态偏移量的可能性。

[\PosCorrGain]

数据类型：num

位置校正增益。为0到1之间的一个数值，默认为1。

错误处理

系统会生成下列可恢复错误，并在错误处理器中处理这些错误。系统变量ERRNO将被设置成：

ERR_UDPUC_COMM	与相关UdpUc装置进行通信时发生的一项错误。

限制

- 在首次使用EGMRunPose前，由于已经启动过相关控制器，因此相关机器人肯定移动过（不论是点动过还是执行了RAPID的某条移动指令）。
- EGMRunPose移动的起点必须是一个精确点。
- EGMRunPose只能用在RAPID运动任务中。
- 如果执行了指令EGMActJoint而非EGMRunPose，那么将会出现以下错误：*41826 EGM mode mismatch*。
- 如果并未指定从\x到\rz中的任何一个开关，那么系统就不会执行移动，而RAPID则会继续执行下一条RAPID指令。

语法

```
EGMRunPose
  [EGMid ':='] <variable (VAR) of egmident>','
  [Mode ':='] < expression (IN) of egmstopmode>
  ['\'x]
  ['\'y]
  ['\'z]
  ['\'rx]
  ['\'ry]
  ['\'rz]
```

下一页继续

1.71 EGMRunPose – 执行一次含一个姿态目标点的EGM移动

Externally Guided Motion

续前页

```
['\'CondTime ':=' <expression (IN) of num>]
['\'RampInTime ':=' <expression (IN) of num>]
['\'RampOutTime ':=' <expression (IN) of num>]
['\'Offset ':=' <expression (IN) of pose>]
['\'PosCorrGain ':=' <expression (IN) of num>] ';'
```

相关信息

信息，关于	请参阅
Externally Guided Motion	应用手册 - 控制器软件*IRC5*
数据类型egmstopmode	第*1392*页的*egmstopmode – 定义EGM所需的停止模式*

1.72 EGMSetupAI – 为EGM设置模拟输入信号

手册用法

EGMSetupAI的作用是为一项特定的EGM进程（EGMid）设置模拟输入信号，以作为指引相关机器人（最多6根附加轴）前往的位置目标点数值的来源。

基本示例

```
VAR egmident egmID1;

EGMGetId egmID1;
EGMSetupAI ROB_1, egmID1, "default" \Pose \aiR1x:=ai_01
     \aiR2y:=ai_02 \aiR3z:=ai_03 \aiR4rx:=ai_04 \aiR5ry:=ai_05
     \aiR6rz:=ai_06;
```

变元

```
EGMSetupAI MecUnit, EGMid, ExtConfigName [\Joint] | [\Pose] |
     [\PathCorr] [\APTR] | [\LATR] [\aiR1x] [\aiR2y] [\aiR3z]
     [\aiR4rx] [\aiR5ry] [\aiR6rz] [\aiE1] [\aiE2] [\aiE3] [\aiE4]
     [\aiE5] [\aiE6]
```

`MecUnit`

数据类型：mecunit

机械单元名称。

`EGMid`

数据类型：egmident

EGM标识。

`ExtConfigName`

数据类型：string

相关系统参数中定义的外部运动接口数据的名称。

更多信息请参见主题*Motion*下类型为*External Motion Interface Data*的技术参考手册 - 系统参数。

`[\Joint]`

数据类型：switch

选择用于位置引导的关节移动。

必须有 \Joint、\Pose、或\PathCorr的至少一个开关。

`[\Pose]`

数据类型：switch

选择用于位置引导的姿势移动。

必须有 \Joint、\Pose、或\PathCorr的至少一个开关。

`[\PathCorr]`

数据类型：switch

选择路径纠正。

必须有 \Joint、\Pose、或\PathCorr的至少一个开关。

下一页继续

1.72 EGMSetupAI - 为EGM设置模拟输入信号

Externally Guided Motion

续前页

[\APTR]

数据类型：switch

建立一个当前点跟踪器类型的传感器，用于路径纠正。例如WeldGuide或AWC。

\APTR或\LATR中至少要有一个。

[\LATR]

数据类型：switch

建立一个智能预测跟踪器类型的传感器，用于路径纠正。例如Laser Tracker。

\APTR或\LATR中至少要有一个。

[\aiR1x] [\aiR2y] [\aiR3z]

数据类型：signalai

指定为姿态移动提供x、y和z值（以毫米为单位）的信号。

指定为关节移动提供机器人关节1到3角度（以度为单位）的信号。

[\aiR4rx] [\aiR5ry] [\aiR6rz]

数据类型：signalai

指定为姿态移动提供相关机器人的x、y和z旋转值（以度为单位）的信号。

指定为关节移动提供机器人关节4到6角度（以度为单位）的信号。

[\aiE1] [\aiE2] [\aiE3] [\aiE4] [\aiE5] [\aiE6]

数据类型：signalai

指定提供附加轴关节1到6之位置的信号。

错误处理

系统会生成下列可恢复错误，并在错误处理器中处理这些错误。系统变量ERRNO将被设置成：

名称	错误原因
ERR_NO_ALIASIO_DEF	该信号变量是在RAPID中声明的一个变量，与用指令AliasIO在I / O配置中定义的I / O信号无关。
ERR_NORUNUNIT	与I / O单元间没有接触。
ERR_SIG_NOT_VALID	无法访问该I / O信号（仅对ICI现场总线有效）。

限制

- EGMSetupAI只能用在RAPID运动任务中。
- 该机械单元必须是一台机器人。
- 必须至少指定一个信号，否则系统会发出一项错误，并停止执行RAPID。

语法

```
EGMSetupAI
  [MecUnit ':='] <variable (VAR) of mecunit> ','
  [EGMid ':='] <variable (VAR) of egmident> ','
  [ExtConfigName ':='] <expression (IN) of string>
  [['\'Joint] | ['\'Pose] | ['\'PathCorr ]]
  [['\'APTR] | ['\'LAT]] ','
  ['\'aiR1x ':=' <variable (VAR) of signalai>]
```

下一页继续

```
['\'aiR2y ':=' <variable (VAR) of signalai>]
['\'aiR3z ':=' <variable (VAR) of signalai>]
['\'aiR4rx ':=' <variable (VAR) of signalai>]
['\'aiR5ry ':=' <variable (VAR) of signalai>]
['\'aiR6rz ':=' <variable (VAR) of signalai>]
['\'aiE1 ':=' <variable (VAR) of signalai>]
['\'aiE2 ':=' <variable (VAR) of signalai>]
['\'aiE3 ':=' <variable (VAR) of signalai>]
['\'aiE4 ':=' <variable (VAR) of signalai>]
['\'aiE5 ':=' <variable (VAR) of signalai>]
['\'aiE6 ':=' <variable (VAR) of signalai>] ';'
```

相关信息

信息，关于	请参阅
Externally Guided Motion	应用手册 - 控制器软件*IRC5*

1.73 EGMSetupAO - 为EGM设施模拟输出信号

手册用法

EGMSetupAO的作用是为一项特定的EGM进程（EGMid）设置AO信号，以作为指引相关机器人（最多6根附加轴）前往的位置目标点数值的来源。

基本示例

```
VAR egmident egmID1;

EGMGetId egmID1;
EGMSetupAO ROB_1, egmID1, "default" \Pose \aoR1x:=ao_01
     \aoR2y:=ao_02 \aoR3z:=ao_03 \aoR4rx:=ao_04 \aoR5ry:=ao_05
     \aoR6rz:=ao_06;
```

变元

```
EGMSetupAO MecUnit, EGMid, ExtConfigName [\Joint] | [\Pose] |
     [\PathCorr] [\APTR] | [\LATR] [\aoR1x] [\aoR2y] [\aoR3Z]
     [\aoR4rx] [\aoR5ry] [\aoR6rz] [\aoE1] [\aoE2] [\aoE3] [\aoE4]
     [\aoE5] [\aoE6]
```

MecUnit

数据类型：mecunit

机械单元名称。

EGMid

数据类型：egmident

EGM标识。

ExtConfigName

数据类型：string

相关系统参数中定义的外部运动接口数据的名称。

更多信息请参见主题*Motion*下类型为*External Motion Interface Data*的技术参考手册 - 系统参数。

[\Joint]

数据类型：switch

选择关节移动。

至少得有开关\Joint或\Pose中的一个。

[\Pose]

数据类型：switch

选择姿态移动。

至少得有开关\Joint或\Pose中的一个。

[\PathCorr]

数据类型：switch

选择路径纠正。

必须有 \Joint、\Pose、或\PathCorr的至少一个开关。

下一页继续

[\APTR]

数据类型：switch

建立一个当前点跟踪器类型的传感器，用于路径纠正。例如WeldGuide或AWC。

\APTR或\LATR中至少要有一个。

[\LATR]

数据类型：switch

建立一个智能预测跟踪器类型的传感器，用于路径纠正。例如Laser Tracker。

\APTR或\LATR中至少要有一个。

[\aoR1x] [\aoR2y] [\aoR3z]

数据类型：signalao

指定为姿态移动提供x、y和z值（以毫米为单位）的信号。

指定为关节移动提供机器人关节1到3角度（以度为单位）的信号。

[\aoR4rx] [\aoR5ry] [\aoR6rz]

数据类型：signalao

指定为姿态移动提供相关机器人的x、y和z旋转值（以度为单位）的信号。

指定为关节移动提供机器人关节4到6角度（以度为单位）的信号。

[\aoE1] [\aoE2] [\aoE3] [\aoE4] [\aoE5] [\aoE6]

数据类型：signalao

指定提供附加轴关节1到6之位置的信号。

错误处理

系统会生成下列可恢复错误，并在错误处理器中处理这些错误。系统变量ERRNO将被设置成：

名称	错误原因
ERR_NO_ALIASIO_DEF	该信号变量是在RAPID中声明的一个变量，与用指令AliasIO在I / O配置中定义的I / O信号无关。
ERR_NORUNUNIT	与I / O单元间没有接触。
ERR_SIG_NOT_VALID	无法访问该I / O信号（仅对ICI现场总线有效）。

限制

- EGMSetupAO只能用在RAPID运动任务中。
- 该机械单元必须是一台机器人。
- 必须至少指定一个信号，否则系统会发出一项错误，并停止执行RAPID。

语法

```
EGMSetupAO
  [MecUnit ':='] <variable (VAR) of mecunit> ','
  [EGMid ':='] <variable (VAR) of egmident> ','
  [ExtConfigName ':='] <expression (IN) of string>
  [['\'Joint] | ['\'Pose] | ['\'PathCorr]]
  [['\'APTR] | ['\'LATR]]
  ['\'aoR1x ':=' <variable (VAR) of signalao>]
```

下一页继续

1.73 EGMSetupAO – 为EGM设施模拟输出信号

Externally Guided Motion

续前页

```
['\'aoR2y ':='] <variable (VAR) of signalao>]
['\'aoR3z ':='] <variable (VAR) of signalao>]
['\'aoR4rx ':='] <variable (VAR) of signalao>]
['\'aoR5ry ':='] <variable (VAR) of signalao>]
['\'aoR6rz ':='] <variable (VAR) of signalao>]
['\'aoE1 ':='] <variable (VAR) of signalao>]
['\'aoE2 ':='] <variable (VAR) of signalao>]
['\'aoE3 ':='] <variable (VAR) of signalao>]
['\'aoE4 ':='] <variable (VAR) of signalao>]
['\'aoE5 ':='] <variable (VAR) of signalao>]
['\'aoE6 ':='] <variable (VAR) of signalao>] ';'
```

相关信息

信息，关于	请参阅
Externally Guided Motion	应用手册 - 控制器软件*IRC5*

1.74 EGMSetupGI – 为EGM设置编组输入信号

手册用法

EGMSetupGI的作用是为一项特定的EGM进程（EGMid）设置编组输入信号，以作为指引相关机器人（最多6根附加轴）前往的位置目标点数值的来源。

基本示例

```
VAR egmident egmID1;

EGMGetId egmID1;
EGMSetupGI ROB_1, egmID1, "default" \Pose \giR1x:=gi_01
     \giR2y:=gi_02 \giR3z:=gi_03 \giR4rx:=gi_04 \giR5ry:=gi_05
     \giR6rz:=gi_06;
```

变元

```
EGMSetupGI MecUnit, EGMid, ExtConfigName [\Joint] | [\Pose] |
     [\PathCorr] [\APTR] | [\LATR] [\giR1x] [\giR2y] [\giR3Z]
     [\giR4rx] [\giR5ry] [\giR6rz] [\giE1] [\giE2] [\giE3] [\giE4]
     [\giE5] [\giE6]
```

MecUnit

数据类型：mecunit

机械单元名称。

EGMid

数据类型：egmident

EGM标识。

ExtConfigName

数据类型：string

相关系统参数中定义的外部运动接口数据的名称。

更多信息请参见主题*Motion*下类型为*External Motion Interface Data*的技术参考手册 - 系统参数。

[\Joint]

数据类型：switch

选择关节移动。

至少得有开关\Joint或\Pose中的一个。

[\Pose]

数据类型：switch

选择姿态移动。

至少得有开关\Joint或\Pose中的一个。

[\PathCorr]

数据类型：switch

选择路径纠正。

必须有 \Joint、\Pose、或\PathCorr的至少一个开关。

下一页继续

1.74 EGMSetupGI - 为EGM设置编组输入信号

Externally Guided Motion

续前页

[\APTR]

数据类型：switch

建立一个当前点跟踪器类型的传感器，用于路径纠正。例如WeldGuide或AWC。

\APTR或\LATR中至少要有一个。

[\LATR]

数据类型：switch

建立一个智能预测跟踪器类型的传感器，用于路径纠正。例如Laser Tracker。

\APTR或\LATR中至少要有一个。

[\giR1x] [\giR2y] [\giR3z]

数据类型：signalgi

指定为姿态移动提供x、y和z值（以毫米为单位）的信号。

指定为关节移动提供机器人关节1到3角度（以度为单位）的信号。

[\giR4rx] [\giR5ry] [\giR6rz]

数据类型：signalgi

指定为姿态移动提供相关机器人的x、y和z旋转值（以度为单位）的信号。

指定为关节移动提供机器人关节4到6角度（以度为单位）的信号。

[\giE1] [\giE2] [\giE3] [\giE4] [\giE5] [\giE6]

数据类型：signalgi

指定提供附加轴关节1到6之位置的信号。

错误处理

系统会生成下列可恢复错误，并在错误处理器中处理这些错误。系统变量ERRNO将被设置成：

名称	错误原因
ERR_NO_ALIASIO_DEF	该信号变量是在RAPID中声明的一个变量，与用指令AliasIO在I / O配置中定义的I / O信号无关。
ERR_NORUNUNIT	与I / O单元间没有接触。
ERR_SIG_NOT_VALID	无法访问该I / O信号（仅对ICI现场总线有效）。

限制

- EGMSetupGI只能用在RAPID运动任务中。
- 该机械单元必须是一台机器人。
- 编组信号只能处理正值，因此它们在EGM中用处有限。
- 必须至少指定一个信号，否则系统会发出一项错误，并停止执行RAPID。

语法

```
EGMSetupGI
  [MecUnit ':='] <variable (VAR) of mecunit> ','
  [EGMid ':='] <variable (VAR) of egmident> ','
  [ExtConfigName ':='] <expression (IN) of string>
  [['\'Joint] | ['\'Pose] | ['\'PathCorr]]
  [['\'APTR] | ['\'LATR]]
```

下一页继续

```
['\'giR1x ':=' <variable (VAR) of signalgi>]
['\'giR2y ':=' <variable (VAR) of signalgi>]
['\'giR3z ':=' <variable (VAR) of signalgi>]
['\'giR4rx ':=' <variable (VAR) of signalgi>]
['\'giR5ry ':=' <variable (VAR) of signalgi>]
['\'giR6rz ':=' <variable (VAR) of signalgi>]
['\'giE1 ':=' <variable (VAR) of signalgi>]
['\'giE2 ':=' <variable (VAR) of signalgi>]
['\'giE3 ':=' <variable (VAR) of signalgi>]
['\'giE4 ':=' <variable (VAR) of signalgi>]
['\'giE5 ':=' <variable (VAR) of signalgi>]
['\'giE6 ':=' <variable (VAR) of signalgi>] ';'
```

相关信息

信息，关于	请参阅
Externally Guided Motion	应用手册 - 控制器软件*IRC5*

1.75 EGMSetupLTAPP – 为EGM设置相应的LTAPP协议

手册用法

EGMSetupLTAPP的作用是为一项特定的EGM进程（EGMid）设置一项*LTAPP*协议，以作为路径校正的来源。

基本示例

下例说明了指令EGMSetupLTAPP。

例 1

```
VAR egmident EGMid1;
EGMGetId EGMid1;
EGMSetupLTAPP ROB_1, EGMid1, "pathCorr", "OptSim", 1\LATR;
```

该程序会登记一项EGM进程，然后设置一个使用通信协议LTAPP且类型为先行的OptSim传感器，并以该传感器作为数据来源（传感器）。该传感器应使用关节类型定义编号1来进行跟踪。

变元

```
EGMActMove MecUnit, EGMid, ExtConfigName, Device, JointType [\APTR]
     | [\LATR]
```

MecUnit

数据类型：mecunit

机械单元名称。

EGMid

数据类型：egmident

EGM标识。

ExtConfigName

数据类型：string

相关系统参数中定义的外部运动接口数据的名称。

更多信息请参见主题*Motion*下类型为*External Motion Interface Data*的技术参考手册 - 系统参数。

Device

数据类型：string

LTAPP装置名称。

JointType

数据类型：num

定义路径校正期间相关传感器设备应采用的关节类型（用一个数字表达）。

[\APTR]

数据类型：switch

建立一个当前点跟踪器类型的传感器，用于路径纠正。例如WeldGuide或AWC。

\APTR或\LATR中至少要有一个。

下一页继续

[\LATR]

数据类型：switch

建立一个智能预测跟踪器类型的传感器，用于路径纠正。例如Laser Tracker。

\APTR或\LATR中至少要有一个。

程序执行

EGMSetupLTAPP会关联到用于某个EGM标识使用的传感器特征数据上，之后用户便可在不同的EGMActMove和EGMMove指令中使用该EGM标识。

语法

```
EGMSetupLTAPP
  [MecUnit ':='] <variable (VAR) of mecunit> ','
  [EGMid ':='] <variable (VAR) of egmident> ','
  [ExtConfigName ':='] < expression (IN) of string> ','
  [Device ':='] < expression (IN) of string> ','
  [JointType ':='] < expression (IN) of num>
  [['\'APTR] | ['\'LATR]] ';'
```

相关信息

信息，关于	请参阅
Externally Guided Motion	应用手册 - 控制器软件*IRC5*

1.76 EGMSetupUC – 为EGM设置UdpUc协议

手册用法

EGMSetupUC的作用是为一项特定的EGM进程（EGMid）设置UdpUc协议，以作为指引相关机器人（最多6根附加轴）前往的位置目标点数值的来源。

基本示例

```
VAR egmident egmID1;
VAR string egmSensor:="egmSensor:";
EGMGetId egmID1;
EGMSetupUC ROB_1, egmID1, "default", egmSensor\Pose;
```

变元

```
EGMSetupUC MecUnit, EGMid, ExtConfigName, UCDevice [\Joint] |
     [\Pose] | [\PathCorr] [\APTR] | [\LATR] [\CommTimeout]
```

MecUnit

数据类型：mecunit

机械单元名称。

EGMid

数据类型：egmident

EGM标识。

ExtConfigName

数据类型：string

相关系统参数中定义的外部运动接口数据的名称。

更多信息请参见主题*Motion*下类型为*External Motion Interface Data*的技术参考手册 - 系统参数。

UCDevice

数据类型：string

UdpUc装置名称。

[\Joint]

数据类型：switch

选择用于位置引导的关节移动。

必须有 \Joint、\Pose、或\PathCorr的至少一个开关。

[\Pose]

数据类型：switch

选择用于位置引导的姿势移动。

必须有 \Joint、\Pose、或\PathCorr的至少一个开关。

[\PathCorr]

数据类型：switch

选择路径纠正。

必须有 \Joint、\Pose、或\PathCorr的至少一个开关。

下一页继续

[\APTR]

数据类型：switch

建立一个当前点跟踪器类型的传感器，用于路径纠正。例如WeldGuide或AWC。

\APTR或\LATR中至少要有一个。

[\LATR]

数据类型：switch

建立一个智能预测跟踪器类型的传感器，用于路径纠正。例如Laser Tracker。

\APTR或\LATR中至少要有一个。

[\CommTimeout]

数据类型：num

与外部UdpUC装置通信的超时时间（以秒为单位）。

错误处理

系统会生成下列可恢复错误，并在错误处理器中处理这些错误。系统变量ERRNO将被设置成：

ERR_UDPUC_COMM	与相关UdpUc装置进行通信时发生的一项错误。

限制

- EGMSetupUC只能用在RAPID运动任务中。
- 该机械单元必须是一台机器人。

语法

```
EGMSetupUC
  [MecUnit ':='] <variable (VAR) of mecunit> ','
  [EGMid ':='] <variable (VAR) of egmident> ','
  [ExtConfigName ':='] <expression (IN) of string> ','
  [UCDevice ':='] <expression (IN) of string>
  [['\'Joint] | ['\'Pose] | ['\'PathCorr]]
  [['\'APTR] | ['\'LATR]]
  ['\'CommTimeout ':=' <expression (IN) of num>] ';'
```

相关信息

信息，关于	请参阅
Externally Guided Motion	应用手册 - 控制器软件*IRC5*

1.77 EGMStop - 停止一次EGM移动

手册用法

EGMStop会停止一项特定的EGM进程（EGMid）。

基本示例

在RAPID运动任务中：

```
VAR egmident egmID1;
PERS pose pose1:=[[0,0,0], [1,0,0,0]];
CONST egm_minmax egm_minmax_lin:=[-0.1,0.1];
CONST egm_minmax egm_minmax_rot:=[-0.1,0.2];
CONST pose posecor:=[[1200,400,900], [0,0,1,0]];
CONST pose posesens:=[[12.3313,-0.108707,416.142],
     [0.903899,-0.00320735,0.427666,0.00765917]];

EGMGetId egmID1;
EGMSetupAI ROB_1, egmID1 \Pose \aiR1x:=ai_01 \aiR2y:=ai_02
     \aiR3z:=ai_03 \aiR4rx:=ai_04 \aiR5ry:=ai_05 \aiR6rz:=ai_06;
EGMActPose egmID1 \Tool:=tool0 \WObj:=wobj0, posecor,
     EGM_FRAME_WOBJ, posesens, EGM_FRAME_TOOL \x:=egm_minmax_lin
     \y:=egm_minmax_lin \z:=egm_minmax_lin \rx:=egm_minmax_rot
     \ry:=egm_minmax_rot \rz:=egm_minmax_rot \LpFilter:=20;
EGMRunPose egmID1, EGM_STOP_HOLD \x \y \z \rx \ry \rz
     \RampInTime:=0.05;
```

在TRAP例程中：

```
EGMStop egmID1, EGM_STOP_HOLD;
```

变元

```
EGMStop EGMid, Mode [\RampOutTime]
```

EGMid

数据类型：egmident

EGM标识。

Mode

数据类型：egmstopmode

定义如何结束移动（EGM_STOP_HOLD, EGM_STOP_RAMP_DOWN）

[\RampOutTime]

数据类型：num

定义以多快的速率来停止EGM。

如果参数Mode被设置成EGM_STOP_HOLD，那么该参数就毫无意义。

限制

- EGMStop只能用在RAPID运动任务中。

下一页继续

语法

```
EGMStop
  [EGMid ':='] <variable (VAR) of egmident>','
  [Mode ':='] < expression (IN) of egmstopmode>
  ['\'RampOutTime ':=' <expression (IN) of num>] ';'
```

相关信息

信息，关于	请参阅
Externally Guided Motion	应用手册 - 控制器软件*IRC5*

1.78 EOffsOff - 停用附加轴的偏移量

手册用法

EOffsOff（*External Offset Off*）用于停用附加轴的偏移量。

通过指令EOffsSet或EOffsOn启用附加轴的偏移量，并用于所有运动，直至附加轴的一些其他偏移量启用，或直至附加轴的偏移量停用。

本指令仅可用于主任务T_ROB1，或者如果在MultiMove系统中，则可用于运动任务中。

基本示例

以下实例介绍了指令EOffsOff：

例 1

```
EOffsOff;
```

附加轴的偏移量停用

例 2

```
MoveL p10, v500, z10, tool1;
EOffsOn \ExeP:=p10, p11;
MoveL p20, v500, z10, tool1;
MoveL p30, v500, z10, tool1;
EOffsOff;
MoveL p40, v500, z10, tool1;
```

将偏移量定义为p10和p11处各轴位置之间的差异。该位移会对p20和p30的运动产生影响，但是不会对p40的运动产生影响。

程序执行

重置附加轴的有效偏移量。

语法

```
EOffsOff ';'
```

相关信息

信息，关于	请参阅
定义使用两个位置的偏移量	第*187*页的*EOffsOn - 启用附加轴的偏移量*
定义使用已知值的偏移量	第*189*页的*EOffsSet - 启用附加轴（使用已知值）的偏移量*
停用机械臂的程序位移	第*447*页的*PDispOff - 停用程序位移*

1.79 EOffsOn - 启用附加轴的偏移量

手册用法

EOffsOn（*External Offset On*）用于定义和启用附加轴（使用两个位置）的偏移量。

本指令仅可用于主任务T_ROB1，或者如果在MultiMove系统中，则可用于运动任务中。

基本示例

以下实例介绍了指令EOffsOn：

另请参阅第*188*页的更多示例

例 1

```
MoveL p10, v500, z10, tool1;
EOffsOn \ExeP:=p10, p20;
```

启用附加轴的偏移量。根据位置p10和p20之间的差异，计算各个轴的偏移量。

例 2

```
MoveL p10, v500, fine \Inpos := inpos50, tool1;
EOffsOn *;
```

启用附加轴的偏移量。由于已经在先前指令中使用了准确定义的停止点，因此，无须使用参数\ExeP。根据各个轴的实际位置与指令中所储存的编程点（*）之间的差异，计算位移。

变元

```
EOffsOn [\ExeP] ProgPoint
```

[\ExeP]

Executed Point

数据类型： robtarget

新位置用于计算偏移量。如果省略该参数，则使用程序执行时轴的当前位置。

ProgPoint

Programmed Point

数据类型： robtarget

轴在编程时的原始位置。

程序执行

根据各附加轴\ExeP与ProgPoint之间的差异，计算偏移量。如果尚未指定\ExeP，则转而使用轴在程序执行时的当前位置。由于使用各个轴的实际位置，因此，在执行EOffsOn时不得移动各个轴。

随后，使用偏移量来取代后续定位指令中附加轴的位置，并保持有效，直至启用一些其他偏移量（指令EOffsSet或EOffsOn），或直至停用附加轴的偏移量（指令EOffsOff）。

在同一时间，仅可启用各单个附加轴的一个偏移量。另一方面，如果可相继地编程若干EOffsOn，则将增加不同的偏移量。

自动重置附加轴偏移量：

- 当使用重启模式重置RAPID时

下一页继续

1.79 EOffsOn - 启用附加轴的偏移量
RobotWare - OS
续前页

- 当加载一则新程序或一个新模块时
- 当从起点开始执行程序时
- 当将程序指针移动到main时
- 当将程序指针移动到子程序时
- 移动程序指针造成执行顺序丢失时

更多示例

有关于如何使用指令EOffsOn的更多例子阐述如下。

例 1

```
SearchL sen1, psearch, p10, v100, tool1;
PDispOn \ExeP:=psearch, *, tool1;
EOffsOn \ExeP:=psearch, *;
```

进行搜索，将搜索的机械臂和附加轴位置储存在位置psearch中。在此之后进行的运动，均始于使用机械臂和附加轴程序位移的位置。根据指令中储存的搜索位置与编程点（*）之间的差异，进行计算。

语法

```
EOffsOn
  [ '\' ExeP ':=' < expression (IN) of robtarget> ',']
  [ ProgPoint ':=' ] < expression (IN) of robtarget> ';'
```

相关信息

信息，关于	请参阅
停用附加轴的偏移量	第*186*页的*EOffsOff - 停用附加轴的偏移量*
定义使用已知值的偏移量	第*189*页的*EOffsSet - 启用附加轴（使用已知值）的偏移量*
机械臂运动的位移	第*448*页的*PDispOn - 启用程序位移*
坐标系	技术参考手册 - *RAPID*语言概览

1.80 EOffsSet - 启用附加轴（使用已知值）的偏移量

手册用法

EOffsSet（*External Offset Set*）用于定义和启用附加轴（使用已知值）的偏移量。

本指令仅可用于主任务T_ROB1，或者如果在MultiMove系统中，则可用于运动任务中。

基本示例

以下实例介绍了指令EOffsSet：

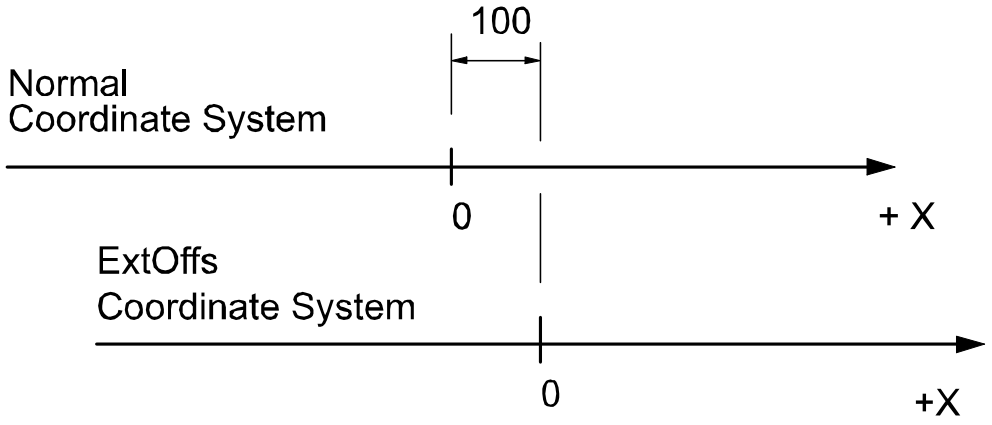
例 1

```
VAR extjoint eax_a_p100 := [100, 0, 0, 0, 0, 0];
...
EOffsSet eax_a_p100;
```

启用附加轴的偏移量eax_a_p100，意味着（假定逻辑附加轴“a”呈线性）：

- 针对逻辑轴“a”，将ExtOffs坐标系转移100 mm（参见下图）。
- 只要偏移量有效，则在x轴方向上的所有位置均将转移100 mm。

下图显示了附加轴的位移。



xx0500002162

变元

```
EOffsSet EAxOffs
```

EAxOffs

External Axes Offset

数据类型：extjoint

将附加轴的偏移量定义为extjoint型数据，表达为：

- 线性轴的mm
- 旋转轴的度数

程序执行

当执行EOffsSet指令时，启用附加轴的偏移量，并保持有效，直至启用一些其他的偏移量（指令EOffsSet或EOffsOn），或者直至停用附加轴的偏移量（指令EOffsOff）。

下一页继续

1.80 EOffsSet - 启用附加轴（使用已知值）的偏移量

RobotWare - OS

续前页

在同一时间，仅能启用附加轴的一个偏移量。运用EOffsSet，无法将偏移量从一个附加轴添加至另一个。

自动重置附加轴偏移量：

- 当使用重启模式重置RAPID时
- 当加载一则新程序或一个新模块时
- 当从起点开始执行程序时
- 当将程序指针移动到main时
- 当将程序指针移动到子程序时
- 移动程序指针造成执行顺序丢失时

语法

```
EOffsSet
  [ EAxOffs ':=' ] < expression (IN) of extjoint> ';'
```

相关信息

信息，关于	请参阅
启用附加轴的偏移量	第*187*页的*EOffsOn - 启用附加轴的偏移量*
停用附加轴的偏移量	第*186*页的*EOffsOff - 停用附加轴的偏移量*
机械臂运动的位移	第*448*页的*PDispOn - 启用程序位移*
extjoint型数据的定义	第*1405*页的*extjoint - 外接头的位置*
坐标系	技术参考手册 - *RAPID*语言概览

1.81 EraseModule - 擦除模块

手册用法

EraseModule用于在执行期间，从程序内存移除模块。

不存在有关于如何加载模块的限制。其可以根据配置或者结合指令Load、StartLoad和WaitLoad，手动实施加载。

无法将模块定义为配置中的*Shared*。

基本示例

以下实例介绍了指令EraseModule：

例 1

```
EraseModule "PART_A";
```

从程序内存中擦除普通程序模块PART_A。

变元

```
EraseModule ModuleName
```

ModuleName

数据类型：string

应当移除模块的名称。请注意，此为模块的名称而非文件的名称。

程序执行

程序执行等待普通程序模块在执行下一个指令之前，完成消除过程。

消除普通程序模块时，剩余的普通程序模块将相连。

限制

不允许消除正在执行中的普通程序模块。

消除过程期间，无法执行软中断程序、系统I/O事件以及其他程序任务。

消除期间，避免机械臂不间断地移动。

执行EraseModule指令期间，程序停止，导致保护停止，且FlexPendant示教器上出现电机关闭和错误消息“20025停止指令超时”。

错误处理

可能会产生下列可恢复错误。错误可以由错误处理程序处理。系统变量ERRNO将设置为：

名称	错误原因
ERR_MODULE	无法清除EraseModule指令中的文件，因为未发现该文件。

语法

```
EraseModule
[ModuleName':=']<expression (IN) of string>';'
```

下一页继续

1.81 EraseModule - 擦除模块

RobotWare - OS

续前页

相关信息

信息，关于	请参阅
卸载普通程序模块	第*843*页的*UnLoad - 执行期间，卸载普通程序模块*
加载同另一程序执行平行的普通程序模块。	第*660*页的*StartLoad - 执行期间，加载普通程序模块* 第*881*页的*WaitLoad - 将加载的模块与任务相连*
接受未解决的参考	技术参考手册 - 系统参数，*Controller*一节

1.82 ErrLog - 写入错误消息

手册用法

ErrLog 用于在FlexPendant示教器上显示错误消息，并将其写入事件日志。必须表明错误编号和五个错误参数。将消息储存在机械臂日志中的过程域中。ErrLog亦可用于显示警告和信息消息。

基本示例

以下实例介绍了指令ErrLog：

例 1

如果你不想创建自己的.xml文件，则可以使用ErrorId 4800，如以下例子：

```
VAR errstr my_title := "myerror";
VAR errstr str1 := "errortext1";
VAR errstr str2 := "errortext2";
VAR errstr str3 := "errortext3";
VAR errstr str4 := "errortext4";
ErrLog 4800, my_title, str1,str2,str3,str4;
```

在FlexPendant示教器上，消息将如下所示：

事件消息：4800

myerror

errortext1

errortext2

errortext3

errortext4

例 2

必须在.xml文件中声明ErrorId。数字必须介于5000 - 9999之间。通过ErrLog指令，将错误消息写入.xml文件，并呈报消息的参数。.xml文件中的ErrorId与ErrLog指令中规定的相同。

注：如果使用介于*5000-9999*之间的*ErrorId*，则必须安装自己的*xml*文件。

.xml文件中消息的实例：

```
<Message number="5210" eDefine="ERR_INPAR_RDONLY">
  <Title>Parameter error</Title>
  <Description>Task:<arg format="%s" ordinal="1" />
    <p />Symbol <arg format="%s" ordinal="2" />is read-only
    <p />Context:<arg format="%s" ordinal="3" /><p />
  </Description>
</Message>
```

指令实例：

```
MODULE MyModule
  PROC main()
    VAR num errorid := 5210;
    VAR errstr arg := "P1";
    ErrLog errorid, ERRSTR_TASK, arg, ERRSTR_CONTEXT,ERRSTR_UNUSED,
         ERRSTR_UNUSED;
```

下一页继续

1.82 ErrLog - 写入错误消息
RobotWare - OS
续前页

```
        ErrLog errorid \W, ERRSTR_TASK, arg,
             ERRSTR_CONTEXT,ERRSTR_UNUSED, ERRSTR_UNUSED;
    ENDPROC
ENDMODULE
```

在FlexPendant示教器上，消息将如下所示：

事件消息：5210

参数错误

任务：T_ROB1

符号P1仅供读取。

范围：MyModule/main/ErrLog

第一个ErrLog指令产生一条错误消息。将该消息储存在过程域的机械臂日志中。其同时在FlexPendant示教器上显示。

第二个指令是警告。仅将消息储存在机械臂日志中。

当完成指令时，在两种情况下，程序均将继续执行。

变元

```
ErrLog ErrorID [\W] | [\I] Argument1 Argument2 Argument3 Argument4
     Argument5
```

ErrorId

数据类型：num

待监测特定错误的数量。如果使用预先安装的xml文件，则错误编号必须介于4800-4814；如果使用自己的xml文件，则错误编号必须介于5000 - 9999。

[\W]

Warning

数据类型：switch

发出仅储存在机械臂事件日志中的警告（并非直接在FlexPendant示教器上显示）。

[\I]

Information

数据类型：switch

发出仅储存在事件日志中的信息消息（并非直接在FlexPendant示教器上显示）。

如果未指定参数\W 或\I，则指令将直接在FlexPendant示教器上产生错误消息，并同时将其储存在事件日志中。

Argument1

数据类型：errstr

错误消息中的第一个参数。可以使用任意errstr型字符串或预定义数据。

Argument2

数据类型：errstr

错误消息中的第二个参数。可以使用任意errstr型字符串或预定义数据。

Argument3

数据类型：errstr

下一页继续

错误消息中的第三个参数。可以使用任意errstr型字符串或预定义数据。

Argument4

数据类型：errstr

错误消息中的第四个参数。可以使用任意errstr型字符串或预定义数据。

Argument5

数据类型：errstr

错误消息中的第五个参数。可以使用任意errstr型字符串或预定义数据。

程序执行

在FlexPendant示教器上显示错误消息（最多5行），并将其写入到事件日志中。

若为参数\W或参数 \I，则将警告或信息消息写入到事件日志中。

如果使用由系统安装的xml文件，则ErrLog会产生介于4800-4814之间的程序错误，如果安装自己的xml文件，则会产生介于5000-9999之间的程序错误。产生的错误取决于指定的ErrorID 。

将消息储存在事件日志的过程域中。

*Additional options*手册中描述了如何安装自己的xml文件，参见下文相关信息。

限制

将总字符串长度（参数1-参数5）限制在195个字符。

语法

```
ErrLog
  [ErrorId ':=' ] < expression (IN) of num> ','
  [ '\'W ] | [' \' I ] ','
  [Argument1 ':=' ] < expression (IN) of errstr> ','
  [Argument2 ':=' ] < expression (IN) of errstr> ','
  [Argument3 ':=' ] < expression (IN) of errstr> ','
  [Argument4 ':=' ] < expression (IN) of errstr> ','
  [Argument5 ':=' ] < expression (IN) of errstr> ';'
```

相关信息

信息，关于	请参阅
errstr型预定义数据	第*1401*页的*errstr - 错误字符串*
在FlexPendant示教器上显示消息	第*744*页的*TPWrite - 写入FlexPendant示教器* 第*832*页的*UIMsgBox - 用户消息对话框，基本类型*
事件日志	操作员手册 - 带 *FlexPendant* 的 *IRC5*
事件日志消息、xml文件说明	应用手册 - 附加功能，事件日志消息一节
如何在使用附加功能时安装XML文件	应用手册 - 附加功能

1.83 ErrRaise - 写入警告，调用错误处理器

手册用法

ErrRaise 用于在程序中创建错误，然后调用程序的错误处理器。将警告写入事件日志。ErrRaise亦可用于错误处理器中，以将当前错误传播至调用程序的错误处理器。

必须规定错误名称、错误编号和五个错误自变量。将消息储存在机械臂日志的过程域中。

基本示例

以下实例介绍了指令ErrRaise：

例 1

如果你不想创建自己的.xml文件，则可以使用ErrorId 4800，如以下例子：

```
MODULE MyModule
  VAR errnum ERR_BATT:=-1;
  PROC main()
    VAR num errorid := 4800;
    VAR errstr my_title := "Backup battery status";
    VAR errstr str1 := "Bacup battery is fully charged";
    BookErrNo ERR_BATT;
    ErrRaise "ERR_BATT", errorid, my_title, ERRSTR_TASK, str1,
         ERRSTR_CONTEXT,ERRSTR_EMPTY;
    ERROR
    IF ERRNO = ERR_BATT THEN
      TRYNEXT;
    ENDIF
  ENDPROC
ENDMODULE
```

在FlexPendant示教器上，消息将如下所示（警告和/或错误）：

事件消息：4800

备用电池状态

任务:主要

备用电池充满电

范围：MyModule/main/ErrRaise

必须通过指令 BookErrNo来登记错误编号。将相关字符串表示为ErrRaise中的第一个参数，ErrorName。

ErrRaise创建错误，然后调用错误处理器。如果错误受到关注，则在过程域的事件日志中生成警告。否则，会产生致命错误，并使程序停止。

ErrRaise亦可用于子程序中的错误处理器中。在这种情况下，继续在调用程序的错误处理器中执行。

例 2

必须在.xml文件中声明ErrorId。数字必须介于5000 - 9999之间。通过ErrRaise指令，将错误消息写入.xml文件，并呈报消息的参数。.xml文件中的ErrorId与ErrRaise指令中规定的相同。

注：如果使用介于*5000-9999*之间的*ErrorId*，则必须安装自己的*xml*文件。

下一页继续

.xml文件中消息的实例：

```
<Message number="7055" eDefine="SYS_ERR_ARL_INPAR_RDONLY">
  <Title>Parameter error</Title>
  <Description>Task:<arg format="%s" ordinal="1" />
    <p />Symbol <arg format="%s" ordinal="2" />is read-only
    <p />Context:<arg format="%s" ordinal="3" /><p /></Description>
</Message>
```

指令实例：

```
MODULE MyModule
  VAR errnum ERR_BATT:=-1;
  PROC main()
    VAR num errorid := 7055;
    BookErrNo ERR_BATT;
    ErrRaise "ERR_BATT", errorid, ERRSTR_TASK,
         ERRSTR_CONTEXT,ERRSTR_UNUSED, ERRSTR_UNUSED,
         ERRSTR_UNUSED;
    ERROR
    IF ERRNO = ERR_BATT THEN
      TRYNEXT;
    ENDIF
  ENDPROC
ENDMODULE
```

在FlexPendant示教器上，消息将如下所示（警告和/或错误）：

事件消息：7055

备用电池状态

任务:主要

备用电池充满电

范围：MyModule/main/ErrRaise

必须通过指令 `BookErrNo`**来登记错误编号。将相关字符串表示为**`ErrRaise`**中的第一个参数，**`ErrorName`**。**

`ErrRaise`**创建错误，然后调用错误处理器。如果错误受到关注，则在过程域的事件日志中生成警告。否则，会产生致命错误，并使程序停止。**

`ErrRaise`**亦可用于子程序中的错误处理器中。在这种情况下，继续在调用程序的错误处理器中执行。**

变元

```
ErrRaise ErrorName ErrorId Argument1 Argument2 Argument3 Argument4
     Argument5
```

`ErrorName`

数据类型：`string`

必须运用指令`BookErrNo`**来登记错误编号。将相应的变量表示为**`ErrorName`**。**

`ErrorId`

数据类型：`num`

待监测特定错误的数量。如果使用预先安装的xml文件，则错误编号必须介于4800-4814；如果使用自己的xml文件，则错误编号必须介于5000 - 9999。

下一页继续

1.83 ErrRaise - 写入警告，调用错误处理器

RobotWare - OS

续前页

Argument1

数据类型：errstr

错误消息中的第一个参数。可以使用任意errstr型字符串或预定义数据。

Argument2

数据类型：errstr

错误消息中的第二个参数。可以使用任意errstr型字符串或预定义数据。

Argument3

数据类型：errstr

错误消息中的第三个参数。可以使用任意errstr型字符串或预定义数据。

Argument4

数据类型：errstr

错误消息中的第四个参数。可以使用任意errstr型字符串或预定义数据。

Argument5

数据类型：errstr

错误消息中的第五个参数。可以使用任意errstr型字符串或预定义数据。

程序执行

如果使用由系统安装的xml文件，则ErrRaise会产生介于4800-4814之间的程序警告，如果安装自己的xml文件，则会产生介于5000-9999之间的程序警告。产生的错误取决于指定的ErrorID。将警告写入域过程的机械臂消息日志中。

当执行ErrRaise时，行为取决于执行位置：

- 当在程序本体中执行指令时，会产生警告，并在错误处理器中继续执行。
- 当在错误处理器中执行指令时，会跳过旧的警告，产生新的警告，并提高控制以调用指令。

限制

将总字符串长度（参数1-参数5）限制在195个字符。

更多示例

有关于如何使用指令ErrRaise的更多例子阐述如下。

例 1

```
VAR errnum ERR_BATT:=-1;
VAR errnum ERR_NEW_ERR:=-1;

PROC main()
  testerrraise;
ENDPROC

PROC testerrraise()
  BookErrNo ERR_BATT;
  BookErrNo ERR_NEW_ERR;
  ErrRaise "ERR_BATT",7055,ERRSTR_TASK,ERRSTR_CONTEXT,
       ERRSTR_UNUSED,ERRSTR_UNUSED,ERRSTR_UNUSED;
```

下一页继续

```
  ERROR
  IF ERRNO = ERR_BATT THEN
    ErrRaise "ERR_NEW_ERR",7156,ERRSTR_TASK,ERRSTR_CONTEXT,
         ERRSTR_UNUSED,ERRSTR_UNUSED, ERRSTR_UNUSED;
  ENDIF
ENDPROC
```

从错误处理器产生新的警告7156。提高控制，以调用程序并停止执行。

语法

```
ErrRaise
  [ErrorName ':=' ] < expression (IN) of string> ','
  [ErrorId ':=' ] < expression (IN) of num> ','
  [Argument1 ':=' ] < expression (IN) of errstr> ','
  [Argument2 ':=' ] < expression (IN) of errstr> ','
  [Argument3 ':=' ] < expression (IN) of errstr> ','
  [Argument4 ':=' ] < expression (IN) of errstr> ','
  [Argument5 ':=' ] < expression (IN) of errstr> ';'
```

相关信息

信息，关于	请参阅
errstr型预定义数据	第*1401*页的*errstr - 错误字符串*
登记错误编号	第*40*页的*BookErrNo - 登记RAPID系统错误编号*
错误处理	技术参考手册 - *RAPID*语言概览
Advanced RAPID	应用手册 - 控制器软件*IRC5*

1.84 ErrWrite - 写入错误消息

手册用法

ErrWrite（*Error Write*）用于在FlexPendant示教器上显示错误消息，并将其写入事件日志。其亦可以用于显示警告和信息消息。

基本示例

以下实例介绍了指令ErrWrite：

例 1

```
ErrWrite "PLC error", "Fatal error in PLC" \RL2:="Call service";
Stop;
```

将消息储存在机械臂日志中。消息亦将在FlexPendant示教器上显示。

例 2

```
ErrWrite \W, "Search error", "No hit for the first search";
RAISE try_search_again;
```

仅将消息储存在机械臂日志中。然后，继续程序执行。

变元

```
ErrWrite [ \W ] | [\I] Header Reason [ \RL2] [ \RL3] [ \RL4]
```

[\W]

Warning

数据类型：switch

发出仅储存在机械臂错误消息日志中的警告（并非直接在FlexPendant示教器上显示）。

[\I]

Information

数据类型：switch

发出仅储存在事件日志中的信息消息（并非直接在FlexPendant示教器上显示）。

如果未指定参数\W 或\I，则指令将直接在FlexPendant示教器上产生错误消息，并同时将其储存在事件日志中。

Header

数据类型：string

错误消息标题（最多46个字符）。

Reason

数据类型：string

错误原因。

[\RL2]

Reason Line 2

数据类型：string

错误原因。

下一页继续

[\RL3]

Reason Line 3

数据类型：string

错误原因。

[\RL4]

Reason Line 4

数据类型：string

错误原因。

程序执行

在FlexPendant示教器上显示错误消息（最多5行），并将其写入到机械臂消息日志中。

若为参数\W或参数 \I，则将警告或信息消息写入到事件日志中。

ErrWrite针对错误、警告（\W）和信息消息(\I).分别产生编号为80001、80002和80003的程序错误

限制

将总字符串长度（数据头+原因+\RL2+\RL3+\RL4）限制在195个字符。

语法

```
ErrWrite
  [ '\'W ] | [ '\' I ] ','
  [ Header ':=' ] < expression (IN) of string>','
  [ Reason ':=' ] < expression (IN) of string>
  [ ’\’RL2 ':=' < expression (IN) of string> ]
  [ ’\’RL3 ':=' < expression (IN) of string> ]
  [ ’\’RL4 ':=' < expression (IN) of string> ] ';'
```

相关信息

信息，关于	请参阅
errstr型预定义数据	第*1401*页的*errstr - 错误字符串*
在FlexPendant示教器上显示消息	第*744*页的*TPWrite - 写入FlexPendant示教器* 第*832*页的*UIMsgBox - 用户消息对话框，基本类型*
事件日志	操作员手册 - 带 *FlexPendant* 的 *IRC5*
写入错误消息 - Err Log	第*193*页的*ErrLog - 写入错误消息*

1.85 EXIT - 终止程序执行

手册用法

EXIT 用于终止程序执行。随后，堵塞程序重启，其为仅可从主程序第一个指令重启的程序。

当出现致命错误或永久地停止程序执行时，应当使用EXIT指令。Stop指令用于临时停止程序执行。在执行指令EXIT后，程序指针消失。为继续程序执行，必须设置程序指针。

基本示例

以下实例介绍了指令EXIT：

例 1

```
ErrWrite "Fatal error","Illegal state";
EXIT;
```

程序执行停止，且无法从程序中的该位置重启。

语法

```
EXIT ';'
```

相关信息

信息，关于	请参阅
临时停止程序执行	第*685*页的*Stop - 停止程序执行*

1.86 ExitCycle - 中断当前循环，并开始下一循环

手册用法

ExitCycle 用于中断当前循环，将程序指针（PP）移回至主程序中第一个指令处。
如果以连续模式执行程序，则其将开始执行下一循环。
如果以循环模式执行，则将在主程序中的第一个指令处停止执行。

基本示例

以下实例介绍了指令ExitCycle：

例 1

```
VAR num cyclecount:=0;
VAR intnum error_intno;

PROC main()
  IF cyclecount = 0 THEN
    CONNECT error_intno WITH error_trap;
    ISignalDI di_error,1,error_intno;
  ENDIF
  cyclecount:=cyclecount+1;
  ! start to do something intelligent
  ...
ENDPROC

TRAP error_trap
  TPWrite "ERROR, I will start on the next item";
  ExitCycle;
ENDTRAP
```

如果设置信号di_error，则将启动下一循环。

程序执行

在对机械单元进行控制的程序任务中执行 ExitCycle，会在实际任务中产生：

- 进行中的机械臂移动停止。
- 清除未在所有路径等级（包括普通和StorePath 等级）下实施的所有机械臂路径。
- 中断在所有执行等级（包括普通和TRAP等级）下开始但并未完成的指令。
- 将程序指针移动到主程序中的第一个指令。
- 继续程序执行，以执行下一循环。

在一些其他的程序任务而非控制机械单元中执行 ExitCycle，会在实际任务中产生：

- 中断对所有执行等级（包括普通和TRAP等级）开始但并未完成的指令。
- 将程序指针移动到主程序中的第一个指令。
- 继续程序执行，以执行下一循环。

程序和系统中的所有其他模式事宜未受ExitCycle影响，例如：

- 变量或永久数据对象的实际值。
- 所有运动设置，例如， StorePath-RestoPath 顺序、全局区域等等。

下一页继续

1.86 ExitCycle - 中断当前循环，并开始下一循环

RobotWare - OS

续前页

- 打开文件、路径等。
- 定义中断等。

当使用程序调用中的ExitCycle，且通过“移动PP到程序...”或“调用程序...”来定义入口程序时，ExitCycle中断当前的循环，并将程序指针移动回入口程序中的第一个指令（而非先前指定的主程序）。

语法

```
ExitCycle';'
```

相关信息

信息，关于	请参阅
在出现致命错误后停止	第*202*页的*EXIT - 终止程序执行*
终止程序执行	第*202*页的*EXIT - 终止程序执行*
停止程序动作	第*685*页的*Stop - 停止程序执行*
完成程序的执行	第*515*页的*RETURN - 完成程序的执行*

1.87 FOR - 重复给定的次数

手册用法

当一个或多个指令重复多次时，使用FOR。

基本示例

以下实例介绍了指令FOR：

另请参阅第*205*页的更多示例

例 1

```
FOR i FROM 1 TO 10 DO
  routine1;
ENDFOR
```

重复routine1无返回值程序10次。

变元

```
FOR Loop counter FROM Start value TO End value [STEP Step value]
     DO ... ENDFOR
```

Loop counter

Identifier

将包含当前循环计数器数值的数据名称。自动声明该数据。

如果循环计数器名称与实际范围中存在的任意数据相同，则将现有数据隐藏在FOR循环中，且在任何情况下均不受影响。

Start value

数据类型：Num

循环计数器的期望起始值（通常为整数值）。

End value

数据类型：Num

循环计数器的期望结束值（通常为整数值）。

Step value

数据类型：Num

循环计数器在各循环的增量（或减量）值（通常为整数值）。

如果未指定该值，则自动将步进值设置为1（或者如果起始值大于结束值，则设置为-1）。

更多示例

有关于如何使用指令FOR的更多例子阐述如下。

例 1

```
FOR i FROM 10 TO 2 STEP -2 DO
  a{i} := a{i-1};
ENDFOR
```

将数组中的数值向上调整，以便a{10}:=a{9}、 a{8}:=a{7}等等。

下一页继续

1.87 FOR - 重复给定的次数

RobotWare - OS

续前页

程序执行

1 评估起始值、结束值和步进值的表达式。

2 向循环计数器分配起始值。

3 检查循环计数器的数值，以查看其数值是否介于起始值和结束值之间，或者是否等于起始值或结束值。如果循环计数器的数值在此范围之外，则FOR循环停止，且程序继续执行紧接ENDFOR的指令。

4 执行FOR循环中的指令。

5 按照步进值，使循环计数器增量（或减量）。

6 重复FOR循环，从点3开始。

限制

仅可在FOR循环内评估循环计数器（数据类型为num），随之隐藏其他具有相同名称的数据和路径。其仅可通过FOR循环中的指令来进行读取（未更新）。

无法使用起始值、结束值或停止值的小数值，以及FOR循环的确切终止条件（不确定最后的循环是否在运行中）。

备注

如果重复次数有待继续，只要评估给定表达式的TRUE值，则应当转而使用WHILE指令。

语法

```
FOR <loop variable> FROM <expression> TO <expression>
  [ STEP <expression> ] DO
  <statement list>
ENDFOR
```

相关信息

信息，关于	请参阅
表达式	技术参考手册 - *RAPID*语言概览
只要...便重复	第*907*页的*WHILE - 只要...便重复*
标识符	技术参考手册 - *RAPID*语言概览

1.88 FricIdInit - 开始摩擦识别

手册用法

FricIdInit标记一系列移动指令的起点，重复此类移动指令，以计算机械臂内摩擦。

示例

关于使用圆周运动以计算该运动的机械臂内部摩擦的基本实例：

```
PERS num friction_levels{6};

! Start of the friction calculation sequence
FricIdInit;

! Execute the move sequence
MoveC p10, p20, Speed, z0, Tool;
MoveC p30, p40, Speed, z0, Tool;

! Repeat the sequence and calculate the friction
FricIdEvaluate friction_levels;

! Activate compensation for the calculated friction levels
FricIdSetFricLevels friction_levels;
```

操作前提

必须将系统参数*Friction FFW On*设置为TRUE。否则，指令FricIdInit将无效。

限制

- FricIdInit仅适用于TCP机械臂。
- 仅可根据运动任务来执行。
- 机械臂必须在基础路径等级上运动。
- 摩擦微调无法与同步移动组合起来，即是说不允许在FricIdInit与FricIdEvaluate之间实现SyncMoveOn。
- 针对所作摩擦微调的移动序列必须从一个精确点开始，到一个精确点结束。若非如此，系统则会在微调期间自动插入精确点。

语法

```
FricIdInit ';'
```

相关信息

信息，关于	请参阅
Advanced robot motion	应用手册 - 控制器软件*IRC5*

1.89 FricIdEvaluate - 评估摩擦识别

手册用法

当计算机械臂各个轴的摩擦时，FricIdEvaluate使机械臂重复指令FricIdInit和FricIdEvaluate之间的运动。

示例

关于使用圆周运动以计算该运动的机械臂内部摩擦的基本实例：

```
PERS num friction_levels{6};

! Start of the friction calculation sequence
FricIdInit;

! Execute the move sequence
MoveC p10, p20, Speed, z0, Tool;
MoveC p30, p40, Speed, z0, Tool;

! Repeat the sequence and calculate the friction
FricIdEvaluate friction_levels;

! Activate compensation for the calculated friction levels
FricIdSetFricLevels friction_levels;
```

变元

```
FricIdEvaluate FricLevels [\MechUnit] [\BwdSpeed] [\NoPrint]
     [\FricLevelMax] [\FricLevelMin] [\OptTolerance]
```

FricLevels

Friction levels

数据类型：array of num

当完成FricIdEvaluate时，数组FricLevels将包含适用于机械臂所有轴的调整摩擦等级。必须声明，该数组拥有的元素数与机械臂所拥有的轴的数量相同。注意，必须调用指令FricIdSetFricLevels，从而使此类值生效。

[\MechUnit]

Mechanical unit

数据类型：mecunit

参数MechUnit是可选择的项。如果省略参数，则将调整预定义RAPID变量ROB_ID所代表的机械单元的摩擦力，其可作为当前程序任务中TCP机械臂的引用。摩擦补偿仅适用于TCP机械臂。

[\BwdSpeed]

Backward speed

数据类型： speeddata

在调整过程中每次迭代后，机械臂沿编程路径向后运动。默认以编程速度向后运动。向后运动期间，为加快进程，可选参数BwdSpeed可用于指定更高的速度。这不会影响调整结果。

下一页继续

[\NoPrint]

数据类型：switch

如果使用参数NoPrint，则未在FlexPendant示教器上写入关于摩擦识别迭代进程的文本。

[\FricLevelMax]

Friction level max

数据类型：num

通常，通过尝试配置摩擦值1%到500%之间的数值，发现最佳摩擦值。在极少情况下，可能会产生错误消息（接头速度错误）。为避免该情形，使用参数FricLevelMax，并将其值设置在500以下。例如，如果将FricLevelMax设置为400，则测试介于1%到400%之间的数值。

容许的数值为101-500。

[\FricLevelMin]

Friction level min

数据类型：num

通常，通过尝试配置摩擦值1%到500%之间的数值，发现最佳摩擦值。使用参数FricLevelMin，设置高于1%的起始值。例如，如果将FricLevelMin设置为80，则测试介于80%到500%之间的数值。

容许的数值为1-99。

[\OptTolerance]

Optimization tolerance

数据类型：num

通常，通过尝试不同的数值，直至实现较小的公差，以此寻找最佳摩擦值。为加快进程，可增加该数值。该数值的增加可能会降低结果的准确性。

容许的数值为1-10。默认值为1。

错误处理

可能会产生下列可恢复错误。错误可以由错误处理程序处理。系统变量 ERRNO 将设置为：

名称	错误原因
ERR_FRICTUNE_FATAL	摩擦调整期间，出现错误。

注意

如果摩擦调整不存在任何影响，则检查系统参数*Friction FFW On*是否设置为TRUE。

操作前提

必须将系统参数*Friction FFW On*设置为TRUE。否则，指令FricIdEvaluate将无效。

限制

- FricIdEvaluate仅适用于TCP机械臂。

下一页继续

1.89 FricIdEvaluate - 评估摩擦识别

RobotWare - OS

续前页

- 仅可根据运动任务来执行FricIdEvaluate。
- 机械臂必须在基础路径等级上运动。
- 针对MultiMove系统，每次仅可针对一个机械臂进行摩擦调整。若干机械臂可以同时执行FricIdEvaluate，但是，只要另一个机械臂忙于进行摩擦调整，则机械臂将自动静止，并等待其转向。
- 摩擦微调无法与同步移动组合起来，即是说不允许在FricIdInit与FricIdEvaluate之间实现SyncMoveOn。
- 针对所作摩擦微调的移动序列必须从一个精确点开始，到一个精确点结束。若非如此，系统则会在微调期间自动插入精确点。

语法

```
FricIdEvaluate
  [ FricLevels ':='] < persistent array {*} (PERS) of num >
  ['\' MechUnit ':=' < variable (VAR) of mecunit >]
  ['\' BwdSpeed ':=' < expression (IN) of speeddata >]
  ['\' NoPrint]
  ['\' FricLevelMax ':=' < expression (VAR) of num >]
  ['\' FricLevelMin ':=' < expression (VAR) of num >]
  ['\' OptTolerance ':=' < expression (VAR) of num >] ';'
```

相关信息

信息，关于	请参阅
Advanced robot motion	应用手册 - 控制器软件*IRC5*

1.90 FricIdSetFricLevels - 在摩擦识别后设置摩擦等级

手册用法

FricIdSetFricLevels用于设置机械单元各个轴的摩擦等级。

示例

关于使用圆周运动以计算该运动的机械臂内部摩擦的基本实例：

```
PERS num friction_levels{6};

! Start of the friction calculation sequence
FricIdInit;

! Execute the move sequence
MoveC p10, p20, Speed, z0, Tool;
MoveC p30, p40, Speed, z0, Tool;

! Repeat the sequence and calculate the friction
FricIdEvaluate friction_levels;

! Activate compensation for the calculated friction levels
FricIdSetFricLevels friction_levels;
```

变元

```
FricIdSetFricLevels FricLevels [\MechUnit]
```

FricLevels

Friction levels

数据类型：array of num

数组FricLevels以默认摩擦百分比的形式，规定各个轴的摩擦等级。数值必须保持0-500的间隔。

[\MechUnit]

Mechanical unit

数据类型：mecunit

参数MechUnit是可选择的项。如果省略参数，则将设置预定义RAPID变量ROB_ID所代表的机械单元的摩擦等级。摩擦补偿仅适用于TCP机械臂。

程序执行

摩擦等级的设置将保持有效，直至：

- 从开始进行程序执行（PP到Main）
- 对FricIdSetFricLevels进行另一次调用。
- 加载新的程序
- 使用重启模式重置系统，重启控制器。

操作前提

必须将系统参数*Friction FFW On*设置为TRUE。否则，指令FricIdSetFricLevels将无效。

下一页继续

1.90 FricIdSetFricLevels - 在摩擦识别后设置摩擦等级

RobotWare - OS

续前页

限制

- FricIdSetFricLevels仅适用于TCP机械臂。

语法

```
FricIdSetFricLevels
  [ FricLevels ':='] < array {*} (IN) of num >
  ['\' MechUnit ':=' < variable (VAR) of mecunit >] ';'
```

相关信息

信息，关于	请参阅
Advanced robot motion	应用手册 - 控制器软件*IRC5*

1.91 GetDataVal - 获得数据对象的值

手册用法

GetDataVal（*Get Data Value*）使其有可能从通过字符串变量而确定的数据对象中获得一个值。

基本示例

以下实例介绍了指令GetDataVal：

例 1

```
VAR num value;
...
GetDataVal "reg"+ValToStr(ReadNum(mycom)),value;
```

由此可获得寄存器的数值，并且可从串行通道mycom获得上述寄存器的编号。将数值储存在变量value中。

例 2

```
VAR datapos block;
VAR string name;
VAR num valuevar;
...
SetDataSearch "num" \Object:="my.*" \InMod:="mymod";
WHILE GetNextSym(name,block) DO
  GetDataVal name\Block:=block,valuevar;
  TPWrite name+" "\Num:=valuevar;
ENDWHILE
```

此会话将打印出始于模块mymod中my的所有num变量，并将在FlexPendant示教器上显示出其数值。

例 3

```
VAR num NumArrConst_copy{2};
...
GetDataVal "NumArrConst", NumArrConst_copy;
TPWrite "Pos1 = " \Num:=NumArrConst_copy{1};
TPWrite "Pos2 = " \Num:=NumArrConst_copy{2};
```

该会话将在数组NumArrConst.中打印出num变量。

变元

```
GetDataVal Object [\Block]|[\TaskRef]|[\TaskName] Value
```

Object

数据类型：string

数据对象的名称。

[\Block]

数据类型：datapos

数据对象的封闭块。仅可通过GetNextSym功能来取得该封闭块。

如果省略该参数，则将获得当前程序执行范围中可见数据对象的值。

下一页继续

1.91 GetDataVal - 获得数据对象的值

RobotWare - OS

续前页

[\TaskRef]

Task Reference

数据类型：taskid

用以搜索指定数据对象的程序任务识别号。使用此参数时，可搜索其他任务中的PERS或TASKPERS声明，任何其他声明均将引起错误。

对于系统中的所有程序任务，数据类型taskid的预定义变量将有效。可变识别号将为“任务名”+“Id”，例如，针对T_ROB1任务，可变识别号将为T_ROB1Id。

[\TaskName]

数据类型：string

用以搜索指定数据对象的程序任务名称。使用此参数时，可搜索其他任务中的PERS或TASKPERS声明，任何其他声明均将引起错误。

Value

数据类型：anytype

关于储存获取数值的变量。数据类型必须与所发现数据对象的数据类型相同。可从常量、变量或永久数据对象中得出获取值，但是必须将其储存在变量中。

错误处理

可能会产生下列可恢复错误。错误可以由错误处理程序处理。系统变量ERRNO将设置为：

名称	错误原因
ERR_SYM_ACCESS	• 不存在数据对象。 • 数据对象是程序数据或程序参数，且不得位于当前有效程序中。 • 在其他任务中搜索其他声明，然后搜索PERS或TASK PERS。
ERR_INVDIM	在参数Value中使用的数据对象和变量具有不同的维度。
ERR_SYMBOL_TYPE	在参数Value中使用的数据对象和变量具有不同的类型。如果使用ALIAS数据类型，则亦将出现此错误，尽管上述类型可能具有相同的基础数据类型。

当使用参数TaskRef或TaskName时，可搜索其他任务中的PERS或TASK PERS声明，任何其他声明均将引起错误，且系统变量ERRNO得以设置为ERR_SYM_ACCESS。搜索在其他任务中被声明为LOCAL的PERS，亦将引起错误，且系统变量ERRNO得以设置为ERR_SYM_ACCESS。

限制

关于半值数据类型，不可能搜索相关的值数据类型。例如，如果搜索dionum，则将不会获得关于信号signaldi的搜索匹配，且如果搜索num，则将不会获得关于信号signalgi或signalai的搜索匹配。

不可能获得在RAPID模式下建立、声明为LOCAL的变量值。

语法

```
GetDataVal
  [ Object ':=' ] < expression (IN) of string >
  ['\'Block' :='<variable (VAR) of datapos>]
```

下一页继续

```
|[ ’\’TaskRef’ :=’ <variable (VAR) of taskid>]
|[ ’\’TaskName’ :=’ <expression (IN) of string>] ’,’]
[ Value ’:=’ ] <variable (VAR) of anytype>]’;’
```

相关信息

信息，关于	请参阅
定义在搜索会话中设置的符号	第*582*页的*SetDataSearch* - 定义在搜索序列中设置的符号
获取下一个匹配的符号	第*1095*页的*GetNextSym* - 获取下一个匹配的符号
设置数据对象的值	第*586*页的*SetDataVal* - 设置数据对象的值
设置许多数据对象的值	第*578*页的*SetAllDataVal* - 在定义设置下，设置所有数据对象的值
相关的数据类型datapos	第*1382*页的*datapos* - 数据类型的封闭块
Advanced RAPID	应用手册 - 控制器软件*IRC5*

1.92 GetSysData - 获取系统数据

手册用法

GetSysData获取指定数据类型的当前系统数据的数值和可选符号名称。

使用该指令，可能获得在实际或相关运动任务或任何名称的运动任务中，关于机械臂的当前有效工具、工件、有效负载或总负载的数据和名称。

基本示例

以下实例介绍了指令GetSysData：

例 1

```
PERS tooldata curtoolvalue := [TRUE, [[0, 0, 0], [1, 0, 0, 0]],
     [2, [0, 0, 2], [1, 0, 0, 0], 0, 0, 0]];
VAR string curtoolname;
GetSysData curtoolvalue;
```

将当前有效工具数据复制到永久变量curtoolvalue。

例 2

```
GetSysData curtoolvalue \ObjectName := curtoolname;
```

同时将当前有效工具名称复制到变量curtoolname。

例 3

```
PERS loaddata curload;
PERS loaddata piece:=[2.8,[-38.2,-10.1,-73.6],[1,0,0,0],0,0,0];
PERS loaddata
     tool2piece:=[13.1,[104.5,13.5,115.9],[1,0,0,0],0,0,0.143];
PERS tooldata tool2 := [TRUE, [[138.695,150.023,98.9783],
     [0.709396,-0.704707,-0.00856676,0.00851007]],
     [10,[105.2,-3.8,118.7], [1,0,0,0],0,0,0.123]];
VAR string name;
..
IF GetModalPayloadMode() = 1 THEN
  GripLoad piece;
  MoveL p3, v1000, fine, tool2;
  ..
  ..
  ! Get current payload
  GetSysData curload \ObjectName := name;
ELSE
  MoveL p30, v1000, fine, tool2\TLoad:=tool2piece;
  ..
  ..
  ! Get current total load
  GetSysData curload \ObjectName := name;
ENDIF
```

如果ModalPayLoadMode为1，则将当前的有效负载和名称复制到变量name。

如果ModalPayLoadMode为0，则将当前的总负载和名称复制到变量name。

变元

```
GetSysData [\TaskRef]|[\TaskName] DestObject[\ObjectName ]
```

下一页继续

[\TaskRef]

Task Reference

数据类型：taskid

用以读取当前有效系统数据的程序任务识别号。

对于系统中的所有程序任务，数据类型taskid的预定义变量将有效。可变识别号将为“任务名”+“Id”，例如，针对T_ROB1任务，可变识别号将为T_ROB1Id。

[\TaskName]

数据类型：string

用以读取当前有效系统数据的程序任务名称。

如果未指定自变量\TaskRef或\TaskName，则使用当前任务。

DestObject

数据类型： anytype

用于储存当前有效系统数据值的永久变量。

本参数的数据类型同时指定了待获取系统数据的类型（工具、工件或有效负载/总负载）。如果使用关于运动指令的TLoad可选参数，则获取总负载，如果使用加载数据数据类型，则获取有效负载。

数据类型	系统数据的类型
tooldata	工具
wobjdata	工件坐标
loaddata	有效负载/总负载

无法使用数组或记录分量。

[\ObjectName]

数据类型：string

选项参数（变量或永久数据对象）同时获取当前有效系统的数据名称。

程序执行

当运行指令GetSysData时，将当前数据值储存在参数DestObject中的指定永久变量。

如果使用参数\ObjectName，则将当前数据的名称储存在参数ObjectName中的指定变量或永久数据对象中。

通过执行任意运动指令，启用关于工具、工件或总负载的当前系统数据。通过执行指令GripLoad，启用有效负载。

下一页继续

1.92 GetSysData - 获取系统数据

RobotWare - OS

续前页

错误处理

可能会产生下列可恢复错误。错误可以由错误处理程序处理。系统变量 ERRNO 将设置为：

名称	错误原因
ERR_NOT_MOVETASK	参数\TaskRef或\TaskName指定了非运动任务。 **注意** 如果参数\TaskRef或\TaskName指定可执行功能 GetSysData（自身非运动任务的引用）的非运动任务，则将不会产生错误。随后，将从相关的运动任务中获取当前系统数据。

语法

```
GetSysData
  ['\' TaskRef ':=' <variable (VAR) of taskid>]
  |['\' TaskName' :=' <expression (IN) of string>]
  [ DestObject' :='] < persistent(PERS) of anytype>
  ['\'ObjectName' :=' < variable or persistent (INOUT) of string>
       ] ';'
```

相关信息

信息，关于	请参阅
工具的定义	第*1502*页的*tooldata - 工具数据*
工件的定义	第*1523*页的*wobjdata - 工件数据*
有效负载的定义	第*1421*页的*loaddata - 加载数据*
设置系统数据	第*594*页的*SetSysData - 设置系统数据*
激活和停用载荷的系统参数 *ModalPayLoadMode*。 （主题 Controller，类型 System Misc，行动值，*ModalPayLoadMode*）	技术参考手册 - 系统参数
关于如何使用TLoad总负载的例子。	第*388*页的*MoveL - 使机械臂沿直线移动*

1.93 GetTrapData - 获取当前TRAP的中断数据

手册用法

GetTrapData用于软中断程序，以获取导致软中断程序被执行的中断的所有信息。

在使用指令ReadErrData之前，于指令IError所生成的软中断程序中使用。

基本示例

以下实例介绍了指令GetTrapData：

另请参阅第*219*页的更多示例

例 1

```
VAR trapdata err_data;
GetTrapData err_data;
```

将中断信息储存在非值变量err_data中。

变元

```
GetTrapData TrapEvent
```

TrapEvent

数据类型：trapdata

关于储存引起待执行软中断的相关信息的变量。

限制

本指令仅可用于TRAP程序。

更多示例

有关于指令GetTrapData的更多例子阐述如下。

例 1

```
VAR errdomain err_domain;
VAR num err_number;
VAR errtype err_type;
VAR trapdata err_data;
...
TRAP trap_err
  GetTrapData err_data;
  ReadErrData err_data, err_domain, err_number, err_type;
ENDTRAP
```

当错误被困于软中断程序trap_err时，将错误域、错误编号和错误类型保存在适当的trapdata型非值变量中。

语法

```
GetTrapData
  [TrapEvent ':='] <variable (VAR) of trapdata>';'
```

相关信息

信息，关于	请参阅
中断概述	技术参考手册 - *RAPID*语言概览

下一页继续

1.93 GetTrapData - 获取当前TRAP的中断数据

RobotWare - OS

续前页

信息，关于	请参阅
更多关于中断管理的信息	技术参考手册 - *RAPID*语言概览
当前TRAP的中断数据	第*1509*页的*trapdata - 当前TRAP的中断数据*
调整关于错误的中断	第*234*页的*IError - 调整关于错误的中断*
获取关于错误的信息	第*496*页的*ReadErrData - 获取关于错误的信息*
Advanced RAPID	应用手册 - 控制器软件*IRC5*

1.94 GOTO - 转到新的指令

手册用法

GOTO用于将程序执行转移到相同程序内的另一线程（标签）。

基本示例

以下实例介绍了指令GOTO：

例 1

```
GOTO next;
...
next:
```

通过以下指令，继续程序执行。

例 2

```
reg1 := 1;
next:
...
reg1 := reg1 + 1;
IF reg1<=5 GOTO next;
```

将执行转移至next四次（reg1= 2、3、4、5）。

例 3

```
IF reg1>100 THEN
  GOTO highvalue
ELSE
  GOTO lowvalue
ENDIF
lowvalue:
...
GOTO ready;
highvalue:
...
ready:
```

如果reg1 大于100，则将执行转移至标签 highvalue，否则，将执行转移至标签 lowvalue。

变元

```
GOTO Label
```

Label

Identifier

继续程序执行的标签。

限制

仅可能将程序执行转移到相同程序内的一个标签。

如果GOTO指令亦位于该指令的相同分支内，则仅可能在IF或TEST指令内，将程序执行转移至标签。

下一页继续

1.94 GOTO - 转到新的指令

RobotWare - OS

续前页

如果GOTO指令亦位于该指令内，则仅可能在FOR或WHILE指令内，将程序执行转移至标签。

语法

```
GOTO <identifier>';'
```

相关信息

信息，关于	请参阅
标签	第*311*页的*Label - 线程名称*
改变程序流的其他指令	技术参考手册 - *RAPID*语言概览

1.95 GripLoad - 定义机械臂的有效负载

手册用法

GripLoad用于定义机械手固定机械臂的有效负载。

描述

GripLoad指定机械臂承载的负载。指定的负载用于设置机械臂的动态模型，以便可以以最佳的方式来控制机械臂运动。

使用指令GripLoad，连接/断开有效负载，该指令在机械手的重量上增加或减去有效负载的重量。

警告

重要的是，始终定义实际工具负载以及使用时机械臂（例如，抓取部分）的有效负载。负载数据定义不正确可能会导致机械臂机械结构过载。

指定不正确的负载数据时，其常常会引起以下后果：

- 机械臂将不会用于其最大容量
- 路径准确性受损，包括过度风险
- 机械结构过载风险

基本示例

有关于指令GripLoad的例子阐述如下。

例 1

```
Set gripper;
WaitTime 0.3;
GripLoad piece1;
```

在机械臂抓握负载的同时，指定有效负载的连接，piece1。

例 2

```
Reset gripper;
WaitTime 0.3;
GripLoad load0;
```

在机械臂释放有效负载的同时，规定断开有效负载。

变元

```
GripLoad Load
```

Load

数据类型：loaddata

描述当前有效负载的负载数据。

可能通过使用与系统输入SimMode（模拟模式）相连的数字信号输入来试运行程序，且有效负载为零。如果将数字信号输入设置为1，则不考虑GripLoad指令中的loaddata，且仅使用当前tooldata中的loaddata。

程序执行

指定的负载影响机械臂的性能。

下一页继续

1.95 GripLoad - 定义机械臂的有效负载
RobotWare - OS
续前页

自动设置默认负载（load0）为0 kg。

- 当使用重启模式重置RAPID时
- 当加载一则新程序或一个新模块时
- 当从起点开始执行程序时
- 当将程序指针移动到main时
- 当将程序指针移动到子程序时
- 移动程序指针造成执行顺序丢失时

更新由当前程序任务控制的机械单元的有效负载。如果非运动任务使用GripLoad，则更新由相关运动任务控制的机械单元的有效负载。

语法

```
GripLoad
  [ Load ':=' ] < persistent (PERS) of loaddata > ';'
```

相关信息

信息，关于	请参阅
工具负载、有效负载或手臂负载的负载识别	操作员手册 - 带 *FlexPendant* 的 *IRC5*，编程与测试-服务程序一节
确定机械单元的有效负载	第*327*页的*MechUnitLoad - 确定机械单元的有效负载*
负载数据的定义	第*1421*页的*loaddata - 加载数据*
用于在仿真模式下无载荷运行机器人的系统输入信号 *SimMode*。 （主题 I/O，类型 System Input，行动值，*SimMode*）	技术参考手册 - 系统参数

1.96 HollowWristReset - 重置IRB5402和IRB5403的中空腕

手册用法

HollowWristReset（*Reset hollow wris*t）重置中空腕机械臂（例如，IRB5402和IRB5403）上腕接头的位置。

在腕接头4和5缠绕一转或多转后，指令使其避免重新缠绕腕接头4和5。在执行HollowWristReset指令后，腕接头可能继续朝相同的方向缠绕。

描述

HollowWristReset使其更易于应用程序。编程时，无须确保腕位置位于±2转以内，且可以节约循环时间，因为机械臂无须花时间重新缠绕腕。在HollowWristReset重置腕位置之前，接头4和5的缠绕限制是±144转。在计划机械臂程序时，机械臂程序员必须意识到此限制且将其纳入考虑。为确保在运行“腕缠绕”程序若干次后，不超过144转的限值，应当始终让机械臂完全停止，并重置在各个程序（或必要时在循环/程序/模块等）中的绝对位置。注意，在执行HollowWristReset指令期间，所有轴均必须保持停止。在程序执行期间，只要考虑此类限制，则可以独立于接头6而无限制地缠绕接头4和5。

使用HollowWristReset而非IndReset来重置中空腕，因为该指令可保持接头6的接头限值，以防止涂料管/电缆过度缠绕。

基本示例

以下实例介绍了指令HollowWristReset：

例 1

```
MoveL p10,v800,fine,paintgun1\WObj:=workobject1;
HollowWristReset;
```

通过停止点来停止所有有效轴，并重置腕。

限制

执行HollowWristReset指令时，必须停止所有有效轴。

在任意腕接头达到±144转限值（51840度/904拉德）之前，必须重置腕接头。

出现程序停止、紧急停止、电源故障停止等时，控制器维持路径环境，以便能够返回路径，并使机械臂从路径停止点继续程序执行。在手动模式下，如果已经将机械臂移出停止和重启之间的路径，则通过FlexPendant示教器上的以下消息来通知操纵员：“未处于路径上！已经在程序停止后移动机械臂。机械臂是否应当返回起始路径？是/否/取消”。这提供了在重启之前返回路径的机会。在自动模式下，机械臂自动地返回路径。

HollowWristReset清除路径环境。这意味着，如果同时执行HollowWristReset指令，在程序重启时，将无法返回路径。如果手动执行该指令（程序编辑器中的“调试+调用程序...”），则应当仅在无需返回路径时执行。也就是说，在以逐步执行的方式彻底完成程序或指令，以及通过点动将机械臂移出路径之后。

语法

```
HollowWristReset ´;’
```

下一页继续

1.96 HollowWristReset - 重置IRB5402和IRB5403的中空腕
RobotWare - OS
续前页

相关信息

信息，关于	请参阅
相关的系统参数	技术参考手册 - 系统参数
返回路径	技术参考手册 - *RAPID*语言概览

1.97 ICap - 将CAP事件与软中断子程序关联起来

手册用法

ICap将用于把中断编号（已与软中断子程序关联起来）与特定CAP事件相关联，有关可发生事件的清单，请参见下文参数部分。当采用ICap时，将在特定进程事件与用户定义软中断子程序之间建立关联。换言之，当发生相关CAP事件时，相关软中断子程序将被执行。

我们建议将软中断放置到背景任务中。

基本示例

下文给出了CAP事件CAP_START与软中断子程序start_trap相关联的示例。

```
VAR intnum start_intno:=0;
...

TRAP start_trap
  ! This routine will be executed when the event CAP_START is
       reported from the core
  ! Do what you want to do
ENDTRAP

PROC main()
  IDelete start_intno;
  CONNECT start_intno WITH start_trap;
  ICap start_intno, CAP_START;
  CapL p1, v100, cdata, weavestart, weave, z50, gun1;
ENDPROC
```

变元

```
ICap Interrupt Event
```

Interrupt

数据类型：intnum

中断识别号。通过指令CONNECT，先前已经同软中断程序相连。

Event

数据类型：num

与中断关联起来的CAP事件编号。这些事件为预定义常量。

可发生的CAP事件

参见应用手册 - *Continuous Application Platform*中的阶段与事件之间关联章节。

事件	阶段	事件编号	描述
CAP_START		0	在CAP进程启动后，即刻会发生此事件。如果采用*flying start*，那么，不会分配任何事件，因为已经发生TCP移动。重启时，将分配此事件。
START_PRE	PRE	1	如果设置了监控PRE阶段的事件，那么就会激活该事件。如果采用*flying start*，那么，不会分配任何事件，因为已经发生TCP移动。重启时，将分配此事件。

下一页继续

1.97 ICap - 将CAP事件与软中断子程序关联起来
Continuous Application Platform (CAP)
续前页

事件	阶段	事件编号	描述
PRE_STARTED	PRE	2	当满足PRE监控列表的所有要求时，即，当启动PRE_START阶段时，会发生此事件。如果采用*flying start*，那么，不会分配任何事件，因为已经发生TCP移动。重启时，将分配此事件。
START_MAIN	START	3	当PRE_START阶段结束以及MAIN阶段启动时，会发生此事件。
MAIN_STARTED	START	4	当满足START监控列表的所有条件时，即，当MAIN阶段启动时，将发生此事件。
STOP_WEAVESTART	MAIN	5	在各摆动启动前，会发生此事件，但只有在命令摆动启动的情况下才会发生。如果采用*flying start*，那么，不会分配任何事件，因为已经发生TCP移动。重启时，将分配此事件。
WEAVESTART_REGAIN	MAIN	6	摆动启动后，当机器人恢复到路径上时，会发生此事件。如果采用*flying start*，那么，不会分配任何事件，因为已经发生TCP移动。重启时，将分配此事件。
MOTION_DELAY	MAIN	7	在运动启动延迟（如存在）后，会发生此事件。如果采用*flying start*，那么，不会分配任何事件，因为已经发生TCP移动。重启时，将分配此事件。
STARTSPEED_TIME	MAIN	8	当启动速度所用时间用完时，会发生此事件，这正是切换到主运动数据之时。
MAIN_MOTION	MAIN	9	当在进程运行的情况下启动主运动时，会发生此事件。
MOVE_STARTED	MAIN	10	在机器人开始沿进程路径移动后，即刻会发生此事件。如果采用*flying start*，那么，不会分配任何事件，因为已经发生TCP移动。重启时，将分配此事件。
RESTART	MAIN	11	当命令重启时，会发生此事件。
NEW_INSTR	MAIN	12	当从RAPID程序取得新CapL或CapC指令时，会发生此事件。
AT_POINT	MAIN	13	在进程路径中的各机器人位置处，会发生此事件，但起点和终点除外。
AT_RESTARTPOINT	MAIN	14	在停止后，当机器人在进程路径上点动回来（重启距离）时，会发生此事件。
LAST_SEGMENT	MAIN	15	在最后一个程序段的起点，将发生此事件。
PROCESS_END_POINT	MAIN	16	当机器人到达进程终点时（即，假设进程结束所在的点），会发生此事件。如果采用*flying end*，那么不会分配任何事件。
END_MAIN	END_MAIN	17	在开始结束序列监控时所处的进程路径点处，即，当机器人到达进程终点时，会发生此事件。
MAIN_ENDED	END_MAIN	18	当满足END_MAIN监控列表的所有条件时，即，当主进程被视为结束时，会发生此事件。
PATH_END_POINT		19	当机器人到达路径终点时（即，最后一个CAP指令的区域精准点或中间（对于*flying end*）），会发生此事件。
PROCESS_ENDED		20	在最后一个CAP指令的区域精准点或中间（对于*flying end*）处，两个进程均结束时，会发生此事件。

下一页继续

事件	阶段	事件编号	描述
END_POST1	END_POST1	21	到了结束POST1阶段之时，即，到了从POST1转换到POST2阶段之时，会发生此事件。如果采用*flying end*，那么不会分配任何事件。
POST1_ENDED	END_POST1	22	当满足END_POST1监控列表的所有条件时，即，当成功结束POST1阶段并启动POST2阶段时，会发生此事件。如果采用*flying end*，那么不会分配任何事件。
END_POST2	END_POST2	23	POST2阶段结束时，即，到了最终完成进程之时，会发生此事件。如果采用*flying end*，那么不会分配任何事件。
POST2_ENDED	END_POST2	24	当满足END_POST2监控列表的所有条件时，即，当成功结束POST2阶段从而结束整个进程时，会发生此事件。如果采用*flying end*，那么不会分配任何事件。
CAP_STOP		25	此事件为一项必要事件。如果采用任何其他事件，那么也必须定义此事件。在因错误或程序停止，控制柜停止工作后，将即刻执行此事件/软中断。错误可为CAP中检测到的可恢复错误、CAP中检测到的致命错误或让控制柜停止的内部错误。此软中断中执行的代码应让所有外部设备处于安全状态，比如，将重置所有外部I/O信号。
CAP_PF_RESTART	MAIN	26	当命令重启时，会发生此事件。
EQUIDIST	MAIN	27	如果以`CapEquiDist`指令发出命令，那么将发送此事件。
AT_ERRORPOINT	MAIN	28	在重启后，当TCP到达监控错误的位置时，会发生此事件。
FLY_START	MAIN	29	采用*flying start*时，会发生此事件。只有*flying start*才会发生此事件。
FLY_END	MAIN	30	采用*flying end*时，会发生此事件。只有*flying end*才会发生此事件。
LAST_INSTR_ENDED	MAIN	31	在*flying end*过程中，当最后一个CAP指令的RAPID执行完成时，会发生此事件。只有*flying end*才会发生此事件。
END_PRE	PRE	32	如果设置了监控PRE阶段的事件，那么就会激活该事件。如果采用*flying start*，那么，不会分配任何事件，因为已经发生TCP移动。重启时，将分配此事件。
PRE_ENDED	PRE	33	如果设置了监控PRE阶段的事件，那么就会激活该事件。如果采用*flying start*，那么，不会分配任何事件，因为已经发生TCP移动。重启时，将分配此事件。
START_POST1	POST1	34	如果设置了监控POST1阶段的事件，那么就会激活此事件。如果采用*flying start*，那么，不会分配任何事件，因为已经发生TCP移动。重启时，将分配此事件。
POST1_STARTED	POST1	35	如果设置了监控POST1阶段的事件，那么就会激活此事件。如果采用*flying start*，那么，不会分配任何事件，因为已经发生TCP移动。重启时，将分配此事件。

下一页继续

1.97 ICap - 将CAP事件与软中断子程序关联起来

Continuous Application Platform (CAP)

续前页

事件	阶段	事件编号	描述
START_POST2	POST2	36	如果设置了监控POST1阶段的事件，那么就会激活此事件。如果采用*flying start*，那么，不会分配任何事件，因为已经发生TCP移动。重启时，将分配此事件。
POST2_STARTED	POST2	37	如果设置了监控POST1阶段的事件，那么就会激活此事件。如果采用*flying start*，那么，不会分配任何事件，因为已经发生TCP移动。重启时，将分配此事件。

限制

在未首先删除同一中断标识变量的情况下，该中断标识变量不可使用一次以上。因此，应按下列方案之一的规定来处理中断。

```
PROC setup_events ()
  VAR intnum start_intno;
  IDelete start_intno;
  CONNECT start_intno WITH start_trap;
  ICap start_intno, CAP_START;
ENDPROC
```

在程序开始时，完成所有中断启动。然后，将这些指令保持在主程序流之外。ICap指令只应被执行一次，比如从启动系统事件子程序执行。建议将软中断放置在背景任务中。

语法

```
ICap
  [Interrupt ':='] < variable (IN) of intnum > ','
  [Event ':='] < variable (IN) of num > ';'
```

相关信息

信息，关于	请参阅
Continuous Application Platform	应用手册 - *Continuous Application Platform*
将中断与软中断关联起来	第*124*页的*CONNECT - 将中断与软中断程序相连*
取消与软中断关联的中断	第*231*页的*IDelete - 取消中断*
数据类型intnum	第*1414*页的*intnum - 中断识别号*

1.98 IDelete - 取消中断

手册用法

IDelete（中断删除）用于取消（删除）中断预定。

如果中断仅临时禁用，则应当使用指令ISleep或IDisable。

基本示例

以下实例介绍了指令IDelete：

例 1

```
IDelete feeder_low;
```

取消中断feeder_low。

变元

```
IDelete Interrupt
```

Interrupt

数据类型：intnum

中断识别号。

程序执行

彻底擦除中断的定义。为了再次进行定义，必须首先将其重新连接至软中断程序。

建议从停止点开始IDelete。否则，在达到运动路径终点之前，中断将停用。

无须擦除中断；在以下情形时，中断会自动停用：

- 加载新的程序
- 从起点重启程序
- 将程序指针移动到程序起点

语法

```
IDelete [ Interrupt ´:=’ ] < variable (VAR) of intnum > ´;’
```

相关信息

信息，关于	请参阅
中断概述	技术参考手册 - *RAPID*语言概览
更多关于中断管理的信息	技术参考手册 - *RAPID*语言概览
临时停用一个中断	第*302*页的*ISleep - 停用一个中断*
临时停用所有中断	第*232*页的*IDisable - 禁用中断*

1.99 IDisable - 禁用中断

手册用法

IDisable（中断禁用）用于使所有临时禁用所有中断。其可能用于程序中尤为敏感的部分，如果中断妨碍正常的程序执行，则可能不允许出现中断。

基本示例

以下实例介绍了指令IDisable：

例 1

```
IDisable;
FOR i FROM 1 TO 100 DO
  character[i]:=ReadBin(sensor);
ENDFOR
IEnable;
```

只要在读取串行通道，则不允许任何中断。

程序执行

将IDisable指令生效期间出现的中断列入等待队列。再次允许中断时，立即开始产生中断，并在队列中以“先进先出”的顺序执行。

IEnable默认有效。IEnable自动设置

- 当使用重启模式重置RAPID时
- 当加载一则新程序或一个新模块时
- 当从起点开始执行程序时
- 当将程序指针移动到main时
- 当将程序指针移动到子程序时
- 移动程序指针造成执行顺序丢失时
- 执行一个循环（通过main）或执行ExitCycle后

语法

```
IDisable´;’
```

相关信息

信息，关于	请参阅
中断概述	技术参考手册 - *RAPID*语言概览
更多关于中断管理的信息	技术参考手册 - *RAPID*语言概览
允许中断	第*233*页的*IEnable - 启用中断*

1.100 IEnable - 启用中断

手册用法

IEnable（中断启用）用于在程序执行期间启用中断。

基本示例

以下实例介绍了指令IEnable：

例 1

```
IDisable;
FOR i FROM 1 TO 100 DO
  character[i]:=ReadBin(sensor);
ENDFOR
IEnable;
```

只要在读取串行通道，则不允许任何中断。读取完毕时，再次允许中断。

程序执行

将IDisable指令生效期间出现的中断列入等候队列。当再次允许中断时（IEnable），随即开始产生中断，并以“先进先出”的顺序在队列中执行。随后，以普通的程序继续程序执行，且此后出现的中断应立即予以处理。

当程序始于起点时，始终允许中断。由ISleep指令禁用的中断，不受IEnable指令的影响。

语法

```
IEnable´;’
```

相关信息

信息，关于	请参阅
中断概述	技术参考手册 - *RAPID*语言概览
更多关于中断管理的信息	技术参考手册 - *RAPID*语言概览
不允许中断	第*232*页的*IDisable - 禁用中断*

1.101 IError - 调整关于错误的中断

手册用法

IError（*Interrupt Errors*）用于在出现错误时，下达中断指令和启用中断。

可通过IError记录错误、警告或状态变更。

基本示例

以下实例介绍了指令IError：

另请参阅第*235*页的更多示例

例 1

```
VAR intnum err_int;
...
PROC main()
  CONNECT err_int WITH err_trap;
  IError COMMON_ERR, TYPE_ALL, err_int;
```

每当系统中产生错误、警告或状态变更时，则在RAPID以及TRAP程序err_trap执行过程中，下达中断指令。

变元

```
IError ErrorDomain [\ErrorId] ErrorType Interrupt
```

ErrorDomain

数据类型：errdomain

有待监测的错误域。参见errdomain型预定义数据。使用COMMON_ERR，指定任意域。

[\ErrorId]

数据类型：num

待监测特定错误的数量视情况而定。必须指定完整错误编号首位数（错误域）以外的错误编号。

例如，10008程序重启，且必须指定为0008或仅为8。

ErrorType

数据类型：errtype

有待监测的事件类型，例如，错误、警告或状态变更。参见errtype型预定义数据。使用TYPE_ALL，指定任意类型。

Interrupt

数据类型：intnum

中断识别号。通过指令CONNECT，先前已经同软中断程序相连。

程序执行

当指定类型以及指定错误编号（视情况）的指定域中出现错误，则自动地调用相关的软中断程序。已经执行过时，继续从出现中断的位置进行程序执行。

下一页继续

更多示例

有关于指令IError的更多例子阐述如下。

```
VAR intnum err_interrupt;
VAR trapdata err_data;
VAR errdomain err_domain;
VAR num err_number;
VAR errtype err_type;
PROC main()
  CONNECT err_interrupt WITH trap_err;
  IError COMMON_ERR, TYPE_ERR, err_interrupt;
  ...
  IDelete err_interrupt;
  ...
ENDPROC
TRAP trap_err
  GetTrapData err_data;
  ReadErrData err_data, err_domain, err_number, err_type;
  ! Set domain no 1 ... 11
  SetGO go_err1, err_domain;
  ! Set error no 1 ...9999
  SetGO go_err2, err_number;
ENDTRAP
```

出现错误（仅限错误，不包括警告或状态变更）时，在软中断程序中检索错误编号，其数值用于设置2组数字信号输出。

限制

不可能下达关于内部错误的中断指令。

在NORMAL类任务中，将在程序停止期间舍弃事件，因此，在普通任务中，并不能获取所有事件。要获取所有事件，任务必须为静态或半静态类型。

在没有首先删除的情况下，同一变量无法多次用于中断识别号。因此，应当按照下述替代选择之一，对中断进行处理。

```
VAR intnum err_interrupt;
PROC main ( )
  CONNECT err_interrupt WITH err_trap;
  IError COMMON_ERR, TYPE_ERR, err_interupt;
  WHILE TRUE DO
  :
  :
  ENDWHILE
ENDPROC
```

在程序开始时启用中断。随后，使开始时的此类指令保持在程序主流程之外。

```
VAR intnum err_interrupt;
PROC main ( )
  CONNECT err_interrupt WITH err_trap;
  IError COMMON_ERR, TYPE_ERR, err_interupt;
  :
  :
  IDelete err_interrupt;
ENDPROC
```

下一页继续

1.101 IError - 调整关于错误的中断

RobotWare - OS

续前页

在程序结束时删除中断，随后重新启用。注意，在这种情况下，中断会在短暂的时间内无效。

语法

```
IError
  [ErrorDomain ':='] <expression (IN) of errdomain>
  ['\'ErrorId':=' <expression (IN) of num>\\ ',’
  [ErrorType' :='] <expression (IN) of errtype> ´,’
  [Interrupt' :='] <variable (VAR) of intnum>';’
```

相关信息

信息，关于	请参阅
中断概述	技术参考手册 - *RAPID*语言概览
更多关于中断管理的信息	技术参考手册 - *RAPID*语言概览
错误域、预定义常量	第*1393*页的*errdomain - 错误域*
错误类型、预定义常量	第*1402*页的*errtype - 错误类型*
获取当前TRAP的中断数据	第*219*页的*GetTrapData - 获取当前TRAP的中断数据*
获取关于错误的信息	第*496*页的*ReadErrData - 获取关于错误的信息*
Advanced RAPID	应用手册 - 控制器软件*IRC5*

1.102 IF - 如果满足条件，那么...；否则...

手册用法

根据是否满足条件，执行不同的指令时，使用IF。

基本示例

有关于指令IF的基本例子阐述如下。

另请参阅第237页的更多示例

例 1

```
IF reg1 > 5 THEN
  Set do1;
  Set do2;
ENDIF
```

仅当reg1大于5时，设置信号do1和do2 。

例 2

```
IF reg1 > 5 THEN
  Set do1;
  Set do2;
ELSE
  Reset do1;
  Reset do2;
ENDIF
```

根据reg1是否大于5，设置或重置信号do1和do2 。

变元

```
IF Condition THEN ...
  {ELSEIF Condition THEN ...}
[ELSE ...]
ENDIF
```

Condition

数据类型：bool

必须满足关于待执行THEN和ELSE/ELSEIF之间指令的条件。

更多示例

有关于如何使用指令IF的更多例子阐述如下。

例 1

```
IF counter > 100 THEN
  counter := 100;
ELSEIF counter < 0 THEN
  counter := 0;
ELSE
  counter := counter + 1;
ENDIF
```

通过1，使counter增量。但是，如果counter的数值超出限值0-100，则向counter分配相应的限值。

下一页继续

1.102 IF - 如果满足条件，那么...；否则...

RobotWare - OS

续前页

程序执行

依次测试条件，直至满足其中一个条件。通过与该条件相关的指令，继续程序执行。如果未满足任何条件，则通过符合ELSE的指令，继续程序执行。如果满足多个条件，则仅执行与第一个此类条件相关的指令。

语法

```
IF <conditional expression> THEN
  <statement list>
{ ELSEIF <conditional expression> THEN
  <statement list> | <EIT> }
[ ELSE
  <statement list> ]
ENDIF
```

相关信息

信息，关于	请参阅
条件（逻辑表达式）	技术参考手册 - *RAPID*语言概览

1.103 Incr - 增量为1

手册用法

Incr用于向数值变量或者永久数据对象增加1。

基本示例

以下实例介绍了指令Incr：

另请参阅第*239*页的更多示例

例 1

```
Incr reg1;
```

将1增加至reg1，即reg1:=reg1+1。

变元

```
Incr Name | Dname
```

Name

数据类型：num

待改变变量或者永久数据对象的名称。

Dname

数据类型：dnum

待改变变量或者永久数据对象的名称。

更多示例

有关于指令Incr的更多例子阐述如下。

例 1

```
VAR num no_of_parts:=0;
...
WHILE stop_production=0 DO
  produce_part;
  Incr no_of_parts;
  TPWrite "No of produced parts= "\Num:=no_of_parts;
ENDWHILE
```

更新FlexPendant示教器上各循环所产生的零件数。只要未设置输入信号stop_production，则继续进行生产。

例 2

```
VAR dnum no_of_parts:=0;
...
WHILE stop_production=0 DO
  produce_part;
  Incr no_of_parts;
  TPWrite "No of produced parts= "\Dnum:=no_of_parts;
ENDWHILE
```

更新FlexPendant示教器上各循环所产生的零件数。只要未设置输入信号stop_production，则继续进行生产。

下一页继续

1.103 Incr - 增量为1

RobotWare - OS

续前页

语法

```
Incr
  [ Name ':=' ] < var or pers (INOUT) of num >
  | [ Dname' :=' ] < var or pers (INOUT) of dnum >' ;'
```

相关信息

信息，关于	请参阅
将变量减少1	第*142*页的*Decr - 减量为1*
向变量增加任意数值。	第*26*页的*Add*-增加数值
运用任意表达式（例如，乘法）来更改数据	第*32*页的*":="* - 分配一个数值

1.104 IndAMove - 独立的绝对位置运动

手册用法

IndAMove（*Independent Absolute Movement*）用于将轴变更为独立模式，并将轴移动到特定位置。

独立轴是独立于机器人系统中的其他轴而运动的轴。由于立即继续程序执行，因此在独立轴运动期间，可能执行其他的指令（包括定位指令）。

如果轴在转数内运动，则应当转而使用指令IndRMove。如果在距离当前位置较近处运动，则必须使用指令IndDMove。

本指令仅可用于主任务T_ROB1，或者如果在*MultiMove*系统中，则可用于运动任务中。

基本示例

以下实例介绍了指令IndAMove：

另请参阅第*242*页的更多示例

例 1

```
IndAMove Station_A,2\ToAbsPos:=p4,20;
```

以20度/秒的速度，使Station_A的轴2运动至位置p4。

变元

```
IndAMove MecUnit Axis [\ToAbsPos] | [\ToAbsNum] Speed [\Ramp]
```

MecUnit

Mechanical Unit

数据类型：mecunit

机械单元名称

Axis

数据类型：num

机械单元当前轴的编号（1-6）

[\ToAbsPos]

To Absolute Position

数据类型： robtarget

指定为robtarget的轴位置。仅使用该特定Axis的分量。将数值用作以度数表示的绝对位置值（针对线性轴，以mm计）。

如果使用指令EOffsSet或EOffsOn来取代轴，则轴位置将受到影响。

针对机械臂轴，转而使用参数\ToAbsNum。

[\ToAbsNum]

To Absolute Numeric value

数据类型：num

用度数确定轴位置（针对线性轴，以mm计）。

使用该参数，位置将不受任何位移的影响，例如，EOffsSet或PDispOn。

下一页继续

1.104 IndAMove - 独立的绝对位置运动

Independent Axis

续前页

与\ToAbsPos的功能相同，但是，将本位置定义为一个数值，使其易于手动改变位置。

Speed

数据类型：num

轴速度，以度/秒计（针对线性轴，以mm/s计）。

[\Ramp]

数据类型：num

减小最大性能的加速度和减速度（1-100%，100% = 最大性能）。

程序执行

执行IndAMove时，指定轴以编程速度运动至指定轴位置。如果编程\Ramp，则加速度/减速度将会降低。

为将轴改变回正常模式，则使用IndReset指令。可以改变同轴逻辑位置的关系，以便从该位置擦除大量的转数，例如，为避免下一运动的回转。

通过执行另一个IndAMove指令（或另一个IndXMove指令），可以改变速度。如果选择相反方向的速度，则轴会停止，然后加速至新的速度和方向。

为逐步执行指令，仅以独立模式来设置轴。执行下一指令时，轴开始运动，且只要进行程序执行便继续运动。有关更多的信息，请参见*RAPID*参考手册 - *RAPID*概述中运动和*I/O*原则 - 程序执行期间的定位 - 独立的轴一节。

当程序指针移动至程序起点或新的程序时，所有的轴均自动地设置为普通，且未改变测量系统（相当于执行指令IndReset\Old）。

注意

在IndCMove运行后，IndAMove指令可导致轴旋转回在IndCMove指令中实施的运动。为防止出现这种情况，在IndAMove之前，使用IndReset指令，或者使用IndRMove指令。

限制

在独立模式下，无法轻推各个轴。如果试图手动执行该轴，则轴将不会运动，且将显示错误消息。执行IndReset指令，或者使程序指针移动至主要部分，以退出独立模式。

如果当轴处于独立模式时出现上电失败，则无法重启程序。显示错误消息，且程序必须始于起点。

本指令不适用于耦合机械臂腕轴（参见*RAPID*参考手册 - *RAPID*概述中运动和*I/O*原则 - 程序执行期间的定位 - 独立的轴一节）。

更多示例

有关于指令IndAMove的更多例子阐述如下。

例 1

```
ActUnit Station_A;
weld_stationA;
IndAMove Station_A,1\ToAbsNum:=90,20\Ramp:=50;
ActUnit Station_B;
```

下一页继续

```
weld_stationB_1;
WaitUntil IndInpos(Station_A,1 ) = TRUE;
WaitTime 0.2;
DeactUnit Station_A;
weld_stationB_2;
```

启用Station_A，并在工作站A中开始焊接。

随后，当在工作站B中焊接机械臂时，Station_A（轴1）得以运动至90度位置。轴的速度为20度/秒。通过将加速度/减速度降低至最大性能的50%，以此改变速度。

当工作站A达到此位置时，其将停用，且当机械臂继续在工作站B中进行焊接时，可在工作站中进行重新装载。

错误处理

可能会产生下列可恢复错误。错误可以由错误处理程序处理。系统变量ERRNO将设置为：

名称	错误原因
ERR_AXIS_ACT	未启用轴。

语法

```
IndAMove
  [ MecUnit':=' ] < variable (VAR) of mecunit>' ,'
  [ Axis':=' ] < expression (IN) of num>
  [ '\'ToAbsPos':=' < expression (IN) of robtarget> ]
  | [ '\' ToAbsNum':=' < expression (IN) of num> ] ','
  [ Speed ':=' ] < expression (IN) of num>
  [ '\' Ramp':=' < expression (IN) of num > ] ';'
```

相关信息

信息，关于	请参阅
普通的独立轴	技术参考手册 - *RAPID*语言概览
Independent Axis	应用手册 - 控制器软件*IRC5*
变更回正常模式	第*250*页的*IndReset* - 独立重置
重置测量系统	第*250*页的*IndReset* - 独立重置
其他独立轴运动	第*254*页的*IndReset* - 独立的相对位置运动 第*247*页的*IndDMove* - 独立的德尔塔位置运动 第*244*页的*IndCMove* - 独立的连续运动
检查独立轴的速度状态	第*1121*页的*IndSpeed* - 独立的速度状态
检查独立轴的位置状态	第*1119*页的*IndInpos* - 在位置状态中的独立轴
定义独立接头	技术参考手册 - 系统参数

1.105 IndCMove - 独立的连续运动

手册用法

IndCMove（*Independent Continuous Movement*）用于将轴变更为独立模式，并以指定速度，开始轴的连续运动。

独立轴是独立于机器人系统中的其他轴而运动的轴。由于立即继续程序执行，因此在独立轴运动期间，可能执行其他的指令（包括定位指令）。

本指令仅可用于主任务T_ROB1，或者如果在*MultiMove*系统中，则可用于运动任务中。

基本示例

以下实例介绍了指令IndCMove：

另请参阅第*245*页的更多示例

例 1

```
IndCMove Station_A,2,-30.5;
```

Station_A的轴2开始以30.5度/秒的速度，朝反方向运动。

变元

```
IndCMove MecUnit Axis Speed [\Ramp]
```

MecUnit

Mechanical Unit

数据类型：mecunit

机械单元名称

Axis

数据类型：num

机械单元当前轴的编号（1-6）。

Speed

数据类型：num

轴速度，以度/秒计（针对线性轴，以mm/s计）。

通过速度参数的标记，指定运动方向。

[\Ramp]

数据类型：num

减小最大性能的加速度和减速度（1-100%，100% = 最大性能）。

程序执行

执行IndCMove时，指定轴开始以编程速度运动。根据速度参数的符号，指定运动方向。如果编程\Ramp，则加速度/减速度将会降低。

为将轴改变回正常模式，使用IndReset指令。可改变与此相关的轴的逻辑位置-可擦除大量完整的转数，例如，为避免下一运动的回转。

通过执行进一步的IndCMove指令，可改变速度。如果选择反方向的速度，则轴停止，随后加速至新的速度和方向。为使轴停止，可使用速度参数0。随后，其将仍然处于独立模式。

下一页继续

逐步执行指令期间，仅以独立模式来设置轴。执行下一指令时，轴开始运动，且只要继续程序执行便继续运行。有关更多的信息，请参见*RAPID*参考手册 - *RAPID*概述中运动和*I/O*原则 - 程序执行期间的定位 - 独立的轴一节。

当程序指针移动至程序起点或新的程序时，所有的轴均自动地设置为正常模式，且未改变测量系统（相当于执行指令IndReset\Old）。

限制

当进一步从其逻辑零位置（常常为工作区的中部）运动时，轴位置的分辨率恶化。为再次实现高分辨率，可通过指令IndReset，将逻辑工作区设置为零。有关更多的信息，请参见*RAPID*参考手册 - *RAPID*概述中运动和*I/O*原则 - 程序执行期间的定位 - 独立的轴一节。

在独立模式下，无法轻推各个轴。如果试图手动执行该轴，则轴将不会运动，且将显示错误消息。执行IndReset指令，或者使程序指针移动至主要部分，以退出独立模式。

如果当轴处于独立模式时出现上电失败，则无法重启程序。显示错误消息，且程序必须始于起点。

本指令不适用于耦合机械臂腕轴（参见*RAPID*参考手册 - *RAPID*概述中运动和*I/O*原则 - 程序执行期间的定位 - 独立的轴一节）。

更多示例

有关于指令IndCMove的更多例子阐述如下。

```
IndCMove Station_A,2,20;
WaitUntil IndSpeed(Station_A,2 \InSpeed) = TRUE;
WaitTime 0.2;
MoveL p10, v1000, fine, tool1;
IndCMove Station_A,2,-10\Ramp:=50;
MoveL p20, v1000, z50, tool1;
IndRMove Station_A,2 \ToRelPos:=p1 \Short,10;
MoveL p30, v1000, fine, tool1;
WaitUntil IndInpos(Station_A,2 ) = TRUE;
WaitTime 0.2;
IndReset Station_A,2 \RefPos:=p40\Short;
MoveL p40, v1000, fine, tool1;
```

Station_A的轴2开始以20度/秒的速度，朝正方向运动。当该轴达到选定的速度时，机械臂轴开始运动。

当机械臂达到位置p10时，外轴改变方向，并以10度/秒的速度旋转。通过将加速度/减速度降低至最大性能的50%，改变速度。以此同时，机械臂朝p20执行。

随后，Station_A的轴2在当前转数内、于位置p1处尽快停止。

当轴2已达到此位置时，机械臂已停止在位置p30，轴2再次返回正常模式。此轴的测量系统偏移量为轴转数的整数变化量，以便实际位置尽可能接近p40。

随后，当机械臂移动至位置p40时，将通过指令MoveL p40，经由前往位置p40（最大±180度）的最短路径，移动Station_A的轴2。

下一页继续

1.105 IndCMove - 独立的连续运动

Independent Axis

续前页

错误处理

可能会产生下列可恢复错误。错误可以由错误处理程序处理。系统变量 ERRNO 将设置为：

名称	错误原因
ERR_AXIS_ACT	未启用轴。

语法

```
IndCMove
  [ MecUnit':=' ] < variable (VAR) of mecunit>' ,'
  [ Axis':=' ] < expression (IN) of num> ',' 
  [ Speed ':=' ] < expression (IN) of num>
  [ '\' Ramp':=' < expression (IN) of num > ] ';'
```

相关信息

信息，关于	请参阅
普通的独立轴	技术参考手册 - *RAPID*语言概览
Independent Axis	应用手册 - 控制器软件*IRC5*
变更回正常模式	第*250*页的*IndReset* - 独立重置
重置测量系统	第*250*页的*IndReset* - 独立重置
其他独立轴运动	第*241*页的*IndAMove* - 独立的绝对位置运动 第*254*页的*IndReset* - 独立的相对位置运动 第*247*页的*IndDMove* - 独立的德尔塔位置运动
检查独立轴的速度状态	第*1121*页的*IndSpeed* - 独立的速度状态
检查独立轴的位置状态	第*1119*页的*IndInpos* - 在位置状态中的独立轴
定义独立接头	技术参考手册 - 系统参数

1.106 IndDMove - 独立的德尔塔位置运动

手册用法

IndDMove（*Independent Delta Movement*）用于将轴变更为独立模式，并将轴移动到特定距离处。

独立轴是独立于机器人系统中的其他轴而运动的轴。由于立即继续程序执行，因此在独立轴运动期间，可能执行其他的指令（包括定位指令）。

如果需要将轴移动至特定位置，则必须转而使用指令IndAMove或IndRMove。

本指令仅可用于主任务T_ROB1，或者如果在*MultiMove*系统中，则可用于运动任务中。

基本示例

以下实例介绍了指令IndDMove：

另请参阅第*248*页的更多示例

例 1

```
IndDMove Station_A,2,-30,20;
```

以20度/秒的速度，使Station_A的轴2朝负方向运动30度。

变元

```
IndDMove MecUnit Axis Delta Speed [\Ramp]
```

MecUnit

Mechanical Unit

数据类型：mecunit

机械单元名称

Axis

数据类型：num

机械单元当前轴的编号（1-6）。

Delta

数据类型：num

当前轴有待运动的距离，以度数表达（对于线性轴，则以mm表达）。符号可指定运动方向。

Speed

数据类型：num

轴速度，以度/秒计（针对线性轴，以mm/s计）。

[\Ramp]

数据类型：num

减小最大性能的加速度和减速度（1-100%，100% = 最大性能）。

程序执行

执行IndDMove时，指定轴以编程速度运动至指定距离处。根据Delta参数的符号，指定运动方向。如果编程\Ramp，则加速度/减速度将会降低。

下一页继续

1.106 IndDMove - 独立的德尔塔位置运动

Independent Axis

续前页

如果轴正在运动，则当执行指令IndDMove时，根据轴的瞬时位置来计算新位置。如果执行距离为0的IndDMove指令，且轴已经位于运动位置，则轴将停止，随后运动回执行指令时的轴位置。

为将轴改变回正常模式，使用IndReset指令。可改变与此相关的轴的逻辑位置-可从该位置擦除大量完整的转数，例如，为避免下一运动的回转。

通过运行进一步的IndDMove指令（或另一个IndXMove指令），可以改变速度。如果选择相反方向的速度，则轴会停止，然后加速至新的速度和方向。

逐步执行指令期间，仅以独立模式来设置轴。执行下一指令时，轴开始运动，且只要继续程序执行便继续运行。有关更多的信息，请参见*RAPID*参考手册 - *RAPID*概述中运动和*I/O*原则 - 程序执行期间的定位 - 独立的轴一节。

当程序指针移动至程序起点或新的程序时，所有的轴均自动地设置为正常模式，且未改变测量系统（相当于运行指令IndReset \Old）。

限制

在独立模式下，无法轻推各个轴。如果试图手动执行该轴，则轴将不会运动，且将显示错误消息。执行IndReset指令，或者使程序指针移动至主要部分，以退出独立模式。

如果当轴处于独立模式时出现上电失败损失，则无法重启程序。显示错误消息，且程序必须始于起点。

本指令不适用于耦合机械臂腕轴（参见*RAPID*参考手册 - *RAPID*概述中运动和*I/O*原则 - 程序执行期间的定位 - 独立的轴一节）。

更多示例

有关于指令IndDMove的更多例子阐述如下。

例 1

```
IndAMove ROB_1,6\ToAbsNum:=90,20;
WaitUntil IndInpos(ROB_1,6) = TRUE;
WaitTime 0.2;
IndDMove Station_A,2,-30,20;
WaitUntil IndInpos(ROB_1,6) = TRUE;
WaitTime 0.2;
IndDMove ROB_1,6,400,20;
```

使机械臂的轴6运动至以下位置：

- 90度
- 60度
- 460度（1转+100度）

错误处理

可能会产生下列可恢复错误。错误可以由错误处理程序处理。系统变量ERRNO将设置为：

名称	错误原因
ERR_AXIS_ACT	未启用轴。

语法

```
IndDMove
```

下一页继续

```
[ MecUnit':=' ] < variable (VAR) of mecunit> ','
[ Axis':=' ] < expression (IN) of num> ','
[ Delta':=' ] < expression (IN) of num>','
[ Speed ':=' ] < expression (IN) of num>
[ '\' Ramp':=' < expression (IN) of num > ] ';'
```

相关信息

信息，关于	请参阅
普通的独立轴	技术参考手册 - *RAPID*语言概览
Independent Axis	应用手册 - 控制器软件*IRC5*
变更回正常模式	第*250*页的*IndReset* - 独立重置
重置测量系统	第*250*页的*IndReset* - 独立重置
其他独立轴运动	第*241*页的*IndAMove* - 独立的绝对位置运动 第*254*页的*IndReset* - 独立的相对位置运动 第*244*页的*IndCMove* - 独立的连续运动
检查独立轴的速度状态	第*1121*页的*IndSpeed* - 独立的速度状态
检查独立轴的位置状态	第*1119*页的*IndInpos* - 在位置状态中的独立轴
定义独立接头	技术参考手册 - 系统参数

1.107 IndReset - 独立重置

手册用法

IndReset（*Independent Reset*）用于将独立轴改变回正常模式。与此同时，可使旋转轴的测量系统运动许多转。

本指令仅可用于主任务T_ROB1，或者如果在*MultiMove*系统中，则可用于运动任务中。

基本示例

以下实例介绍了指令IndReset：

另请参阅第*252*页的更多示例

```
IndCMove Station_A,2,5;
MoveL *,v1000,fine,tool1;
IndCMove Station_A,2,0;
WaitUntil IndSpeed(Station_A,2\ZeroSpeed);
WaitTime 0.2
IndReset Station_A,2;
```

Station_A的轴2首先以独立模式运动，然后改变回正常模式。本轴将维持其位置。

> **注意**
>
> 当执行指令IndReset时，不得使当前的独立轴和正常轴运动。这便是先前位置是停止点的原因，并以零速执行IndCMove指令。此外，暂停0.2秒，用以确保已实现正确的状态。

变元

```
IndReset MecUnit Axis [\RefPos] | [\RefNum] [\Short] | [\Fwd]
     |[\Bwd] | \Old]
```

MecUnit

Mechanical Unit

数据类型：mecunit

机械单元名称

Axis

数据类型：num

机械单元当前轴的编号（1-6）。

[\RefPos]

Reference Position

数据类型： robtarget

将参考轴位置指定为robtarget。仅使用特定Axis的分量。该位置必须位于正常工作范围以内。

针对机械臂轴，转而使用参数\RefNum。

仅确定本参数连同参数\Short、\Fwd或\Bwd。不允许连同参数\Old。

下一页继续

[\RefNum]

Reference Numeric value

数据类型：num

以度数来确定参考轴位置（对于线性轴，则以mm来确定）。本位置必须位于正常工作范围以内。

仅确定本参数连同参数\Short、\Fwd或\Bwd。不允许连同参数\Old。

与\RefPos的功能相同，但是，将本位置定义为一个数值，使其易于手动改变位置。

[\Short]

数据类型：switch

测量系统将改变轴侧转数的整数，以便轴尽可能地接近指定的\RefPos或\RefNum位置。如果在IndReset之后，执行相同位置的定位指令，则轴将运行最短的路径，即小于±180度，以到达该位置。

[\Fwd]

Forward

数据类型：switch

测量系统将改变轴侧转数的整数，以便参考位置将位于指定\RefPos或\RefNum位置的正向侧。如果在IndReset之后，执行相同位置的定位指令，则轴将朝正方向转360度以内，以到达该位置。

[\Bwd]

Backward

数据类型：switch

测量系统将改变轴侧转数的整数，以便参考位置将位于指定\RefPos或\RefNum位置的反向侧。如果在IndReset之后，执行相同位置的定位指令，则轴将朝反方向转360度以内，以到达该位置。

[\Old]

数据类型：switch

保持在原位置。

注意

在远离零的位置，降低分辨率。

如果未指定参数\Short、\Fwd、\Bwd或\Old，则将\Old用作默认值。

程序执行

当执行IndReset时，其将独立轴转变回正常模式。与此同时，可使轴的测量系统运动整数转。

本指令亦可用于正常模式，以改变测量系统。

注意

本位置仅用于调整测量系统，即轴将不会运动到本位置。

下一页继续

1.107 IndReset - 独立重置

Independent Axis

续前页

限制

仅当以正常模式运行的所有有效轴均静止时，方可执行本指令。在与同一运动规划器相连的各机械单元中，所有有效轴均需静止。将要改变为正常模式的独立模式轴，亦必须保持静止。对于正常模式下的各轴，通过执行移动指令以及参数fine，以实现上述要求。通过IndCMove以及Speed:=0（随后等待0.2秒）、IndRMove、IndAMove或IndDMove指令，独立轴停止。

当从逻辑位置0移开时，各位置的分辨率得以降低。应当运用指令IndReset以及除\Old外的参数，将从位置0逐渐旋转的轴设置为零。

无法改变线性轴的测量系统。

为确保在轴以及相对测量测量系统（同步开关）的IndReset之后正确启动，必须在IndReset指令之后，额外增加0.12秒的延时。

仅机械臂轴6可用作独立轴。IndReset指令亦可用于IRB 1600、2600和4600模型上的轴4（而非用于ID版本）。如果IndReset用于机械臂轴4，则轴6不得处于独立模式。

如果该指令位于移动指令之后，则必须使用停止点（zonedata fine）而非飞越点来编程移动指令，否则，将无法在电源故障后重启。

无法在与任意下列特殊系统事件相连的RAPID程序中执行IndReset：PowerOn、Stop、QStop、Restart或者Step。

IndReset仅切换轴的独立状态。其无法用于停止独立移动。为停止独立移动，其必须达到停止条件，或者用户必须移动PP到Main。

更多示例

有关于指令IndReset的更多例子阐述如下。

例 1

```
IndAMove Station_A,1\ToAbsNum:=750,50;
WaitUntil IndInpos(Station_A,1);
WaitTime 0.2;
IndReset Station_A,1 \RefNum:=0 \Short;
IndAMove Station_A,1\ToAbsNum:=750,50;
WaitUntil IndInpos(Station_A,1);
WaitTime 0.2;
IndReset Station_A,1 \RefNum:=300 \Short;
```

Station_A中的轴1首先独立地移动至750度位置（2转和30度）。与此同时，当其改变至普通模式时，将逻辑位置设置为30度。

随后，Station_A中的轴1移动至750度位置（2转和30度）。与此同时，当其改变至普通模式时，将逻辑位置设置为390度（1转和30度）。

错误处理

可能会产生下列可恢复错误。错误可以由错误处理程序处理。系统变量ERRNO将设置为：

名称	错误原因
ERR_AXIS_ACT	未启用轴。
ERR_AXIS_MOVING	轴正在移动。

下一页继续

语法

```
IndReset
  [ MecUnit':=' ] < variable (VAR) of mecunit> ','
  [ Axis ':=' ] < expression (IN) of num>
  [ '\' RefPos ':=' < expression (IN) of robtarget> ] |
  [ '\' RefNum ':=' < expression (IN) of num> ]
  [ '\' Short ] | [ '\' Fwd ] | [ '\' Bwd ] | [ '\' Old ]';'
```

相关信息

信息，关于	请参阅
普通的独立轴	技术参考手册 - *RAPID*语言概览
Independent Axis	应用手册 - 控制器软件*IRC5*
将轴改变为独立模式	第*241*页的*IndAMove* - 独立的绝对位置运动 第*244*页的*IndCMove* - 独立的连续运动 第*247*页的*IndDMove* - 独立的德尔塔位置运动 第*254*页的*IndReset* - 独立的相对位置运动
检查独立轴的速度状态	第*1121*页的*IndSpeed* - 独立的速度状态
检查独立轴的位置状态	第*1119*页的*IndInpos* - 在位置状态中的独立轴
定义独立接头	技术参考手册 - 系统参数

1.108 IndReset - 独立的相对位置运动

手册用法

IndRMove（*Independent Relative Movement*）用于将旋转轴改变为独立模式，并将轴移动至一转内的特定位置。

独立轴是独立于机器人系统中的其他轴而运动的轴。由于立即继续程序执行，因此在独立轴运动期间，可能执行其他的指令（包括定位指令）。

如果轴有待移动至绝对位置（若干转），或者如果轴呈线性，则转而使用指令IndAMove。如果在距离当前位置一定的距离处开始移动，则必须使用指令IndDMove。

本指令仅可用于主任务T_ROB1，或者如果在*MultiMove*系统中，则可用于运动任务中。

基本示例

以下实例介绍了指令IndRMove：

另请参阅第*256*页的更多示例

例 1

```
IndRMove Station_A,2\ToRelPos:=p5 \Short,20;
```

Station_A的轴2以20度/秒的速度，在一转（最大旋转±180度）以内，经最短的路径移动至位置p5。

变元

```
IndRMove MecUnit Axis [\ToRelPos] | [\ToRelNum] [\Short] | [\Fwd]
     | [\Bwd] Speed [\Ramp]
```

MecUnit

Mechanical Unit

数据类型：mecunit

机械单元名称

Axis

数据类型：num

机械单元当前轴的编号（1-6）。

[\ToRelPos]

To Relative Position

数据类型： robtarget

指定为robtarget的轴位置。仅使用此特定Axis的分量。将数值用作轴旋转一转以内的位置值（以度数表示）。这意味着，轴没有运动到一转。

如果使用指令EOffsSet或EOffsOn来取代轴，则轴位置将受到影响。

针对机械臂轴，转而使用参数\ToRelNum。

[\ToRelNum]

To Relative Numeric value

数据类型：num

用度数来定义轴位置。

下一页继续

使用该参数，位置将不受任何位移的影响，例如，EOffsSet或PDispOn。

与\ToRelPos的功能相同，但是，将本位置定义为一个数值，使其易于手动改变位置。

[\Short]

数据类型：switch

轴经最短的路径移动至新位置。这意味着，任意方向的最大旋转将为180度。因此，移动方向取决于轴的当前位置。

[\Fwd]

Forward

数据类型：switch

本轴朝正方向移动至新位置。这意味着，最大旋转将为360度，且始终位于正方向（位置值有所增加）。

[\Bwd]

Backward

数据类型：switch

本轴朝反方向移动至新位置。这意味着，最大旋转将为360度，且始终位于反方向（位置值有所降低）。

如果省略\Short、\Fwd或\Bwd参数，则\Short用作默认值。

Speed

数据类型：num

轴速度以度/秒表示。

[\Ramp]

数据类型：num

减小最大性能的加速度和减速度（1-100%，100% = 最大性能）。

程序执行

执行IndRMove时，指定轴以编程速度运动至指定轴位置，但是，最多仅旋转一转。如果编程\Ramp，则加速度/减速度将会降低。

为将轴改变回正常模式，使用IndReset指令。可改变与此相关的轴的逻辑位置-可从该位置擦除大量完整的转数，例如，为避免下一运动的回转。

通过运行进一步的IndRMove指令（或另一个IndXMove指令），可以改变速度。如果选择相反方向的速度，则轴会停止，然后加速至新的速度和方向。

逐步执行指令期间，仅以独立模式来设置轴。执行下一指令时，轴开始运动，且只要继续程序执行便继续运行。有关更多的信息，请参见*RAPID*参考手册 - *RAPID*概述中运动和*I/O*原则 - 程序执行期间的定位 - 独立的轴一节。

当程序指针移动至程序起点或新的程序时，所有的轴均自动地设置为正常模式，且未改变测量系统（相当于运行指令IndReset \Old）。

下一页继续

1.108 IndReset - 独立的相对位置运动

Independent Axis

续前页

限制

在独立模式下，无法轻推各个轴。如果试图手动执行该轴，则轴将不会运动，且将显示错误消息。执行IndReset指令，或者使程序指针移动至主要部分，以退出独立模式。

如果当轴处于独立模式时出现上电失败，则无法重启程序。显示错误消息，且程序必须始于起点。

本指令不适用于耦合机械臂腕轴（参见*RAPID*参考手册 - *RAPID*概述中运动和*I/O*原则 - 程序执行期间的定位 - 独立的轴一节）。

更多示例

有关于指令IndRMove的更多例子阐述如下。

例 1

```
IndRMove Station_A,1\ToRelPos:=p5 \Fwd,20\Ramp:=50;
```

Station_A的轴1以20度/秒的速度，在一转（最多旋转360度）以内，开始朝正方向移动至位置p5。通过加速度/减速度来改变速度，使其降低至最大性能的50%。

```
IndAMove Station_A,1\ToAbsNum:=90,20;
WaitUntil IndInpos(Station_A,1 ) = TRUE;
IndRMove Station_A,1\ToRelNum:=80 \Fwd,20;
WaitTime 0.2;
WaitUntil IndInpos(Station_A,1 ) = TRUE;
WaitTime 0.2;
IndRMove Station_A,1\ToRelNum:=50 \Bwd,20;
WaitUntil IndInpos(Station_A,1 ) = TRUE;
WaitTime 0.2;
IndRMove Station_A,1\ToRelNum:=150 \Short,20;
WaitUntil IndInpos(Station_A,1 ) = TRUE;
WaitTime 0.2;
IndAMove Station_A,1\ToAbsNum:=10,20;
```

Station_A的轴1移动至以下位置：

- 90度
- 440度（1转+80度）
- 410度（1转+50度）
- 510度（1转+150度）
- 10度

错误处理

可能会产生下列可恢复错误。错误可以由错误处理程序处理。系统变量 ERRNO 将设置为：

名称	错误原因
ERR_AXIS_ACT	未启用轴。

语法

```
IndRMove
  [ MecUnit ':=' ] < variable (VAR) of mecunit> ','
  [ Axis ':=' ] < expression (IN) of num>
  [ '\' ToRelPos ':=' < expression (IN) of robtargets> ]
```

下一页继续

```
| [ '\' ToRelNum ':=' < expression (IN) of num> ]
[ '\' Short ] | [ '\' Fwd ] | [ '\' Bwd ] ','
[ Speed ':=' ] < expression (IN) of num>
[ '\' Ramp ':=' < expression (IN) of num > ] ';'
```

相关信息

信息，关于	请参阅
普通的独立轴	技术参考手册 - *RAPID*语言概览
Independent Axis	应用手册 - 控制器软件*IRC5*
变更回正常模式	第*250*页的*IndReset* - 独立重置
重置测量系统	第*250*页的*IndReset* - 独立重置
其他独立轴运动	第*241*页的*IndAMove* - 独立的绝对位置运动 第*247*页的*IndDMove* - 独立的德尔塔位置运动 第*244*页的*IndCMove* - 独立的连续运动
检查独立轴的速度状态	第*1121*页的*IndSpeed* - 独立的速度状态
检查独立轴的位置状态	第*1119*页的*IndInpos* - 在位置状态中的独立轴
定义独立接头	技术参考手册 - 系统参数

1.109 InitSuperv - 重置CAP的所有监控

手册用法

InitSuperv将用于初始化CAP监控。这意味着，将清除所有监控列表，将取消所有I/O预定。

示例

```
PROC main()
  InitSuperv;
  SetupSuperv diWR_EST, ACT,SUPERV_MAIN;
  SetupSuperv diGA_EST, ACT,SUPERV_MAIN;
  CapL p2, v100, cdata1, weavestart, weave,fine, tWeldGun;
ENDPROC
```

InitSuperv将用于在设置新监控前，清除所有监控列表。

限制

InitSuperv指令只应被执行一次，比如，从最开始的位置执行。

语法

```
InitSuperv ';'
```

相关信息

信息，关于	请参阅
Continuous Application Platform	应用手册 - *Continuous Application Platform*
SetupSuperv指令	第*599*页的*SetupSuperv* - 设置*CAP*信号监控条件
RemoveSuperv指令	第*505*页的*RemoveSuperv* - 撤除一个信号的条件

1.110 InvertDO - 转化数字信号输出信号值

手册用法

InvertDO（*Invert Digital Output*）转化数字信号输出的信号值（0 -> 1和1 -> 0）。

基本示例

以下实例介绍了指令InvertDO：

例 1

```
InvertDO do15;
```

转化信号do15的当前值.

变元

```
InvertDO Signal
```

Signal

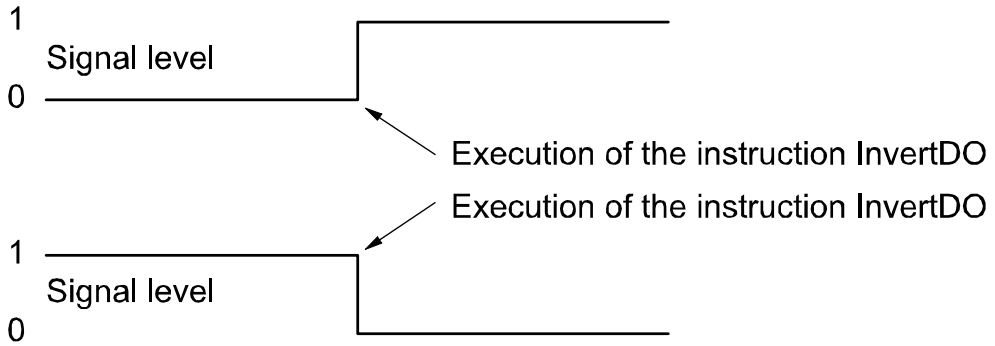
数据类型：signaldo

待转化信号的名称。

程序执行

转化信号的当前值（参见下图）。

下图显示了数字信号输出信号的转化。



xx0500002164

错误处理

可能会产生下列可恢复错误。错误可以由错误处理程序处理。系统变量ERRNO将设置为：

名称	错误原因
ERR_NO_ALIASIO_DEF	该信号变量是在RAPID中声明的一个变量，与用指令AliasIO在I / O配置中定义的I / O信号无关。
ERR_NORUNUNIT	与I / O单元间没有接触。
ERR_SIG_NOT_VALID	无法访问该I / O信号（仅对ICI现场总线有效）。

语法

```
InvertDO
  [ Signal ':=' ] < variable (VAR) of signaldo > ';'
```

下一页继续

1.110 InvertDO - 转化数字信号输出信号值

RobotWare - OS

续前页

相关信息

信息，关于	请参阅
输入/输出指令	技术参考手册 - *RAPID*语言概览
输入/输出功能性概述	技术参考手册 - *RAPID*语言概览
I/O配置	技术参考手册 - 系统参数

1.111 IOBusStart - Start of I/O bus

手册用法

IOBusStart用于创建特定的I/O总线。

基本示例

以下实例介绍了指令IOBusStart：

例 1

```
IOBusStart "IBS";
```

本指令以名称IBS，创建I/O总线。

变元

```
IOBusStart BusName
```

BusName

数据类型：string

创建I/O总线的名称。

程序执行

通过参数BusName中指定的名称，创建I/O总线。

错误处理

可能会产生下列可恢复错误。错误可以由错误处理程序处理。系统变量 ERRNO 将设置为：

名称	错误原因
ERR_NAME_INVALID	I/O总线名称不存在。

语法

```
IOBusStart
  [ BusName ':=' ] < expression (IN) of string>';'
```

相关信息

信息，关于	请参阅
如何获取I/O总线状态	第262页的*IOBusState - 获取I/O总线的当前状态*
I/O配置	技术参考手册 - 系统参数

1.112 IOBusState - 获取I/O总线的当前状态

手册用法

IOBusState用于读取特定I/O总线的状态。其物理状态和逻辑状态确定I/O总线的状态。

基本示例

以下实例介绍了指令IOBusState：

例 1

```
VAR busstate bstate;

IOBusState "IBS", bstate \Phys;
TEST bstate
CASE IOBUS_PHYS_STATE_RUNNING:
  ! Possible to access the signals on the IBS bus
DEFAULT:
  ! Actions for not up and running IBS bus
ENDTEST
```

本指令返回busstate型bstate变量中IBS的物理I/O总线状态。

例 2

```
VAR busstate bstate;

IOBusState "IBS", bstate \Logic;
TEST bstate
CASE IOBUS_LOG_STATE_STARTED:
  ! The IBS bus is started
DEFAULT:
  ! Actions for stopped IBS bus
ENDTEST
```

本指令返回busstate型bstate变量中IBS的逻辑I/O总线状态。

变元

```
IOBusState BusName State [\Phys] | [\Logic]
```

BusName

数据类型：string

用以获取相关状态的I/O总线名称。

State

数据类型：busstate

I/O总线状态得以返回的变量。参见下文程序执行时的busstate型预定义数据。

[\Phys]

Physical

数据类型：switch

如果使用该参数，则读取I/O总线的物理状态。

下一页继续

[\Logic]

Logical

数据类型：switch

如果使用该参数，则读取I/O总线的逻辑状态。

程序执行

参数State中的返回，即在参数BusName中指定的I/O总线的状态。

I/O总线逻辑状态描述了用户可下达关于总进线的指令的状态。当使用可选参数\Logic时，下表确定了I/O总线的状态。

返回值	符号常量	备注
10	IOBUS_LOG_STATE_STOPPED	由于错误，总线停止 2)
11	IOBUS_LOG_STATE_STARTED	总线启用 1)

I/O总线物理状态描述了现场总线驱动程序能够下达关于总进线的指令的状态。当使用可选参数\Phys时，下表确定了I/O总线的状态。

返回值	符号常量	备注
20	IOBUS_PHYS_STATE_HALTED	总线停用 3)
21	IOBUS_PHYS_STATE_RUNNING	总线正在使用中 1)
22	IOBUS_PHYS_STATE_ERROR	总线不工作 2)
23	IOBUS_PHYS_STATE_STARTUP	总线处于启动模式，且未与任何I/O单元相连。
24	IOBUS_PHYS_STATE_INIT	仅创建总线 3)

注意

当未使用任何可选参数\Phys或\Logic时，下表确定了I/O总线的状态。

返回值	符号常量	备注
0	BUSSTATE_HALTED	总线停用 3)
1	BUSSTATE_RUN	总线正在使用中 1)
2	BUSSTATE_ERROR	总线不工作 2)
3	BUSSTATE_STARTUP	总线处于启动模式，且未与任何I/O单元相连。
4	BUSSTATE_INIT	仅创建总线 3)

1) 如果I/O总线正在使用中，则指令IOBusState中的参数State中返回的状态，可以为IOBUS_LOG_STATE_STARTED、IOBUS_PHYS_STATE_RUNNING或BUSSTATE_RUN，具体取决于IOBusState中是否使用可选参数。

2) 如果I/O总线因某些错误而停止，则参数State中返回的状态可以为IOBUS_LOG_STATE_STOPPED、IOBUS_PHYS_STATE_ERROR或BUSSTATE_ERROR，具体取决于是否在IOBusState中使用可选参数。

3) 通过Robotware - OS的当前版本，不可能在RAPID程序中获得该状态。

下一页继续

1.112 IOBusState - 获取I/O总线的当前状态

RobotWare - OS

续前页

错误处理

可能会产生下列可恢复错误。错误可以由错误处理程序处理。系统变量 ERRNO 将设置为：

名称	错误原因
ERR_NAME_INVALID	I/O总线名称不存在。

语法

```
IOBusState
  [ BusName ':=' ] < expression (IN) of string> ','
  [ State ':=' ] < variable (VAR) of busstate>
  [ '\' Phys] | [ '\' Logic] ';'
```

相关信息

信息，关于	请参阅
I/O总线状态的定义	第*1348*页的*busstate - I/O总线的状态*
I/O总线的起点	第*261*页的*IOBusStart - Start of I/O bus*
输入/输出功能性概述	技术参考手册 - *RAPID*语言概览
I/O配置	技术参考手册 - 系统参数

1.113 IODisable - 停用I/O单元

手册用法

IODisable用于在程序执行期间停用I/O单元。

如果用系统参数来定义I/O单元，则I/O单元在启动后自动启用。当出于某种原因而被需要时，I/O单元可在程序执行期间停用或启用。

> **注意**
>
> 当Unit Trustlevel设置为Required时，不可能停用I/O单元。

基本示例

以下实例介绍了指令IODisable：

另请参阅第*266*页的更多示例

例 1

```
CONST string board1:="board1";
IODisable board1, 5;
```

停用名为board1的I/O单元。最多等待5秒。

变元

```
IODisable UnitName MaxTime
```

UnitName

数据类型：string

I/O单元的名称（必须以系统参数来呈现单元名称）。

MaxTime

数据类型：num

容许的最大等待时间以秒来表达。如果在I/O单元完成停用步骤之前，用尽该时间，则将调用错误处理器，如果存在这样的错误处理器，则错误代码为ERR_IODISABLE。如果不存在错误处理器，则将停止程序执行。I/O单元停用步骤将始终得以继续进行，无论MaxTime或错误如何。

停用一个I/O单元耗时大约0-5 s。

程序执行

指定的I/O单元开始停用步骤。当完成停用步骤时，本指令就绪。如果在I/O单元完成停用步骤之前，用尽MaxTime，则将产生可恢复的错误。

在I/O单元停用之后，关于该单元的任意输出设置均将产生错误。

错误处理

可能会产生下列可恢复错误。错误可以由错误处理程序处理。系统变量ERRNO将设置为：

名称	错误原因
ERR_IODISABLE	在停用I/O单元之前，等待时间到期。
ERR_NAME_INVALID	I/O单元名称不存在。

下一页继续

1.113 IODisable - 停用I/O单元
RobotWare - OS
续前页

名称	错误原因
ERR_TRUSTLEVEL	如果将Unit Trustlevel设置为Required，则无法停用I/O单元。

更多示例

有关于指令IODisable的更多例子阐述如下。

例 1

```
PROC go_home()
  VAR num recover_flag :=0;
  ...
  ! Start to deactivate I/O unit board1
  recover_flag := 1;
  IODisable "board1", 0;
  ! Move to home position
  MoveJ home, v1000,fine,tool1;
  ! Wait until deactivation of I/O unit board1 is ready
  recover_flag := 2;
  IODisable "board1", 5;
  ...
  ERROR
    IF ERRNO = ERR_IODISABLE THEN
      IF recover_flag = 1 THEN
        TRYNEXT;
      ELSEIF recover_flag = 2 THEN
        IF RemaningRetries() > 0 THEN
          RETRY;
        ELSE
          RAISE;
        ENDIF
      ENDIF
    ELSE
      ErrWrite "IODisable error", "Not possible to deactivate I/O
           unit board1";
      Stop;
    ENDIF
ENDPROC
```

为节省循环时间，I/O单元board1在机械臂移动至home位置期间停用。通过位于home位置的机械臂，完成试验，以确立I/O单元board1是否完全停用。在最多重试次数（4次，等待时间为5 s）之后，出现错误消息时，机械臂将停止执行。

相同的原则可以用在IOEnable（同IODisable相比，这将节省更多的循环时间）。

语法

```
IODisable
  [ UnitName ':=' ] < expression (IN) of string> ','
  [ MaxTime ':=' ] < expression (IN) of num> ';'
```

下一页继续

相关信息

信息，关于	请参阅
启用I/O单元	第*268*页的*IOEnable - 启用I/O单元*
输入/输出指令	技术参考手册 - *RAPID*语言概览
输入/输出功能概述	技术参考手册 - *RAPID*语言概览
I/O配置	技术参考手册 - 系统参数

1.114 IOEnable - 启用I/O单元

手册用法

IOEnable用于在程序执行期间启用I/O单元。

如果用系统参数来定义I/O单元，则I/O单元在启动后自动启用。当出于某种原因而被需要时，I/O单元可在程序执行期间停用或启用。

当启用I/O单元时，控制器行动取决于系统参数中确定的Unit Trustlevel。

基本示例

以下实例介绍了指令IOEnable：

另请参阅第*269*页的更多示例

例 1

```
CONST string board1:="board1";
IOEnable board1, 5;
```

启用名为board1的I/O单元。最多等待5秒。

变元

```
IOEnable UnitName MaxTime
```

UnitName

数据类型：string

I/O单元的名称（必须以系统参数来呈现I/O单元名称）。

MaxTime

数据类型：num

容许的最大等待时间以秒来表达。如果在I/O单元完成启用步骤之前，用尽该时间，则将调用错误处理器，如果存在这样的错误处理器，则错误代码为ERR_IOENABLE。如果不存在错误处理器，则将停止执行。I/O单元启用步骤将始终得以继续进行，无论MaxTime或错误如何。

启用一个I/O单元耗时大约2-5 s。

程序执行

指定的I/O单元开始启用步骤。当完成启用步骤时，本指令就绪。如果在I/O单元完成启用步骤之前，用尽MaxTime，则将产生可恢复的错误。

在IODisable - IOEnable顺序之后，关于当前I/O单元的所有输出值将得以设置为原来的值（在IODisable之前）。

错误处理

可能会产生下列可恢复错误。错误可以由错误处理程序处理。系统变量 ERRNO 将设置为：

名称	错误原因
ERR_IOENABLE	在启用I/O单元之前，用尽暂停时间。
ERR_NAME_INVALID	I/O单元名称不存在。
ERR_BUSSTATE	在启用I/O单元之前，I/O总线处于错误状态或进入错误状态。

下一页继续

更多示例

IOEnable亦可用于检查某些I/O单元是否因某些原因而断开。

有关于如何使用指令IOEnable的更多例子阐述如下。

例 1

```
VAR num max_retry:=0;
...
IOEnable "board1", 0;
SetDO board1_sig3, 1;
...
ERROR
  IF ERRNO = ERR_IOENABLE THEN
    IF RemaningRetries() > 0 THEN
      WaitTime 1;
      RETRY;
    ELSE
      RAISE;
    ENDIF
  ELSE
    ErrWrite "IOEnable error", "Not possible to activate I/O unit
         board1";
    Stop;
  ENDIF
```

在使用关于I/O单元board1的信号之前，通过在0秒之后暂停，以此尝试启用I/O单元，从而完成实验。如果试验失败，则跳接至错误处理器。在错误处理器中，程序执行等待1 秒，并进行新的尝试。在经过4重复尝试后，将错误ERR_IOENABLE传播至本程序的调用方。

语法

```
IOEnable
  [ UnitName ':=' ] < expression (IN) of string>' ,'
  [ MaxTime' :=' ] < expression (IN) of num > ';'
```

相关信息

信息，关于	请参阅
停用I/O单元	第265页的*IODisable - 停用I/O单元*
输入/输出指令	技术参考手册 - *RAPID*语言概览
输入/输出功能性概述	技术参考手册 - *RAPID*语言概览
I/O配置	技术参考手册 - 系统参数

1.115 IPathPos - 获取摆动时的中心线机器人位置。

手册用法

IPathPos将用于在以CAP进行摆动期间检索中心线位置。

此函数主要与跟踪功能结合使用。必须同时启动左侧和右侧的摆动信号和同步信号。

基本示例

```
connect intpt, TRP_ipathpos IPathPos p_robt, sen_pos, intpt;
```

当p_robt获得新计算值时，将发送中断intpt，将执行TRAP TRP_ipathpos。

变元

```
IPathPos p_robt, sen_pos, intpt [\NoDispl]
```

p_robt

数据类型： robtarget

p_robt将保持计算的robtarget的最新值。

sen_pos

数据类型：pos

不采用sen_pos。

intpt

数据类型：intno

intpt指定了每次为p_robt赋予新值时将收到的中断。

[\NoDispl]

数据类型：switch

如果指定\NoDispl，那么，PERS p_robt中返回的值将不包括可能采用RAPID指令PDispSet和PDispOn指定的位移。

限制

需要启动摆动和摆动同步（可进行也可不进行跟踪）。

语法

```
IPathPos
  [p_robt ':='] < persistent (PERS) of robtarget > ','
  [sen_pos ':='] < persistent (PERS) of pos > ','
  [Interrupt ':='] < variable (IN) of intnum >
  ['\' NoDispl ] ';'
```

相关信息

信息，关于	请参阅
Continuous Application Platform	应用手册 - *Continuous Application Platform*
CapWeaveSync指令	第*95*页的*CapWeaveSync - 设置摆动同步信号和电平*
CapAPTrSetup指令	第*65*页的*CapAPTrSetup - 设置At-Point-Tracker*

下一页继续

续前页

信息，关于	请参阅
CapLATrSetup指令	第86页的*CapLATrSetup - 设置Look-Ahead-Tracker*

1.116 IPers - 在永久变量数值改变时中断

手册用法

IPers（*Interrupt Persistent*）用于在永久变量的数值发生改变时，下达中断指令和启用中断。

基本示例

以下实例介绍了指令IPers：

例 1

```
VAR intnum pers1int;
PERS num counter := 0;

PROC main()
  CONNECT pers1int WITH iroutine1;
  IPers counter, pers1int;
  ...
  IDelete pers1int;
ENDPROC

TRAP iroutine1
  TPWrite "Current value of counter = " \Num:=counter;
ENDTRAP
```

下达关于每当永久变量counter改变时出现的中断的指令。随后，调用iroutine1软中断程序。

变元

```
IPers Name Interrupt
```

Name

数据类型：anytype

将产生中断的永久变量。

可以使用所有类型的数据，例如，原子、记录、记录成分、数组或数组元素。

Interrupt

数据类型：intnum

中断识别号。通过指令CONNECT，先前已经同软中断程序相连。

程序执行

当永久变量数值改变时，调用相关的软中断程序。当已经执行此程序时，继续从出现中断的位置进行程序执行。

如果永久变量数值在程序停止期间改变，则当程序再次启动时，将不会出现任何中断。

限制

在没有首先删除中断识别号的相同变量时，不能多次使用该相同变量。参见指令-ISignalDI。

如果预定参数Name中指定的记录成分或数组元素等数据，则每当数据任意部分发生改变时，将会出现中断。

下一页继续

当执行软中断程序以及读取永久变量数值时，无法确保读取的数值便是引起中断的数值。

语法

```
IPers
  [ Name ':=' ] < persistent (PERS) of anytype > ','
  [ Interrupt' :=' ] < variable (VAR) of intnum > ';'
```

相关信息

信息，关于	请参阅
中断和中断管理概述	技术参考手册 - *RAPID*语言概览
输入信号中断	第*290*页的*ISignalDI* - 下达数字信号输入信号中断指令
中断识别号	第*1414*页的*intnum* - 中断识别号
Advanced RAPID	应用手册 - 控制器软件*IRC5*

1.117 IRMQMessage - 下达数据类型的RMQ中断指令

手册用法

IRMQMessage（*Interrupt RAPID Message Queue Message*）用于在使用RMQ功能时，下达和启用特定数据类型的中断指令。

基本示例

以下实例介绍了指令IRMQMessage：

另请参阅第*274*页的*IRMQMessage - 下达数据类型的RMQ中断指令*

例 1

```
VAR intnum rmqint;
VAR string dummy;
...
PROC main()
  CONNECT rmqint WITH iroutine1;
  IRMQMessage dummy, rmqint;
```

下达每当收到包含数据类型string的新rmqmessage时出现的中断指令。随后，调用iroutine1TRAP程序。

变元

```
IRMQMessage InterruptDataType Interrupt
```

InterruptDataType

数据类型：anytype

当收到指定数据类型的rmqmessage时，变量、永久或常量等数据类型的引用将产生中断。

Interrupt

数据类型：intnum

中断识别号。通过指令CONNECT，先前已经同TRAP程序相连。

程序执行

当收到指定数据类型的RMQ消息时，调用相关的TRAP程序。当已经执行时，继续从出现中断的位置进行程序执行。

无论维度数量如何，所有包含相同类型数据的消息，均将由相同中断进行处理。如果使用不同的纬度，则使用RMQGetMsgHeader以进行匹配。

包含未连接中断的数据类型的任意消息，将产生警告。

如果收到消息，且该消息符合对同TRAP程序以及指令IRMQMessage相连的预期答案和消息的描述，则RMQSendWait指令具有最高优先权。

并非所有数据类型都能用于参数InterruptDataType（参见限制）。

本中断被认作是安全中断。安全中断无法同指令ISleep一同处于休眠模式。程序停止时，安全中断事件将列入队列，并逐步执行，且当再次以连续模式启动时，将执行中断。弃用安全中断的唯一时刻便是中断队列已满。随后，将报告错误。本中断将无法经历程序重置，例如，PP到Main。

下一页继续

更多示例

有关于如何使用指令IRMQMessage的更多例子阐述如下。

例 1

```
MODULE ReceiverMod
  VAR intnum intno1;
  VAR rmqheader rmqheader1;
  VAR rmqslot rmqslot1;
  VAR rmqmessage rmqmessage1;

  PROC main()
    VAR string interrupt_on_str := stEmpty;
    CONNECT intno1 WITH RecMsgs;
    ! Set up interrupts for data type string
    IRMQMessage interrupt_on_str, intno1;

    ! Perform cycle
    WHILE TRUE DO
      ...
    ENDWHILE
  ENDPROC
  TRAP RecMsgs
    VAR string receivestr;
    VAR string client_name;
    VAR num userdef;

    ! Get the message from the RMQ
    RMQGetMessage rmqmessage1;
    ! Get information about the message
    RMQGetMsgHeader rmqmessage1 \Header:=rmqheader1
         \SenderId:=rmqslot1 \UserDef:=userdef;

    IF rmqheader1.datatype = "string" AND rmqheader1.ndim = 0 THEN
      ! Get the data received in rmqmessage1
      RMQGetMsgData rmqmessage1, receivestr;
      client_name := RMQGetSlotName(rmqslot1);
      TPWrite "Rec string: " + receivestr;
      TPWrite "User Def: " + ValToStr(userdef);
      TPWrite "From: " + client_name;
    ELSE
      TPWrite "Faulty data received!"
    ENDIF

  ENDTRAP
ENDMODULE
```

例子表明如何设置指定数据类型的中断。当收到消息时，执行TRAPRecMsgs，并将从消息中收到的数据在FlexPendant示教器上打印出来。如果收到的数据类型或数据维度与预期不同，则将其在FlexPendant示教器上打印出来。

下一页继续

1.117 IRMQMessage - 下达数据类型的RMQ中断指令

FlexPendant Interface, PC Interface, or Multitasking

续前页

限制

不允许以同步模式执行IRMQMessage。这将会引起致命的运行时间错误。

不可能设置中断，发送或接收非值、半值数据类型或数据类型motsetdata.等数据类型的实例。

在没有首先删除的情况下，同一变量无法多次用于中断识别号。因此，应当按照下述替代选择之一，对中断进行处理。

```
VAR intnum rmqint;
PROC main ()
  VAR mytype dummy;
  CONNECT rmq1int WITH iroutine1;
  IRMQMessage dummy, rmqint;
  WHILE TRUE DO
  ...
  ENDWHILE
ENDPROC
```

在程序开始时启用所有中断。随后，使开始指令保持在程序主流程之外。

```
VAR intnum rmqint;
PROC main ( )
  VAR mytype dummy;
  CONNECT rmqint WITH iroutine1;
  IRMQMessage dummy, rmqint;
  ...
  IDelete rmqint;
ENDPROC
```

在程序结束时删除中断，随后重新启用。注意，在这种情况下，中断会在短暂的时间内无效。

语法

```
IRMQMessage
  [ InterruptDataType ':=' ] < reference (REF) of anytype >
  [ Interrupt ':=' ] < variable (VAR) of intnum >';'
```

相关信息

信息，关于	请参阅
RAPID消息队列功能描述	应用手册 - 控制器软件*IRC5*
将数据发送至RAPID任务或机械臂应用开发器客户端的队列。	第*520*页的*RMQFindSlot - 从槽名中寻找槽识别号*
从RAPID消息队列中获取第一个消息。	第*522*页的*RMQGetMessage - 获取RMQ消息*
将数据发送至RAPID任务或机械臂应用开发器客户端的队列，并等待客户端的回答。	第*537*页的*RMQSendWait - 发送RMQ数据消息，并等待响应*
从rmqmessage提取标题数据。	第*528*页的*RMQGetMsgHeader - 从RMQ消息获取标题信息*
将数据发送至RAPID任务或机械臂应用开发器客户端的队列。	第*534*页的*RMQSendMessage - 发送RMQ数据消息*
从rmqmessage提取数据。	第*525*页的*RMQGetMsgData - 从RMQ消息获取数据部分*

下一页继续

续前页

信息，关于	请参阅
从指定槽识别号中获取槽名。	第*1215*页的*RMQGetSlotName - 获取RMQ客户端的名称*

1.118 ISignalAI - 模拟信号输入信号的中断

手册用法

ISignalAI（中断信号模拟信号输入）用于下达和启用模拟信号输入信号的中断指令。

基本示例

以下实例介绍了指令ISignalAI：

例 1

```
VAR intnum sig1int;
PROC main()
  CONNECT sig1int WITH iroutine1;
  ISignalAI \Single, ai1, AIO_BETWEEN, 1.5, 0.5, 0, sig1int;
```

下达0.5和1.5之间的模拟信号输入信号ai1逻辑值首次出现的中断指令。随后，调用iroutine1软中断程序。

例 2

```
ISignalAI ai1, AIO_BETWEEN, 1.5, 0.5, 0.1, sig1int;
```

下达在0.5和1.5之间的模拟信号输入信号ai1的逻辑值每次出现的中断指令，且同储存的参考值相比，绝对信号差异大于0.1。

例 3

```
ISignalAI ai1, AIO_OUTSIDE, 1.5, 0.5, 0.1, sig1int;
```

下达模拟信号输入信号ai1的逻辑值每当低于0.5或高于1.5时出现的中断指令，且同储存的参考值相比，绝对信号差异大于0.1。

变元

```
ISignalAI [\Single] | [\SingleSafe] Signal Condition HighValue
     LowValue DeltaValue [\DPos] | [\DNeg] Interrupt
```

[\Single]

数据类型：switch

确定中断是否仅出现一次或循环出现。如果参数Single得以设置，则中断最多出现一次。如果省略Single和SingleSafe参数，则每当满足条件时便会出现中断。

[\SingleSafe]

数据类型：switch

确定中断单一且安全。关于单一的定义，请参见Single参数的描述。安全中断无法同指令ISleep一同处于休眠模式。程序停止时，安全中断事件将列入队列，并逐步执行，且当再次以连续模式启动时，将执行中断。弃用安全中断的唯一时刻便是中断队列已满。随后，将报告错误。本中断将无法经历程序重置，例如，PP到Main。

Signal

数据类型：signalai

将产生中断的信号的名称。

下一页继续

Condition

数据类型：aiotrigg

确定HighValue和LowValue如何定义待满足的条件：

值	符号常量	备注
1	AIO_ABOVE_HIGH	如果高于指定高值，则信号将产生中断
2	AIO_BELOW_HIGH	如果低于指定高值，则信号将产生中断
3	AIO_ABOVE_LOW	如果高于指定低值，则信号将产生中断
4	AIO_BELOW_LOW	如果低于指定低值，则信号将产生中断
5	AIO_BETWEEN	如果介于指定低值与高值之间，则信号将产生中断
6	AIO_OUTSIDE	如果低于指定低值或高于指定高值，则信号将产生中断
7	AIO_ALWAYS	信号将始终产生中断

HighValue

数据类型：num

可定义条件的高逻辑值。

LowValue

数据类型：num

可定义条件的低逻辑值。

DeltaValue

数据类型：num

在产生新的中断之前，定义最小逻辑信号差异。在产生新的中断之前，同储存的参考值进行比较，当前信号值必须大于指定的DeltaValue。

[\DPos]

数据类型：switch

确定仅正逻辑信号差异将产生新的中断。

[\DNeg]

数据类型：switch

确定仅负逻辑信号差异将产生新的中断。

如果未使用任何\DPos和\DNeg参数，则正负差异均将产生新的中断。

Interrupt

数据类型：intnum

中断识别号。通过指令CONNECT，先前已经同软中断程序相连。

程序执行

当信号满足指定条件（包括Condition和DeltaValue）时，调用相关的软中断程序。当已经执行时，继续从出现中断的位置进行程序执行。

产生中断的条件

在下达中断预定指令之前，每当对信号进行取样时，读取、保存信号值，随后将其用作DeltaValue条件的参考值。

下一页继续

1.118 ISignalAI - 模拟信号输入信号的中断
RobotWare - OS
续前页

在中断预定期间，如果指定DeltaValue = 0，且在中断预定时间之后，对信号进行取样。随后，根据Condition，并考虑DeltaValue，将信号值同HighValue和LowValue进行比较，以确定是否应当产生中断。如果新读取值满足指定的HighValue和LowValueCondition，但是同最后储存的参考值相比，其差异小于或等于DeltaValue参数，因此，无中断出现。如果信号差异未处于指定方向，则将不会出现中断（参数\DPos或\DNeg）。

如果满足以下条件，则通过随后用于任意样本的新读取值，更新针对DeltaValue条件而储存的参考值：

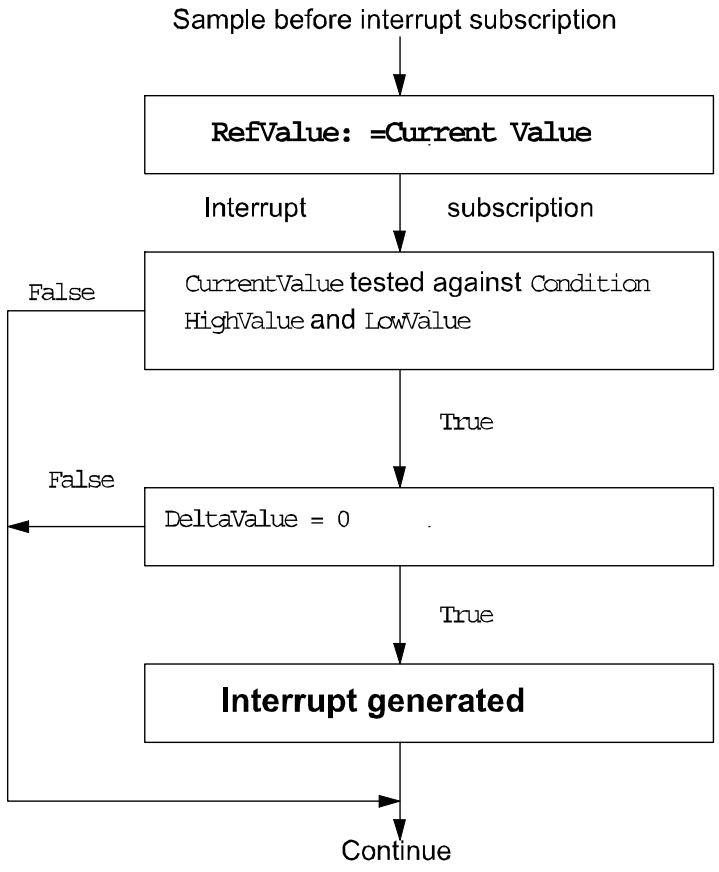
- 参数Condition以及指定的HighValue和LowValue（限值以内）。
- 参数DeltaValue（独立于指定开关\DPos或\DNeg，位于任意方向上的充足信号变更）

将在样本时间而非中断预定时间更新参考值。

如果信号差异的方向符合指定的参数（任意方向、\DPos0或\DNeg），则亦将在用以更新参考值的样本处产生中断。

当使用\Single开关时，最多将产生一个中断。如果未使用开关\Single（循环中断），则在信号值各样本处，对指定条件（包含Condition和DeltaValue）进行新的试验。比较当前信号值与最后储存的参考值，以决定是否应当产生中断。

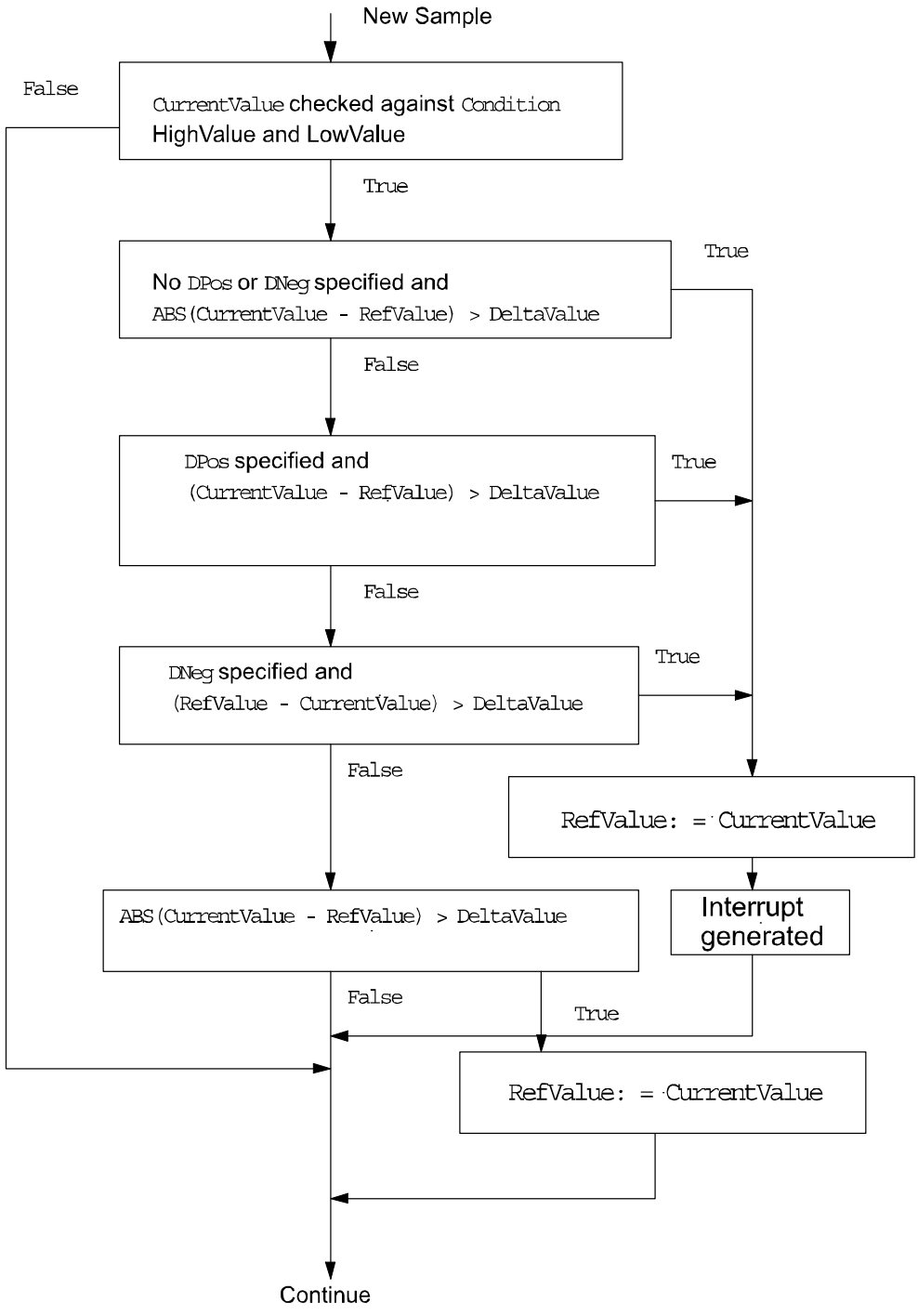
关于在中断预定期间产生中断的条件



xx0500002165

下一页继续

关于在中断预定之后，各样本产生中断的条件



xx0500002166

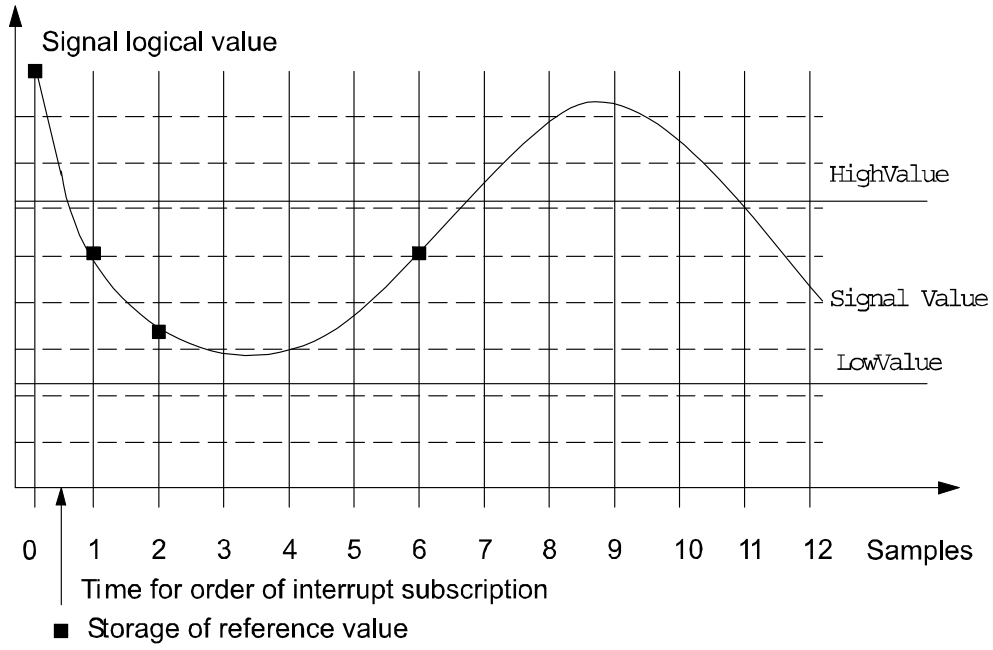
下一页继续

1.118 ISignalAI - 模拟信号输入信号的中断

RobotWare - OS

续前页

产生中断实例1



xx0500002167

假定下达指令的中断介于样本0与1之间，则以下指令将产生以下结果：

```
ISignalAI ai1, AIO_BETWEEN, 6.1, 2.2, 1.0, sig1int;
```

因为信号值介于`HighValue`**和**`LowValue`** 之间，且与样本0相比，信号差异高于**`DeltaValue`**，因此，样本1将产生中断。**

因为信号值介于`HighValue`** 和**`LowValue`**之间，且与样本1相比，信号差异高于**`DeltaValue`**，因此，样本2将产生中断。**

因为信号差异小于`DeltaValue`**，因此，样本3、4和5将不会产生任何中断。**

样本6将产生中断。

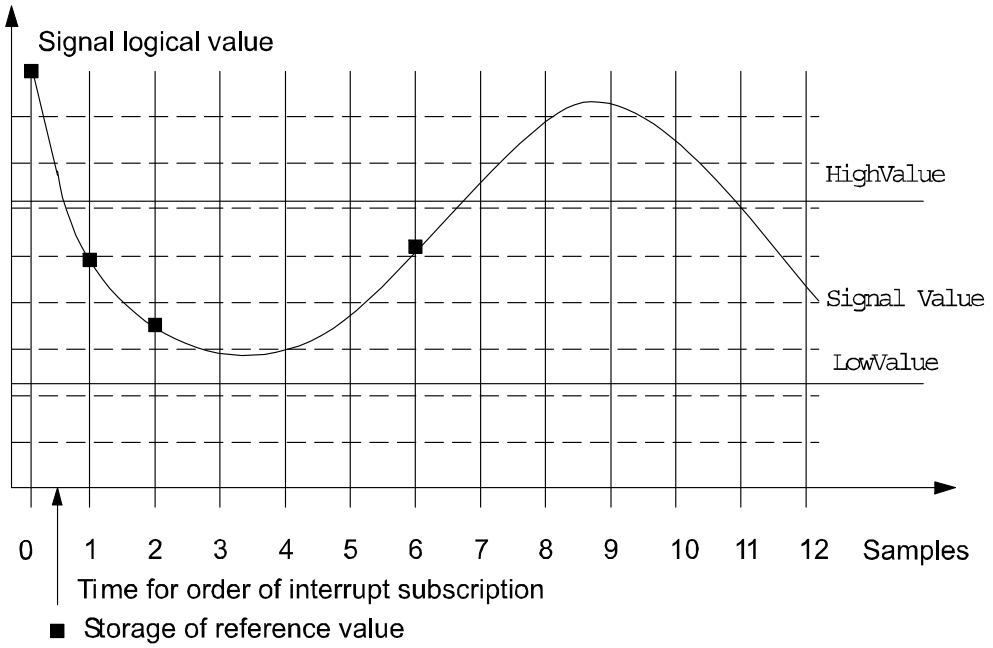
因为信号高于`HighValue`**，因此，样本7到10将不会产生任何中断。**

因为同样本6相比，信号差异等于`DeltaValue`**，因此，样本11将不会产生任何中断。**

因为同样本6相比，信号差异小于`DeltaValue`**，因此，样本12将不会产生任何中断。**

下一页继续

产生中断实例2



xx0500002168

假定下达指令的中断介于样本0与1之间，则以下指令将产生以下结果：

```
ISignalAI ai1, AIO_BETWEEN, 6.1, 2.2, 1.0 \DPos, sig1int;
```

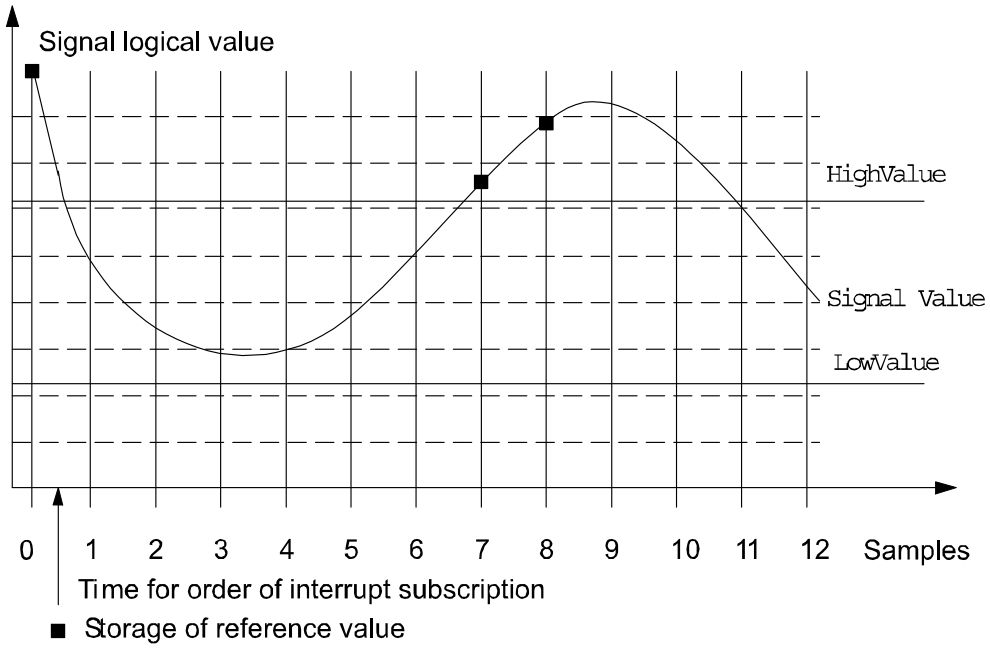
因为信号处于限值以内，且介于当前值与最后储存参考值之间的绝对信号差异大于1.0，因此，将新参考值储存在样本1和2处。因为信号朝负方向改变，因此，将不会产生任何中断。

因为信号值介于`HighValue`**和**`LowValue`**之间，且与样本2相比，正方向的信号差异高于**`DeltaValue`**，因此，样本6将产生中断。**

下一页继续

1.118 ISignalAI - 模拟信号输入信号的中断
RobotWare - OS
续前页

产生中断实例3



xx0500002169

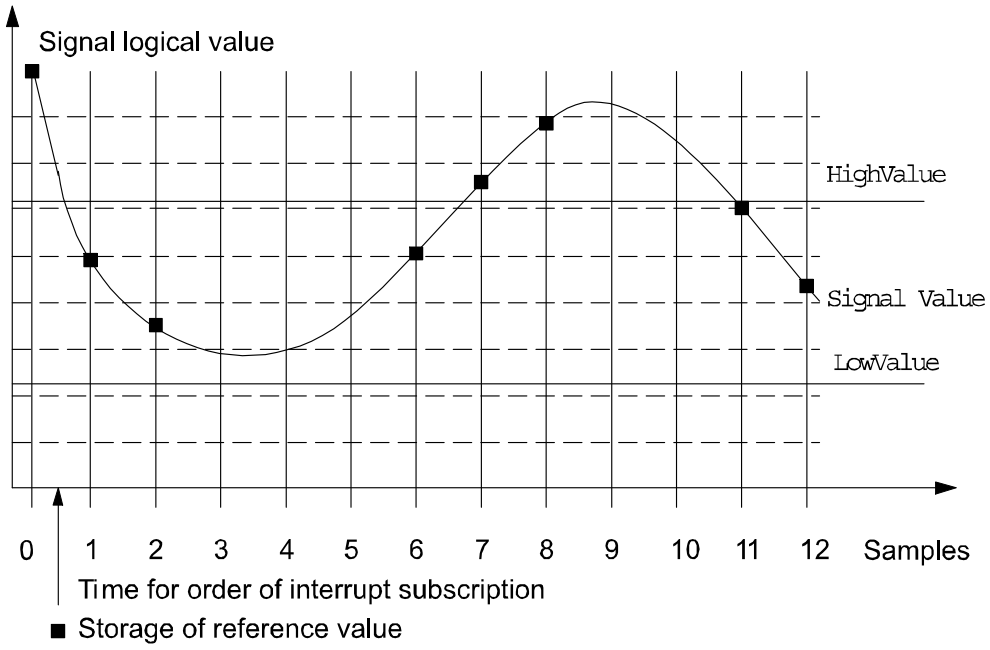
假定下达指令的中断介于样本0与1之间，则以下指令将产生以下结果：

```
ISignalAI \Single, ai1, AIO_OUTSIDE, 6.1, 2.2, 1.0 \DPos, sig1int;
```

因为信号处于限值以内，且介于当前值与最后储存参考值之间的绝对信号差异大于1.0，因此，将新参考值储存在样本7处。

因为信号值高于HighValue，且与样本7相比，正方向的信号差异高于DeltaValue，因此，样本8将产生中断。

产生中断实例4



xx0500002170

下一页继续

假定下达指令的中断介于样本0与1之间，则以下指令将产生以下结果：

```
ISignalAI ai1, AIO_ALWAYS, 6.1, 2.2, 1.0 \DPos, sig1int;
```

因为信号处于限值以内，且介于当前值与最后储存参考值之间的绝对信号差异大于1.0，因此，将新参考值储存在样本1和2处。

因为同样本2相比，正方向的信号差异大于DeltaValue，因此，样本6将会产生中断。

因为同先前样本相比，正方向的信号差异大于DeltaValue，因此，样本7和8将会产生中断。

因为信号处于限值以内，且介于当前值与最后储存参考值之间的绝对信号差异大于1.0，因此，将新参考值储存在样本11和12处。

错误处理

如果模拟信号输入信号预定中断，则在满足下达中断预定指令期间指定条件的模拟值中，其所有变更均将产生中断。如果模拟值过于嘈杂，则可产生许多中断，即便模拟值中的一位或两位发生改变。

为避免因模拟信号输入值的小变更产生中断，将DeltaValue设置为大于0的水平。随后，将不再产生中断，直至模拟数值的变更大于指定DeltaValue。

可能会产生下列可恢复错误。错误可以由错误处理程序处理。系统变量ERRNO将设置为：

名称	错误原因
ERR_NO_ALIASIO_DEF	信号变量是RAPID中声明的变量，且尚未同I/O配置以及指令AliasIO中确定的I/O信号相连。
ERR_AO_LIM	针对指定模拟信号输入信号Signal而编程的HighValue或LowValue参数超出限值。
ERR_NORUNUNIT	与I / O单元间没有接触。

限制

HighValue和LowValue参数应当介于针对信号而确定的逻辑最大值与逻辑最小值之间。

HighValue必须高于LowValue。

DeltaValue必须为0或者正值。

中断识别号的限制与ISignalDI的限制相同。

语法

```
ISignalAI
  [ '\' Single ] | [ '\' SingleSafe ] ','
  [ Signal ':=' ] <variable (VAR) of signalai> ','
  [ Condition ':=' ] <expression (IN) of aiotrigg> ','
  [ HighValue ':=' ] <expression (IN) of num> ','
  [ LowValue ':=' ] <expression (IN) of num> ','
  [ DeltaValue ':=' ] <expression (IN) of num>
  [['\'DPos] | [ '\' DNeg] ',']
  [ Interrupt ':=' ] <variable (VAR) of intnum> ';'
```

下一页继续

1.118 ISignalAI - 模拟信号输入信号的中断
RobotWare - OS
续前页

相关信息

信息，关于	请参阅
中断和中断管理概述	技术参考手册 - *RAPID*语言概览
常量的定义	第*1343*页的*aiotrigg - 模拟I/O*触发条件
信号输出信号的中断	第*287*页的*ISignalAO* - 模拟信号输出信号的中断
数字信号输入信号的中断	第*290*页的*ISignalDI* - 下达数字信号输入信号中断指令
数字信号输出信号的中断	第*293*页的*ISignalDO* - 数字信号输出信号的中断
中断识别号	第*1414*页的*intnum - 中断识别号*
相关的系统参数（过滤器）	技术参考手册 - 系统参数

1.119 ISignalAO - 模拟信号输出信号的中断

手册用法

ISignalAO（中断信号模拟信号输出）用于下达和启用模拟信号输出信号的中断指令。

基本示例

以下实例介绍了指令ISignalAO：

例 1

```
VAR intnum sig1int;
PROC main()
  CONNECT sig1int WITH iroutine1;
  ISignalAO \Single, ao1, AIO_BETWEEN, 1.5, 0.5, 0, sig1int;
```

下达关于在0.5和1.5之间的模拟信号输出信号ao1逻辑值首次出现中断的指令。随后，调用iroutine1软中断程序。

例 2

```
ISignalAO ao1, AIO_BETWEEN, 1.5, 0.5, 0.1, sig1int;
```

下达关于在0.5和1.5之间的模拟信号输出信号ao1的逻辑值每次出现中断的指令，且同储存的参考值相比，绝对信号差异大于0.1。

例 3

```
ISignalAO ao1, AIO_OUTSIDE, 1.5, 0.5, 0.1, sig1int;
```

下达关于低于0.5 或高于1.5的模拟信号输出信号ao1的逻辑值每次出现中断的指令，且同先前储存的参考值相比，绝对信号差异大于0.1。

变元

```
ISignalAO [\Single] | [\SingleSafe] Signal Condition HighValue
     LowValue DeltaValue [\DPos] | [\DNeg] Interrupt
```

[\Single]

数据类型：switch

确定中断是否仅出现一次或循环出现。如果参数Single得以设置，则中断最多出现一次。如果省略Single和SingleSafe参数，则每当满足条件时便会出现中断。

[\SingleSafe]

数据类型：switch

确定中断单一且安全。关于单一的定义，请参见Single参数的描述。安全中断无法同指令ISleep一同处于休眠模式。程序停止时，安全中断事件将列入队列，并逐步执行，且当再次以连续模式启动时，将执行中断。弃用安全中断的唯一时刻便是中断队列已满。随后，将报告错误。本中断将无法经历程序重置，例如，PP到Main。

Signal

数据类型：signalao

将产生中断的信号的名称。

下一页继续

1.119 ISignalAO - 模拟信号输出信号的中断
RobotWare - OS
续前页

Condition

数据类型：aiotrigg

确定HighValue和LowValue如何定义待满足的条件：

值	符号常量	备注
1	AIO_ABOVE_HIGH	如果高于指定高值，则信号将产生中断
2	AIO_BELOW_HIGH	如果低于指定高值，则信号将产生中断
3	AIO_ABOVE_LOW	如果高于指定低值，则信号将产生中断
4	AIO_BELOW_LOW	如果低于指定低值，则信号将产生中断
5	AIO_BETWEEN	如果介于指定低值与高值之间，则信号将产生中断
6	AIO_OUTSIDE	如果低于指定低值或高于指定高值，则信号将产生中断
7	AIO_ALWAYS	信号将始终产生中断

HighValue

数据类型：num

可定义条件的高逻辑值。

LowValue

数据类型：num

可定义条件的低逻辑值。

DeltaValue

数据类型：num

在产生新的中断之前，定义最小逻辑信号差异。在产生新的中断之前，同先前储存的参考值进行比较，当前信号值必须大于指定的DeltaValue。

[\DPos]

数据类型：switch

确定仅正逻辑信号差异将产生新的中断。

[\DNeg]

数据类型：switch

确定仅负逻辑信号差异将产生新的中断。

如果未使用\DPos和\DNeg参数，则正负差异均将产生新的中断。

Interrupt

数据类型：intnum

中断识别号。通过指令CONNECT，先前已经同软中断程序相连。

程序执行

关于以下内容的信息，请参见指令ISignalAI：

- 程序执行
- 产生中断的条件
- 更多示例

相同原则适用于ISignalAO以及ISignalAI。

下一页继续

错误处理

可能会产生下列可恢复错误。错误可以由错误处理程序处理。系统变量 ERRNO 将设置为：

名称	错误原因
ERR_NO_ALIASIO_DEF	信号变量是RAPID中声明的变量，且尚未同I/O配置以及指令AliasIO中确定的I/O信号相连。
ERR_AO_LIM	针对指定模拟信号输入信号Signal而编程的HighValue或LowValue参数超出限值。
ERR_NORUNUNIT	与I / O单元间没有接触。

限制

HighValue和LowValue参数应当介于针对信号而确定的逻辑最大值与逻辑最小值之间。

HighValue必须高于LowValue。

DeltaValue必须为0或者正值。

中断识别号的限制与ISignalDO的限制相同。

语法

```
ISignalAO
[ ’\’ Single ] | [ ’\’ SingleSafe ] ’,’
[ Signal’:=’ ]<variable (VAR) of signalao>’,’
[ Condition’:=’ ]<expression (IN) of aiotrigg>’,’
[ HighValue’:=’ ]<expression (IN) of num>’,’
[ LowValue’:=’ ]<expression (IN) of num>’,’
[ DeltaValue’:=’ ]<expression (IN) of num>
[[’\’DPos] | [ ’\’DNeg] ’,’]
[ Interrupt’:=’ ]<variable (VAR) of intnum>’;’
```

相关信息

信息，关于	请参阅
中断和中断管理概述	技术参考手册 - *RAPID*语言概览
常量的定义	第*1343*页的*aiotrigg - 模拟I/O触发条件*
模拟信号输入信号的中断	第*278*页的*ISignalAI - 模拟信号输入信号的中断*
数字信号输入信号的中断	第*290*页的*ISignalDI - 下达数字信号输入信号中断指令*
数字信号输出信号的中断	第*293*页的*ISignalDO - 数字信号输出信号的中断*
中断识别号	第*1414*页的*intnum - 中断识别号*
相关的系统参数（过滤器）	技术参考手册 - 系统参数

1.120 ISignalDI - 下达数字信号输入信号中断指令

手册用法

ISignalDI（中断信号数字信号输入）用于下达和启用数字信号输入信号的中断指令。

基本示例

以下实例介绍了指令ISignalDI：

例 1

```
VAR intnum sig1int;
PROC main()
  CONNECT sig1int WITH iroutine1;
  ISignalDI di1,1,sig1int;
```

下达关于每当数字信号输入信号di1设置为1时出现中断的指令。随后，调用iroutine1软中断程序。

例 2

```
ISignalDI di1,0,sig1int;
```

下达关于每当数字信号输入信号di1设置为0时出现中断的指令。

例 3

```
ISignalDI \Single, di1,1,sig1int;
```

仅下达数字信号输入信号di1首次设置为1时出现中断的指令。

变元

```
ISignalDI [ \Single] | [ \SingleSafe] Signal TriggValue Interrupt
```

[\Single]

数据类型：switch

确定中断是否仅出现一次或者循环出现。

如果参数Single得以设置，则中断最多出现一次。如果省略Single和SingleSafe参数，则每当满足条件时便会出现中断。

[\SingleSafe]

数据类型：switch

确定中断单一且安全。关于单一的定义，请参见Single参数的描述。安全中断无法同指令ISleep一同处于休眠模式。程序停止时，安全中断事件将列入队列，并逐步执行，且当再次以连续模式启动时，将执行中断。弃用安全中断的唯一时刻便是中断队列已满。随后，将报告错误。本中断将无法经历程序重置，例如，PP到Main。

Signal

数据类型：signaldi

将产生中断的信号的名称。

TriggValue

数据类型：dionum

信号因出现中断而必须改变的值。

下一页继续

将该值指定为0或1或符号值（例如，high/low）。在转变为0或1之后，边缘触发信号。

TriggValue 2或符号值edge可用于产生关于正侧面（0 -> 1）和负侧面（1 -> 0）的中断。

Interrupt

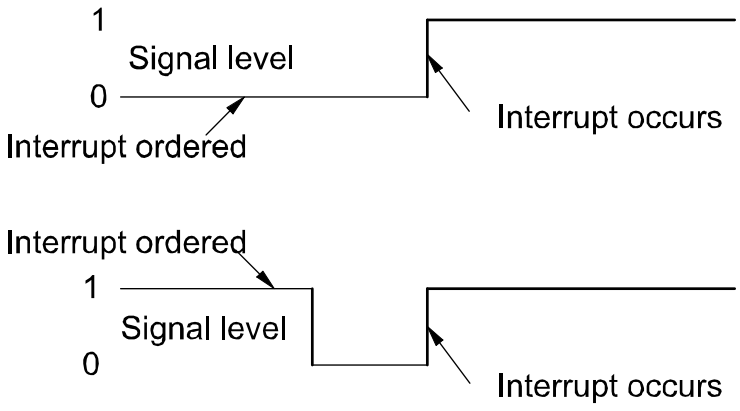
数据类型：intnum

中断识别号。通过指令CONNECT，先前已经同软中断程序相连。

程序执行

当信号承载指定值时，调用相关的软中断程序。当已经执行时，继续从出现中断的位置进行程序执行。

如果在下达中断指令之前，信号改变指定值，则不会出现中断。下图阐明了在信号等级1下，数字信号输入信号的中断。



xx0500002189

错误处理

可能会产生下列可恢复错误。错误可以由错误处理程序处理。系统变量 ERRNO 将设置为：

名称	错误原因
ERR_NO_ALIASIO_DEF	信号变量是RAPID中声明的变量，且尚未同I/O配置以及指令AliasIO中确定的I/O信号相连。
ERR_NORUNUNIT	与I / O单元间没有接触。

限制

在没有首先删除的情况下，同一变量无法多次用于中断识别号。因此，应当按照下述替代选择之一，对中断进行处理。

```
VAR intnum sig1int;
PROC main ()
  CONNECT sig1int WITH iroutine1;
  ISignalDI di1, 1, sig1int;
  WHILE TRUE DO
  ...
  ENDWHILE
ENDPROC
```

下一页继续

1.120 ISignalDI - 下达数字信号输入信号中断指令

RobotWare - OS

续前页

在程序开始时启用所有中断。随后，使开始指令保持在程序主流程之外。

```
VAR intnum sig1int;
PROC main ()
  CONNECT sig1int WITH iroutine1;
  ISignalDI di1, 1, sig1int;
  ...
  IDelete sig1int;
ENDPROC
```

在程序结束时删除中断，随后重新启用。注意，在这种情况下，中断会在短暂的时间内无效。

语法

```
ISignalDI
  [ '\' Single ] | [ '\' SingleSafe ] ','
  [ Signal ':=' ] < variable (VAR) of signaldi > ','
  [ TriggValue' :=' ] < expression (IN) of dionum > ','
  [ Interrupt' :=' ] < variable (VAR) of intnum > ';'
```

相关信息

信息，关于	请参阅
中断和中断管理概述	技术参考手册 - *RAPID*语言概览
输出信号的中断	第*293*页的*ISignalDO - 数字信号输出信号的中断*
中断识别号	第*1414*页的*intnum - 中断识别号*

1.121 ISignalDO - 数字信号输出信号的中断

手册用法

ISignalDO（中断信号数字信号输出）用于下达和启用数字信号输出信号中断的指令。

基本示例

以下实例介绍了指令ISignalDO：

例 1

```
VAR intnum sig1int;
PROC main()
  CONNECT sig1int WITH iroutine1;
  ISignalDO do1,1,sig1int;
```

下达关于每当数字信号输出信号do1设置为1时出现中断的指令。随后，调用iroutine1软中断程序。

例 2

```
ISignalDO do1,0,sig1int;
```

下达关于每当数字信号输出信号do1设置为0时出现中断的指令。

例 3

```
ISignalDO\Single, do1,1,sig1int;
```

仅下达关于数字信号输出信号do1首次设置为1时出现中断的指令。

变元

```
ISignalDO [ \Single ] | [ \SingleSafe ] Signal TriggValue Interrupt
```

`[ \Single ]`

数据类型：switch

确定中断是否仅出现一次或者循环出现。

如果参数Single得以设置，则中断最多出现一次。如果省略Single和SingleSafe参数，则每当满足条件时便会出现中断。

`[ \SingleSafe ]`

数据类型：switch

确定中断单一且安全。关于单一的定义，请参见Single参数的描述。安全中断无法同指令ISleep一同处于休眠模式。程序停止时，安全中断事件将列入队列，并逐步执行，且当再次以连续模式启动时，将执行中断。弃用安全中断的唯一时刻便是中断队列已满。随后，将报告错误。本中断将无法经历程序重置，例如，PP到Main。

`Signal`

数据类型：signaldo

将产生中断的信号的名称。

`TriggValue`

数据类型：dionum

信号因出现中断而必须改变的值。

下一页继续

1.121 ISignalDO - 数字信号输出信号的中断

RobotWare - OS

续前页

将该值指定为0或1或符号值（例如，high/low）。在转变为0或1之后，边缘触发信号。

TriggValue 2或符号值edge可用于产生关于正侧面（0 -> 1）和负侧面（1 -> 0）的中断。

Interrupt

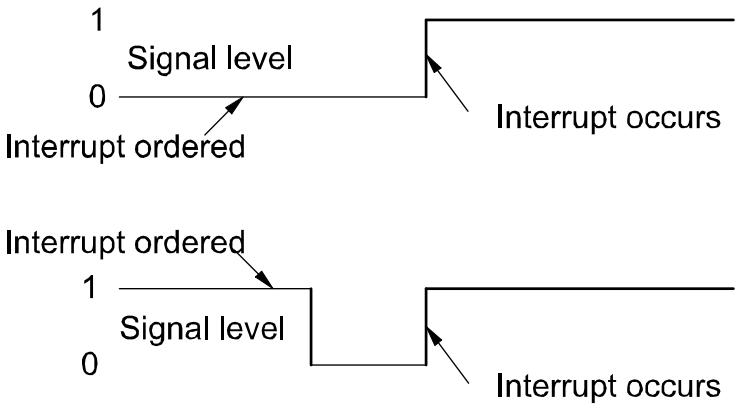
数据类型：intnum

中断识别号。通过指令CONNECT，先前已经同软中断程序相连。

程序执行

当信号承载指定值0或1时，调用相关的软中断程序。当已经执行时，继续从出现中断的位置进行程序执行。

如果在下达中断指令之前，信号改变指定值，则不会出现中断。下图阐明了在信号等级1下，数字信号输出信号的中断。



xx0500002190

错误处理

可能会产生下列可恢复错误。错误可以由错误处理程序处理。系统变量 ERRNO 将设置为：

名称	错误原因
ERR_NO_ALIASIO_DEF	信号变量是RAPID中声明的变量，且尚未同I/O配置以及指令AliasIO中确定的I/O信号相连。
ERR_NORUNUNIT	与I / O单元间没有接触。

限制

在没有首先删除的情况下，同一变量无法多次用于中断识别号。因此，应当按照下述替代选择之一，对中断进行处理。

```
VAR intnum sig1int;
PROC main ()
  CONNECT sig1int WITH iroutine1;
  ISignalDO do1, 1, sig1int;
  WHILE TRUE DO
  ...
  ENDWHILE
ENDPROC
```

下一页继续

在程序开始时启用所有中断。随后，使开始指令保持在程序主流程之外。

```
VAR intnum sig1int;
PROC main ()
  CONNECT sig1int WITH iroutine1;
  ISignalDO do1, 1, sig1int;
  ...
  IDelete sig1int;
ENDPROC
```

在程序结束时删除中断，随后重新启用。注意，在这种情况下，中断会在短暂的时间内无效。

语法

```
ISignalDO
  [ '\' Single ] | [ '\' SingleSafe ] ','
  [ Signal ':=' ] < variable (VAR) of signaldo > ','
  [ TriggValue' :=' ] < expression (IN) of dionum > ','
  [ Interrupt' :=' ] < variable (VAR) of intnum > ';'
```

相关信息

信息，关于	请参阅
中断和中断管理概述	技术参考手册 - *RAPID*语言概览
输入信号中断	第*290*页的*ISignalDI* - 下达数字信号输入信号中断指令
中断识别号	第*1414*页的*intnum* - 中断识别号

1.122 ISignalGI - 下达一组数字信号输入信号中断的指令

手册用法

ISignalGI（中断信号组数字信号输入）用于下达和启用一组数字信号输入信号中断的指令。

基本示例

以下实例介绍了指令ISignalGI：

例 1

```
VAR intnum sig1int;
PROC main()
  CONNECT sig1int WITH iroutine1;
  ISignalGI gi1,sig1int;
```

当数字信号输入组信号变更数值时，下达中断指令。

变元

```
ISignalGI [ \Single ] | [ \SingleSafe ] Signal Interrupt
```

[\Single]

数据类型：switch

确定中断是否仅出现一次或者循环出现。

如果参数Single得以设置，则中断最多出现一次。如果省略Single和SingleSafe参数，则每当满足条件时便会出现中断。

[\SingleSafe]

数据类型：switch

确定中断单一且安全。关于单一的定义，请参见Single参数的描述。安全中断无法同指令ISleep一同处于休眠模式。程序停止时，安全中断事件将列入队列，并逐步执行，且当再次以连续模式启动时，将执行中断。弃用安全中断的唯一时刻便是中断队列已满。随后，将报告错误。本中断将无法经历程序重置，例如，PP到Main。

Signal

数据类型：signalgi

产生中断的组输入信号的名称。

Interrupt

数据类型：intnum

中断识别号。通过指令CONNECT，先前已经同软中断程序相连。

程序执行

当组信号变更数值时，调用相关的软中断程序。当已经执行此程序时，继续从出现中断的位置进行程序执行。

如果信号在下达中断指令之前变更，则不会出现中断。

当数字组输入信号设置为一个数值时，可产生若干次中断。原因在于，在探测机器人系统的同时，未探测到组信号所涵盖的单个位的变更。为避免一个组信号变更的多次中断，可确定信号的过滤器时间。

下一页继续

错误处理

可能会产生下列可恢复错误。错误可以由错误处理程序处理。系统变量ERRNO将设置为：

名称	错误原因
ERR_NO_ALIASIO_DEF	信号变量是RAPID中声明的变量，且尚未同I/O配置以及指令AliasIO中确定的I/O信号相连。
ERR_NORUNUNIT	与I / O单元间没有接触。

限制

可成组使用的信号的最大数量为32。

无法在指令中使用数值条件，以确定中断应当出现在该特定值变更中。必须通过读取执行TRAP时的组信号数值，使用用户程序来对上述情况进行处理。

中断将以位中断的形式产生，例如，关于组内单个数字信号输入信号变更的中断。如果组信号变更数值中的各个位在各设置之间存在延迟，则将会产生若干中断。当使用ISignalGI时，关于I/O板如何工作的知识，对于获得正确的功能而言必不可少。如果组输入设置产生了若干中断，则转而使用关于选通信号的ISignalDI，该信号在组信号中的所有位均已设置时设置。

在没有首先删除的情况下，同一变量无法多次用于中断识别号。因此，应当按照下述替代选择之一，对中断进行处理。

```
VAR intnum sig1int;
PROC main ()
  CONNECT sig1int WITH iroutine1;
  ISignalGI gi1, sig1int;
  WHILE TRUE DO
  ...
  ENDWHILE
ENDPROC
```

在程序开始时启用所有中断。随后，使开始指令保持在程序主流程之外。

```
VAR intnum sig1int;
PROC main ()
  CONNECT sig1int WITH iroutine1;
  ISignalGI gi1, sig1int;
  ...
  IDelete sig1int;
ENDPROC
```

在程序结束时删除中断，随后重新启用。应当注意，在这种情况下，中断会在短暂的时间内无效。

语法

```
ISignalGI
  [ '\' Single ] | [ '\' SingleSafe ] ',’
  [ Signal ':=' ] < variable (VAR) of signalgi > ',’
  [ Interrupt':=' ] < variable (VAR) of intnum > ';’
```

下一页继续

1.122 ISignalGI - 下达一组数字信号输入信号中断的指令
RobotWare - OS
续前页

相关信息

信息，关于	请参阅
中断和中断管理概述	技术参考手册 - *RAPID*语言概览
输入信号中断	第*290*页的*ISignalDI* - 下达数字信号输入信号中断指令
组输出信号的中断	第*299*页的*ISignalGO* - 下达一组数字信号输出信号中断的指令
中断识别号	第*1414*页的*intnum* - 中断识别号
滤波时间	技术参考手册 - 系统参数

1.123 ISignalGO - 下达一组数字信号输出信号中断的指令

手册用法

ISignalGO（中断信号组数字信号输出）用于下达和启用一组数字信号输出信号中断的指令。

基本示例

以下实例介绍了指令ISignalGO：

例 1

```
VAR intnum sig1int;
PROC main()
  CONNECT sig1int WITH iroutine1;
  ISignalGO go1,sig1int;
```

当数字信号输出组信号变更数值时，下达中断指令。

变元

```
ISignalGO [ \Single ] | [ \SingleSafe ] Signal Interrupt
```

[\Single]

数据类型：switch

确定中断是否仅出现一次或者循环出现。

如果参数\Single得以设置，则中断最多出现一次。如果省略Single和SingleSafe参数，则每当满足条件时便会出现中断。

[\SingleSafe]

数据类型：switch

确定中断单一且安全。关于单一的定义，请参见Single参数的描述。安全中断无法同指令ISleep一同处于休眠模式。程序停止时，安全中断事件将列入队列，并逐步执行，且当再次以连续模式启动时，将执行中断。弃用安全中断的唯一时刻便是中断队列已满。随后，将报告错误。本中断将无法经历程序重置，例如，PP到Main。

Signal

数据类型：signalgo

产生中断的组输出信号的名称。

Interrupt

数据类型：intnum

中断识别号。通过指令CONNECT，先前已经同软中断程序相连。

程序执行

当组信号变更数值时，调用相关的软中断程序。当已经执行此程序时，继续从出现中断的位置进行程序执行。

如果信号在下达中断指令之前变更，则不会出现中断。

下一页继续

1.123 ISignalGO - 下达一组数字信号输出信号中断的指令
RobotWare - OS
续前页

错误处理

可能会产生下列可恢复错误。错误可以由错误处理程序处理。系统变量 ERRNO 将设置为：

名称	错误原因
ERR_NO_ALIASIO_DEF	信号变量是RAPID中声明的变量，且尚未同I/O配置以及指令 AliasIO中确定的I/O信号相连。
ERR_NORUNUNIT	与I / O单元间没有接触。

限制

可成组使用的信号的最大数量为32。

无法在指令中使用数值条件，以确定中断应当出现在该特定值变更中。必须通过读取执行TRAP时的组信号数值，使用用户程序来对上述情况进行处理。

在没有首先删除的情况下，同一变量无法多次用于中断识别号。因此，应当按照下述替代选择之一，对中断进行处理。

```
VAR intnum sig1int;
PROC main ()
  CONNECT sig1int WITH iroutine1;
  ISignalGO go1, sig1int;
  WHILE TRUE DO
  ...
  ENDWHILE
ENDPROC
```

在程序开始时启用所有中断。随后，使开始指令保持在程序主流程之外。

```
VAR intnum sig1int;
PROC main ()
  CONNECT sig1int WITH iroutine1;
  ISignalGO go1, sig1int;
  ...
  IDelete sig1int;
ENDPROC
```

在程序结束时删除中断，随后重新启用。注意，在这种情况下，中断会在短暂的时间内无效。

语法

```
ISignalGO
  [ '\' Single ] | [ '\' SingleSafe ] ','
  [ Signal ':=' ] < variable (VAR) of signalgo > ','
  [ Interrupt':=' ] < variable (VAR) of intnum > ';'
```

相关信息

信息，关于	请参阅
中断和中断管理概述	技术参考手册 - *RAPID*语言概览
输出信号的中断	第*293*页的*ISignalDO* - 数字信号输出信号的中断
组输入信号的中断	第*296*页的*ISignalGI* - 下达一组数字信号输入信号中断的指令

下一页继续

续前页

信息，关于	请参阅
中断识别号	第*1414*页的*intnum - 中断识别号*

1.124 ISleep - 停用一个中断

手册用法

ISleep（中断睡眠）用于暂时停用单个中断。

在停用期间，无软中断执行的情况下，可舍弃产生的任何指定类型的中断。

基本示例

以下实例介绍了指令ISleep。

另请参阅第*302*页的更多示例

例 1

```
ISleep sig1int;
```

停用中断sig1int。

变元

```
ISleep Interrupt
```

Interrupt

数据类型：intnum

中断的变量（中断识别号）。

程序执行

在未执行任何软中断的情况下，舍弃产生的所有指定类型的中断，直至已通过指令IWatch，重新启用中断。忽略ISleep生效时产生的中断。

更多示例

有关于指令ISleep的更多例子阐述如下。

例 1

```
VAR intnum timeint;
PROC main()
  CONNECT timeint WITH check_serialch;
  ITimer 60, timeint;
  ...
  ISleep timeint;
  WriteBin ch1, buffer, 30;
  IWatch timeint;
  ...
TRAP check_serialch
  WriteBin ch1, buffer, 1;
  IF ReadBin(ch1\Time:=5) < 0 THEN
    TPWrite "The serial communication is broken";
    EXIT;
  ENDIF
ENDTRAP
```

通过每60秒产生的中断，监测经由ch1串行通道的通信。软中断程序检查通信是否正在工作。当通信正在进行时，不允许中断。

下一页继续

错误处理

可能会产生下列可恢复错误。错误可以由错误处理程序处理。系统变量 ERRNO 将设置为：

名称	错误原因
ERR_UNKINO	尚不知晓中断编号。 不允许关于既未下达，亦未启用的中断的指令。
ERR_INOISSAFE	如果尝试暂时停用安全中断，则采用ISleep。

语法

```
ISleep
  [ Interrupt ':=' ] < variable (VAR) of intnum > ';'
```

相关信息

信息，关于	请参阅
中断概述	技术参考手册 - *RAPID*语言概览
启用中断	第*309*页的*IWatch - 启用中断*
停用所有中断	第*232*页的*IDisable - 禁用中断*
取消中断	第*231*页的*IDelete - 取消中断*

1.125 ITimer - 下达定时中断的指令

手册用法

ITimer（*Interrupt Timer*）用于下达和启用定时中断的指令。

例如，该指令可用于检查外围设备在每一分钟的状态。

基本示例

以下实例介绍了指令ITimer：

另请参阅第*305*页的更多示例

例 1

```
VAR intnum timeint;
PROC main()
  CONNECT timeint WITH iroutine1;
  ITimer 60, timeint;
```

下达每60秒循环出现中断的指令。随后，调用软中断程序iroutine1。

例 2

```
ITimer \Single, 60, timeint;
```

在60秒之后，下达仅出现一次中断的指令。

变元

```
ITimer [ \Single ] | [ \SingleSafe ] Time Interrupt
```

[\Single]

数据类型：switch

确定中断是否仅出现一次或者循环出现。

如果参数Single得以设置，则中断仅出现一次。如果省略Single和SingleSafe参数，则每到指定时间便会出现中断。

[\SingleSafe]

数据类型：switch

确定中断单一且安全。关于单一的定义，请参见Single参数的描述。无法通过指令ISleep，将安全中断置于睡眠。程序停止时，将安全中断事件列入队列，并逐步执行，且当再次以连续模式起动时，将执行中断。

Time

数据类型：num

在出现中断之前，必须耗费的时间。

以秒来指定数值。如果Single或SingleSafe得以设置，则时间可能不低于0.01秒。循环中断的相关时间为0.1秒。

Interrupt

数据类型：intnum

中断的变量（中断识别号）。通过指令CONNECT，先前已经同软中断程序相连。

下一页继续

程序执行

在中断下达后的给定时间，自动调用相关的软中断程序。当已经执行该程序时，从出现中断的位置继续进行程序执行。

如果周期性地出现中断，则重新计算自出现中断时开始的时间。

更多示例

有关于指令ITimer的更多例子阐述如下。

例 1

```
VAR intnum timeint;
PROC main()
  CONNECT timeint WITH check_serialch;
  ITimer 60, timeint;
  ...
TRAP check_serialch
  WriteBin ch1, buffer, 1;
  IF ReadBin(ch1\Time:=5) < 0 THEN
    TPWrite "The serial communication is broken";
    EXIT;
  ENDIF
ENDTRAP
```

通过每60秒产生的中断，监测经由ch1串行通道的通信。软中断程序检查是否正在进行通信。否则，结束程序执行，并会出现错误消息。

限制

在没有首先删除中断识别号的相同变量时，不能多次使用该相同变量。参见指令-ISignalDI。

语法

```
ITimer
  [ '\' Single ] | [ '\' SingleSafe ] ','
  [ Time ':=' ] < expression (IN) of num >','
  [ Interrupt' :=' ] < variable (VAR) of intnum > ';'
```

相关信息

信息，关于	请参阅
中断和中断管理概述	技术参考手册 - *RAPID*语言概览

1.126 IVarValue - 下达变量值中断指令

手册用法

IVarValue（*Interrupt Variable Value*）为，当通过串行传感器接口访问的变量数值发生变化时，用于下达中断指令并启用中断。

例如，该指令可用于从焊缝跟踪器获取焊缝体积或间隙值。

基本示例

以下实例介绍了指令IVarValue：

例 1

```
LOCAL PERS num
     adptVlt{25}:=[1,1.2,1.4,1.6,1.8,2,2.16667,2.33333,2.5,...];
LOCAL PERS num
     adptWfd{25}:=[2,2.2,2.4,2.6,2.8,3,3.16667,3.33333,3.5,...];
LOCAL PERS num
     adptSpd{25}:=10,12,14,16,18,20,21.6667,23.3333,25[,...];
LOCAL CONST num GAP_VARIABLE_NO:=11;
PERS num gap_value;
VAR intnum IntAdap;

PROC main()
  ! Setup the interrupt. The trap routine AdapTrp will be called
  ! when the gap variable with number ´GAP_VARIABLE_NO’ in the
  !sensor interface has been changed. The new value will be
  ! available in the PERS gp_value variable.
  ! Connect to the sensor device "sen1:" (defined in sio.cfg).
  SenDevice "sen1:";

  CONNECT IntAdap WITH AdapTrp;
  IVarValue "sen1:", GAP_VARIABLE_NO, gap_value, IntAdap;

  ! Start welding
  ArcL\On,*,v100,adaptSm,adaptWd,adaptWv,z10,tool\j\Track:=track;
  ArcL\On,*,v100,adaptSm,adaptWd,adaptWv,z10,tool\j\Track:=track;

ENDPROC

TRAP AdapTrap
  VAR num ArrInd;
  !Scale the raw gap value received
  ArrInd:=ArrIndx(gap_value);

  ! Update active welddata PERS variable ‘adaptWd’ with new data
  ! from the arrays of predefined parameter arrays. The scaled gap
  ! value is used as index in the voltage, wirefeed and
  ! speed arrays.
  adaptWd.weld_voltage:=adptVlt{ArrInd};
  adaptWd.weld_wirefeed:=adptWfd{ArrInd};
  adaptWd.weld_speed:=adptSpd{ArrInd};
```

下一页继续

```
    !Request a refresh of AW parameters using the new data i adaptWd
    ArcRefresh;

  ENDTRAP
```

变元

```
IVarValue device VarNo Value Interrupt [\Unit] [\DeadBand]
     [\ReportAtTool] [\SpeedAdapt] [\APTR]
```

device

数据类型：string

在sio.cfg中配置了I/O设备名称，以供传感器使用。

VarNo

数据类型：num

监督变量的数量。

Value

数据类型：num

将控制VarNo新值的PERS变量。

Interrupt

数据类型：intnum

中断的变量（中断识别号）。通过指令CONNECT，先前已经同软中断程序相连。

[\Unit]

数据类型： num

使VarNo在检查之前以及在保存到Value之前加倍的传感器数值的比例因子。

[\DeadBand]

数据类型： num

如果Varno的数值（由传感器返回）处于+/- DeadBand以内，则不会产生中断。

[\ReportAtTool]

数据类型：switch

此可选变元仅可用于前向预测类型的传感器，例如，光学跟踪传感器。该变元指定不应立即评估变量值，而是应该在机器人的TCP打到位置时进行，即提前预测得到了补偿时。

[\SpeedAdapt]

数据类型：num

\SpeedAdapt 是一个比例因数，用于在 Arc 和 Cap 指令中改变的比例。此参数根据下列条件与 VarNo 的传感器值相乘。

处理速度 = \SpeedAdapt * value(VarNo)

[\APTR]

数据类型：switch

下一页继续

1.126 IVarValue - 下达变量值中断指令

Optical Tracking

续前页

指定变量的订阅应该与该点的跟踪器配对，例如 WeldGuide，在变元 `device` 中指定。

程序执行

在中断下达后的给定时间，自动调用相关的软中断程序。当已经执行该程序时，从出现中断的位置继续进行程序执行。

限制

在没有首先删除中断识别号的相同变量时，不能使用该相同变量五次以上。

> **小心**
>
> 过高的中断频率将停止整个RAPID的执行。

语法

```
IVarValue
  [ device ':=' ] < expression (IN) of string> ','
  [ VarNo ':=' ] < expression (IN) of num > ','
  [ Value ':=' ] < persistent (PERS) of num > ','
  [ Interrupt ':=' ] < variable (VAR) of intnum > ','
  [ '\' Unit ':=' ] < expression (IN) of num > ','
  [ '\' DeadBand ':=' ] < expression (IN) of num > ','
  [ '\' ReportAtTool ] ','
  [ '\' SpeedAdapt ':=' ] < expression (IN) of num > ','
  [ '\' APTR ] ';'
```

相关信息

信息，关于	请参阅
与传感器设备相连	第574页的*SenDevice - 与传感器设备相连*
中断和中断管理概述	技术参考手册 - *RAPID*语言概览
光学跟踪	应用手册 - *Continuous Application Platform*
光学跟踪弧	应用手册 - *Arc and Arc Sensor*

1.127 IWatch - 启用中断

手册用法

IWatch（中断观察）用于启用先前下达指令，但是却通过ISleep停用的中断。

基本示例

以下实例介绍了指令IWatch：

另请参阅第309页的更多示例

例 1

```
IWatch sig1int;
```

先前停用的中断 sig1int得以启用。

变元

```
IWatch Interrupt
```

Interrupt

数据类型：intnum

中断的变量（中断识别号）。

程序执行

重新启用指定类型的中断。忽略ISleep指令生效期间产生的中断。

更多示例

有关于指令IWatch的更多例子阐述如下。

例 1

```
VAR intnum sig1int;
PROC main()
  CONNECT sig1int WITH iroutine1;
  ISignalDI di1,1,sig1int;
  ...
  ISleep sig1int;
  weldpart1;
  IWatch sig1int;
```

在执行weldpart1程序期间，信号di1不允许中断。

错误处理

可能会产生下列可恢复错误。错误可以由错误处理程序处理。系统变量ERRNO将设置为：

名称	错误原因
ERR_UNKINO	尚不知晓中断编号。 不允许关于既未下达，亦未启用的中断的指令。

语法

```
IWatch
  [ Interrupt ':=' ] < variable (VAR) of intnum > ';'
```

下一页继续

1.127 IWatch - 启用中断
RobotWare - OS
续前页

相关信息

信息，关于	请参阅
中断概述	技术参考手册 - *RAPID*语言概览
停用中断	第*302*页的*ISleep* - 停用一个中断

1.128 Label - 线程名称

手册用法

Label用于命名程序中的程序。使用GOTO 指令，该名称随后可用于移动相同程序内的程序执行。

基本示例

以下实例介绍了指令Label：

例 1

```
GOTO next;
...
next:
```

通过next之后指令，继续程序执行。

变元

```
Label:
```

Label

Identifier

期望给出线程的名称。

程序执行

执行该指令时，未出现任何情况。

限制

标记不得与以下内容相同

- 同一程序内的所有其他标记。
- 同一程序内的所有数据名称。

标记会隐藏在其所在程序内具有相同名称的全局数据和程序。

语法

```
<identifier>':'
```

相关信息

信息，关于	请参阅
标识符	技术参考手册 - *RAPID*语言概览
将程序执行移动至标记	第*221*页的*GOTO* - 转到新的指令

1.129 Load - 执行期间，加载普通程序模块

手册用法

Load用于在执行期间，将普通程序模块加载到程序内存中。

将已加载的普通程序模块添加至程序内存中业已存在的模块。

可以用静态（默认）或动态模式加载程序或系统程序模块。

通过指令UnLoad，可卸载静态和动态加载的模块。

静态模式

下表描述了不同的操作如何影响静态加载程序或系统程序模块。

模块类型	从FlexPendant示教器，设置PP到Main。	打开新的RAPID程序
普通程序模块	未受影响	卸载
系统程序模块	未受影响	未受影响

动态模型

下表描述了不同的操作如何影响动态加载程序或系统程序模块。

模块类型	从FlexPendant示教器，设置PP到Main。	打开新的RAPID程序
普通程序模块	卸载	卸载
系统程序模块	卸载	卸载

基本示例

以下实例介绍了指令Load：

另请参阅第*313*页的更多示例

例 1

```
Load \Dynamic, diskhome \File:="PART_A.MOD";
```

将来自diskhome的普通程序模块PART_A.MOD加载到程序内存中。diskhome 是预定义字符串常量"HOME:"。用动态模式来加载普通程序模块。

例 2

```
Load \Dynamic, diskhome \File:="PART_A.MOD";
Load \Dynamic, diskhome \File:="PART_B.MOD" \CheckRef;
```

将普通程序模块PART_A.MOD加载到程序内存中，随后，加载PART_B.MOD。如果PART_A.MOD包含PART_B.MOD的参考，则仅当最后一个模块加载完毕后，方可用\CheckRef来检查未解决的参考。如果\CheckRef用于PART_A.MOD，则将出现链接错误，且将不会加载模块。

变元

```
Load [\Dynamic] FilePath [\File] [\CheckRef]
```

[\Dynamic]

数据类型：switch

开关以动态模式启用模块负载。否则，负载采用静态模式。

下一页继续

FilePath

数据类型：string

将文件路径和文件名称加载到程序内存中。当使用参数\File时，应当排除文件名称。

[\File]

数据类型：string

当参数FilePath 中排除文件名称时，则必须通过该参数来进行定义。

[\CheckRef]

数据类型：switch

在加载模块后，检查程序任务中未解决的参考。如未使用，则不对未解决的参考进行检查。

程序执行

程序执行等待普通程序模块在继续下一个指令之前，完成加载。

如果未使用参数\CheckRef，则针对加载操作，将始终接受未解决的引用，但是在执行未解决的引用期间，其将成为运行时错误。

在普通程序模块加载后，其将连接和初始化。已加载模块的初始化将所有模块等级下的变量设置为其单位值。

如果加载操作出现任何错误，包括使用开关\CheckRef时的未解决参考，则程序内存中已加载的模块将不再有效。

为取得易于理解和维持的良好程序结构，应当从主模块完成普通程序模块的所有加载和卸载，且执行期间，其始终存在于程序内存中。

有关将包含主无返回值程序的程序加载至主程序（通过另一个主无返回值程序）的操作，请参见下文第*313*页的更多示例中的例子。

更多示例

有关于如何使用指令Load的更多例子阐述如下。

更多一般的例子

```
Load \Dynamic, "HOME:/DOORDIR/DOOR1.MOD";
```

将来自路径DOORDIR中的HOME:的普通程序模块DOOR1.MOD加载到程序内存中。以动态模式来加载普通程序模块。

```
Load "HOME:" \File:="DOORDIR/DOOR1.MOD";
```

与上述情况相同，但是采用另一种语法，并且以静态模式加载模块。

```
Load\Dynamic, "HOME:/DOORDIR/DOOR1.MOD";
%"routine_x"%;
UnLoad "HOME:/DOORDIR/DOOR1.MOD";
```

执行（后期绑定）期间，无返回值程序routine_x将得以绑定。

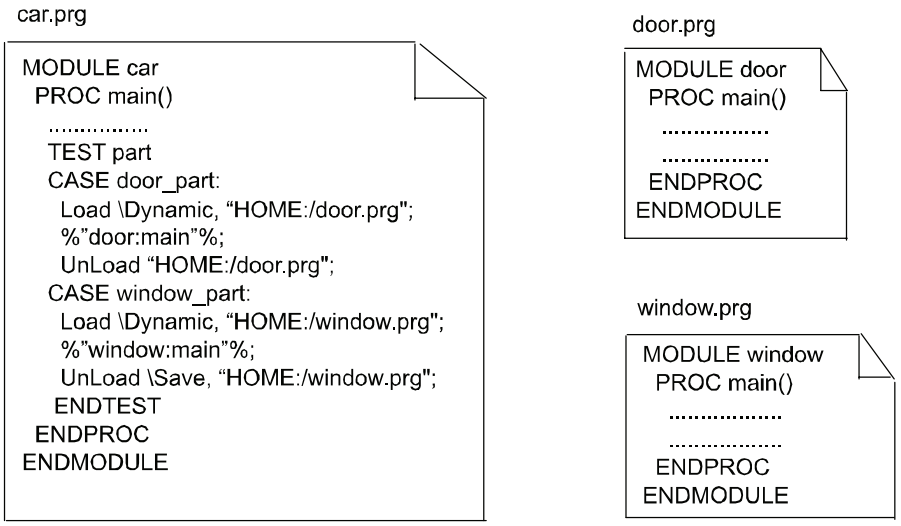
下一页继续

1.129 Load - 执行期间，加载普通程序模块

RobotWare - OS

续前页

加载包含主无返回值程序的程序。



xx0500002104

上述例子表明了如何加载包括main无返回值程序在内的程序。已经单独地开发和测试该程序，随后，运用某种类型的主程序框架，将Load或StartLoad... WaitLoad加载到系统中。在该例子中，car.prg将加载其他程序，即door.prg或window.prg。

在程序car.prg中，将加载位于"HOME:"的door.prg或window.prg。因为在加载之后，系统将door.prg和window.prg中的main无返回值程序视为模块中的**LOCAL**，并按照以下方式进行过程调用：%"door:main"%或%"window: main"%。当你想要访问模块door或模块window中该实例无返回值程序main内其他模块中的**LOCAL**无返回值程序时，使用该语法。

卸载模块以及\Save参数，将再次使main无返回值程序在已保存程序中全面存在。

将模块car或window加载到系统中时，如果将程序指针设置到程序任意部分的main，则程序指针将始终设置到主程序中的全局main无返回值程序，在本例子中则设置到car.prg。

限制

加载期间，避免机械臂不间断地移动。

错误处理

可能会产生下列可恢复错误。错误可以由错误处理程序处理。系统变量ERRNO将设置为：

名称	错误原因
ERR_FILNOTFND	无法发现Load指令中指定的文件。
ERR_IOERROR	读取Load指令中的文件尚存在问题。
ERR_PRGMEMFULL	因为程序内存已满，因此，无法加载模块。
ERR_LOADED	已经将模块加载到程序内存中。
ERR_SYNTAX	已加载的模块包含语法错误。

下一页继续

名称	错误原因
ERR_LINKREF	• 已加载的模块会引起致命的链接错误。 • 如果将Load与开关\CheckRef一同使用，以检查所有参考错误，且程序内存包含未解决的参考。

如果出现一些上述错误，则实际模块将得以卸载，且在ERROR处理器中不可用。

语法

```
Load
  [´\´Dynamic´,´]
  [FilePath´:=´]<expression (IN) of string>
  [´\´File´:=´ <expression (IN) of string>]
  [´\´CheckRef]´;´
```

相关信息

信息，关于	请参阅
卸载普通程序模块	第*843*页的*UnLoad - 执行期间，卸载普通程序模块*
加载同另一程序执行平行的普通程序模块。	第*660*页的*StartLoad - 执行期间，加载普通程序模块* 第*881*页的*WaitLoad - 将加载的模块与任务相连*
检查程序参考	第*97*页的*CheckProgRef - 检查程序参考*

1.130 LoadId - 工具或有效负载的负载识别

手册用法

通过执行用户定义的RAPID程序，LoadId（*Load Identification*）可用于工具（如为roomfix TCP协议，则同时包括机械手工具）或有效负载（通过指令GripLoad启用）的负载识别。

> **注意**
>
> 一种确定工具负载或有效负载的简单方法是使用交互式对话RAPID程序LoadIdentify。可以从目录ProgramEditor/Debug/Call Routine.../LoadIdentify启动该程序。

基本示例

以下实例介绍了指令LoadId：

另请参阅第*319*页的更多示例

例 1

```
VAR bool invalid_pos := TRUE;
VAR jointtarget joints;
VAR bool valid_joints{12};
CONST speeddata low_ori_speed := [20, 5, 20, 5];
VAR bool slow_test_flag := TRUE;
PERS tooldata grip3 := [ TRUE, [[97.4, 0, 223.1], [0.924, 0, 0.383
     ,0]], [0, [0, 0, 0], [1, 0, 0, 0], 0, 0, 0]];
! Check if valid robot type
IF ParIdRobValid(TOOL_LOAD_ID) <> ROB_LOAD_VAL THEN
  EXIT;
ENDIF
! Check if valid robot position
WHILE invalid_pos = TRUE DO
  joints := CJointT();
  IF ParIdPosValid (TOOL_LOAD_ID, joints, valid_joints) = TRUE THEN
    ! Valid position
    invalid_pos := FALSE;
  ELSE
    ! Invalid position
    ! Adjust the position by program movements (horizontal tilt
        house)
    MoveAbsJ joints, low_ori_speed, fine, tool0;
  ENDIF
ENDWHILE
! Do slow test for check of free working area
! Load modules into the system
Load \Dynamic, "RELEASE:/system/mockit.sys";
Load \Dynamic, "RELEASE:/system/mockit1.sys";
IF slow_test_flag = TRUE THEN
  %"LoadId"% TOOL_LOAD_ID, MASS_WITH_AX3, grip3 \SlowTest;
ENDIF
```

下一页继续

```
! Do measurement and update all load data in grip3
%"LoadID"% TOOL_LOAD_ID, MASS_WITH_AX3, grip3;
! Unload modules
UnLoad "RELEASE:/system/mockit.sys";
UnLoad "RELEASE:/system/mockit1.sys";
```

工具grip3.的负载识别

条件

在负载测量以及LoadId之前，应当满足以下条件：

- 确保所有负载均正确地加载在机械臂上
- 检查机械臂类型以及ParIdRobValid是否有效
- 检查位置以及ParIdPosValid是否有效：
  - 轴 3、5和6不要接近其相应的工作范围
  - 倾斜机壳几乎水平，即轴4位于零位置。
- 在运行LoadId之前，应当在系统参数和LoadId的参数中定义以下数据。

  下表阐明了工具的负载识别。

LoadId之前的负载识别模式/已定义的数据	移动TCP，质量已知	移动TCP，质量未知	Roomfix TCP，质量已知	Roomfix TCP，质量未知
上臂负载（系统参数）		定义		定义
工具中的质量	定义		定义	

  下表阐明了有效负载的负载识别。

LoadId之前的负载识别模式/已定义的数据	移动TCP，质量已知	移动TCP，质量未知	Roomfix TCP，质量已知	Roomfix TCP，质量未知
上臂负载（系统参数）		定义		定义
工具中的负载数据	定义	定义	定义	定义
有效负载中的质量	定义		定义	
工具中的工具坐标系	定义	定义		
工件中的用户坐标系			定义	定义
工件中的工件坐标系			定义	定义

- 运行模式和速度覆盖：
  - 采用手动减速模式进行慢速测试
  - 采用自动模式（或手动模式全速）的负载测量，速度覆盖为100%

变元

```
LoadId ParIdType LoadIdType Tool [\PayLoad] [\WObj] [\ConfAngle]
     [\SlowTest] [\Accuracy]
```

下一页继续

1.130 LoadId - 工具或有效负载的负载识别

RobotWare-OS

续前页

ParIdType

数据类型：paridnum

负载识别的类型如下表所述。

值	符号常量	备注
1	TOOL_LOAD_ID	确定工具负载
2	PAY_LOAD_ID	确定有效负载（参考指令GripLoad）

LoadIdType

数据类型：loadidnum

负载识别的类型如下表所述。

值	符号常量	备注
1	MASS_KNOWN	工具或有效负载中各自的已知质量（必须指定规定工具或有效负载中的质量）。
2	MASS_WITH_AX3	工具或有效负载中各自的未知质量。将通过轴3的运动，完成工具或有效负载中质量的识别。

Tool

数据类型： tooldata

关于待确定工具的永久变量。如果参数\PayLoad得以指定，则使用工具的永久变量。

关于工具的负载识别，不得指定以下参数\PayLoad或\WObj 。

[\ PayLoad]

数据类型：loaddata

待确定有效负载的永久变量。

必须始终确定关于有效负载负载识别的选项参数。

[\ WObj]

数据类型： wobjdata

关于使用中工件的永久变量。

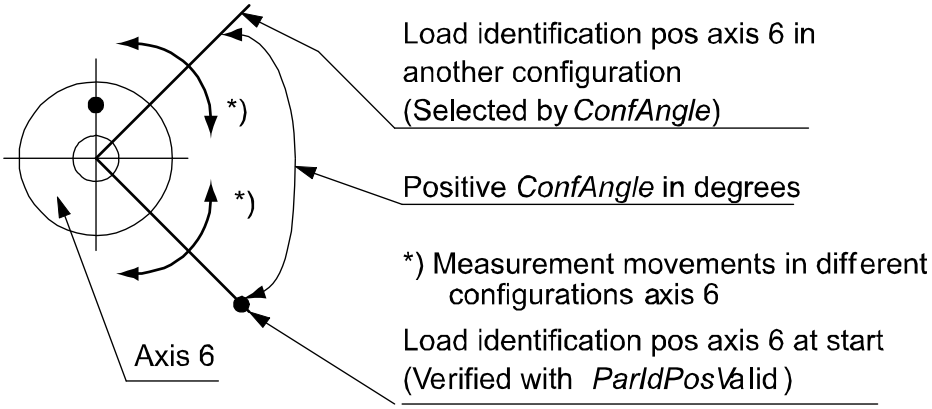
必须始终通过roomfix TCP，确定关于有效负载负载识别的选项参数。

下一页继续

[\ ConfAngle]

数据类型：num

关于特定配置角±度数规格的选项参数，用于参数识别。



xx0500002198

如果未指定该参数，则默认+ 90度。最小为+或-30度。最佳为+或-90度。

[\ SlowTest]

数据类型：switch

选项参数用以确定是否仅应当完成用以检查自由工作区的慢速测试。参见下表：

LoadId ... \SlowTest	仅进行慢速测试
LoadId ...	仅进行测量，并更新工具或有效负载

[\ Accuracy]

数据类型：num

关于计算出的测量精度结果的变量，以占整个负载识别计算结果的百分比表示（100%意味着最大精度）。

程序执行

机械臂将承载大量关于轴5和6且相对较小的运输和测量运动。为识别质量，同时将使轴3运动。

在所有测量、运动和负载计算之后，负载数据返回参数Tool或PayLoad。计算以下负载数据：

- 质量以kg计（如果质量未知，且未受影响）
- 重心x、y、z和力矩轴
- 惯性ix、iy和iz，以kgm计

更多示例

有关于指令LoadId的更多例子阐述如下。

例 1

```
PERS tooldata grip3 := [ FALSE, [[97.4, 0, 223.1], [0.924, 0, 0.383
     ,0]], [6, [10, 10, 100], [0.5, 0.5, 0.5, 0.5], 1.2, 2.7,
     0.5]];
PERS loaddata piece5 := [ 5, [0, 0, 0], [1, 0, 0, 0], 0, 0, 0];
```

下一页继续

1.130 LoadId - 工具或有效负载的负载识别

RobotWare-OS

续前页

```
PERS wobjdata wobj2 := [ TRUE, TRUE, "", [ [34, 0, -45], [0.5,
     -0.5, 0.5 ,-0.5] ], [ [0.56, 10, 68], [0.5, 0.5, 0.5 ,0.5] ]
     ];
VAR num load_accuracy;
! Load modules into the system
Load \Dynamic, "RELEASE:/system/mockit.sys";
Load \Dynamic, "RELEASE:/system/mockit1.sys";
! Do measurement and update all payload data except mass in piece5
%"LoadId"% PAY_LOAD_ID, MASS_KNOWN, grip3 \PayLoad:=piece5
     \WObj:=wobj2 \Accuracy:=load_accuracy;
TPWrite " Load accuracy for piece5 (%) = " \Num:=load_accuracy;
! Unload modules
UnLoad "RELEASE:/system/mockit.sys";
UnLoad "RELEASE:/system/mockit1.sys";
```

安装roomfix TCP时，有效负载piece5以及已知质量的负载识别。

限制

通常，采用服务程序LoadIdentify，完成机械臂的工具或有效负载的负载识别。同时可能通过该RAPID指令LoadId，进行上述识别。在加载或执行程序以及LoadId之前，必须将以下模块加载至系统：

```
Load \Dynamic, "RELEASE:/system/mockit.sys";
Load \Dynamic, "RELEASE:/system/mockit1.sys";
```

随后，可能通过后期绑定调用（参见上文例1）来调用LoadId。

在任意类型停止（例如，程序停止、紧急停止或电影故障）之后，不可能重启负载识别运动。随后，必须从开始进行负载识别运动。

错误处理

可能会产生下列可恢复错误。错误可以由错误处理程序处理。系统变量ERRNO将设置为：

名称	错误原因
ERR_PID_MOVESTOP	执行RAPID NOSTEPIN程序LoadId期间的任意错误。 将程序指针提高到用户调用的LoadId。
ERR_PID_RAISE_PP	
ERR_LOADID_FATAL	

语法

```
LoadId
  [ ParIdType ':=' ] <expression (IN) of paridnum> ',
  [ LoadIdType ':=' ] <expression (IN) of loadidnum> ','
  [ Tool ':=' ] <persistent (PERS) of tooldata>
  [ '\' PayLoad ':=' <persistent (PERS) of loaddata> ]
  [ '\' WObj ':=' <persistent (PERS) of wobjdata> ]
  [ '\' ConfAngle ':=' <expression (IN) of num> ]
  [ '\' SlowTest ]
  [ '\' Accuracy ':=' <variable (VAR) of num> ] ';'
```

下一页继续

相关信息

信息，关于	请参阅
预定义程序Load Identify	操作员手册 - 带 *FlexPendant* 的 *IRC5*
参数识别的类型	第*1444*页的*paridnum - 参数识别的类型*
`ParIdRobValid`的结果	第*1446*页的*paridvalidnum - ParIdRobValid*的结果
负载识别的类型	第*1426*页的*loadidnum - 负载识别的类型*
有效的机械臂类型	第*1167*页的*ParIdRobValid - 用于参数识别的有效机械臂类型*
有效的机械臂位置	第*1164*页的*ParIdPosValid - 用于参数识别的有效机械臂位置*

1.131 MakeDir - 创建新路径

手册用法

MakeDir将用于生成新路径。用户必须具备生成新路径时所处的父路径的编写和执行许可。

基本示例

以下实例介绍了指令MakeDir：

例 1

```
MakeDir "HOME:/newdir";
```

本例子在HOME下，创建名为newdir的新路径：

变元

```
MakeDir Path
```

Path

数据类型：string

新路径的名称指定完整或相关的路径。

错误处理

可能会产生下列可恢复错误。错误可以由错误处理程序处理。系统变量 ERRNO 将设置为：

名称	错误原因
ERR_FILEACC	无法创建路径。

语法

```
MakeDir
[ Path':=' ] < expression (IN) of string>';'
```

相关信息

信息，关于	请参阅
删除路径	第*503*页的*RemoveDir - 删除路径*
重命名文件	第*507*页的*RenameFile - 重命名文件*
删除文件	第*504*页的*RemoveFile - 删除文件*
复制文件	第*126*页的*CopyFile - 复制文件*
检查文件类型	第*1126*页的*IsFile - 检查文件的类型*
检查文件大小	第*1077*页的*FileSize - 检索文件的大小*
检查文件系统大小	第*1082*页的*FSSize - 检索文件系统的大小*
文件和串行通道处理	应用手册 - 控制器软件*IRC5*

1.132 ManLoadIdProc - IRBP机械臂的负载识别

手册用法

ManLoadIdProc（*Manipulator Load Identification Procedure*）通过执行用户定义的RAPID程序，用于外机械臂有效负载的负载识别。

本指令仅可用于主任务T_ROB1，或者如果在*MultiMove*系统中，则可用于运动任务中。

> **注意**
>
> 一种识别有效负载的更为简单的方法是使用服务程序ManLoadIdentify。可以从目录程序编辑器、调试、调用程序、**ManLoadIdentify**开始该服务程序。

基本示例

以下实例介绍了指令ManLoadIdProc：

```
PERS loaddata myload := [6,[0,0,0],[1,0,0,0],0,0,0];
VAR bool defined;
ActUnit STN1;
ManLoadIdProc \ParIdType := IRBP_L
  \MechUnit := STN1
  \PayLoad := myload
  \ConfigAngle := 60
  \AlreadyActive
  \DefinedFlag := defined;
DeactUnit STN1;
```

安装在机械单元STN1上的有效负载myload的负载识别。外机械臂为IRBP-L型。将配置角设置为60度。在负载识别和停用前后，启用机械臂。在更新和定义识别myload之后，将其设置为TRUE。

变元

```
ManLoadIdProc [\ParIdType] [\MechUnit] | [\MechUnitName]
     [\AxisNumber] [\PayLoad] [\ConfigAngle] [\DeactAll] |
     [\AlreadyActive] [DefinedFlag] [DoExit]
```

[\ ParIdType]

数据类型：paridnum

参数识别的类型。发现数据类型paridnum下的预定义常量。

[\ MechUnit]

数据类型：mecunit

用于负载识别的机械单元。无法与参数\MechUnitName一同使用。

[\ MechUnitName]

数据类型：string

用于负载识别的机械单元可作为字符串。无法与参数\MechUnit一同使用。

[\ AxisNumber]

数据类型：num

下一页继续

1.132 ManLoadIdProc - IRBP机械臂的负载识别
RobotWare-OS
续前页

机械单元内的轴编号，其保存有待识别的负载。

[\ PayLoad]

数据类型：loaddata

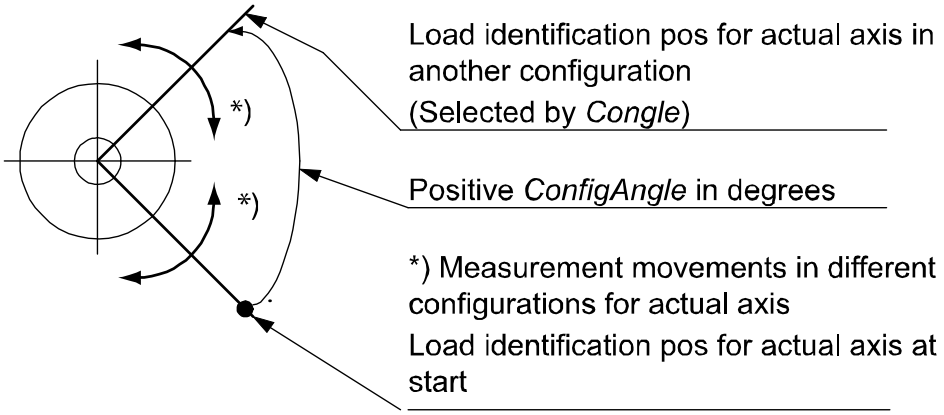
关于待识别有效负载的变量。必须指定分量质量。

在完成识别后，将对该变量进行更新。

[\ ConfigAngle]

数据类型：num

特定配置角±度数的规格将用于参数识别。



xx0500002197

最小+或-30度。最佳+或-90度。

[\ DeactAll]

数据类型：switch

如果使用该开关，则在完成识别之前，系统中的所有机械单元均将停用。随后，将启用用以识别的机械单元。无法与参数\AlreadyActive一同使用。

[\ AlreadyActive]

数据类型：switch

如果用以识别的机械单元有效，则使用该开关。无法与参数\DeactAll一同使用。

[\ DefinedFlag]

数据类型：bool

如果已经完成识别，则将该参数设置为TRUE，否则设置为FALSE。

[\ DoExit]

数据类型：bool

如果设置为TRUE，则将通过EXIT命令来结束负载识别，以强制用户在继续执行之前，设置PP到Main。如果不存在或设置为FALSE，则将不会完成任何EXIT。注意，ManLoadIdProc始终清除当前路径。

程序执行

可选择所有的参数。如果未给出参数，则用户将要求从FlexPendant示教器获得相关值（\DoExit除外）。

下一页继续

将始终要求用户给出质量，且如果机械臂为IRBP R型，则z以mm计。

机械单元将实施大量相对较小的运输和测量运动。

在所有测量、运动和负载计算之后，负载数据返回参数Payload（如使用）。计算以下负载数据。

机械臂类型/计算出的负载数据	IRBP-K	IRBP-L IRBP-C IRBP_T	IRBP-R	IRBP-A IRBP-B IRBP-D
loaddata中的参数PayLoad - cog.x，cog.y，cog.z，以mm计	cog.x cog.y	cog.x cog.y	cog.x cog.y	cog.x cog.y cog.z
loaddata中的参数PayLoad - ix，iy，iz，以kgm^{2}计	iz	iz	ix iy iz	ix iy iz

将在FlexPendant示教器上显示计算出的数据。

限制

通常，通过服务程序ManLoadIdentify，完成外机械臂负载的负载识别。同时可能通过该RAPID指令ManLoadIdProc，进行该识别。

在负载识别之前，将清除进行中的所有路径。如果使用参数\DoExit:=TRUE，则程序指针将在负载识别之后丢失。

在任意类型停止（例如，程序停止、紧急停止或电影故障）之后，不可能重启负载识别运动。必须从开始重启负载识别运动。

错误处理

可能会产生下列可恢复错误。错误可以由错误处理程序处理。系统变量ERRNO将设置为：

名称	错误原因
ERR_PID_MOVESTOP	执行RAPID NOSTEPIN程序ManLoadIdProc期间的任意错误。 将程序指针提高到用户调用的ManLoadIdProc。
ERR_PID_RAISE_PP	
ERR_LOADID_FATAL	

语法

```
ManLoadIdProc
  [ '\'ParIdType ':=' <expression (IN) of paridnum>]
  [ '\'MechUnit ':=' <variable (VAR) of mecunit> ]
  | ['\' MechUnitName ':=' <expression (IN) of string>]
  [ '\' AxisNumber ':=' <expression (IN) of num> ]
  [ '\' PayLoad ':=' <var or pers (INOUT) of loaddata>
  [ '\' ConfigAngle ':=' <expression (IN) of num>]
  [ '\' DeactAll] | [ '\' AlreadyActive]
  [ '\' DefinedFlag ':=' <variable (VAR) of bool> ]
  [ '\' DoExit ':=' <expression (IN) of bool> ] ';'
```

相关信息

信息，关于	请参阅
参数识别的类型	第*1444*页的*paridnum - 参数识别的类型*

下一页继续

1.132 ManLoadIdProc - IRBP机械臂的负载识别

RobotWare-OS

续前页

信息，关于	请参阅
机械单元	第*1428*页的*mecunit - 机械单元*
有效载荷	第*1421*页的*loaddata - 加载数据*

1.133 MechUnitLoad - 确定机械单元的有效负载

手册用法

MechUnitLoad用于确定外机械单元的有效负载。通过指令GripLoad，确定机械臂的有效负载。

应当将该指令用于所有机械单元，并使动态模型（ABB定位器和导轨）伺服，以实现最佳运动性能。

在执行指令ActUnit之后，应当始终执行MechUnitLoad指令。

本指令仅可用于主任务T_ROB1，或者如果在*MultiMove*系统中，则可用于运动任务中。

描述

MechUnitLoad指定机械单元承载的负载。指定的负载用于设置机械单元的动态模型，以便可以以最佳的方式来控制机械单元的运动。

使用指令MechUnitLoad，连接/断开有效负载，该指令在机械单元的重量上增加或减去有效负载的重量。

警告

重要的是，始终定义机械单元（即固定装置和工件）的实际有效负载。负载数据定义不正确可能会导致机械结构过载。

指定不正确的负载数据时，其常常会引起以下后果：

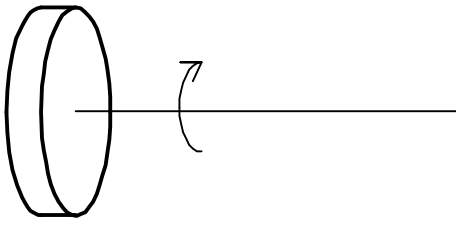
- 机械单元将不会用于其最大容量
- 路径准确性受损，包括过度风险
- 机械结构过载风险

基本示例

以下实例介绍了指令MechUnitLoad：

图示

下图显示了在名为STN1的IRBP L型机械单元上的轴1。



xx0500002142

例 1

```
ActUnit SNT1;
MechUnitLoad STN1, 1, load0;
```

启用机械单元STN1，并定义有效负载load0，其相当于轴1上无负载（完全没有）。

例 2

```
ActUnit STN1;
```

下一页继续

1.133 MechUnitLoad - 确定机械单元的有效负载

RobotWare - OS

续前页

```
MechUnitLoad STN1, 1, fixture1;
```

启用机械单元STN1，并定义有效负载fixture1，其相当于安装在轴1上的固定装置。

例 3

```
ActUnit STN1;
MechUnitLoad STN1, 1, workpiece1;
```

启用机械单元STN1，并定义有效负载workpiece1，其相当于安装在轴1上固定装置和工件。

变元

```
MechUnitLoad MechUnit AxisNo Load
```

MechUnit

Mechanical Unit

数据类型：mecunit

机械单元名称

AxisNo

Axis Number

数据类型：num

用于维持负载的机械单元内的轴编号。轴编号始于1。

Load

数据类型：loaddata

可描述当前待定义有效负载的负载数据，即固定装置或固定装置连同工件取决于工件是否安装在机械单元上。

程序执行

在执行MechUnitLoad之后，当机械臂和其他轴已经停止时，则针对指定机械单元和轴，定义指定负载。这意味着有效负载由控制系统进行控制和监测。

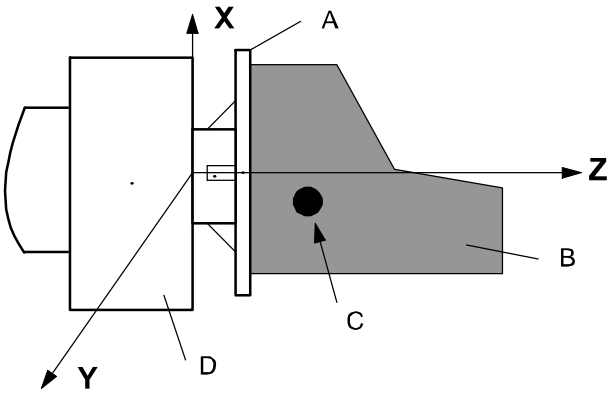
当使用特定机械单元类型的重启模式重置系统，默认有效负载为该机械单元类型的预定义最大有效负载。

当使用另一个有效负载时，应当通过该指令，重新定义机械单元和轴的实际有效负载。在激活机械单元后，应当始终完成上述操作。

已定义的有效负载将经历重启，且亦将在从点动控制窗口手动启用其他机械单元之后，经历程序重启。

下一页继续

以下图表显示了安装在机械单元末端执行器（机械单元的末端执行器坐标系）上的有效负载。



xx0500002143

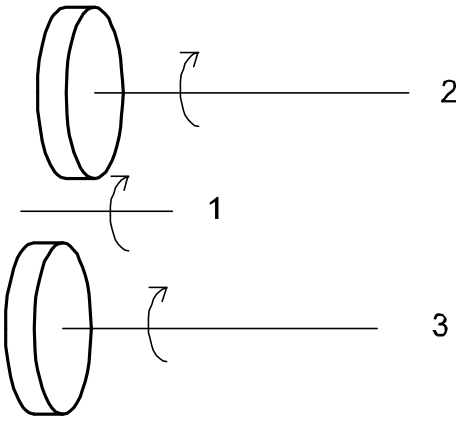
A	末端执行器
B	固定装置和工件
C	有效负载的重心（固定装置+工件）
D	机械单元

更多示例

有关于如何使用指令MechUnitLoad的更多例子阐述如下。

图示

下图显示了名为INTERCH的IRBP K型机械单元以及三个轴（1、2和3）。



xx0500002144

例 1

```
MoveL homeside1, v1000, fine, gun1;
...
ActUnit INTERCH;
```

启用整个机械单元INTERCH 。

例 2

```
MechUnitLoad INTERCH, 2, workpiece1;
```

定义机械单元INTERCH轴2上的有效负载workpiece1。

下一页继续

1.133 MechUnitLoad - 确定机械单元的有效负载

RobotWare - OS

续前页

例 3

```
MechUnitLoad INTERCH, 3, workpiece2;
```

定义机械单元INTERCH轴3上的有效负载workpiece2。

例 4

```
MoveL homeside2, v1000, fine, gun1;
```

机械单元INTERCH 的轴运动至开关位置homeside2，且有效负载施加在轴2和3上。

例 5

```
ActUnit STN1;
MechUnitLoad STN1, 1, workpiece1;
```

启用机械单元STN1。定义机械单元STN1轴1上的有效负载workpiece1。

限制

如果该指令位于移动指令之后，则必须使用停止点（zonedata fine）而非飞越点来编程移动指令，否则，将无法在电源故障后重启。

无法在与任意下列特殊系统事件关联的RAPID程序中执行MechUnitLoad：PowerOn、Stop、QStop, 、Restart或者 Step。

语法

```
MechUnitLoad
  [MechUnit ':=' ] <variable (VAR) of mecunit> ','
  [AxisNo ':=' ] <expression (IN) of num> ','
  [Load ':=' ] <persistent (PERS) of loaddata> ';'
```

相关信息

信息，关于	请参阅
外机械单元有效负载的识别	产品手册 - *IRBP /D2009*
机械单元数据的定义	第*1428*页的*mecunit* - 机械单元
负载数据的定义	第*1421*页的*loaddata* - 加载数据
定义机械臂的有效负载	第*223*页的*GripLoad* - 定义机械臂的有效负载

1.134 MotionProcessModeSet - 设置运动过程模式

手册用法

MotionProcessModeSet用于设置TCP机械臂的运动过程模式（*Motion Process Mode*）。

本指令仅可用于主任务T_ROB1，或者如果在*MultiMove*系统中，则可用于任何运动任务中。

基本示例

下列示例说明了指令MotionProcessModeSet：

```
MotionProcessModeSet OPTIMAL_CYCLE_TIME_MODE;
! Do cycle-time critical movement
..
MotionProcessModeSet ACCURACY_MODE;
! Do cutting with high accuracy
..
```

改变用于运行中TCP机械臂的运动过程模式。

变元

```
MotionProcessModeSet Mode
```

Mode

数据类型：motionprocessmode

待使用的运动过程模式。其为数据类型motionprocessmode的整常数。

程序执行

运行过程模式适用于TCP机械臂，直至执行新的MotionProcessModeSet指令，参见第*332*页的相关信息。

自动设置关于运动过程模式的默认配置值：

- 当使用重启模式重置RAPID时
- 当加载一则新程序或一个新模块时
- 当从起点开始执行程序时
- 当将程序指针移动到main时
- 当将程序指针移动到子程序时
- 移动程序指针造成执行顺序丢失时

预定义数据

预定义数据类型motionprocessmode的以下符号常量，并可将其用于确定参数Mode中的整数。

由题为*Motion*的*Robot*型系统参数*Use Motion Process Mode*所定义的默认模式。

符号常量	常量值	描述
OPTIMAL_CYCLE_TIME_MODE	1	该模式给出了最短的循环时间。
LOW_SPEED_ACCURACY_MODE	2	此模式主要为大机器人提高路径准确度。
LOW_SPEED_STIFF_MODE	3	建议低速接触应用采用该模式，其最大伺服刚度很重要。

下一页继续

1.134 MotionProcessModeSet - 设置运动过程模式

RobotWare - OS

续前页

符号常量	常量值	描述
ACCURACY_MODE	4	此模式主要为小机器人提高路径准确度。
MPM_USER_MODE_1	5	用户定义模式
MPM_USER_MODE_2	6	
MPM_USER_MODE_3	7	
MPM_USER_MODE_4	8	

限制

将当机械臂保持静止时，方可改变模式，否则，将强制执行精点。

语法

```
MotionProcessModeSet
  [ Mode ':=' ] < expression (IN) of motionprocessmode> ';'
```

相关信息

信息，关于	请参阅
Advanced robot motion	应用手册 - 控制器软件*IRC5*
*Motion Process Mode*参数的配置。	技术参考手册 - 系统参数
调节伺服。	第*826*页的*TuneServo* - 调节伺服

1.135 MotionSup - 禁用/启用运动监控

手册用法

MotionSup（*Motion Supervision*）用于在程序执行期间，停用或启用关于机械臂运动的运动监控功能。

本指令仅可用于主任务T_ROB1，或者如果在*MultiMove*系统中，则可用于运动任务中。

描述

运动监控系指一系列关于机械臂高灵敏度且基于模型的监控的功能。其包含关于负载监控、堵塞监控和碰撞检测的功能。因为将监控设计地非常灵敏，因此，如果存在施加于机械臂的较大过程力，则监控可能跳闸。

如果未正确地定义负载，则使用负载识别功能来详细说明。如果较大的外过程力存在于应用中的绝大部分，诸如去毛刺期间，则使用系统参数，以提高运动监控的监控水平，直至不再触发。如果外力仅仅是暂时的，例如，存在于关闭大型点焊枪期间，则应当使用MotionSup指令，以提高对应用中存在干扰的部分的监控水平（或关闭功能）。

基本示例

以下实例介绍了指令MotionSup：

例 1

```
! If the motion supervision is active in the system parameters,
! then it is active by default during program execution
...
! If the motion supervision is deactivated through the system
! parameters,
! then it cannot be activated through the MotionSup instruction
...
! Deactivate motion supervision during program execution
MotionSup \Off;
...
! Activate motion supervision again during program execution
MotionSup \On;
...
! Tune the supervision level to 200% (makes the function less
! sensitive) of the level in
! the system parameters
MotionSup \On \TuneValue:= 200;
...
```

变元

```
MotionSup[\On] | [\Off] [\TuneValue]
```

【 \On 】

数据类型：switch

程序执行期间，启用运动监控功能（如果已经在系统参数中启用）。

下一页继续

1.135 MotionSup - 禁用/启用运动监控

Collision Detection

续前页

【\Off】

数据类型：switch

程序执行期间，停用运动监控功能。

必须指定参数\On或\Off之一。

【\TuneValue】

数据类型：num

调整运动监控灵敏度等级，以占系统参数水平的百分比（1-300%）计。水平越高，表示灵敏度越稳固。该参数仅可与参数\On结合。

程序执行

如果在系统参数和RAPID程序中启用功能运动监控，且由于碰撞而触发运动监控，那么

- 机械臂将尽快停止
- 机械臂将倒退，以消除所有剩余力
- 出现错误消息时，程序执行将停止

如果在系统参数中启用运动监控，则在程序执行期间（TuneValue 100%），运动监控默认启用。自动设置此类值。

- 当使用重启模式重置RAPID时
- 当加载一则新程序或一个新模块时
- 当从起点开始执行程序时
- 当将程序指针移动到main时
- 当将程序指针移动到子程序时
- 移动程序指针造成执行顺序丢失时

限制

当一个或多个接头以独立接头模式运行时，关于外轴的运动监控将不会启用。当以软伺服模式使用机械臂时，可能有必要关闭运动监控，以避免意外跳闸。

语法

```
MotionSup
[ ’\’ On] | [ ’\’ Off ]
[’\’ Tunevalue’:=’< expression (IN) of num> ] ’;´
```

相关信息

信息，关于	请参阅
关于功能的一般描述	技术参考手册 - *RAPID*语言概览
运用系统参数进行调整	技术参考手册 - 系统参数
运动设置数据	第*1430*页的*motsetdata* - 运动设置数据

1.136 MoveAbsJ - 移动机械臂至绝对接头位置

手册用法

MoveAbsJ（*Move Absolute Joint*）用于将机械臂和外轴移动至轴位置中指定的绝对位置。

使用实例：

- 终点是一个奇点
- 关于IRB 6400C的模糊位置，例如，关于机械臂上工具的运动

使用MoveAbsJ运动期间，机械臂的位置不会受到给定工具和工件以及有效程序位移的影响。机械臂运用该数据，以计算负载、TCP速度和拐角路径。可在邻近运动指令中使用相同的工具。

机械臂和外轴沿非线性路径运动至目的位置。所有轴均同时达到目的位置。

本指令仅可用于主任务T_ROB1，或者如果在*MultiMove*系统中，则可用于运动任务中。

基本示例

以下实例介绍了指令MoveAbsJ：

另请参阅第*338*页的更多示例

例 1

```
MoveAbsJ p50, v1000, z50, tool2;
```

通过速度数据v1000和区域数据z50，机械臂以及工具tool2 得以沿非线性路劲运动至绝对轴位置p50。

例 2

```
MoveAbsJ *, v1000\T:=5, fine, grip3;
```

机械臂以及工具grip3沿非线性路径运动至停止点，该停止点储存为指令（标有*）中的绝对轴位置。整个运动耗时5秒。

变元

```
MoveAbsJ [\Conc] ToJointPos [\ID] [\NoEOffs] Speed [\V] | [\T] Zone
    [\Z] [\Inpos] Tool [\WObj] [\TLoad]
```

[\Conc]

Concurrent

数据类型：switch

当机械臂正在运动时，执行后续指令。如果不需要同步，则通常不适用参数，但是，当同外部设备通信时，参数则用于缩短循环时间。

运用参数\Conc，将连续运动指令的数量限制为5。在包含StorePath-RestoPath运动指令以及参数的程序段中，不允许\Conc。

如果省略该参数，且ToJointPos并非停止点，则在机械臂达到编程区之前，执行后续指令一段时间。

不能将该参数用于MultiMove系统中的协调同步移动。

ToJointPos

To Joint Position

下一页继续

1.136 MoveAbsJ - 移动机械臂至绝对接头位置
RobotWare - OS
续前页

数据类型：jointtarget

机械臂和外轴的目的绝对接头位置。其定义为指定位置，或直接储存在指令中（在指令中标有*）。

[\ID]

Synchronization id

数据类型： identno

如果运动同步或协调同步，则参数[\ID]在MultiMove系统中具有强制性。在其他情况下，不允许该参数。指定id编号必须与所有合作程序任务中的编号相同。通过运用id编号，运动不会在进行时混淆。

[\NoEOffs]

No External Offsets

数据类型： switch

如果参数\NoEOffs得以设置，则关于MoveAbsJ的运动将不受外轴有效偏移量的影响。

Speed

数据类型： speeddata

适用于运动的速度数据。速度数据规定了关于工具中心点、工具方位调整和外轴的速率。

[\V]

Velocity

数据类型：num

该参数用于规定指令中TCP的速率，以mm/s计。随后，取代速度数据中指定的相关速率。

[\T]

Time

数据类型：num

该参数用于规定机械臂运动的总时间，以秒计。随后，取代相关的速度数据。

Zone

数据类型：zonedata

相关移动的区域数据。区域数据描述了所生成拐角路径的大小。

[\Z]

Zone

数据类型：num

该参数用于规定指令中机械臂TCP的位置精度。角路径的长度以mm计，其替代区域数据中指定的相关区域。

[\Inpos]

In position

数据类型： stoppointdata

下一页继续

该参数用于规定停止点中机械臂TCP位置的收敛准则。停止点数据取代Zone参数中的指定区域。

Tool

数据类型： tooldata

运动期间使用的工具。

在工具数据中确定TCP的位置以及工具上的负载。TCP位置用于计算运动的速率和角路径。

[\WObj]

Work Object

数据类型： wobjdata

运动期间使用的工件。

如果由机械臂固定工具，则可以省略该参数。如果机械臂固定工件，即工具保持静止，或采用协调的外轴，则必须指定参数。

如果采用固定工具或协调的外轴，则将在工件中确定系统用以计算运动速率和角路径的数据。

[\TLoad]

Total load

数据类型：loaddata

\TLoad主动轴描述了移动中使用的总负载。总负载就是相关的工具负载加上该工具正在处理的有效负载。如果使用了\TLoad自变数，那么就不考虑当前tooldata中的loaddata。

如果\TLoad自变数被设置成load0，那么就不考虑\TLoad自变数，而是以当前tooldata中的loaddata作为代替。

想要使用\TLoad自变数，就必需将系统参数ModalPayLoadMode的数值设置成0。如果将ModalPayLoadMode设置成0，那么就再也无法使用指令GripLoad。

可用服务例程“负载标识”（LoadIdentify）来识别总负载。如果系统参数ModalPayLoadMode被设置成0，且系统正在运行该服务例程，那么操作员便可将相关工具的loaddata复制到一个现有的或新的loaddata永久变量中。

如果使用了关联到系统输入项SimMode（仿真模式）上的一个数字输入信号，那么便可在没有任何有效负载的情况下试运行该程序。如果该数字输入信号被设置成1，那么就不考虑可选自变数\TLoad中的loaddata，而是以当前tooldata中的loaddata作为代替。

注意

处理有效负载的默认功能是使用指令GripLoad，因此系统参数ModalPayLoadMode的默认值为1。

程序执行

MoveAbsJ的运动不受有效程序位移的影响，且如果通过开关\NoEOffs执行，则外轴将不会出现偏移量。未采用开关\NoEOffs时，目的目标中的外轴将受外轴有效偏移量的影响。

下一页继续

1.136 MoveAbsJ - 移动机械臂至绝对接头位置

RobotWare - OS

续前页

通过插入轴角，将工具移动至目的绝对接头位置。这意味着，各轴均以恒定轴速率运动，且所有轴均同时达到目的接头位置，其形成一条非线性路径。

通常，TCP以适当的编程速率运动。重定位工具，并在TCP运动的同时，使外轴移动。如果无法达到重定位或外轴的编程速率，则TCP的速率将会降低。

当运动转移至下一段路径时，通常会产生角路径。如果在区域数据中指定停止点，则仅当机械臂和外轴已达到适当的接头位置时，方才继续程序执行。

更多示例

有关于如何使用指令MoveAbsJ的更多例子阐述如下。

例 1

```
MoveAbsJ *, v2000\V:=2200, z40 \Z:=45, grip3;
```

工具grip3沿非线性路径运动至指令中储存的绝对接头位置。将数据设置为v2000和z40时，开始运动。TCP的速率和区域半径分别为2200 mm/s和45 mm。

例 2

```
MoveAbsJ p5, v2000, fine \Inpos := inpos50, grip3;
```

工具grip3沿非线性路径运动至绝对接头位置p5。当满足关于停止点fine的50%的位置条件和50%的速度条件时，机械臂认为该工具位于点内。其最多等待2秒，以满足各条件。参见数据类型为stoppointdata的预定义数据inpos50。

例 3

```
MoveAbsJ \Conc, *, v2000, z40, grip3;
```

工具grip3沿非线性路径运动至指令中储存的绝对接头位置。当机械臂运动时，执行后续逻辑指令。

例 4

```
MoveAbsJ \Conc, * \NoEOffs, v2000, z40, grip3;
```

与上面相同的运动，但是该运动不受外轴有效偏移量的影响。

例 5

```
GripLoad obj_mass;
MoveAbsJ start, v2000, z40, grip3 \WObj:= obj;
```

机械臂使同固定工具grip3相关的工件obj沿非线性路径运动至绝对轴位置start。

错误处理

可能会产生下列可恢复错误。错误可以由错误处理程序处理。系统变量 ERRNO 将设置为：

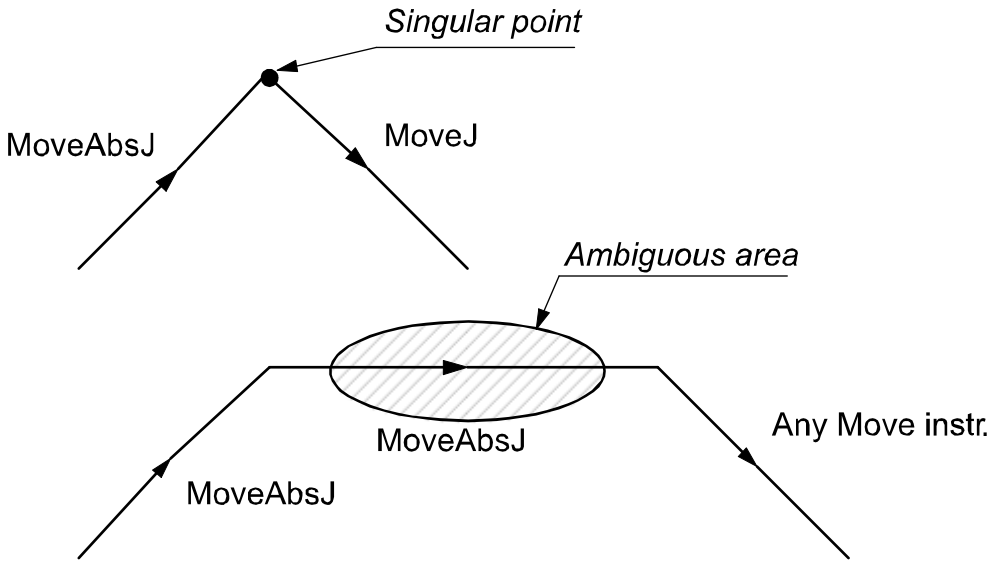
名称	错误原因
ERR_CONC_MAX	已经超过运用参数\Conc的连续运动指令的数量。

下一页继续

限制

为反转涉及的指令MoveAbsJ，并避免关于奇点或模糊区域的问题，后续指令必须满足以下特定要求（参见下图）。

本图显示了关于MoveAbsJ后向执行的限制。



xx0500002201

语法

```
MoveAbsJ
  [ '\' Conc ',' ]
  [ ToJointPos ':=' ] < expression (IN) of jointtarget >
  [ '\' ID ':=' < expression (IN) of identno >]
  [ '\' NoEoffs ] ','
  [ Speed ':=' ] < expression (IN) of speeddata >
  [ '\' V ':=' < expression (IN) of num > ]
  | [ '\' T ':=' < expression (IN) of num > ] ','
  [Zone ':=' ] < expression (IN) of zonedata>
  ['\' Z ':=' ] < expression (IN) of num >
  [ '\' Inpos' :=' < expression (IN) of stoppointdata > ] ','
  [ Tool ':=' ] < persistent (PERS) of tooldata >
  [ '\' WObj ':=' < persistent (PERS) of wobjdata > ]
  [ '\' TLoad ':=' < persistent (PERS) of loaddata > ] ';'
```

相关信息

信息，关于	请参阅
其他定位指令	技术参考手册 - *RAPID*语言概览
接头位置的定义	第*1418*页的*jointtarget* - 接头位置数据
载荷的定义	第*1421*页的*loaddata* - 加载数据
速度的定义	第*1480*页的*speeddata* - 速度数据
停止点数据的定义	第*1483*页的*stoppointdata* - 停止点数据
工具的定义	第*1502*页的*tooldata* - 工具数据

下一页继续

1.136 MoveAbsJ - 移动机械臂至绝对接头位置

RobotWare - OS

续前页

信息，关于	请参阅
工件的定义	第*1523*页的*wobjdata - 工件数据*
区域数据的定义	第*1531*页的*zonedata - 区域数据*
一般动作	技术参考手册 - *RAPID*语言概览
同时进行的程序执行	技术参考手册 - *RAPID*语言概览
关于如何使用TLoad总负载的例子。	第*388*页的*MoveL - 使机械臂沿直线移动*
定义机器人的载荷	第*223*页的*GripLoad - 定义机械臂的有效负载*
LoadIdentify，载荷识别服务例行程序	操作员手册 - 带 *FlexPendant* 的 *IRC5*
用于在仿真模式下无载荷运行机器人的系统输入信号 *SimMode*。 （主题 I/O，类型 System Input，行动值，*SimMode*）	技术参考手册 - 系统参数
激活和停用载荷的系统参数 *ModalPayLoadMode*。 （主题 Controller，类型 System Misc，行动值，*ModalPayLoadMode*）	技术参考手册 - 系统参数

1.137 MoveC - 使机械臂沿圆周移动

手册用法

MoveC用于将工具中心点（TCP）沿圆周移动至给定目的地。移动期间，该周期的方位通常相对保持不变。

本指令仅可用于主任务T_ROB1，或者如果在*MultiMove*系统中，则可用于运动任务中。

基本示例

以下实例介绍了指令MoveC：

另请参阅第*344*页的更多示例

例 1

```
MoveC p1, p2, v500, z30, tool2;
```

工具的TCPtool2沿圆周移动至位置p2，其速度数据为v500 且区域数据为z30。根据起始位置、圆周点p1和目的点p2，确定该循环。

例 2

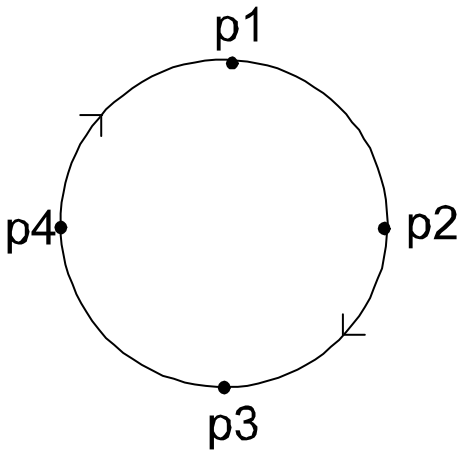
```
MoveC *, *, v500 \T:=5, fine, grip3;
```

工具的TCPgrip3沿圆周移动至指令中储存的精点（标有第二个*）。同时将圆周点储存在指令中（标有第一个*）。完整的运动耗时5秒。

例 3

```
MoveL p1, v500, fine, tool1;
MoveC p2, p3, v500, z20, tool1;
MoveC p4, p1, v500, fine, tool1;
```

本图显示了如何通过两个MoveC 指令，实施一个完整的周期。



xx0500002212

变元

```
MoveC [\Conc] CirPoint ToPoint [\ID] Speed [\V] | [\T] Zone [\Z]
     [\Inpos] Tool [\WObj] [\Corr] [\TLoad]
```

[\Conc]

Concurrent

下一页继续

1.137 MoveC - 使机械臂沿圆周移动
RobotWare - OS
续前页

数据类型：switch

当机械臂正在运动时，执行后续指令。通常不使用参数，但是，当使用飞越点时，可使用参数以避免由过载CPU所引起的多余停止。当编程点在高速度下极为接近时，该参数适用。当不需要同外设备与机械臂运动之间的外设备和同步进行通信时，参数亦适用。

运用参数\Conc，将连续运动指令的数量限制为5。在包含StorePath-RestoPath运动指令以及参数的程序段中，不允许\Conc。

如果省略该参数，且ToPoint并非停止点，则在机械臂达到编程区之前，执行后续指令一段时间。

不能将该参数用于MultiMove系统中的协调同步移动。

CirPoint

数据类型：robtarget

相关机器人的圆弧点。圆弧点是指相关起点与终点间的圆弧上的某个位置。若要获得最好的准确度，则宜把该点放在相关起点与终点的正中间处。如果该点太靠近起点或终点，那么相关机器人就可能发出一条警告。将圆弧点定义为一个已命名的位置，或将其直接保存在相关指令（在指令中用一个*标注）中。勿使用外轴的这一位置。

ToPoint

数据类型：robtarget

机器人和外部轴的目标点。定义为已命名的位置或直接存储在指令中（在指令中加 * 标记）。

[\ID]

Synchronization id

数据类型： identno

如果运动同步或协调同步，则参数[\ID]在MultiMove系统中具有强制性。在其他情况下，不允许该参数。指定id编号必须与所有合作程序任务中的编号相同。通过运用id编号，运动不会在进行时混淆。

Speed

数据类型： speeddata

适用于移动的速度数据。速度数据定义了相关TCP的速度、工具的重定方位和外轴。

[\V]

Velocity

数据类型：num

该参数用于规定指令中TCP的速率，以mm/s计。随后，取代速度数据中指定的相关速率。

[\T]

Time

数据类型：num

该参数用于规定机械臂和外轴运动期间的总时间，以秒计。随后，取代相关的速度数据。

下一页继续

Zone

数据类型：zonedata

相关移动的区域数据。区域数据描述了所生成拐角路径的大小。

[\Z]

Zone

数据类型：num

该参数用于规定指令中机械臂TCP的位置精度。角路径的长度以mm计，其替代区域数据中指定的相关区域。

[\Inpos]

In position

数据类型：stoppoint data

该参数用于规定停止点中机械臂TCP位置的收敛准则。停止点数据取代Zone参数中的指定区域。

Tool

数据类型： tooldata

相关机器人移动时使用的工具。工具中心点是将移过去的指定目标点。

[\WObj]

Work Object

数据类型： wobjdata

该工件（对象坐标系）与相关指令中的机器人位置相关联。

用户可以忽略该自变数，且如果忽略了该自变数，那么相关位置就会与全局坐标系关联起来。另一方面，如果使用了一个固定TCP或若干协同外轴，那么就必须为一个圆圈（相对于待执行工件的圆圈）指定该自变数。

[\Corr]

Correction

数据类型：switch

如果存在该参数，则将通过指令CorrWrite而写入修正条目的修正数据添加到路径和目的位置。

[\TLoad]

Total load

数据类型：loaddata

\TLoad主动轴描述了移动中使用的总负载。总负载就是相关的工具负载加上该工具正在处理的有效负载。如果使用了\TLoad自变数，那么就不考虑当前tooldata中的loaddata。

如果\TLoad自变数被设置成load0，那么就不考虑\TLoad自变数，而是以当前tooldata中的loaddata作为代替。

想要使用\TLoad自变数，就必需将系统参数ModalPayLoadMode的数值设置成0。如果将ModalPayLoadMode设置成0，那么就再也无法使用指令GripLoad。

下一页继续

1.137 MoveC - 使机械臂沿圆周移动
RobotWare - OS
续前页

可用服务例程“负载标识”（LoadIdentify）来识别总负载。如果系统参数ModalPayLoadMode被设置成0，且系统正在运行该服务例程，那么操作员便可将相关工具的loaddata复制到一个现有的或新的loaddata永久变量中。

如果使用了关联到系统输入项SimMode（仿真模式）上的一个数字输入信号，那么便可在没有任何有效负载的情况下试运行该程序。如果该数字输入信号被设置成1，那么就不考虑可选自变数\TLoad中的loaddata，而是以当前tooldata中的loaddata作为代替。

注意

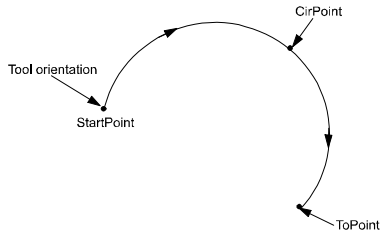
处理有效负载的默认功能是使用指令GripLoad，因此系统参数ModalPayLoadMode的默认值为1。

程序执行

将机械臂和外部单元移动至以下目的点：

- 以恒定编程速率，沿圆周移动工具的TCP。
- 以恒定速率，将工具从起始位置的方位调整为目的点处的方位。
- 调整同圆周路径相关的姿态。因此，如果路径起点和目的点的相关姿态相同，则在运动期间，相关姿态保持不变（参见下图）。

本图表明了圆周运动期间的工具方位。



xx0500002214

未达到圆周点中的姿态。仅用于区分姿态调整的两种可能姿态。沿路径调整姿态的准确性仅取决于起点和目的点处的姿态。

在指令CirPathMode中描述了圆周路径期间，关于工具方位的不同模式。

以恒定速率执行不协调的外轴，从而使其与机械臂轴同时到达目的点。未使用循环位置中的位置。

如果不可能达到关于调整姿态或外轴的编程速率，则将降低TCP的速率。

当运动转移至下一段路径时，通常会产生角路径。如果在区域数据中指定停止点，则仅当机械臂和外轴已达到适当的位置时，方才继续程序执行。

更多示例

有关于如何使用指令MoveC的更多例子阐述如下。

例 1

```
MoveC *, *, v500 \V:=550, z40 \Z:=45, grip3;
```

工具的TCPgrip3沿圆周运动至指令中储存的位置。将数据设置为v500和 z40时，开始运动。TCP的速率和区域半径分别为550 mm/s和45 mm。

下一页继续

例 2

```
MoveC p5, p6, v2000, fine \Inpos := inpos50, grip3;
```

工具的TCPgrip3沿圆周运动至停止点p6。当满足关于停止点fine的50%的位置条件和50%的速度条件时，机械臂认为该工具位于点内。其最多等待2秒，以满足各条件。参见数据类型为stoppointdata.的预定义数据inpos50。

例 3

```
MoveC \Conc, *, *, v500, z40, grip3;
```

工具的TCPgrip3沿圆周运动至指令中储存的位置。圆周点亦储存在指令中。当机械臂运动时，执行后续逻辑指令。

例 4

```
MoveC cir1, p15, v500, z40, grip3 \WObj:=fixture;
```

工具的TCPgrip3通过圆周点cir1，沿圆周运动至位置p15。在关于fixture的工件坐标系中指定了此类点。

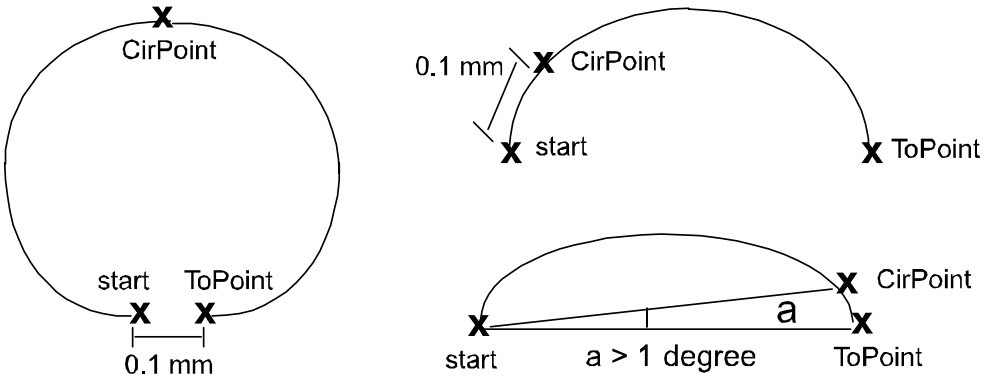
错误处理

如果已经超出了使用参数\Conc的连续运动指令数，则将系统变量ERRNO设置为ERR_CONC_MAX。

可用错误处理器来处理该错误。

限制

关于如何放置CirPoint和ToPoint，存在一些限制，如下图所示。



xx0500002213

- 起点与ToPoint之间的最小距离为0.1 mm
- 起点与CirPoint之间的最小距离为0.1 mm
- 起点的CirPoint与ToPoint之间的最小角度为1度

精度接近低限，例如，如果圆周上的起点和ToPoint相互接近，则圆周倾斜所引起的错误会远高于已编程各点的精度。

确保机械臂可在程序执行期间达到圆周点，并划分圆周运动顺序（如必要）。

下一页继续

1.137 MoveC - 使机械臂沿圆周移动

RobotWare - OS

续前页

当不允许机械臂在圆周路径上停止时，将执行模式由向前改变为向后或相反，且将产生错误消息。

> **警告**
>
> 指令MoveC （或包括圆周运动在内的所有其他指令）不得始于圆周点和终点之间的TCP起点。否则，机械臂将不会采用编程路径（以不同于编程方向的另一方向，围绕圆周路径进行定位）。
>
> 为将风险降至最低，需将系统参数*Restrict placing of circlepoints*设置为TRUE（类型为*Motion Planner*，题为*Motion*）。参数增加了对未转动240度以上的圆周路径以及位于圆周路径中部的圆周点的监控。

语法

```
MoveC
  [ '\' Conc ',' ]
  [ CirPoint ':=' ] < expression (IN) of robtarget> ','
  [ ToPoint ':=' ] < expression (IN) of robtarget> ','
  [ '\' ID ':=' < expression (IN) of identno>] ','
  [ Speed ':=' ] < expression (IN) of speeddata>
  [ '\' V ':=' < expression (IN) of num> ]
  | [ '\' T ':=' < expression (IN) of num> ] ','
  [ Zone ':=' ] < expression (IN) of zonedata>
  [ '\' Z ':=' < expression (IN) of num> ]
  [ '\' Inpos ':=' < expression (IN) of stoppointdata> ] ','
  [ Tool ':=' ] < persistent (PERS) of tooldata>
  [ '\' WObj ':=' < persistent (PERS) of wobjdata> ]
  [ '\' Corr ]
  [ '\' TLoad ':=' < persistent (PERS) of loaddata > ] ';'
```

相关信息

信息，关于	请参阅
其他定位指令	技术参考手册 - *RAPID*语言概览
载荷的定义	第*1421*页的*loaddata - 加载数据*
速度的定义	第*1480*页的*speeddata - 速度数据*
停止点数据的定义	第*1483*页的*stoppointdata - 停止点数据*
工具的定义	第*1502*页的*tooldata - 工具数据*
工件的定义	第*1523*页的*wobjdata - 工件数据*
区域数据的定义	第*1531*页的*zonedata - 区域数据*
写入纠正条目	第*137*页的*CorrWrite - 写入修正发电机*
圆周路径期间的工具方位调整	第*98*页的*CirPathMode - 圆周路径期间的工具方位调整*
一般动作	技术参考手册 - *RAPID*语言概览
坐标系	技术参考手册 - *RAPID*语言概览
同时进行的程序执行	技术参考手册 - *RAPID*语言概览
系统参数	技术参考手册 - 系统参数

下一页继续

信息，关于	请参阅
关于如何使用TLoad总负载的例子。	第*388*页的*MoveL - 使机械臂沿直线移动*
定义机器人的载荷	第*223*页的*GripLoad - 定义机械臂的有效负载*
LoadIdentify，载荷识别服务例行程序	操作员手册 - 带 *FlexPendant* 的 *IRC5*
用于在仿真模式下无载荷运行机器人的系统输入信号 *SimMode*。 （主题 I/O，类型 System Input，行动值，*SimMode*）	技术参考手册 - 系统参数
激活和停用载荷的系统参数 *ModalPayLoadMode*。 （主题 Controller，类型 System Misc，行动值，*ModalPayLoadMode*）	技术参考手册 - 系统参数

1.138 MoveCAO - 使机械臂沿圆周运动，设置拐角处的模拟信号输出

手册用法

MoveCAO（*Move Circular Analog Output*）用于将工具中心点移动至给定目的地。将指定的模拟信号输出设置在目的点角路径中部。运动期间，相对于圆周，姿态通常保持不变。

本指令仅可用于主任务T_ROB1，或者如果在*MultiMove*系统中，则可用于运动任务中。

基本示例

以下实例介绍了指令MoveCAO：

例 1

```
MoveCAO p1, p2, v500, z30, tool2, ao1, 1.1;
```

工具的TCPtool2沿圆周运动至位置p2，其速度数据为v500 ，区域数据为z30。根据起点位置、圆周点p1和目的点p2来定义圆周。将输出ao1设置在p2处角路径中部。

变元

```
MoveCAO CirPoint ToPoint [\ID] Speed [\T] Zone Tool [\WObj] Signal
    Value [\TLoad]
```

CirPoint

数据类型：robtarget

机械臂的圆周点。圆周点位于起点和目的点之间的圆周上。为获得最佳精度，应当将其放置在起点和目的点之间大约中间的位置。如果放置位置过于接近起点和目的点，则机械臂可能会发出警告。将圆周点定义为指定位置，或直接储存在指令中（在指令中标记有*）。未使用外轴的位置。

ToPoint

数据类型：robtarget

机器人和外部轴的目标点。定义为已命名的位置或直接存储在指令中（在指令中加 * 标记）。

[\ID]

Synchronization id

数据类型：identno

如果运动同步或协调同步，则参数[\ID]在MultiMove系统中具有强制性。在其他情况下，不允许该参数。指定id编号必须与所有合作程序任务中的编号相同。通过运用id编号，运动不会在进行时混淆。

Speed

数据类型：speeddata

适用于移动的速度数据。速度数据定义了相关TCP的速度、工具的重定方位和外轴。

[\T]

Time

数据类型：num

下一页继续

该参数用于规定机械臂和外轴运动期间的总时间，以秒计。随后，取代相关的速度数据。

Zone

数据类型： zonedata

相关移动的区域数据。区域数据描述了所生成拐角路径的大小。

Tool

数据类型： tooldata

相关机器人移动时使用的工具。工具中心点是将移过去的指定目标点。

[\WObj]

Work Object

数据类型： wobjdata

该工件（对象坐标系）与相关指令中的机器人位置相关联。

可省略该参数，随后，位置与世界坐标系相关。另一方面，如果使用固定式TCP或协调的外轴，则必须指定该参数，从而执行与工件相关的圆周。

Signal

数据类型：signalao

待改变模拟信号输出信号的名称。

Value

数据类型：num

信号的期望值。

[\TLoad]

Total load

数据类型：loaddata

\TLoad主动轴描述了移动中使用的总负载。总负载就是相关的工具负载加上该工具正在处理的有效负载。如果使用了\TLoad自变数，那么就不考虑当前tooldata中的loaddata。

如果\TLoad自变数被设置成load0，那么就不考虑\TLoad自变数，而是以当前tooldata中的loaddata作为代替。

想要使用\TLoad自变数，就必需将系统参数ModalPayLoadMode的数值设置成0。如果将ModalPayLoadMode设置成0，那么就再也无法使用指令GripLoad。

可用服务例程“负载标识”（LoadIdentify）来识别总负载。如果系统参数ModalPayLoadMode被设置成0，且系统正在运行该服务例程，那么操作员便可将相关工具的loaddata复制到一个现有的或新的loaddata永久变量中。

如果使用了关联到系统输入项SimMode（仿真模式）上的一个数字输入信号，那么便可在没有任何有效负载的情况下试运行该程序。如果该数字输入信号被设置成1，那么

下一页继续

1.138 MoveCAO - 使机械臂沿圆周运动，设置拐角处的模拟信号输出
RobotWare - OS
续前页

就不考虑可选自变数\TLoad中的loaddata，而是以当前tooldata中的loaddata作为代替。

> **注意**
>
> 处理有效负载的默认功能是使用指令GripLoad，因此系统参数ModalPayLoadMode的默认值为1。

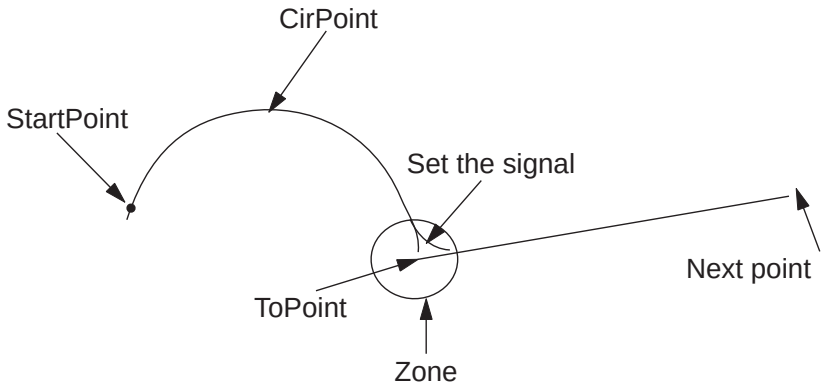
程序执行

更多关于圆周运动的信息，请参见指令MoveC，第341页的*MoveC - 使机械臂沿圆周移动*。

将模拟信号输出信号设置在飞焊点角路径的中间位置，如下图所示。

本图显示了MoveCAO角路径中模拟信号输出信号的设置。

```
MoveCAO p2, p2, v500, z30, tool2, ao1, 1.1;
```



xx1400001116

关于停止点，我们建议使用关于MoveC和SetAO的“普通”编程顺序。但是，当使用指令MoveCAO中的停止点时，模拟信号输出信号得以在机械臂达到停止点时设置。

以连续执行模式逐步向前而非逐步向后地设置指定的I/O信号。

限制

符合指令MoveC的一般限制，参见第341页的*MoveC - 使机械臂沿圆周移动*。

错误处理

系统会生成下列可恢复错误，并在错误处理器中处理这些错误。系统变量ERRNO将被设置成：

如果关于指定模拟信号输出信号Signal的编程Value参数超出限值，则ERR_AO_LIM。

如果信号变量是RAPID中声明的变量，则ERR_NO_ALIASIO_DEF。尚未同I/O配置以及指令AliasIO中确定的I/O信号相连。

如果与I/O单元无接触，则ERR_NORUNUNIT。

如果无法访问I/O信号（仅对ICI现场总线有效），则ERR_SIG_NOT_VALID。

语法

```
MoveCAO
  [ CirPoint ':=' ] < expression (IN) of robtarget > ','
```

下一页继续

```
[ ToPoint ':=' ] < expression (IN) of robtarget > ','
[ '\' ID ':=' < expression (IN) of identno >]','
[ Speed ':=' ] < expression (IN) of speeddata >
[ '\' T ':=' < expression (IN) of num > ] ','
[ Zone ':=' ] < expression (IN) of zonedata > ','
[ Tool ':=' ] < persistent (PERS) of tooldata >
[ '\' WObj ':=' < persistent (PERS) of wobjdata > ] ','
[ Signal ':=' ] < variable (VAR) of signalao > ] ','
[ Value ':=' ] < expression (IN) of num > ]
[ '\' TLoad ':=' < persistent (PERS) of loaddata > ] ';'
```

相关信息

信息，关于	请参阅
其他定位指令	技术参考手册 - *RAPID*语言概览
使机械臂沿圆周移动	第*341*页的*MoveC - 使机械臂沿圆周移动*
载荷的定义	第*1421*页的*loaddata - 加载数据*
速度的定义	第*1480*页的*speeddata - 速度数据*
工具的定义	第*1502*页的*tooldata - 工具数据*
工件的定义	第*1523*页的*wobjdata - 工件数据*
区域数据的定义	第*1531*页的*zonedata - 区域数据*
一般动作	技术参考手册 - *RAPID*语言概览
坐标系	技术参考手册 - *RAPID*语言概览
关于I/O设置的运动	技术参考手册 - *RAPID*语言概览
关于如何使用TLoad总负载的例子。	第*388*页的*MoveL - 使机械臂沿直线移动*
定义机器人的载荷	第*223*页的*GripLoad - 定义机械臂的有效负载*
LoadIdentify，载荷识别服务例行程序	操作员手册 - 带 *FlexPendant* 的 *IRC5*
用于在仿真模式下无载荷运行机器人的系统输入信号 *SimMode*。 （题为I/O System，类型为System Input，行动值为*SimMode*）	技术参考手册 - 系统参数
激活和停用载荷的系统参数 *ModalPayLoadMode*。 （主题 Controller，类型 System Misc，行动值，*ModalPayLoadMode*）	技术参考手册 - 系统参数

1.139 MoveCDO - 使机械臂沿圆周运动，设置拐角处的数字信号输出

手册用法

MoveCDO（*Move Circular Digital Output*）用于将工具中心点移动至给定目的地。将指定的数字信号输出设置/重置在目的点角路径中部。运动期间，相对于圆周，姿态通常保持不变。

本指令仅可用于主任务T_ROB1，或者如果在*MultiMove*系统中，则可用于运动任务中。

基本示例

以下实例介绍了指令MoveCDO：

例 1

```
MoveCDO p1, p2, v500, z30, tool2, do1,1;
```

工具的TCPtool2沿圆周运动至位置p2，其速度数据为v500 ，区域数据为z30。根据起点位置、圆周点p1和目的点p2来定义圆周。将输出do1设置在p2处角路径中部。

变元

```
MoveCDO CirPoint ToPoint [\ID] Speed [\T] Zone Tool [\WObj] Signal
    Value [\TLoad]
```

CirPoint

数据类型：robtarget

机械臂的圆周点。圆周点位于起点和目的点之间的圆周上。为获得最佳精度，应当将其放置在起点和目的点之间大约中间的位置。如果放置位置过于接近起点和目的点，则机械臂可能会发出警告。将圆周点定义为指定位置，或直接储存在指令中（在指令中标记有*）。未使用外轴的位置。

ToPoint

数据类型：robtarget

机器人和外部轴的目标点。定义为已命名的位置或直接存储在指令中（在指令中加 * 标记）。

[\ID]

Synchronization id

数据类型： identno

如果运动同步或协调同步，则参数[\ID]在MultiMove系统中具有强制性。在其他情况下，不允许该参数。指定id编号必须与所有合作程序任务中的编号相同。通过运用id编号，运动不会在进行时混淆。

Speed

数据类型： speeddata

适用于移动的速度数据。速度数据定义了相关TCP的速度、工具的重定方位和外轴。

[\T]

Time

数据类型：num

下一页继续

该参数用于规定机械臂和外轴运动期间的总时间，以秒计。随后，取代相关的速度数据。

Zone

数据类型：zonedata

相关移动的区域数据。区域数据描述了所生成拐角路径的大小。

Tool

数据类型： tooldata

相关机器人移动时使用的工具。工具中心点是将移过去的指定目标点。

[\WObj]

Work Object

数据类型： wobjdata

该工件（对象坐标系）与相关指令中的机器人位置相关联。

可省略该参数，随后，位置与世界坐标系相关。另一方面，如果使用固定式TCP或协调的外轴，则必须指定该参数，从而执行与工件相关的圆周。

Signal

数据类型：signaldo

待改变数字信号输出信号的名称。

Value

数据类型：dionum

信号的期望值（0或1）。

[\TLoad]

Total load

数据类型：loaddata

\TLoad主动轴描述了移动中使用的总负载。总负载就是相关的工具负载加上该工具正在处理的有效负载。如果使用了\TLoad自变数，那么就不考虑当前tooldata中的loaddata。

如果\TLoad自变数被设置成load0，那么就不考虑\TLoad自变数，而是以当前tooldata中的loaddata作为代替。

想要使用\TLoad自变数，就必需将系统参数ModalPayLoadMode的数值设置成0。如果将ModalPayLoadMode设置成0，那么就再也无法使用指令GripLoad。

可用服务例程“负载标识”（LoadIdentify）来识别总负载。如果系统参数ModalPayLoadMode被设置成0，且系统正在运行该服务例程，那么操作员便可将相关工具的loaddata复制到一个现有的或新的loaddata永久变量中。

如果使用了关联到系统输入项SimMode（仿真模式）上的一个数字输入信号，那么便可在没有任何有效负载的情况下试运行该程序。如果该数字输入信号被设置成1，那么

下一页继续

1.139 MoveCDO - 使机械臂沿圆周运动，设置拐角处的数字信号输出
RobotWare - OS
续前页

就不考虑可选自变数\TLoad中的loaddata，而是以当前tooldata中的loaddata作为代替。

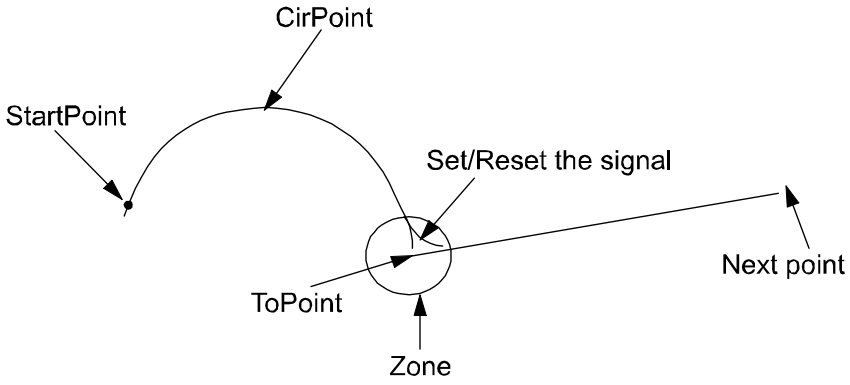
> **注意**
>
> 处理有效负载的默认功能是使用指令GripLoad，因此系统参数ModalPayLoadMode的默认值为1。

程序执行

更多关于圆周运动的信息，请参见指令MoveC。

将数字信号输出信号设置/重置在飞焊点角路径的中间位置，如下图所示。

本图显示了MoveCDO角路径中数字信号输出信号的设置/重置。



xx0500002215

关于停止点，我们建议使用关于MoveC+SetDO的“普通”编程顺序。但是，当使用指令MoveCDO中的停止点时，数字信号输出信号得以在机械臂达到停止点时设置/重置。

以连续执行模式逐步向前而非逐步向后地设置/重置指定的I/O信号。

限制

符合指令MoveC.的一般限制

错误处理

系统会生成下列可恢复错误，并在错误处理器中处理这些错误。系统变量ERRNO将被设置成：

如果信号变量是RAPID中声明的变量，则ERR_NO_ALIASIO_DEF。尚未同I/O配置以及指令AliasIO中确定的I/O信号相连。

如果与I/O单元无接触，则ERR_NORUNUNIT。

如果无法访问I/O信号（仅对ICI现场总线有效），则ERR_SIG_NOT_VALID。

语法

```
MoveCDO
  [ CirPoint ':=' ] < expression (IN) of robtarget > ','
  [ ToPoint ':=' ] < expression (IN) of robtarget > ','
  [ '\' ID ':=' < expression (IN) of identno >]','
  [ Speed ':=' ] < expression (IN) of speeddata >
```

下一页继续

```
[ '\' T ':=' < expression (IN) of num > ] ','
[ Zone ':=' ] < expression (IN) of zonedata > ','
[ Tool ':=' ] < persistent (PERS) of tooldata >
[ '\' WObj ':=' < persistent (PERS) of wobjdata > ] ','
[ Signal ':=' ] < variable (VAR) of signaldo > ] ','
[ Value ':=' ] < expression (IN) of dionum > ]
[ '\' TLoad ':=' < persistent (PERS) of loaddata > ] ';'
```

相关信息

信息，关于	请参阅
其他定位指令	技术参考手册 - *RAPID*语言概览
使机械臂沿圆周移动	第*341*页的*MoveC - 使机械臂沿圆周移动*
载荷的定义	第*1421*页的*loaddata - 加载数据*
速度的定义	第*1480*页的*speeddata - 速度数据*
工具的定义	第*1502*页的*tooldata - 工具数据*
工件的定义	第*1523*页的*wobjdata - 工件数据*
区域数据的定义	第*1531*页的*zonedata - 区域数据*
一般动作	技术参考手册 - *RAPID*语言概览
坐标系	技术参考手册 - *RAPID*语言概览
关于I/O设置的运动	技术参考手册 - *RAPID*语言概览
关于如何使用TLoad总负载的例子。	第*388*页的*MoveL - 使机械臂沿直线移动*
定义机器人的载荷	第*223*页的*GripLoad - 定义机械臂的有效负载*
LoadIdentify，载荷识别服务例行程序	操作员手册 - 带 *FlexPendant* 的 *IRC5*
用于在仿真模式下无载荷运行机器人的系统输入信号 *SimMode*。 （主题 I/O，类型 System Input，行动值，*SimMode*）	技术参考手册 - 系统参数
激活和停用载荷的系统参数 *ModalPayLoadMode*。 （主题 Controller，类型 System Misc，行动值，*ModalPayLoadMode*）	技术参考手册 - 系统参数

1.140 MoveCGO - 机械臂沿圆周运动，设置拐角处的组输出信号

手册用法

MoveCGO（*Move Circular Group Output*）用于将工具中心点移动至给定目的地。将指定的组输出信号设置在目的点角路径中部。运动期间，相对于圆周，姿态通常保持不变。

本指令仅可用于主任务T_ROB1，或者如果在*MultiMove*系统中，则可用于运动任务中。

基本示例

以下实例介绍了指令MoveCGO：

例 1

```
MoveCGO p1, p2, v500, z30, tool2, go1 \Value:=5;
```

工具的TCPtool2沿圆周运动至位置p2，其速度数据为v500 ，区域数据为z30。根据起点位置、圆周点p1和目的点p2来定义圆周。将组输出信号go1设置在p2处角路径中部。

变元

```
MoveCGO CirPoint ToPoint [\ID] Speed [\T] Zone Tool [\WObj] Signal
     [\Value] | [\DValue] [\TLoad]
```

CirPoint

数据类型： robtarget

机械臂的圆周点。圆周点位于起点和目的点之间的圆周上。为获得最佳精度，应当将其放置在起点和目的点之间大约中间的位置。如果放置位置过于接近起点和目的点，则机械臂可能会发出警告。将圆周点定义为指定位置，或直接储存在指令中（在指令中标记有*）。未使用外轴的位置。

ToPoint

数据类型： robtarget

机器人和外部轴的目标点。定义为已命名的位置或直接存储在指令中（在指令中加 * 标记）。

[\ID]

Synchronization id

数据类型： identno

如果运动同步或协调同步，则参数[\ID]在MultiMove系统中具有强制性。在其他情况下，不允许该参数。指定id编号必须与所有合作程序任务中的编号相同。通过运用id编号，运动不会在进行时混淆。

Speed

数据类型： speeddata

适用于移动的速度数据。速度数据定义了相关TCP的速度、工具的重定方位和外轴。

[\T]

Time

数据类型：num

下一页继续

该参数用于规定机械臂和外轴运动期间的总时间，以秒计。随后，取代相关的速度数据。

Zone

数据类型： zonedata

相关移动的区域数据。区域数据描述了所生成拐角路径的大小。

Tool

数据类型： tooldata

相关机器人移动时使用的工具。工具中心点是将移过去的指定目标点。

[\WObj]

Work Object

数据类型： wobjdata

该工件（对象坐标系）与相关指令中的机器人位置相关联。

可省略该参数，随后，位置与世界坐标系相关。另一方面，如果使用固定式TCP或协调的外轴，则必须指定该参数，从而执行与工件相关的圆周。

Signal

数据类型：signalgo

待改变组输出信号的名称。

[\Value]

数据类型：num

信号的期望值。

[\DValue]

数据类型：dnum

信号的期望值。

如果未输入参数\Value或\DValue，则将显示错误消息。

[\TLoad]

Total load

数据类型：loaddata

\TLoad主动轴描述了移动中使用的总负载。总负载就是相关的工具负载加上该工具正在处理的有效负载。如果使用了\TLoad自变数，那么就不考虑当前tooldata中的loaddata。

如果\TLoad自变数被设置成load0，那么就不考虑\TLoad自变数，而是以当前tooldata中的loaddata作为代替。

想要使用\TLoad自变数，就必需将系统参数ModalPayLoadMode的数值设置成0。如果将ModalPayLoadMode设置成0，那么就再也无法使用指令GripLoad。

可用服务例程“负载标识”（LoadIdentify）来识别总负载。如果系统参数ModalPayLoadMode被设置成0，且系统正在运行该服务例程，那么操作员便可将相关工具的loaddata复制到一个现有的或新的loaddata永久变量中。

如果使用了关联到系统输入项SimMode（仿真模式）上的一个数字输入信号，那么便可在没有任何有效负载的情况下试运行该程序。如果该数字输入信号被设置成1，那么

下一页继续

1.140 MoveCGO - 机械臂沿圆周运动，设置拐角处的组输出信号
RobotWare - OS
续前页

就不考虑可选自变数\TLoad中的loaddata，而是以当前tooldata中的loaddata作为代替。

> **注意**
>
> 处理有效负载的默认功能是使用指令GripLoad，因此系统参数ModalPayLoadMode的默认值为1。

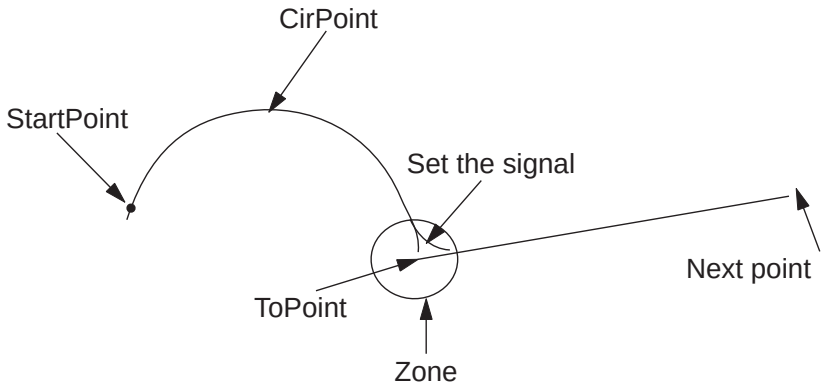
程序执行

更多关于圆周运动的信息，请参见指令MoveC，第341页的*MoveC - 使机械臂沿圆周移动*。

将组输出信号设置在飞焊点角路径的中间位置，如下图所示。

本图显示了MoveCGO角路径中组输出信号的设置。

```
MoveCGO p2, p2, v500, z30, tool2, go1 \Value:=5;
```



xx1400001116

关于停止点，我们建议使用关于MoveC和SetGO的“普通”编程顺序。但是，当使用指令MoveCGO中的停止点时，组输出信号得以在机械臂达到停止点时设置。

以连续执行模式逐步向前而非逐步向后地设置指定的I/O信号。

限制

符合指令MoveC的一般限制，参见第341页的*MoveC - 使机械臂沿圆周移动*。

错误处理

系统会生成下列可恢复错误，并在错误处理器中处理这些错误。系统变量ERRNO将被设置成：

如果关于指定组输出信号Value或DValue参数超出限值，则ERR_GO_LIM。

如果信号变量是RAPID中声明的变量，则ERR_NO_ALIASIO_DEF。尚未同I/O配置以及指令AliasIO中确定的I/O信号相连。

如果与I/O单元无接触，则ERR_NORUNUNIT。

如果无法访问I/O信号（仅对ICI现场总线有效），则ERR_SIG_NOT_VALID。

语法

```
MoveCGO
  [ CirPoint ':=' ] < expression (IN) of robtarget > ','
```

下一页继续

```
[ ToPoint ':=' ] < expression (IN) of robtarget > ','
[ '\' ID ':=' < expression (IN) of identno >]','
[ Speed ':=' ] < expression (IN) of speeddata >
[ '\' T ':=' < expression (IN) of num > ] ','
[ Zone ':=' ] < expression (IN) of zonedata > ','
[ Tool ':=' ] < persistent (PERS) of tooldata >
[ '\' WObj ':=' < persistent (PERS) of wobjdata > ] ','
[ Signal ':=' ] < variable (VAR) of signalgo > ] ','
[ '\' Value ':=' ] < expression (IN) of num > ]
| [ '\' Dvalue' :=' ] < expression (IN) of dnum >
[ '\' TLoad ':=' < persistent (PERS) of loaddata > ] ';'
```

相关信息

信息，关于	请参阅
其他定位指令	技术参考手册 - *RAPID*语言概览
使机械臂沿圆周移动	第*341*页的*MoveC - 使机械臂沿圆周移动*
载荷的定义	第*1421*页的*loaddata - 加载数据*
速度的定义	第*1480*页的*speeddata - 速度数据*
工具的定义	第*1502*页的*tooldata - 工具数据*
工件的定义	第*1523*页的*wobjdata - 工件数据*
区域数据的定义	第*1531*页的*zonedata - 区域数据*
一般动作	技术参考手册 - *RAPID*语言概览
坐标系	技术参考手册 - *RAPID*语言概览
关于I/O设置的运动	技术参考手册 - *RAPID*语言概览
关于如何使用TLoad总负载的例子。	第*388*页的*MoveL - 使机械臂沿直线移动*
定义机器人的载荷	第*223*页的*GripLoad - 定义机械臂的有效负载*
LoadIdentify，载荷识别服务例行程序	操作员手册 - 带 *FlexPendant* 的 *IRC5*
用于在仿真模式下无载荷运行机器人的系统输入信号 *SimMode*。 （题为I/O System，类型为System Input，行动值为*SimMode*）	技术参考手册 - 系统参数
激活和停用载荷的系统参数 *ModalPayLoadMode*。 （主题 Controller，类型 System Misc，行动值，*ModalPayLoadMode*）	技术参考手册 - 系统参数

1.141 MoveCSync - 机械臂沿圆周运动，执行RAPID无返回值程序。

手册用法

MoveCSync（*Move Circular Synchronously*）用于使工具中心点沿圆周运动至给定目的地。下达关于指定RAPID无返回值程序的命令，从而在目的点角路径中部执行。运动期间，相对于圆周，姿态通常保持不变。

本指令仅可用于主任务T_ROB1，或者如果在*MultiMove*系统中，则可用于运动任务中。

基本示例

以下实例介绍了指令MoveCSync：

例 1

```
MoveCSync p1, p2, v500, z30, tool2, "proc1";
```

工具的TCPtool2沿圆周运动至位置p2，其速度数据为v500 ，区域数据为z30。根据起点位置、圆周点p1和目的点p2来定义圆周。在p2处角路径中部执行无返回值程序proc1。

例 2

```
MoveCSync p1, p2, v500, z30, tool2, "MyModule:proc1";
```

与上文例1相同，但在此处，将于角路径中部调用模块MyModule中局部声明的无返回值程序proc1。

变元

```
MoveCSync CirPoint ToPoint [\ID] Speed [\T] Zone Tool [\WObj]
     ProcName [\TLoad]
```

CirPoint

数据类型：robtarget

机械臂的圆周点。圆周点位于起点和目的点之间的圆周上。为获得最佳精度，应当将其放置在起点和目的点之间大约中间的位置。如果放置位置过于接近起点和目的点，则机械臂可能会发出警告。将圆周点定义为指定位置，或直接储存在指令中（在指令中标记有*）。未使用外轴的位置。

ToPoint

数据类型：robtarget

机器人和外部轴的目标点。定义为已命名的位置或直接存储在指令中（在指令中加 * 标记）。

[\ID]

Synchronization id

数据类型： identno

如果移动是同步或协调同步的，变元 [\ID] 在 MultiMove 系统中是强制的。这个变元在任何其他情况下都不允许使用。指定的 ID 号必须与所有协作程序任务中的 ID 号相同。使用此 ID 号，动作才不会在运行时搞混。

Speed

数据类型： speeddata

下一页继续

适用于运动的速度数据。速度数据规定了关于TCP、工具方位调整和外轴的速率。

[\T]

Time

数据类型：num

该参数用于规定机械臂和外轴运动期间的总时间，以秒计。随后，取代相关的速度数据。

Zone

数据类型：zonedata

相关移动的区域数据。区域数据描述了所生成拐角路径的大小。

Tool

数据类型： tooldata

相关机器人移动时使用的工具。工具中心点是将移过去的指定目标点。

[\WObj]

Work Object

数据类型： wobjdata

该工件（对象坐标系）与相关指令中的机器人位置相关联。

可省略该参数，随后，位置与世界坐标系相关。另一方面，如果使用固定式TCP或协调的外轴，则必须指定该参数。

ProcName

Procedure Name

数据类型：string

将在目的点角路径中部执行的RAPID无返回值程序的名称。

将根据软中断等级执行本无返回值程序（参见下文中的程序执行）。

[\TLoad]

Total load

数据类型：loaddata

\TLoad主动轴描述了移动中使用的总负载。总负载就是相关的工具负载加上该工具正在处理的有效负载。如果使用了\TLoad自变数，那么就不考虑当前tooldata中的loaddata。

如果\TLoad自变数被设置成load0，那么就不考虑\TLoad自变数，而是以当前tooldata中的loaddata作为代替。

想要使用\TLoad自变数，就必需将系统参数ModalPayLoadMode的数值设置成0。如果将ModalPayLoadMode设置成0，那么就再也无法使用指令GripLoad。

可用服务例程“负载标识”（LoadIdentify）来识别总负载。如果系统参数ModalPayLoadMode被设置成0，且系统正在运行该服务例程，那么操作员便可将相关工具的loaddata复制到一个现有的或新的loaddata永久变量中。

如果使用了关联到系统输入项SimMode（仿真模式）上的一个数字输入信号，那么便可在没有任何有效负载的情况下试运行该程序。如果该数字输入信号被设置成1，那么

下一页继续

1.141 MoveCSync - 机械臂沿圆周运动，执行RAPID无返回值程序。
RobotWare - OS
续前页

就不考虑可选自变数\TLoad中的loaddata，而是以当前tooldata中的loaddata作为代替。

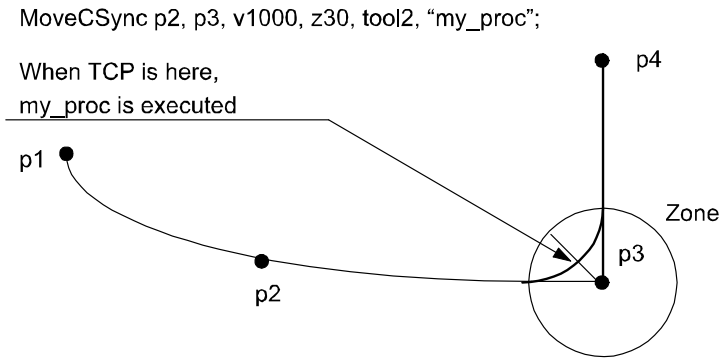
> **注意**
>
> 处理有效负载的默认功能是使用指令GripLoad，因此系统参数ModalPayLoadMode的默认值为1。

程序执行

更多关于圆周运动的信息，请参见指令MoveC 。

下达关于指定RAPID无返回值程序的指令，从而在TCP达到MoveCSync指令目的点中角路径中部时执行，如下图所示。

本图表明，在角路径中部下达关于执行用户定义的RAPID无返回值程序的指令。



xx0500002216

关于停止点，我们建议使用关于MoveC的“普通”编程序列以及序列中的其他RAPID指令。

本表描述了以不同的执行模式来执行指定RAPID无返回值程序。

执行模式	RAPID无返回值程序的执行
连续或循环	根据该描述
逐步向前	位于停止点
逐步向后	完全没有

MoveCSync是指令TriggInt和TriggC的封装。按照软中断等级，执行过程调用。

如果在程序停止后的减速期间，达到目的点中角路径的中部，则将不会调用过程（停止程序执行）。将在下一个程序起点执行过程调用。

限制

符合指令MoveC.的一般限制

当机械臂达到角路径中部时，通常存在2-30 ms的延迟，直至执行指定RAPID程序，具体取决于当时正在实施运动的类型。

在程序停止后，将执行模式从连续或循环切换为逐步向前或向后，由此产生错误。该错误告知用户，模式切换可导致错过路径执行序列中RAPID无返回值程序的执行。

下一页继续

1.141 MoveCSync - 机械臂沿圆周运动，执行RAPID无返回值程序。

RobotWare - OS

续前页

指令MoveCSync无法用于软中断等级。无法通过逐步执行来测试指定RAPID无返回值程序。

语法

```
MoveCSync
  [ CirPoint ':=' ] < expression (IN) of robtarget > ','
  [ ToPoint ':=' ] < expression (IN) of robtarget > ','
  [ '\' ID ':=' < expression (IN) of identno >]','
  [ Speed ':=' ] < expression (IN) of speeddata >
  [ '\' T ':=' < expression (IN) of num > ] ','
  [ Zone ':=' ] < expression (IN) of zonedata > ','
  [ Tool ':=' ] < persistent (PERS) of tooldata >
  [ '\' WObj ':=' < persistent (PERS) of wobjdata > ] ','
  [ ProcName ':=' ] < expression (IN) of string > ]
  [ '\' TLoad ':=' < persistent (PERS) of loaddata > ] ';'
```

相关信息

信息，关于	请参阅
其他定位指令	技术参考手册 - *RAPID*语言概览
使机械臂沿圆周移动	第*341*页的*MoveC - 使机械臂沿圆周移动*
载荷的定义	第*1421*页的*loaddata - 加载数据*
速度的定义	第*1480*页的*speeddata - 速度数据*
工具的定义	第*1502*页的*tooldata - 工具数据*
工件的定义	第*1523*页的*wobjdata - 工件数据*
区域数据的定义	第*1531*页的*zonedata - 区域数据*
一般动作	技术参考手册 - *RAPID*语言概览
坐标系	技术参考手册 - *RAPID*语言概览
定义与位置相关的中断	第*769*页的*TriggInt - 定义与位置相关的中断*
关于事件的机械臂圆周移动	第*747*页的*TriggC - 关于事件的机械臂圆周移动*
关于如何使用TLoad总负载的例子。	第*388*页的*MoveL - 使机械臂沿直线移动*
定义机器人的载荷	第*223*页的*GripLoad - 定义机械臂的有效负载*
LoadIdentify，载荷识别服务例行程序	操作员手册 - 带 *FlexPendant* 的 *IRC5*
用于在仿真模式下无载荷运行机器人的系统输入信号 *SimMode*。 （主题 I/O，类型 System Input，行动值，*SimMode*）	技术参考手册 - 系统参数
激活和停用载荷的系统参数 *ModalPayLoadMode*。 （主题 Controller，类型 System Misc，行动值，*ModalPayLoadMode*）	技术参考手册 - 系统参数

1.142 MoveExtJ - 在没有TCP的情况下，移动一个或数个机械单元

手册用法

MoveExtJ（*Move External Joints*）用于使外轴沿直线运动或旋转外轴。在没有TCP的情况下，外轴可属于一个或多个机械单元。

在没有TCP的情况下，仅当通过定义为运动任务的实际程序任务，且任务控制一个或多个机械单元时，方可使用该指令。

基本示例

以下实例介绍了指令MoveExtJ：

另请参阅第*365*页的更多示例

例 1

```
MoveExtJ jpos10, vrot10, z50;
```

将旋转的外轴移动至接头位置jpos10，其速度为10度/秒，且区域数据为z50。

例 2

```
MoveExtJ \Conc, jpos20, vrot10 \T:=5, fine \InPos:=inpos20;
```

用5秒时间，将外轴移动至接头位置jpos20。立即向前进行程序执行，但是，外轴在位置jpos20停止，直至inpos20中的收敛准则得以满足。

变元

```
MoveExtJ [\Conc] ToJointPos [\ID] [\UseEOffs] Speed [\T] Zone
    [\Inpos]
```

`[ \Conc ]`

Concurrent

数据类型：switch

当外轴正在运动时，执行后续指令。通常不使用参数，但是，当使用飞越点时，可使用参数以避免由过载CPU所引起的多余停止。当编程点在高速度下极为接近时，该参数适用。当不需要同外设备与机械臂运动之间的外设备和同步进行通信时，参数亦适用。

运用参数\Conc，将连续运动指令的数量限制为5。在包含StorePath-RestoPath运动指令以及参数的程序段中，不允许\Conc。

如果省略该参数，且ToJointPos并非停止点，则在外轴达到编程区之前，执行后续指令一段时间。

不能将该参数用于MultiMove系统中的协调同步移动。

`ToJointPos`

To Joint Position

数据类型：jointtarget

外轴的目的绝对接头位置。其定义为指定位置，或直接储存在指令中（在指令中标有*）。

`[ \ID ]`

Synchronization ID

数据类型： identno

下一页继续

如果移动是同步或协调同步的，变元 [\ID] 在 MultiMove 系统中是强制的。这个变元在任何其他情况下都不允许使用。指定的 ID 号必须与所有协作程序任务中的 ID 号相同。使用此 ID 号，动作才不会在运行时搞混。

[\UseEOffs]

Use External Offset

数据类型：switch

当使用参数UseEOffs时，针对MoveExtJ指令启用外轴的偏移量，其通过指令EOffsSet来设置。关于外偏移量的更多信息，请参见指令EOffsSet。

Speed

数据类型： speeddata

适用于运动的速度数据。速度数据规定了线性或旋转外轴的速率。

[\T]

Time

数据类型：num

该参数用于规定外轴运动期间的总时间，以秒计。随后，取代相关的速度数据。

Zone

数据类型：zonedata

关于运动的区域数据。区域数据规定了停止点或飞越点。如为飞越点，则区域半径将描述线性或旋转外轴的减速度和加速度。

[\Inpos]

In position

数据类型： stoppointdata

该参数用于规定关于停止点中外轴位置的收敛准则。停止点数据取代Zone参数中的指定区域。

程序执行

以编程速率，将线性或旋转外轴移动至编程点。

更多示例

```
CONST jointtarget j1 :=
     [[9E9,9E9,9E9,9E9,9E9,9E9],[0,9E9,9E9,9E9,9E9,9E9]];
CONST jointtarget j2 :=
     [[9E9,9E9,9E9,9E9,9E9,9E9],[30,9E9,9E9,9E9,9E9,9E9]];
CONST jointtarget j3 :=
     [[9E9,9E9,9E9,9E9,9E9,9E9],[60,9E9,9E9,9E9,9E9,9E9]];
CONST jointtarget j4 :=
     [[9E9,9E9,9E9,9E9,9E9,9E9],[90,9E9,9E9,9E9,9E9,9E9]];
CONST speeddata rot_ax_speed := [0, 0, 0, 45];

MoveExtJ j1, rot_ax_speed, fine;
MoveExtJ j2, rot_ax_speed, z20;
MoveExtJ j3, rot_ax_speed, z20;
MoveExtJ j4, rot_ax_speed, fine;
```

在该例子中，以45度/秒的速速，将旋转的单轴移动至接头位置0、30、60和90度处。

下一页继续

1.142 MoveExtJ - 在没有TCP的情况下，移动一个或数个机械单元

RobotWare - OS

续前页

错误处理

如果已经超出了使用参数\Conc的连续运动指令数，则将系统变量ERRNO设置为ERR_CONC_MAX。

可用错误处理器来处理该错误。

语法

```
MoveExtJ
  [ '\' Conc ',' ]
  [ ToJointPos ':=' ] < expression (IN) of jointtarget >
  [ '\' ID ':=' < expression (IN) of identno >]','
  [ '\' UseEOffs ',' ]
  [ Speed ':=' ] < expression (IN) of speeddata >
  [ '\' T ':=' < expression (IN) of num > ] ','
  [ Zone ':=' ] < expression (IN) of zonedata >
  [ '\' Inpos ':=' < expression (IN) of stoppointdata >] ';'
```

相关信息

信息，关于	请参阅
其他定位指令	技术参考手册 - *RAPID*语言概览
接头位置的定义	第*1418*页的*jointtarget - 接头位置数据*
速度的定义	第*1480*页的*speeddata - 速度数据*
区域数据的定义	第*1531*页的*zonedata - 区域数据*
一般动作	技术参考手册 - *RAPID*语言概览
同时进行的程序执行	技术参考手册 - *RAPID*语言概览

1.143 MoveJ - 通过接头移动，移动机械臂

手册用法

当该运动无须位于直线中时，MoveJ用于将机械臂迅速地从一点移动至另一点。

机械臂和外轴沿非线性路径运动至目的位置。所有轴均同时达到目的位置。

本指令仅可用于主任务T_ROB1，或者如果在*MultiMove*系统中，则可用于运动任务中。

基本示例

以下实例介绍了指令MoveJ：

另请参阅第*369*页的更多示例

例 1

```
MoveJ p1, vmax, z30, tool2;
```

将工具的工具中心点tool2沿非线性路径移动至位置p1，其速度数据为vmax，且区域数据为z30。

例 2

```
MoveJ *, vmax \T:=5, fine, grip3;
```

将工具的TCPgrip3沿非线性路径移动至储存于指令中的停止点（标记有*）。整个运动耗时5秒。

变元

```
MoveJ [\Conc] ToPoint [\ID] Speed [\V] | [\T] Zone [\Z] [\Inpos]
     Tool [\WObj] [\TLoad]
```

`[ \Conc ]`

Concurrent

数据类型：switch

当机械臂正在运动时，执行后续指令。通常不使用参数，但是，当使用飞越点时，可使用参数以避免由过载CPU所引起的多余停止。当编程点在高速度下极为接近时，该参数适用。当不需要同外设备与机械臂运动之间的外设备和同步进行通信时，参数亦适用。

运用参数\Conc，将连续运动指令的数量限制为5。在包含StorePath-RestoPath运动指令以及参数的程序段中，不允许\Conc。

如果省略该参数，且ToPoint并非停止点，则在机械臂达到编程区之前，执行后续指令一段时间。

不能将该参数用于MultiMove系统中的协调同步移动。

`ToPoint`

数据类型：robtarget

机器人和外部轴的目标点。定义为已命名的位置或直接存储在指令中（在指令中加 * 标记）。

`[ \ID]`

Synchronization id

数据类型： identno

下一页继续

1.143 MoveJ - 通过接头移动，移动机械臂
RobotWare - OS
续前页

如果协调同步运动，且在任何其他情形下均不被允许时，则必须在MultiMove系统中使用该参数。

在所有合作程序任务中，指定id编号必须相同。id编号确保各运动不会在运行时混淆。

Speed

数据类型： speeddata

适用于运动的速度数据。速度数据规定了关于工具中心点、工具方位调整和外轴的速率。

[\V]

Velocity

数据类型：num

该参数用于规定指令中TCP的速率，以mm/s计。随后，取代速度数据中指定的相关速率。

[\T]

Time

数据类型：num

该参数用于规定机械臂运动的总时间，以秒计。随后，取代相关的速度数据。

Zone

数据类型：zonedata

相关移动的区域数据。区域数据描述了所生成拐角路径的大小。

[\Z]

Zone

数据类型：num

该参数用于规定指令中机械臂TCP的位置精度。角路径的长度以mm计，其替代区域数据中指定的相关区域。

[\Inpos]

In position

数据类型： stoppointdata

该参数用于规定停止点中机械臂TCP位置的收敛准则。停止点数据取代Zone参数中的指定区域。

Tool

数据类型： tooldata

移动机械臂时正在使用的工具。工具中心点是指移动至指定目的点的点。

[\WObj]

Work Object

数据类型： wobjdata

指令中机器人位置关联的工件（坐标系）。

可省略该参数，随后，位置与世界坐标系相关。另一方面，如果使用固定式TCP或协调的外轴，则必须指定该参数。

下一页继续

[\TLoad]

Total load

数据类型：loaddata

\TLoad主动轴描述了移动中使用的总负载。总负载就是相关的工具负载加上该工具正在处理的有效负载。如果使用了\TLoad自变数，那么就不考虑当前tooldata中的loaddata。

如果\TLoad自变数被设置成load0，那么就不考虑\TLoad自变数，而是以当前tooldata中的loaddata作为代替。

想要使用\TLoad自变数，就必需将系统参数ModalPayLoadMode的数值设置成0。如果将ModalPayLoadMode设置成0，那么就再也无法使用指令GripLoad。

可用服务例程“负载标识”（LoadIdentify）来识别总负载。如果系统参数ModalPayLoadMode被设置成0，且系统正在运行该服务例程，那么操作员便可将相关工具的loaddata复制到一个现有的或新的loaddata永久变量中。

如果使用了关联到系统输入项SimMode（仿真模式）上的一个数字输入信号，那么便可在没有任何有效负载的情况下试运行该程序。如果该数字输入信号被设置成1，那么就不考虑可选自变数\TLoad中的loaddata，而是以当前tooldata中的loaddata作为代替。

注意

处理有效负载的默认功能是使用指令GripLoad，因此系统参数ModalPayLoadMode的默认值为1。

程序执行

通过插入轴角，将工具中心点移动至目的点。这意味着，各轴均以恒定轴速率运动，且所有轴均同时达到目的点，其形成一条非线性路径。

通常，TCP以适当的编程速率运动（无论是否调整外轴）。调整工具方位，并在TCP运动的同时，使外轴移动。如果无法达到关于调整方位或外轴的编程速率，则TCP的速率将会降低。

当运动转移至下一段路径时，通常会产生角路径。如果指定区域数据中的停止点，则仅当机械臂和外轴已达到适当位置时，方才继续程序执行。

更多示例

有关于如何使用指令MoveJ的更多例子阐述如下。

例 1

```
MoveJ *, v2000\V:=2200, z40 \Z:=45, grip3;
```

工具的TCPgrip3沿非线性路径运动至指令中储存的位置。将数据设置为v2000和z40时，开始运动。TCP的速率和区域半径分别为2200 mm/s和45 mm。

例 2

```
MoveJ p5, v2000, fine \Inpos := inpos50, grip3;
```

工具的TCPgrip3沿非线性路径运动至停止点p5。当满足关于停止点fine的50%的位置条件和50%的速度条件时，机械臂认为该工具位于点内。其最多等待2秒，以满足各条件。参见数据类型为stoppointdata的预定义数据inpos50。

下一页继续

1.143 MoveJ - 通过接头移动，移动机械臂

RobotWare - OS

续前页

例 3

```
MoveJ \Conc, *, v2000, z40, grip3;
```

工具的TCPgrip3沿非线性路径运动至指令中储存的位置。当机械臂运动时，执行后续逻辑指令。

例 4

```
MoveJ start, v2000, z40, grip3 \WObj:=fixture;
```

工具的TCPgrip3沿非线性路径运动至位置start。在关于fixture的工件坐标系中指定该位置。

错误处理

如果已经超出了使用参数\Conc的连续运动指令数，则将系统变量ERRNO设置为ERR_CONC_MAX。

可用错误处理器来处理该错误。

语法

```
MoveJ
  [ '\' Conc ',' ]
  [ ToPoint ':=' ] < expression (IN) of robtarget >
  [ '\' ID ':=' < expression (IN) of identno >]','
  [ Speed ':=' ] < expression (IN) of speeddata >
  [ '\' V ':=' < expression (IN) of num > ]
  | [ '\' ':=' < expression (IN) of num > ] ','
  [ Zone ':=' ] < expression (IN) of zonedata >
  [ '\' Z ':=' < expression (IN) of num > ]
  [ '\' Inpos ':=' < expression (IN) of stoppointdata > ] ','
  [ Tool ':=' ] < persistent (PERS) of tooldata >
  [ '\' WObj ':=' < persistent (PERS) of wobjdata > ]
  [ '\' TLoad ':=' < persistent (PERS) of loaddata > ] ';'
```

相关信息

信息，关于	请参阅
其他定位指令	技术参考手册 - *RAPID*语言概览
载荷的定义	第*1421*页的*loaddata - 加载数据*
速度的定义	第*1480*页的*speeddata - 速度数据*
停止点数据的定义	第*1483*页的*stoppointdata - 停止点数据*
工具的定义	第*1502*页的*tooldata - 工具数据*
工件的定义	第*1523*页的*wobjdata - 工件数据*
区域数据的定义	第*1531*页的*zonedata - 区域数据*
一般动作	技术参考手册 - *RAPID*语言概览
坐标系	技术参考手册 - *RAPID*语言概览
同时进行的程序执行	技术参考手册 - *RAPID*语言概览
关于如何使用TLoad总负载的例子。	第*388*页的*MoveL - 使机械臂沿直线移动*
定义机器人的载荷	第*223*页的*GripLoad - 定义机械臂的有效负载*
LoadIdentify，载荷识别服务例行程序	操作员手册 - 带 *FlexPendant* 的 *IRC5*

下一页继续

信息，关于	请参阅
用于在仿真模式下无载荷运行机器人的系统输入信号 *SimMode*。 （主题 I/O，类型 System Input，行动值，*SimMode*）	技术参考手册 - 系统参数
激活和停用载荷的系统参数 *ModalPayLoadMode*。 （主题 Controller，类型 System Misc，行动值，*ModalPayLoadMode*）	技术参考手册 - 系统参数

1.144 MoveJAO - 通过接头移动来移动机械臂，设置拐角处的模拟信号输出

手册用法

当该运动无须位于直线中时，MoveJAO（*Move Joint Analog Output*）用于将机械臂迅速地从一点移动至另一点。将指定的模拟信号输出信号设置在角路径中部。

机械臂和外轴沿非线性路径运动至目的位置。所有轴均同时达到目的位置。

本指令仅可用于主任务T_ROB1，或者如果在*MultiMove*系统中，则可用于运动任务中。

基本示例

以下实例介绍了指令MoveJAO：

例 1

```
MoveJAO p1, vmax, z30, tool2, ao1, 1.1;
```

工具的工具中心点tool2 沿非线性路径移动至位置p1，其速度数据为vmax，且区域数据为z30。将输出ao1设置在p1处角路径中部。

变元

```
MoveJAO ToPoint [\ID] Speed [\T] Zone Tool [\WObj] Signal Value
     [\TLoad]
```

ToPoint

数据类型： robtarget

机器人和外部轴的目标点。定义为已命名的位置或直接存储在指令中（在指令中加 * 标记）。

[\ID]

Synchronization id

数据类型： identno

如果移动是同步或协调同步的，变元 [\ID] 在 MultiMove 系统中是强制的。这个变元在任何其他情况下都不允许使用。指定的 ID 号必须与所有协作程序任务中的 ID 号相同。使用此 ID 号，动作才不会在运行时搞混。

Speed

数据类型： speeddata

适用于运动的速度数据。速度数据规定了关于工具中心点、工具方位调整和外轴的速率。

[\T]

Time

数据类型：num

该参数用于规定机械臂运动的总时间，以秒计。随后，取代相关的速度数据。

Zone

数据类型： zonedata

相关移动的区域数据。区域数据描述了所生成拐角路径的大小。

下一页继续

`Tool`

数据类型： `tooldata`

移动机械臂时正在使用的工具。工具中心点是指移动至指定目的点的点。

`[ \WObj ]`

Work Object

数据类型： `wobjdata`

指令中机器人位置关联的工件（坐标系）。

可省略该参数，随后，位置与世界坐标系相关。另一方面，如果使用固定式TCP或协调的外轴，则必须指定该参数。

`Signal`

数据类型：`signalao`

待改变模拟信号输出信号的名称。

`Value`

数据类型：`num`

信号的期望值。

`[ \TLoad ]`

Total load

数据类型：`loaddata`

`\TLoad`主动轴描述了移动中使用的总负载。总负载就是相关的工具负载加上该工具正在处理的有效负载。如果使用了`\TLoad`自变数，那么就不考虑当前`tooldata`中的`loaddata`。

如果`\TLoad`自变数被设置成`load0`，那么就不考虑`\TLoad`自变数，而是以当前`tooldata`中的`loaddata`作为代替。

想要使用`\TLoad`自变数，就必需将系统参数`ModalPayLoadMode`的数值设置成0。如果将`ModalPayLoadMode`设置成0，那么就再也无法使用指令`GripLoad`。

可用服务例程“负载标识”（LoadIdentify）来识别总负载。如果系统参数`ModalPayLoadMode`被设置成0，且系统正在运行该服务例程，那么操作员便可将相关工具的`loaddata`复制到一个现有的或新的`loaddata`永久变量中。

如果使用了关联到系统输入项`SimMode`（仿真模式）上的一个数字输入信号，那么便可在没有任何有效负载的情况下试运行该程序。如果该数字输入信号被设置成1，那么就不考虑可选自变数`\TLoad`中的`loaddata`，而是以当前`tooldata`中的`loaddata`作为代替。

> **注意**
>
> 处理有效负载的默认功能是使用指令`GripLoad`，因此系统参数`ModalPayLoadMode`的默认值为1。

程序执行

更多关于接头移动的信息，请参见指令`MoveJ`，*第367页的MoveJ - 通过接头移动，移动机械臂*。

将模拟信号输出信号设置在飞焊点角路径的中间位置，如下图所示。

下一页继续

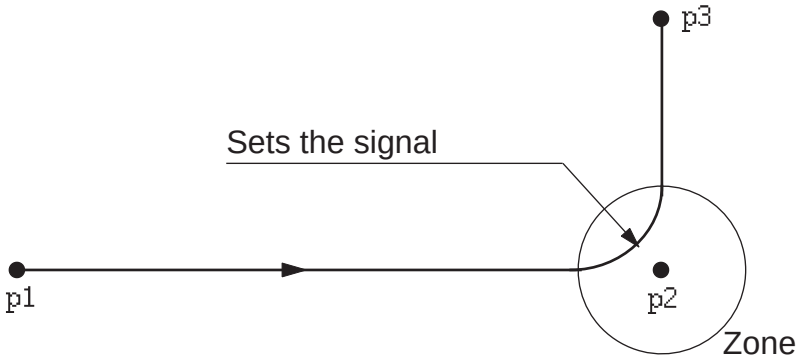
1.144 MoveJAO - 通过接头移动来移动机械臂，设置拐角处的模拟信号输出

RobotWare - OS

续前页

本图显示了MoveJAO角路径中模拟信号输出信号的设置。

```
MoveJAO p2, vmax, z30, tool2, ao1, 1.1;
```



xx1400001118

关于停止点，我们建议使用关于MoveJ和SetAO的“普通”编程顺序。但是，当使用指令MoveJAO中的停止点时，模拟信号输出信号得以在机械臂达到停止点时设置。

以连续执行模式逐步向前而非逐步向后地设置指定的I/O信号。

错误处理

系统会生成下列可恢复错误，并在错误处理器中处理这些错误。系统变量ERRNO将被设置成：

如果关于指定模拟信号输出信号Signal的编程Value参数超出限值，则ERR_AO_LIM。

如果信号变量是RAPID中声明的变量，则ERR_NO_ALIASIO_DEF。尚未同I/O配置以及指令AliasIO中确定的I/O信号相连。

如果与I/O单元无接触，则ERR_NORUNUNIT。

如果无法访问I/O信号（仅对ICI现场总线有效），则ERR_SIG_NOT_VALID。

语法

```
MoveJAO
  [ ToPoint ':=' ] < expression (IN) of robtarget >
  [ '\' ID ':=' < expression (IN) of identno >]','
  [ Speed ':=' ] < expression (IN) of speeddata >
  [ '\' T ':=' < expression (IN) of num > ] ','
  [ Zone ':=' ] < expression (IN) of zonedata > ','
  [ Tool ':=' ] < persistent (PERS) of tooldata>
  [ '\' WObj ':=' < persistent (PERS) of wobjdata > ] ','
  [ Signal ':=' ] < variable (VAR) of signalao>] ','
  [ Value ':=' ] < expression (IN) of num > ]
  [ '\' TLoad ':=' < persistent (PERS) of loaddata > ] ';'
```

相关信息

信息，关于	请参阅
其他定位指令	技术参考手册 - *RAPID*语言概览
通过接头移动，移动机械臂	第*367*页的*MoveJ - 通过接头移动，移动机械臂*
载荷的定义	第*1421*页的*loaddata - 加载数据*
速度的定义	第*1480*页的*speeddata - 速度数据*

下一页继续

信息，关于	请参阅
工具的定义	第*1502*页的*tooldata - 工具数据*
工件的定义	第*1523*页的*wobjdata - 工件数据*
区域数据的定义	第*1531*页的*zonedata - 区域数据*
一般动作	技术参考手册 - *RAPID*语言概览
坐标系	技术参考手册 - *RAPID*语言概览
关于I/O设置的运动	技术参考手册 - *RAPID*语言概览
关于如何使用TLoad总负载的例子。	第*388*页的*MoveL - 使机械臂沿直线移动*
定义机器人的载荷	第*223*页的*GripLoad - 定义机械臂的有效负载*
LoadIdentify，载荷识别服务例行程序	操作员手册 - 带 *FlexPendant* 的 *IRC5*
用于在仿真模式下无载荷运行机器人的系统输入信号 *SimMode*。 （主题 I/O，类型 System Input，行动值，*SimMode*）	技术参考手册 - 系统参数
激活和停用载荷的系统参数 *ModalPayLoadMode*。 （主题 Controller，类型 System Misc，行动值，*ModalPayLoadMode*）	技术参考手册 - 系统参数

1.145 MoveJDO - 通过接头移动来移动机械臂，设置拐角处的数字信号输出

手册用法

当该运动无须位于直线中时，MoveJDO（*Move Joint Digital Output*）用于将机械臂迅速地从一点移动至另一点。在角路径中部设置/重置指定的数字信号输出信号。

机械臂和外轴沿非线性路径运动至目的位置。所有轴均同时达到目的位置。

本指令仅可用于主任务T_ROB1，或者如果在*MultiMove*系统中，则可用于运动任务中。

基本示例

以下实例介绍了指令MoveJDO：

例 1

```
MoveJDO p1, vmax, z30, tool2, do1, 1;
```

工具的工具中心点tool2 沿非线性路径移动至位置p1，其速度数据为vmax，且区域数据为z30。将输出do1设置在p1处角路径中部。

变元

```
MoveJDO ToPoint [\ID] Speed [\T] Zone Tool [\WObj] Signal Value
     [\TLoad]
```

ToPoint

数据类型：robtarget

机器人和外部轴的目标点。定义为已命名的位置或直接存储在指令中（在指令中加 * 标记）。

[\ID]

Synchronization id

数据类型： identno

如果移动是同步或协调同步的，变元 [\ID] 在 MultiMove 系统中是强制的。这个变元在任何其他情况下都不允许使用。指定的 ID 号必须与所有协作程序任务中的 ID 号相同。使用此 ID 号，动作才不会在运行时搞混。

Speed

数据类型： speeddata

适用于运动的速度数据。速度数据规定了关于工具中心点、工具方位调整和外轴的速率。

[\T]

Time

数据类型：num

该参数用于规定机械臂运动的总时间，以秒计。随后，取代相关的速度数据。

Zone

数据类型：zonedata

相关移动的区域数据。区域数据描述了所生成拐角路径的大小。

下一页继续

Tool

数据类型： tooldata

移动机械臂时正在使用的工具。工具中心点是指移动至指定目的点的点。

[\WObj]

Work Object

数据类型： wobjdata

指令中机器人位置关联的工件（坐标系）。

可省略该参数，随后，位置与世界坐标系相关。另一方面，如果使用固定式TCP或协调的外轴，则必须指定该参数。

Signal

数据类型：signaldo

待改变数字信号输出信号的名称。

Value

数据类型：dionum

信号的期望值（0或1）。

[\TLoad]

Total load

数据类型：loaddata

\TLoad主动轴描述了移动中使用的总负载。总负载就是相关的工具负载加上该工具正在处理的有效负载。如果使用了\TLoad自变数，那么就不考虑当前tooldata中的loaddata。

如果\TLoad自变数被设置成load0，那么就不考虑\TLoad自变数，而是以当前tooldata中的loaddata作为代替。

想要使用\TLoad自变数，就必需将系统参数ModalPayLoadMode的数值设置成0。如果将ModalPayLoadMode设置成0，那么就再也无法使用指令GripLoad。

可用服务例程“负载标识”（LoadIdentify）来识别总负载。如果系统参数ModalPayLoadMode被设置成0，且系统正在运行该服务例程，那么操作员便可将相关工具的loaddata复制到一个现有的或新的loaddata永久变量中。

如果使用了关联到系统输入项SimMode（仿真模式）上的一个数字输入信号，那么便可在没有任何有效负载的情况下试运行该程序。如果该数字输入信号被设置成1，那么就不考虑可选自变数\TLoad中的loaddata，而是以当前tooldata中的loaddata作为代替。

注意

处理有效负载的默认功能是使用指令GripLoad，因此系统参数ModalPayLoadMode的默认值为1。

程序执行

更多关于接头移动的信息，请参见指令MoveJ。

将数字信号输出信号设置/重置在飞焊点角路径的中间位置，如下图所示。

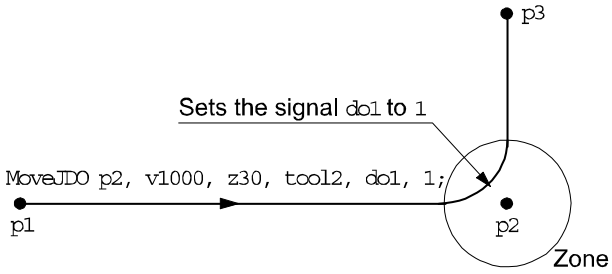
下一页继续

1.145 MoveJDO - 通过接头移动来移动机械臂，设置拐角处的数字信号输出

RobotWare - OS

续前页

本图显示了MoveJDO角路径中数字信号输出信号的设置/重置。



xx0500002196

关于停止点，我们建议使用关于MoveJ+SetDO的“普通”编程顺序。但是，当使用指令MoveJDO中的停止点时，数字信号输出信号得以在机械臂达到停止点时设置/重置。

以连续执行模式逐步向前而非逐步向后地设置/重置指定的I/O信号。

错误处理

系统会生成下列可恢复错误，并在错误处理器中处理这些错误。系统变量ERRNO将被设置成：

如果信号变量是RAPID中声明的变量，则ERR_NO_ALIASIO_DEF。尚未同I/O配置以及指令AliasIO中确定的I/O信号相连。

如果与I/O单元无接触，则ERR_NORUNUNIT。

如果无法访问I/O信号（仅对ICI现场总线有效），则ERR_SIG_NOT_VALID。

语法

```
MoveJDO
  [ ToPoint ':=' ] < expression (IN) of robtarget >
  [ '\' ID ':=' < expression (IN) of identno >]','
  [ Speed ':=' ] < expression (IN) of speeddata >
  [ '\' T ':=' < expression (IN) of num > ] ','
  [ Zone ':=' ] < expression (IN) of zonedata > ','
  [ Tool ':=' ] < persistent (PERS) of tooldata>
  [ '\' WObj ':=' < persistent (PERS) of wobjdata > ] ','
  [ Signal ':=' ] < variable (VAR) of signaldo>] ','
  [ Value ':=' ] < expression (IN) of dionum > ]
  [ '\' TLoad ':=' < persistent (PERS) of loaddata > ] ';'
```

相关信息

信息，关于	请参阅
其他定位指令	技术参考手册 - *RAPID*语言概览
通过接头移动，移动机械臂	第*367*页的*MoveJ - 通过接头移动，移动机械臂*
载荷的定义	第*1421*页的*loaddata - 加载数据*
速度的定义	第*1480*页的*speeddata - 速度数据*
工具的定义	第*1502*页的*tooldata - 工具数据*
工件的定义	第*1523*页的*wobjdata - 工件数据*
区域数据的定义	第*1531*页的*zonedata - 区域数据*
一般动作	技术参考手册 - *RAPID*语言概览

下一页继续

信息，关于	请参阅
坐标系	技术参考手册 - *RAPID*语言概览
关于I/O设置的运动	技术参考手册 - *RAPID*语言概览
关于如何使用TLoad总负载的例子。	第*388*页的*MoveL - 使机械臂沿直线移动*
定义机器人的载荷	第*223*页的*GripLoad - 定义机械臂的有效负载*
LoadIdentify，载荷识别服务例行程序	操作员手册 - 带 *FlexPendant* 的 *IRC5*
用于在仿真模式下无载荷运行机器人的系统输入信号 *SimMode*。 （主题 I/O，类型 System Input，行动值，*SimMode*）	技术参考手册 - 系统参数
激活和停用载荷的系统参数 *ModalPayLoadMode*。 （主题 Controller，类型 System Misc，行动值，*ModalPayLoadMode*）	技术参考手册 - 系统参数

1.146 MoveJGO - 通过接头移动来移动机械臂，设置拐角处的组输出信号

手册用法

当该运动无须位于直线中时，MoveJGO（*Move Joint Group Output*）用于将机械臂迅速地从一点移动至另一点。将指定的组输出信号设置在角路径中部。

机械臂和外轴沿非线性路径运动至目的位置。所有轴均同时达到目的位置。

本指令仅可用于主任务T_ROB1，或者如果在*MultiMove*系统中，则可用于运动任务中。

基本示例

以下实例介绍了指令MoveJGO：

例 1

```
MoveJGO p1, vmax, z30, tool2, go1 \Value:=5;
```

工具的工具中心点tool2 沿非线性路径移动至位置p1，其速度数据为vmax，且区域数据为z30。将组输出信号go1设置在p1处角路径中部。

变元

```
MoveJGO ToPoint [\ID] Speed [\T] Zone Tool [\WObj] Signal [\Value]
     | [\DValue] [\TLoad]
```

ToPoint

数据类型： robtarget

机器人和外部轴的目标点。定义为已命名的位置或直接存储在指令中（在指令中加 * 标记）。

[\ID]

Synchronization id

数据类型： identno

如果移动是同步或协调同步的，变元 [\ID] 在 MultiMove 系统中是强制的。这个变元在任何其他情况下都不允许使用。指定的 ID 号必须与所有协作程序任务中的 ID 号相同。使用此 ID 号，动作才不会在运行时搞混。

Speed

数据类型： speeddata

适用于运动的速度数据。速度数据规定了关于工具中心点、工具方位调整和外轴的速率。

[\T]

Time

数据类型：num

该参数用于规定机械臂运动的总时间，以秒计。随后，取代相关的速度数据。

Zone

数据类型： zonedata

相关移动的区域数据。区域数据描述了所生成拐角路径的大小。

下一页继续

`Tool`

数据类型：`tooldata`

移动机械臂时正在使用的工具。工具中心点是指移动至指定目的点的点。

`[ \WObj ]`

Work Object

数据类型：`wobjdata`

指令中机器人位置关联的工件（坐标系）。

可省略该参数，随后，位置与世界坐标系相关。另一方面，如果使用固定式TCP或协调的外轴，则必须指定该参数。

`Signal`

数据类型：`signalgo`

待改变组输出信号的名称。

`[ \Value ]`

数据类型：`num`

信号的期望值。

`[ \DValue ]`

数据类型：`dnum`

信号的期望值。

如果未输入参数`\Value`或`\DValue`，则将显示错误消息。

`[ \TLoad ]`

Total load

数据类型：`loaddata`

`\TLoad`主动轴描述了移动中使用的总负载。总负载就是相关的工具负载加上该工具正在处理的有效负载。如果使用了`\TLoad`自变数，那么就不考虑当前`tooldata`中的`loaddata`。

如果`\TLoad`自变数被设置成`load0`，那么就不考虑`\TLoad`自变数，而是以当前`tooldata`中的`loaddata`作为代替。

想要使用`\TLoad`自变数，就必需将系统参数`ModalPayLoadMode`的数值设置成0。如果将`ModalPayLoadMode`设置成0，那么就再也无法使用指令`GripLoad`。

可用服务例程“负载标识”（LoadIdentify）来识别总负载。如果系统参数`ModalPayLoadMode`被设置成0，且系统正在运行该服务例程，那么操作员便可将相关工具的`loaddata`复制到一个现有的或新的`loaddata`永久变量中。

如果使用了关联到系统输入项`SimMode`（仿真模式）上的一个数字输入信号，那么便可在没有任何有效负载的情况下试运行该程序。如果该数字输入信号被设置成1，那么就不考虑可选自变数`\TLoad`中的`loaddata`，而是以当前`tooldata`中的`loaddata`作为代替。

注意

处理有效负载的默认功能是使用指令`GripLoad`，因此系统参数`ModalPayLoadMode`的默认值为1。

下一页继续

1.146 MoveJGO - 通过接头移动来移动机械臂，设置拐角处的组输出信号

RobotWare - OS

续前页

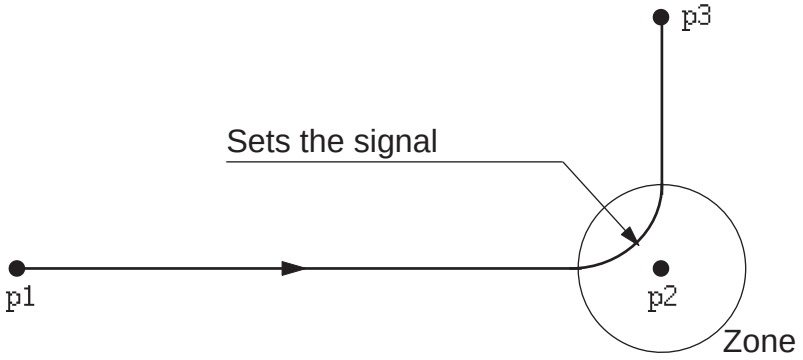
程序执行

更多关于接头移动的信息，请参见指令MoveJ。

将组输出信号设置在飞焊点角路径的中间位置，如下图所示。

本图显示了MoveJGO角路径中组输出信号的设置。

```
MoveJGO p2, vmax, z30, tool2, go1 \Value:=5;
```



xx1400001118

关于停止点，我们建议使用关于MoveJ+SetGO的“普通”编程顺序。但是，当使用指令MoveJGO中的停止点时，组输出信号得以在机械臂达到停止点时设置。

以连续执行模式逐步向前而非逐步向后地设置指定的I/O信号。

错误处理

系统会生成下列可恢复错误，并在错误处理器中处理这些错误。系统变量ERRNO将被设置成：

如果关于指定组输出信号Value或DValue参数超出限值，则ERR_GO_LIM。

如果信号变量是RAPID中声明的变量，则ERR_NO_ALIASIO_DEF。尚未同I/O配置以及指令AliasIO中确定的I/O信号相连。

如果与I/O单元无接触，则ERR_NORUNUNIT。

如果无法访问I/O信号（仅对ICI现场总线有效），则ERR_SIG_NOT_VALID。

语法

```
MoveJGO
  [ ToPoint ':=' ] < expression (IN) of robtarget >
  [ '\' ID ':=' < expression (IN) of identno >]','
  [ Speed ':=' ] < expression (IN) of speeddata >
  [ '\' T ':=' < expression (IN) of num > ] ','
  [ Zone ':=' ] < expression (IN) of zonedata > ','
  [ Tool ':=' ] < persistent (PERS) of tooldata>
  [ '\' WObj ':=' < persistent (PERS) of wobjdata > ] ','
  [ Signal ':=' ] < variable (VAR) of signalgo>] ','
  [ '\' Value ':=' ] < expression (IN) of num > ]
  | [ '\' Dvalue' :=' ] < expression (IN) of dnum >
  [ '\' TLoad ':=' < persistent (PERS) of loaddata > ] ';'
```

下一页继续

相关信息

信息，关于	请参阅
其他定位指令	技术参考手册 - *RAPID*语言概览
通过接头移动，移动机械臂	第*367*页的*MoveJ - 通过接头移动，移动机械臂*
载荷的定义	第*1421*页的*loaddata - 加载数据*
速度的定义	第*1480*页的*speeddata - 速度数据*
工具的定义	第*1502*页的*tooldata - 工具数据*
工件的定义	第*1523*页的*wobjdata - 工件数据*
区域数据的定义	第*1531*页的*zonedata - 区域数据*
一般动作	技术参考手册 - *RAPID*语言概览
坐标系	技术参考手册 - *RAPID*语言概览
关于I/O设置的运动	技术参考手册 - *RAPID*语言概览
关于如何使用TLoad总负载的例子。	第*388*页的*MoveL - 使机械臂沿直线移动*
定义机器人的载荷	第*223*页的*GripLoad - 定义机械臂的有效负载*
LoadIdentify，载荷识别服务例行程序	操作员手册 - 带 *FlexPendant* 的 *IRC5*
用于在仿真模式下无载荷运行机器人的系统输入信号 *SimMode*。 （主题 I/O，类型 System Input，行动值，*SimMode*）	技术参考手册 - 系统参数
激活和停用载荷的系统参数 *ModalPayLoadMode*。 （主题 Controller，类型 System Misc，行动值，*ModalPayLoadMode*）	技术参考手册 - 系统参数

1.147 MoveJSync - 通过接头移动来移动机械臂，执行RAPID无返回值程序。

手册用法

当该运动无须位于直线中时，MoveJSync （*Move Joint Synchronously*）为，用于将机械臂迅速地从一点移动至另一点。下达关于指定RAPID无返回值程序的指令，以便在目的点中角路径中部执行。

机械臂和外轴沿非线性路径运动至目的位置。所有轴均同时达到目的位置。

本指令仅可用于主任务T_ROB1，或者如果在*MultiMove*系统中，则可用于运动任务中。

基本示例

以下实例介绍了指令MoveJSync：

例 1

```
MoveJSync p1, vmax, z30, tool2, "proc1";
```

工具的工具中心点tool2沿非线性路径移动至位置p1，其速度数据为vmax，且区域数据为z30。在p1处角路径中部执行无返回值程序proc1。

例 2

```
MoveJSync p1, vmax, z30, tool2, "MyModule:proc1";
```

与上文例1相同，但在此处，将于角路径中部调用模块MyModule中局部声明的无返回值程序proc1。

变元

```
MoveJSync ToPoint [\ID] Speed [\T] Zone Tool [\WObj] ProcName
     [\TLoad]
```

ToPoint

数据类型：robtarget

机器人和外部轴的目标点。定义为已命名的位置或直接存储在指令中（在指令中加 * 标记）。

[\ID]

Synchronization id

数据类型： identno

如果移动是同步或协调同步的，变元 [\ID] 在 MultiMove 系统中是强制的。这个变元在任何其他情况下都不允许使用。指定的 ID 号必须与所有协作程序任务中的 ID 号相同。使用此 ID 号，动作才不会在运行时搞混。

Speed

数据类型： speeddata

适用于运动的速度数据。速度数据规定了关于工具中心点、工具方位调整和外轴的速率。

[\T]

Time

数据类型：num

该参数用于规定机械臂运动的总时间，以秒计。随后，取代相关的速度数据。

下一页继续

Zone

数据类型：zonedata

相关移动的区域数据。区域数据描述了所生成拐角路径的大小。

Tool

数据类型： tooldata

移动机械臂时正在使用的工具。工具中心点是指移动至指定目的点的点。

[\WObj]

Work Object

数据类型： wobjdata

指令中机器人位置关联的工件（坐标系）。

可省略该参数，随后，位置与世界坐标系相关。另一方面，如果使用固定式TCP或协调的外轴，则必须指定该参数。

ProcName

Procedure Name

数据类型：string

将在目的点中角路径中部执行的RAPID无返回值程序的名称。过程调用为后期绑定调用，因此，继承了其属性。

将根据软中断等级执行本无返回值程序（参见下文中的程序执行）。

[\TLoad]

Total load

数据类型：loaddata

\TLoad主动轴描述了移动中使用的总负载。总负载就是相关的工具负载加上该工具正在处理的有效负载。如果使用了\TLoad自变数，那么就不考虑当前tooldata中的loaddata。

如果\TLoad自变数被设置成load0，那么就不考虑\TLoad自变数，而是以当前tooldata中的loaddata作为代替。

想要使用\TLoad自变数，就必需将系统参数ModalPayLoadMode的数值设置成0。如果将ModalPayLoadMode设置成0，那么就再也无法使用指令GripLoad。

可用服务例程“负载标识”（LoadIdentify）来识别总负载。如果系统参数ModalPayLoadMode被设置成0，且系统正在运行该服务例程，那么操作员便可将相关工具的loaddata复制到一个现有的或新的loaddata永久变量中。

如果使用了关联到系统输入项SimMode（仿真模式）上的一个数字输入信号，那么便可在没有任何有效负载的情况下试运行该程序。如果该数字输入信号被设置成1，那么就不考虑可选自变数\TLoad中的loaddata，而是以当前tooldata中的loaddata作为代替。

注意

处理有效负载的默认功能是使用指令GripLoad，因此系统参数ModalPayLoadMode的默认值为1。

下一页继续

1.147 MoveJSync - 通过接头移动来移动机械臂，执行RAPID无返回值程序。

RobotWare - OS

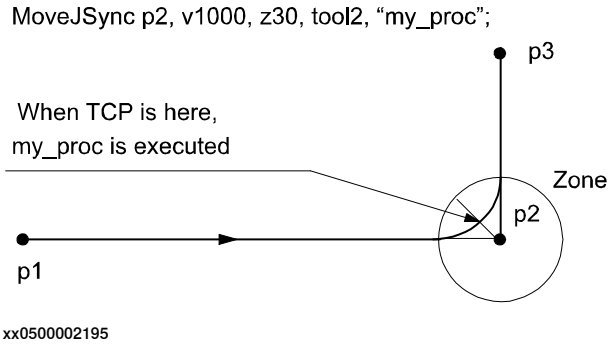
续前页

程序执行

更多关于接头移动的信息，请参见指令MoveJ。

下达关于指定RAPID无返回值程序的指令，从而在TCP达到MoveJSync指令目的点中角路径中部时执行，

如下图所示。



xx0500002195

关于停止点，我们建议使用关于MoveJ 的“普通”编程序列以及序列中的其他RAPID指令。

本表描述了以不同的执行模式来执行指定RAPID无返回值程序。

执行模式	RAPID无返回值程序的执行
连续或循环	根据该描述
逐步向前	位于停止点
逐步向后	完全没有

MoveJSync是指令TriggInt和TriggJ的封装。按照软中断等级，执行过程调用。

如果在程序停止后的减速期间，达到目的点中角路径的中部，则将不会调用过程（停止程序执行）。将在下一个程序起点执行过程调用。

限制

当机械臂达到角路径中部时，通常存在2-30 ms的延迟，直至执行指定RAPID程序，具体取决于当时正在实施运动的类型。

在程序停止后，将执行模式从连续或循环切换为逐步向前或向后，由此产生错误。该错误告知用户，模式切换可导致错过路径执行序列中RAPID无返回值程序的执行。

指令MoveJSync无法用于软中断等级。无法通过逐步执行来测试指定RAPID无返回值程序。

语法

```
MoveJSync
  [ ToPoint ':=' ] < expression (IN) of robtarget >
  [ '\' ID ':=' < expression (IN) of identno >] ','
  [ Speed ':=' ] < expression (IN) of speeddata >
  '\' T ':=' < expression (IN) of num >] ','
  [ Zone ':=' ] < expression (IN) of zonedata > ','
  [ Tool ':=' ] < persistent (PERS) of tooldata >
  [ '\' WObj '=' < persistent (PERS) of wobjdata > ] ','
  [ ProcName '=' ] < expression (IN) of string > ]
  [ '\' TLoad ':=' < persistent (PERS) of loaddata > ] ';'
```

下一页继续

相关信息

信息，关于	请参阅
其他定位指令	技术参考手册 - *RAPID*语言概览
通过接头移动，移动机械臂	第*367*页的*MoveJ - 通过接头移动，移动机械臂*
载荷的定义	第*1421*页的*loaddata - 加载数据*
速度的定义	第*1480*页的*speeddata - 速度数据*
工具的定义	第*1502*页的*tooldata - 工具数据*
工件的定义	第*1523*页的*wobjdata - 工件数据*
区域数据的定义	第*1531*页的*zonedata - 区域数据*
一般动作	技术参考手册 - *RAPID*语言概览
坐标系	技术参考手册 - *RAPID*语言概览
定义与位置相关的中断	第*769*页的*TriggInt - 定义与位置相关的中断*
关于事件的轴式机械臂运动	第*778*页的*TriggJ - 关于事件的轴式机械臂运动*
关于如何使用TLoad总负载的例子。	第*388*页的*MoveL - 使机械臂沿直线移动*
定义机器人的载荷	第*223*页的*GripLoad - 定义机械臂的有效负载*
LoadIdentify，载荷识别服务例行程序	操作员手册 - 带 *FlexPendant* 的 *IRC5*
用于在仿真模式下无载荷运行机器人的系统输入信号 *SimMode*。 （主题 I/O，类型 System Input，行动值，*SimMode*）	技术参考手册 - 系统参数
激活和停用载荷的系统参数 *ModalPayLoadMode*。 （主题 Controller，类型 System Misc，行动值，*ModalPayLoadMode*）	技术参考手册 - 系统参数

1.148 MoveL - 使机械臂沿直线移动

手册用法

MoveL用于将工具中心点沿直线移动至给定目的。当TCP保持固定时，则该指令亦可用于调整工具方位。

本指令仅可用于主任务T_ROB1，或者如果在MultiMove系统中，则可用于运动任务中。

基本示例

以下实例介绍了指令MoveL：

另请参阅第*390*页的更多示例

例 1

```
MoveL p1, v1000, z30, tool2;
```

工具的TCPtool2将直线运动至位置p1，其速度数据为v1000，且区域数据为z30。

例 2

```
MoveL *, v1000\T:=5, fine, grip3;
```

工具的TCPgrip3沿直线移动至储存于指令中的停止点（标记有*）。完整的运动耗时5秒。

变元

```
MoveL [\Conc] ToPoint [\ID] Speed [\V] | [ \T] Zone [\Z] [\Inpos]
     Tool [\WObj] [\Corr] [\TLoad]
```

[\Conc]

Concurrent

数据类型：switch

当机械臂正在运动时，执行后续指令。通常不使用参数，但是，当使用飞越点时，可使用参数以避免由过载CPU所引起的多余停止。当编程点在高速度下极为接近时，该参数适用。当不需要同外设备与机械臂运动之间的外设备和同步进行通信时，参数亦适用。

运用参数\Conc，将连续运动指令的数量限制为5。在包含StorePath-RestoPath运动指令以及参数的程序段中，不允许\Conc。

如果省略该参数，且ToPoint并非停止点，则在机械臂达到编程区之前，执行后续指令一段时间。

不能将该参数用于MultiMove系统中的协调同步移动。

ToPoint

数据类型：robtarget

机器人和外部轴的目标点。定义为已命名的位置或直接存储在指令中（在指令中加 * 标记）。

[\ID]

Synchronization id

数据类型： identno

下一页继续

如果移动是同步或协调同步的，变元 [\ID] 在 MultiMove 系统中是强制的。这个变元在任何其他情况下都不允许使用。指定的 ID 号必须与所有协作程序任务中的 ID 号相同。使用此 ID 号，动作才不会在运行时搞混。

Speed

数据类型： speeddata

适用于移动的速度数据。速度数据定义了工具中心点的速率、工具重定位以及外部轴。

[\V]

Velocity

数据类型：num

该参数用于规定指令中TCP的速率，以mm/s计。随后，取代速度数据中指定的相关速率。

[\T]

Time

数据类型：num

该参数用于规定机械臂运动的总时间，以秒计。随后，取代相关的速度数据。

Zone

数据类型：zonedata

相关移动的区域数据。区域数据描述了所生成拐角路径的大小。

[\Z]

Zone

数据类型：num

该参数用于规定指令中机械臂TCP的位置精度。角路径的长度以mm计，其替代区域数据中指定的相关区域。

[\Inpos]

In position

数据类型： stoppointdata

该参数用于规定停止点中机械臂TCP位置的收敛准则。停止点数据取代Zone参数中的指定区域。

Tool

数据类型： tooldata

移动机械臂时正在使用的工具。工具中心点是指移动至指定目的位置的点。

[\WObj]

Work Object

数据类型： wobjdata

指令中机器人位置关联的工件（坐标系）。

可省略该参数，随后，位置与世界坐标系相关。另一方面，如果使用固定式工具或协调的外轴，则必须指定该参数，从而执行与工件相关的线性运动。

[\Corr]

Correction

下一页继续

1.148 MoveL - 使机械臂沿直线移动
RobotWare - OS
续前页

数据类型：switch

如果存在该参数，则将通过指令CorrWrite而写入修正条目的修正数据添加到路径和目的位置。

[\TLoad]

Total load

数据类型：loaddata

\TLoad主动轴描述了移动中使用的总负载。总负载就是相关的工具负载加上该工具正在处理的有效负载。如果使用了\TLoad自变数，那么就不考虑当前tooldata中的loaddata。

如果\TLoad自变数被设置成load0，那么就不考虑\TLoad自变数，而是以当前tooldata中的loaddata作为代替。

想要使用\TLoad自变数，就必需将系统参数ModalPayLoadMode的数值设置成0。如果将ModalPayLoadMode设置成0，那么就再也无法使用指令GripLoad。

可用服务例程“负载标识”（LoadIdentify）来识别总负载。如果系统参数ModalPayLoadMode被设置成0，且系统正在运行该服务例程，那么操作员便可将相关工具的loaddata复制到一个现有的或新的loaddata永久变量中。

如果使用了关联到系统输入项SimMode（仿真模式）上的一个数字输入信号，那么便可在没有任何有效负载的情况下试运行该程序。如果该数字输入信号被设置成1，那么就不考虑可选自变数\TLoad中的loaddata，而是以当前tooldata中的loaddata作为代替。

注意

处理有效负载的默认功能是使用指令GripLoad，因此系统参数ModalPayLoadMode的默认值为1。

程序执行

将机械臂和外部单元移动至以下目的位置：

- 以恒定编程速率，沿直线移动工具的TCP。
- 以相等的间隔，沿路径调整工具方位。
- 以恒定速率执行不协调的外轴，从而使其与机械臂轴同时到达目的点。

如果不可能达到关于调整姿态或外轴的编程速率，则将降低TCP的速率。

当运动转移至下一段路径时，通常会产生角路径。如果在区域数据中指定停止点，则仅当机械臂和外轴已达到适当的位置时，方才继续程序执行。

更多示例

有关于如何使用指令MoveL的更多例子阐述如下。

例 1

```
MoveL *, v2000 \V:=2200, z40 \Z:=45, grip3;
```

工具的TCPgrip3沿直线运动至指令中储存的位置。将数据设置为v2000和z40时，开始运动。TCP的速率和区域半径分别为2200 mm/s和45 mm。

例 2

```
MoveL p5, v2000, fine \Inpos := inpos50, grip3;
```

下一页继续

工具的TCPgrip3沿直线运动至停止点p5。当满足关于停止点fine的50%的位置条件和50%的速度条件时，机械臂认为该工具位于点内。其最多等待2秒，以满足各条件。参见数据类型为stoppointdata.的预定义数据inpos50。

例 3

```
MoveL \Conc, *, v2000, z40, grip3;
```

工具的TCPgrip3沿直线运动至指令中储存的位置。当机械臂运动时，执行后续逻辑指令。

例 4

```
MoveL start, v2000, z40, grip3 \WObj:=fixture;
```

工具的TCPgrip3沿直线运动至位置start。在关于fixture的工件坐标系中指定该位置。

关于TLoad的例子

```
MoveL p1, v1000, fine, tool2;
! Pick up the payload
Set gripperdo;
MoveL p2, v1000, z30, tool2 \TLoad:=tool2piece;
MoveL p3, v1000, fine, tool2 \TLoad:=tool2piece;
! Release the payload
Reset gripperdo;
MoveL p4, v1000, fine, tool2;
```

工具的TCPtool2沿直线运动至位置p1，由此选取有效负载。通过使用总负载tool2piece，TCP由该位置移动至位置p2和p3。未考虑当前tooldata中的loaddata。释放有效负载，且当运动至位置p4时，再次考虑工具的负载。

错误处理

如果已经超出了使用参数\Conc的连续运动指令数，则将系统变量ERRNO设置为ERR_CONC_MAX。

可用错误处理器来处理该错误。

语法

```
MoveL
  [ '\' Conc ',' ]
  [ ToPoint ':=' ] < expression (IN) of robtarget >
  [ '\' ID ':=' < expression (IN) of identno >] ','
  [ Speed ':=' ] < expression (IN) of speeddata >
  [ '\' V ':=' < expression (IN) of num > ]
  | [ '\' T ':=' < expression (IN) of num > ] ','
  [Zone ':=' ] < expression (IN) of zonedata >
  [ '\' Z ':='< expression (IN) of num > ]
  [ '\' Inpos ':=' < expression (IN) of stoppointdata > ] ','
  [ Tool ':=' ] < persistent (PERS) of tooldata >
  [ '\' WObj ':=' < persistent (PERS) of wobjdata > ]
  [ '\' Corr ]
  [ '\' TLoad ':=' < persistent (PERS) of loaddata > ] ';'
```

下一页继续

1.148 MoveL - 使机械臂沿直线移动
RobotWare - OS
续前页

相关信息

信息，关于	请参阅
其他定位指令	技术参考手册 - *RAPID*语言概览
载荷的定义	第*1421*页的*loaddata* - 加载数据
速度的定义	第*1480*页的*speeddata* - 速度数据
停止点数据的定义	第*1483*页的*stoppointdata* - 停止点数据
工具的定义	第*1502*页的*tooldata* - 工具数据
工件的定义	第*1523*页的*wobjdata* - 工件数据
区域数据的定义	第*1531*页的*zonedata* - 区域数据
写入纠正条目	第*137*页的*CorrWrite* - 写入修正发电机
一般动作	技术参考手册 - *RAPID*语言概览
坐标系	技术参考手册 - *RAPID*语言概览
同时进行的程序执行	技术参考手册 - *RAPID*语言概览
定义机器人的载荷	第*223*页的*GripLoad* - 定义机械臂的有效负载
LoadIdentify，载荷识别服务例行程序	操作员手册 - 带 *FlexPendant* 的 *IRC5*
用于在仿真模式下无载荷运行机器人的系统输入信号 *SimMode*。 （主题 I/O，类型 System Input，行动值，*SimMode*）	技术参考手册 - 系统参数
激活和停用载荷的系统参数 *ModalPayLoadMode*。 （主题 Controller，类型 System Misc，行动值，*ModalPayLoadMode*）	技术参考手册 - 系统参数

1.149 MoveLAO - 使机械臂沿直线运动，设置拐角处的模拟信号输出

手册用法

MoveLAO（*Move Linearly Analog Output*）用于使工具中心点沿直线运动至给定目的地。在角路径中部设置指定模拟信号输出信号。

当TCP保持固定时，该指令亦可用于调整工具方位。

本指令仅可用于主任务T_ROB1，或者如果在*MultiMove*系统中，则可用于运动任务中。

基本示例

以下实例介绍了指令MoveLAO：

例 1

```
MoveLAO p1, v1000, z30, tool2, ao1, 1.1;
```

工具的TCPtool2沿直线移动至位置p1，其速度数据为v1000 ，且区域数据为z30。在p1处角路径中部，设置输出ao1。

变元

```
MoveLAO ToPoint [\ID] Speed [\T] Zone Tool [\WObj] Signal Value
    [\TLoad]
```

ToPoint

数据类型： robtarget

机器人和外部轴的目标点。定义为已命名的位置或直接存储在指令中（在指令中加 * 标记）。

[\ID]

Synchronization id

数据类型： identno

如果移动是同步或协调同步的，变元 [\ID] 在 MultiMove 系统中是强制的。这个变元在任何其他情况下都不允许使用。指定的 ID 号必须与所有协作程序任务中的 ID 号相同。使用此 ID 号，动作才不会在运行时搞混。

Speed

数据类型： speeddata

适用于移动的速度数据。速度数据定义了工具中心点的速率、工具重定位以及外部轴。

[\T]

Time

数据类型：num

该参数用于规定机械臂运动的总时间，以秒计。随后，取代相关的速度数据。

Zone

数据类型： zonedata

相关移动的区域数据。区域数据描述了所生成拐角路径的大小。

Tool

数据类型： tooldata

下一页继续

1.149 MoveLAO - 使机械臂沿直线运动，设置拐角处的模拟信号输出

RobotWare - OS

续前页

移动机械臂时正在使用的工具。工具中心点是指移动至指定目的位置的点。

[\WObj]

Work Object

数据类型： wobjdata

指令中机器人位置关联的工件（坐标系）。

可省略该参数，随后，位置与世界坐标系相关。另一方面，如果使用固定式TCP或协调的外轴，则必须指定该参数。

Signal

数据类型：signalao

待改变模拟信号输出信号的名称。

Value

数据类型：num

信号的期望值。

[\TLoad]

Total load

数据类型：loaddata

\TLoad主动轴描述了移动中使用的总负载。总负载就是相关的工具负载加上该工具正在处理的有效负载。如果使用了\TLoad自变数，那么就不考虑当前tooldata中的loaddata。

如果\TLoad自变数被设置成load0，那么就不考虑\TLoad自变数，而是以当前tooldata中的loaddata作为代替。

想要使用\TLoad自变数，就必需将系统参数ModalPayLoadMode的数值设置成0。如果将ModalPayLoadMode设置成0，那么就再也无法使用指令GripLoad。

可用服务例程“负载标识”（LoadIdentify）来识别总负载。如果系统参数ModalPayLoadMode被设置成0，且系统正在运行该服务例程，那么操作员便可将相关工具的loaddata复制到一个现有的或新的loaddata永久变量中。

如果使用了关联到系统输入项SimMode（仿真模式）上的一个数字输入信号，那么便可在没有任何有效负载的情况下试运行该程序。如果该数字输入信号被设置成1，那么就不考虑可选自变数\TLoad中的loaddata，而是以当前tooldata中的loaddata作为代替。

> **注意**
>
> 处理有效负载的默认功能是使用指令GripLoad，因此系统参数ModalPayLoadMode的默认值为1。

程序执行

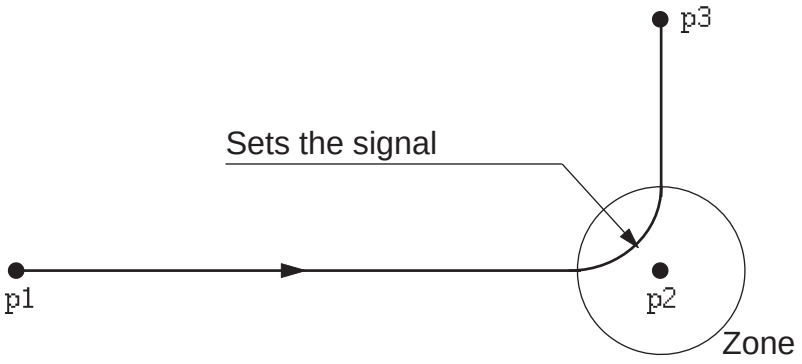
更多关于线性运动的信息，请参见指令MoveL。

将模拟信号输出信号设置在飞焊点角路径的中间位置，如下图所示。

本图显示了MoveLAO角路径中模拟信号输出信号的设置。

```
MoveLAO p2, v1000, z30, tool2, ao1, 1.1;
```

下一页继续



xx1400001118

关于停止点，我们建议使用关于MoveL和SetAO的“普通”编程顺序。但是，当使用指令MoveLAO中的停止点时，模拟信号输出信号得以在机械臂达到停止点时设置。

以连续执行模式逐步向前而非逐步向后地设置指定的I/O信号。

错误处理

系统会生成下列可恢复错误，并在错误处理器中处理这些错误。系统变量ERRNO将被设置成：

如果关于指定模拟信号输出信号Signal的编程Value参数超出限值，则ERR_AO_LIM。

如果信号变量是RAPID中声明的变量，则ERR_NO_ALIASIO_DEF。尚未同I/O配置以及指令AliasIO中确定的I/O信号相连。

如果与I/O单元无接触，则ERR_NORUNUNIT。

如果无法访问I/O信号（仅对ICI现场总线有效），则ERR_SIG_NOT_VALID。

语法

```
MoveLAO
  [ ToPoint ':=' ] < expression (IN) of robtarget >
  [ '\' ID ':=' < expression (IN) of identno >]','
  [ Speed ':=' ] < expression (IN) of speeddata >
  [ '\' T ':=' < expression (IN) of num > ] ','
  [ Zone ':=' ] < expression (IN) of zonedata > ','
  [ Tool ':=' ] < persistent (PERS) of tooldata >
  [ '\' WObj ':=' ] < persistent (PERS) of wobjdata > ','
  [ Signal ':=' ] < variable (VAR) of signalao >] ','
  [ Value ':=' ] < expression (IN) of num > ]
  [ '\' TLoad ':=' < persistent (PERS) of loaddata > ] ';’
```

相关信息

信息，关于	请参阅
其他定位指令	技术参考手册 - *RAPID*语言概览
直线移动机器人	第*388*页的*MoveL - 使机械臂沿直线移动*
载荷的定义	第*1421*页的*loaddata - 加载数据*
速度的定义	第*1480*页的*speeddata - 速度数据*
工具的定义	第*1502*页的*tooldata - 工具数据*
工件的定义	第*1523*页的*wobjdata - 工件数据*

下一页继续

1.149 MoveLAO - 使机械臂沿直线运动，设置拐角处的模拟信号输出

RobotWare - OS

续前页

信息，关于	请参阅
区域数据的定义	第*1531*页的*zonedata - 区域数据*
一般动作	技术参考手册 - *RAPID*语言概览
坐标系	技术参考手册 - *RAPID*语言概览
关于I/O设置的运动	技术参考手册 - *RAPID*语言概览
关于如何使用TLoad总负载的例子。	第*388*页的*MoveL - 使机械臂沿直线移动*
定义机器人的载荷	第*223*页的*GripLoad - 定义机械臂的有效负载*
LoadIdentify，载荷识别服务例行程序	操作员手册 - 带 *FlexPendant* 的 *IRC5*
用于在仿真模式下无载荷运行机器人的系统输入信号 *SimMode*。 （主题 I/O，类型 System Input，行动值，*SimMode*）	技术参考手册 - 系统参数
激活和停用载荷的系统参数 *ModalPayLoadMode*。 （主题 Controller，类型 System Misc，行动值，*ModalPayLoadMode*）	技术参考手册 - 系统参数

1.150 MoveLDO - 使机械臂沿直线运动，设置拐角处的数字信号输出

手册用法

MoveLDO（*Move Linearly Digital Output*）用于使工具中心点沿直线运动至给定目的地。在角路径中部设置/重置指定数字信号输出信号。

当TCP保持固定时，该指令亦可用于调整工具方位。

本指令仅可用于主任务T_ROB1，或者如果在*MultiMove*系统中，则可用于运动任务中。

基本示例

以下实例介绍了指令MoveLDO：

例 1

```
MoveLDO p1, v1000, z30, tool2, do1,1;
```

工具的TCPtool2沿直线移动至位置p1，其速度数据为v1000 ，且区域数据为z30。在p1处角路径中部，设置输出do1。

变元

```
MoveLDO ToPoint [\ID] Speed [\T] Zone Tool [\WObj] Signal Value
     [\TLoad]
```

ToPoint

数据类型： robtarget

机器人和外部轴的目标点。定义为已命名的位置或直接存储在指令中（在指令中加 * 标记）。

[\ID]

Synchronization id

数据类型： identno

如果移动是同步或协调同步的，变元 [\ID] 在 MultiMove 系统中是强制的。这个变元在任何其他情况下都不允许使用。指定的 ID 号必须与所有协作程序任务中的 ID 号相同。使用此 ID 号，动作才不会在运行时搞混。

Speed

数据类型： speeddata

适用于移动的速度数据。速度数据定义了工具中心点的速率、工具重定位以及外部轴。

[\T]

Time

数据类型：num

该参数用于规定机械臂运动的总时间，以秒计。随后，取代相关的速度数据。

Zone

数据类型： zonedata

相关移动的区域数据。区域数据描述了所生成拐角路径的大小。

Tool

数据类型： tooldata

下一页继续

1.150 MoveLDO - 使机械臂沿直线运动，设置拐角处的数字信号输出
RobotWare - OS
续前页

移动机械臂时正在使用的工具。工具中心点是指移动至指定目的位置的点。

[\WObj]

Work Object

数据类型： wobjdata

指令中机器人位置关联的工件（坐标系）。

可省略该参数，随后，位置与世界坐标系相关。另一方面，如果使用固定式TCP或协调的外轴，则必须指定该参数。

Signal

数据类型：signaldo

待改变数字信号输出信号的名称。

Value

数据类型：dionum

信号的期望值（0或1）。

[\TLoad]

Total load

数据类型：loaddata

\TLoad主动轴描述了移动中使用的总负载。总负载就是相关的工具负载加上该工具正在处理的有效负载。如果使用了\TLoad自变数，那么就不考虑当前tooldata中的loaddata。

如果\TLoad自变数被设置成load0，那么就不考虑\TLoad自变数，而是以当前tooldata中的loaddata作为代替。

想要使用\TLoad自变数，就必需将系统参数ModalPayLoadMode的数值设置成0。如果将ModalPayLoadMode设置成0，那么就再也无法使用指令GripLoad。

可用服务例程“负载标识”（LoadIdentify）来识别总负载。如果系统参数ModalPayLoadMode被设置成0，且系统正在运行该服务例程，那么操作员便可将相关工具的loaddata复制到一个现有的或新的loaddata永久变量中。

如果使用了关联到系统输入项SimMode（仿真模式）上的一个数字输入信号，那么便可在没有任何有效负载的情况下试运行该程序。如果该数字输入信号被设置成1，那么就不考虑可选自变数\TLoad中的loaddata，而是以当前tooldata中的loaddata作为代替。

注意

处理有效负载的默认功能是使用指令GripLoad，因此系统参数ModalPayLoadMode的默认值为1。

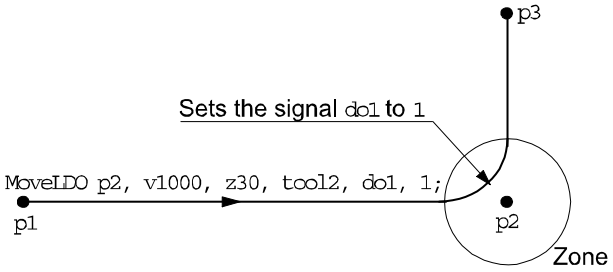
程序执行

更多关于线性运动的信息，请参见指令MoveL。

将数字信号输出信号设置/重置在飞焊点角路径的中间位置，如下图所示。

下一页继续

本图显示了MoveLDO角路径中数字信号输出信号的设置/重置。



xx0500002193

关于停止点，我们建议使用关于MoveL和SetDO的“普通”编程顺序。但是，当使用指令MoveLDO中的停止点时，数字信号输出信号得以在机械臂达到停止点时设置/重置。

以连续执行模式逐步向前而非逐步向后地设置/重置指定的I/O信号。

错误处理

系统会生成下列可恢复错误，并在错误处理器中处理这些错误。系统变量ERRNO将被设置成：

如果信号变量是RAPID中声明的变量，则ERR_NO_ALIASIO_DEF。尚未同I/O配置以及指令AliasIO中确定的I/O信号相连。

如果与I/O单元无接触，则ERR_NORUNUNIT。

如果无法访问I/O信号（仅对ICI现场总线有效），则ERR_SIG_NOT_VALID。

语法

```
MoveLDO
  [ ToPoint ':=' ] < expression (IN) of robtarget >
  [ '\' ID ':=' < expression (IN) of identno >]','
  [ Speed ':=' ] < expression (IN) of speeddata >
  [ '\' T ':=' < expression (IN) of num > ] ','
  [ Zone ':=' ] < expression (IN) of zonedata > ','
  [ Tool ':=' ] < persistent (PERS) of tooldata >
  [ '\' WObj ':=' ] < persistent (PERS) of wobjdata > ','
  [ Signal ':=' ] < variable (VAR) of signaldo >] ','
  [ Value ':=' ] < expression (IN) of dionum > ]
  [ '\' TLoad ':=' < persistent (PERS) of loaddata > ] ';'
```

相关信息

信息，关于	请参阅
其他定位指令	技术参考手册 - *RAPID*语言概览
直线移动机器人	第*388*页的*MoveL - 使机械臂沿直线移动*
载荷的定义	第*1421*页的*loaddata - 加载数据*
速度的定义	第*1480*页的*speeddata - 速度数据*
工具的定义	第*1502*页的*tooldata - 工具数据*
工件的定义	第*1523*页的*wobjdata - 工件数据*
区域数据的定义	第*1531*页的*zonedata - 区域数据*
一般动作	技术参考手册 - *RAPID*语言概览

下一页继续

1.150 MoveLDO - 使机械臂沿直线运动，设置拐角处的数字信号输出

RobotWare - OS

续前页

信息，关于	请参阅
坐标系	技术参考手册 - *RAPID*语言概览
关于I/O设置的运动	技术参考手册 - *RAPID*语言概览
关于如何使用TLoad总负载的例子。	第*388*页的*MoveL - 使机械臂沿直线移动*
定义机器人的载荷	第*223*页的*GripLoad - 定义机械臂的有效负载*
LoadIdentify，载荷识别服务例行程序	操作员手册 - 带 *FlexPendant* 的 *IRC5*
用于在仿真模式下无载荷运行机器人的系统输入信号 *SimMode*。 （主题 I/O，类型 System Input，行动值，*SimMode*）	技术参考手册 - 系统参数
激活和停用载荷的系统参数 *ModalPayLoadMode*。 （主题 Controller，类型 System Misc，行动值，*ModalPayLoadMode*）	技术参考手册 - 系统参数

1.151 MoveLGO - 使机械臂沿直线运动，设置拐角处的组输出信号

手册用法

MoveLGO（*Move Linearly Group Output*）用于使工具中心点沿直线运动至给定目的地。在角路径中部设置指定组输出信号。

当TCP保持固定时，该指令亦可用于调整工具方位。

本指令仅可用于主任务T_ROB1，或者如果在*MultiMove*系统中，则可用于运动任务中。

基本示例

以下实例介绍了指令MoveLGO：

例 1

```
MoveLGO p1, v1000, z30, tool2, go1 \Value:=5;
```

工具的TCPtool2沿直线移动至位置p1，其速度数据为v1000 ，且区域数据为z30。在p1处角路径中部，设置组输出信号go1。

变元

```
MoveLGO ToPoint [\ID] Speed [\T] Zone Tool [\WObj] Signal [\Value]
     | [\DValue] [\TLoad]
```

ToPoint

数据类型： robtarget

机器人和外部轴的目标点。定义为已命名的位置或直接存储在指令中（在指令中加 * 标记）。

[\ID]

Synchronization id

数据类型： identno

如果移动是同步或协调同步的，变元 [\ID] 在 MultiMove 系统中是强制的。这个变元在任何其他情况下都不允许使用。指定的 ID 号必须与所有协作程序任务中的 ID 号相同。使用此 ID 号，动作才不会在运行时搞混。

Speed

数据类型： speeddata

适用于移动的速度数据。速度数据定义了工具中心点的速率、工具重定位以及外部轴。

[\T]

Time

数据类型：num

该参数用于规定机械臂运动的总时间，以秒计。随后，取代相关的速度数据。

Zone

数据类型： zonedata

相关移动的区域数据。区域数据描述了所生成拐角路径的大小。

Tool

数据类型： tooldata

下一页继续

1.151 MoveLGO - 使机械臂沿直线运动，设置拐角处的组输出信号

RobotWare - OS

续前页

移动机械臂时正在使用的工具。工具中心点是指移动至指定目的位置的点。

`[ \WObj ]`

Work Object

数据类型： `wobjdata`

指令中机器人位置关联的工件（坐标系）。

可省略该参数，随后，位置与世界坐标系相关。另一方面，如果使用固定式TCP或协调的外轴，则必须指定该参数。

`Signal`

数据类型：`signalgo`

待改变组输出信号的名称。

`[ \Value ]`

数据类型：`num`

信号的期望值。

`[ \DValue ]`

数据类型：`dnum`

信号的期望值。

如果未输入参数`\Value`或`\DValue`，则将显示错误消息。

`[ \TLoad ]`

Total load

数据类型：`loaddata`

`\TLoad`主动轴描述了移动中使用的总负载。总负载就是相关的工具负载加上该工具正在处理的有效负载。如果使用了`\TLoad`自变数，那么就不考虑当前`tooldata`中的`loaddata`。

如果`\TLoad`自变数被设置成`load0`，那么就不考虑`\TLoad`自变数，而是以当前`tooldata`中的`loaddata`作为代替。

想要使用`\TLoad`自变数，就必需将系统参数`ModalPayLoadMode`的数值设置成0。如果将`ModalPayLoadMode`设置成0，那么就再也无法使用指令`GripLoad`。

可用服务例程“负载标识”（LoadIdentify）来识别总负载。如果系统参数`ModalPayLoadMode`被设置成0，且系统正在运行该服务例程，那么操作员便可将相关工具的`loaddata`复制到一个现有的或新的`loaddata`永久变量中。

如果使用了关联到系统输入项`SimMode`（仿真模式）上的一个数字输入信号，那么便可在没有任何有效负载的情况下试运行该程序。如果该数字输入信号被设置成1，那么就不考虑可选自变数`\TLoad`中的`loaddata`，而是以当前`tooldata`中的`loaddata`作为代替。

> **注意**
>
> 处理有效负载的默认功能是使用指令`GripLoad`，因此系统参数`ModalPayLoadMode`的默认值为1。

下一页继续

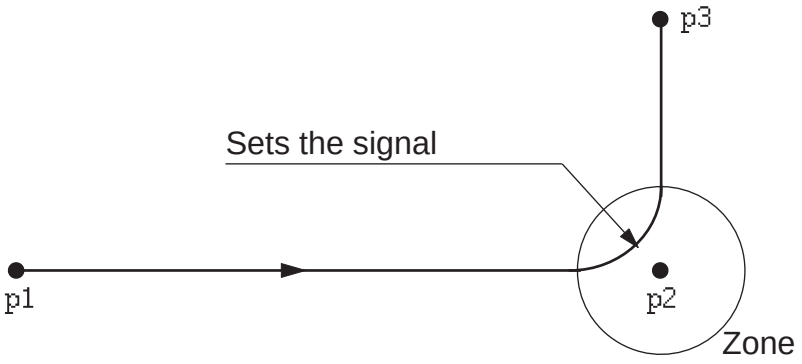
程序执行

更多关于线性运动的信息，请参见指令MoveL。

将指定组输出信号设置在飞焊点角路径的中间位置，如下图所示。

本图显示了MoveLGO角路径中组输出信号的设置。

```
MoveLGO p2, v1000, z30, tool2, go1 \Value:=5;
```



xx1400001118

关于停止点，我们建议使用关于MoveL+SetGO的“普通”编程顺序。但是，当使用指令MoveLGO中的停止点时，组输出信号得以在机械臂达到停止点时设置。

以连续执行模式逐步向前而非逐步向后地设置指定的I/O信号。

错误处理

系统会生成下列可恢复错误，并在错误处理器中处理这些错误。系统变量ERRNO将被设置成：

如果信号变量是RAPID中声明的变量，则ERR_NO_ALIASIO_DEF。尚未同I/O配置以及指令AliasIO中确定的I/O信号相连。

如果与I/O单元无接触，则ERR_NORUNUNIT。

如果无法访问I/O信号（仅对ICI现场总线有效），则ERR_SIG_NOT_VALID。

语法

```
MoveLGO
  [ ToPoint ':=' ] < expression (IN) of robtarget >
  [ '\' ID ':=' < expression (IN) of identno >]','
  [ Speed ':=' ] < expression (IN) of speeddata >
  [ '\' T ':=' < expression (IN) of num > ] ','
  [ Zone ':=' ] < expression (IN) of zonedata > ','
  [ Tool ':=' ] < persistent (PERS) of tooldata >
  [ '\' WObj ':=' ] < persistent (PERS) of wobjdata > ','
  [ Signal ':=' ] < variable (VAR) of signaldo >] ','
  [ '\' Value ':=' ] < expression (IN) of num > ]
  | [ '\' Dvalue' :=' ] < expression (IN) of dnum >
  [ '\' TLoad ':=' < persistent (PERS) of loaddata > ] ';'
```

相关信息

信息，关于	请参阅
其他定位指令	技术参考手册 - *RAPID*语言概览

下一页继续

1.151 MoveLGO - 使机械臂沿直线运动，设置拐角处的组输出信号

RobotWare - OS

续前页

信息，关于	请参阅
直线移动机器人	第*388*页的*MoveL - 使机械臂沿直线移动*
载荷的定义	第*1421*页的*loaddata - 加载数据*
速度的定义	第*1480*页的*speeddata - 速度数据*
工具的定义	第*1502*页的*tooldata - 工具数据*
工件的定义	第*1523*页的*wobjdata - 工件数据*
区域数据的定义	第*1531*页的*zonedata - 区域数据*
一般动作	技术参考手册 - *RAPID*语言概览
坐标系	技术参考手册 - *RAPID*语言概览
关于I/O设置的运动	技术参考手册 - *RAPID*语言概览
关于如何使用TLoad总负载的例子。	第*388*页的*MoveL - 使机械臂沿直线移动*
定义机器人的载荷	第*223*页的*GripLoad - 定义机械臂的有效负载*
LoadIdentify，载荷识别服务例行程序	操作员手册 - 带 *FlexPendant* 的 *IRC5*
用于在仿真模式下无载荷运行机器人的系统输入信号 *SimMode*。 （主题 I/O，类型 System Input，行动值，*SimMode*）	技术参考手册 - 系统参数
激活和停用载荷的系统参数 *ModalPayLoadMode*。 （主题 Controller，类型 System Misc，行动值，*ModalPayLoadMode*）	技术参考手册 - 系统参数

1.152 MoveLSync - 机械臂沿直线运动，执行RAPID无返回值程序

手册用法

MoveLSync（*Move Linearly Synchronously*）用于使工具中心点沿直线运动至给定目的地。下达关于指定RAPID无返回值程序的命令，从而在目的点角路径中部执行。

当TCP保持固定时，该指令亦可用于调整工具方位。

本指令仅可用于主任务T_ROB1，或者如果在*MultiMove*系统中，则可用于运动任务中。

基本示例

以下实例介绍了指令MoveLSync：

例 1

```
MoveLSync p1, v1000, z30, tool2, "proc1";
```

工具的TCPtool2沿直线移动至位置p1，其速度数据为v1000 ，且区域数据为z30。在p1处角路径中部，执行无返回值程序proc1。

例 2

```
MoveLSync p1, v1000, z30, tool2, "proc1";
```

与上文例1相同，但在此处，将于角路径中部调用模块MyModule中局部声明的无返回值程序proc1。

变元

```
MoveLSync ToPoint [\ID] Speed [\T] Zone Tool [\WObj] ProcName
    [\TLoad]
```

ToPoint

数据类型：robtarget

机器人和外部轴的目标点。定义为已命名的位置或直接存储在指令中（在指令中加 * 标记）。

[\ID]

Synchronization id

数据类型： identno

如果移动是同步或协调同步的，变元 [\ID] 在 MultiMove 系统中是强制的。这个变元在任何其他情况下都不允许使用。指定的 ID 号必须与所有协作程序任务中的 ID 号相同。使用此 ID 号，动作才不会在运行时搞混。

Speed

数据类型： speeddata

适用于移动的速度数据。速度数据定义了工具中心点的速率、工具重定位以及外部轴。

[\T]

Time

数据类型：num

该参数用于规定机械臂运动的总时间，以秒计。随后，取代相关的速度数据。

下一页继续

1.152 MoveLSync - 机械臂沿直线运动，执行RAPID无返回值程序
RobotWare - OS
续前页

Zone

数据类型：zonedata

相关移动的区域数据。区域数据描述了所生成拐角路径的大小。

Tool

数据类型： tooldata

移动机械臂时正在使用的工具。工具中心点是指移动至指定目的位置的点。

[\WObj]

Work Object

数据类型： wobjdata

指令中机器人位置关联的工件（坐标系）。

可省略该参数，随后，位置与世界坐标系相关。另一方面，如果使用固定式TCP或协调的外轴，则必须指定该参数。

ProcName

Procedure Name

数据类型：string

将在目的点中角路径中部执行的RAPID无返回值程序的名称。过程调用为后期绑定调用，因此，继承了其属性。

将根据软中断等级执行本无返回值程序（参见下文中的程序执行）。

[\TLoad]

Total load

数据类型：loaddata

\TLoad主动轴描述了移动中使用的总负载。总负载就是相关的工具负载加上该工具正在处理的有效负载。如果使用了\TLoad自变数，那么就不考虑当前tooldata中的loaddata。

如果\TLoad自变数被设置成load0，那么就不考虑\TLoad自变数，而是以当前tooldata中的loaddata作为代替。

想要使用\TLoad自变数，就必需将系统参数ModalPayLoadMode的数值设置成0。如果将ModalPayLoadMode设置成0，那么就再也无法使用指令GripLoad。

可用服务例程“负载标识”（LoadIdentify）来识别总负载。如果系统参数ModalPayLoadMode被设置成0，且系统正在运行该服务例程，那么操作员便可将相关工具的loaddata复制到一个现有的或新的loaddata永久变量中。

如果使用了关联到系统输入项SimMode（仿真模式）上的一个数字输入信号，那么便可在没有任何有效负载的情况下试运行该程序。如果该数字输入信号被设置成1，那么就不考虑可选自变数\TLoad中的loaddata，而是以当前tooldata中的loaddata作为代替。

注意

处理有效负载的默认功能是使用指令GripLoad，因此系统参数ModalPayLoadMode的默认值为1。

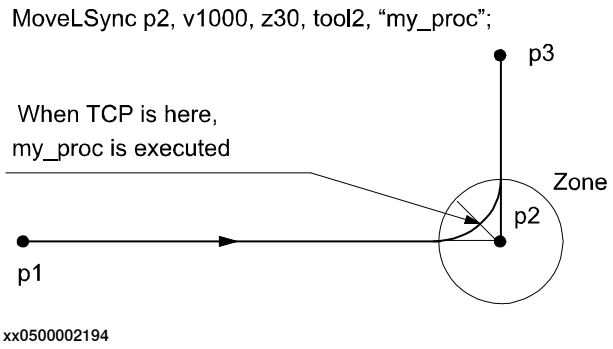
下一页继续

程序执行

更多关于线性运动的信息，请参见指令MoveL。

下达关于指定RAPID无返回值程序的指令，从而在TCP达到MoveLSync指令目的点中角路径中部时执行，如下图所示。

本图表明，在角路径中部下达关于执行用户定义的RAPID无返回值程序的指令。



关于停止点，我们建议使用关于MoveL的“普通”编程序列以及序列中的其他RAPID指令。

本表描述了以不同的执行模式来执行指定RAPID无返回值程序。

执行模式：	RAPID无返回值程序的执行
连续或循环	根据该描述
逐步向前	位于停止点
逐步向后	完全没有

MoveLSync是指令TriggInt和TriggL的封装。按照软中断等级，执行过程调用。

如果在程序停止后的减速期间，达到目的点中角路径的中部，则将不会调用过程（停止程序执行）。将在下一个程序起点执行过程调用。

限制

当机械臂达到角路径中部时，通常存在2-30 ms的延迟，直至执行指定RAPID程序，具体取决于当时正在实施运动的类型。

在程序停止后，将执行模式从连续或循环切换为逐步向前或向后，由此产生错误。该错误告知用户，模式切换可导致错过路径执行序列中RAPID无返回值程序的执行。

指令MoveLSync无法用于软中断等级。无法通过逐步执行来测试指定RAPID无返回值程序。

语法

```
MoveLSync
  [ ToPoint ':=' ] < expression (IN) of robtarget >
  [ '\' ID ':=' < expression (IN) of identno >]','
  [ Speed ':=' ] < expression (IN) of speeddata >
  [ '\' T ':=' < expression (IN) of num > ] ','
  [ Zone ':=' ] < expression (IN) of zonedata > ','
  [ Tool ':=' ] < persistent (PERS) of tooldata >
  [ '\' WObj ':=' < persistent (PERS) of wobjdata > ] ','
  [ ProcName ':=' ] < expression (IN) of string > ]
  [ '\' TLoad ':=' < persistent (PERS) of loaddata > ] ';'
```

下一页继续

1.152 MoveLSync - 机械臂沿直线运动，执行RAPID无返回值程序
RobotWare - OS
续前页

相关信息

信息，关于	请参阅
其他定位指令	技术参考手册 - *RAPID*语言概览
直线移动机器人	第*388*页的*MoveL - 使机械臂沿直线移动*
载荷的定义	第*1421*页的*loaddata - 加载数据*
速度的定义	第*1480*页的*speeddata - 速度数据*
工具的定义	第*1502*页的*tooldata - 工具数据*
工件的定义	第*1523*页的*wobjdata - 工件数据*
区域数据的定义	第*1531*页的*zonedata - 区域数据*
一般动作	技术参考手册 - *RAPID*语言概览
坐标系	技术参考手册 - *RAPID*语言概览
定义与位置相关的中断	第*769*页的*TriggInt - 定义与位置相关的中断*
关于事件的机械臂线性运动	第*785*页的*TriggL - 关于事件的机械臂线性运动*
关于如何使用TLoad总负载的例子。	第*388*页的*MoveL - 使机械臂沿直线移动*
定义机器人的载荷	第*223*页的*GripLoad - 定义机械臂的有效负载*
LoadIdentify，载荷识别服务例行程序	操作员手册 - 带 *FlexPendant* 的 *IRC5*
用于在仿真模式下无载荷运行机器人的系统输入信号 *SimMode*。 （主题 I/O，类型 System Input，行动值，*SimMode*）	技术参考手册 - 系统参数
激活和停用载荷的系统参数 *ModalPayLoadMode*。 （主题 Controller，类型 System Misc，行动值，*ModalPayLoadMode*）	技术参考手册 - 系统参数

1.153 MToolRotCalib - 移动工具旋转校准

手册用法

MToolRotCalib *(Moving Tool Rotation Calibration)* 用于校准移动工具的旋转。

机械臂及其运动的位置始终与其工具坐标系相关，即TCP和工具方位。为获得最佳精度，重要的是尽可能正确地定义工具坐标系。

通过使用FlexPendant示教器的手动方法，亦可完成校准（在操作手册 - 带*FlexPendant*示教器的*IRC5*，编程与测试一节中有做描述）。

描述

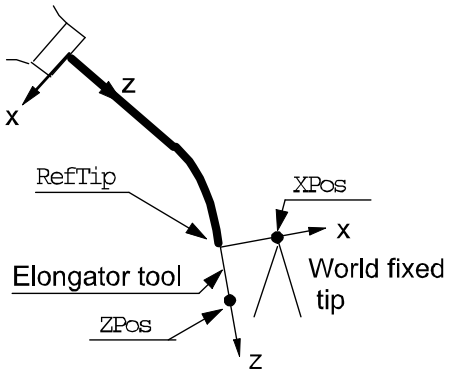
为定义工具方位，需要位于机械臂工作场所内的世界固定电极头。

在使用指令MToolRotCalib之前，必须满足一些先决条件：

- 必须将有待校准的工具安装在机械臂上，并通过正确的部件robhold(TRUE)进行定义。
- 如果使用具有绝对精度的机械臂，则应当已经确定工具的负载和重心。LoadIdentify可用于负载定义。
- 必须已经确定工具的TCP值。可通过指令MToolTCPCalib.来完成校准
- 在机械臂点动之前，必须启用tool0、wobj0和 PDispOff 。
- 手动控制尽可能靠近世界固定电极头（工具坐标系起点）的实际工具的TCP，并定义参考点RefTip.的 jointtarget
- 在不改变工具方位的前提下，手动控制机械臂，以便世界固定电极头指向工具坐标系正z轴上的某点，并定义关于点ZPos.的jointtarget
- 在不改变工具方位的前提下，随意地手动控制机械臂，以便世界固定电极头指向工具坐标系正x轴上的某点，并定义关于点XPos.的jointtarget

可使用某种类型的延伸器工具，来帮助指出正z轴和x轴。

关于RefTip、ZPos和可选XPos的接头位置的定义，请参见下图。



xx0500002192

注意

不建议修改指令MToolRotCalib中的位置RefTip、ZPos和XPos。

下一页继续

1.153 MToolRotCalib - 移动工具旋转校准

RobotWare - OS

续前页

基本示例

以下实例介绍了指令MToolRotCalib：

例 1

```
! Created with the world fixed tip pointing at origin, positive
! z-axis, and positive x-axis of the wanted tool coordinate
! system.
CONST jointtarget pos_tip := [...];
CONST jointtarget pos_z := [...];
CONST jointtarget pos_x := [...];

PERS tooldata tool1:= [ TRUE, [[20, 30, 100], [1, 0, 0 ,0]], [0.001,
     [0, 0, 0.001], [1, 0, 0, 0], 0, 0, 0]];

! Instructions for creating or ModPos of pos_tip, pos_z, and pos_x
MoveAbsJ pos_tip, v10, fine, tool0;
MoveAbsJ pos_z, v10, fine, tool0;
MoveAbsJ pos_x, v10, fine, tool0;

! Only tool calibration in the z direction
MToolRotCalib pos_tip, pos_z, tool1;
```

计算tool1z方向的工具方位(tframe.rot)。计算工具方位的x和y方向，以同腕坐标系相符。

例 2

```
! Calibration with complete tool orientation
MToolRotCalib pos_tip, pos_z \XPos:=pos_x, tool1;
```

计算tool1的完整工具方位（tframe.rot）。

变元

```
MToolRotCalib RefTip ZPos [\XPos]Tool
```

RefTip

数据类型：jointtarget

工具的TCP的所在点指向世界固定电极头。

ZPos

数据类型：jointtarget

确定正z方向的延伸器点。

[\XPos]

数据类型：jointtarget

确定x正方向的延伸器点。如果省略该点，则工具的x和y方向将符合腕坐标系中的相关轴。

Tool

数据类型： tooldata

有待校准的工具的永久变量。

下一页继续

程序执行

本系统计算和更新指定tooldata中的工具方位（tframe.rot）。计算基于指定2或3jointtarget。tooldata中的剩余数据，例如，TCP(tframe.trans)，不会发生改变。

语法

```
MToolRotCalib
  [ RefTip ':=' ] < expression (IN) of jointtarget > ','
  [ ZPos ':=' ] < expression (IN) of jointtarget >
  [ '\' XPos ':=' < expression (IN) of jointtarget > ] ','
  [ Tool ':=' ] < persistent (PERS) of tooldata > ';'
```

相关信息

信息，关于	请参阅
关于移动工具的TCP的校准	第*412*页的*MToolTCPCalib - 关于移动工具的TCP的校准*
关于固定工具的TCP的校准	第*682*页的*SToolTCPCalib - 关于固定工具的TCP的校准*
关于固定工具的TCP和旋转的校准	第*679*页的*SToolRotCalib - 关于固定工具的TCP和旋转的校准*

1.154 MToolTCPCalib - 关于移动工具的TCP的校准

手册用法

`MToolTCPCalib`（*Moving Tool TCP Calibration*）用于校准移动工具的工具中心点-TCP。

机械臂及其运动的位置始终与其工具坐标系相关，即TCP和工具方位。为获得最佳精度，重要的是尽可能正确地定义工具坐标系。

通过使用FlexPendant示教器的手动方法，亦可完成校准（在操作手册 - 带*FlexPendant*示教器的*IRC5*，编程与测试一节中有做描述）。

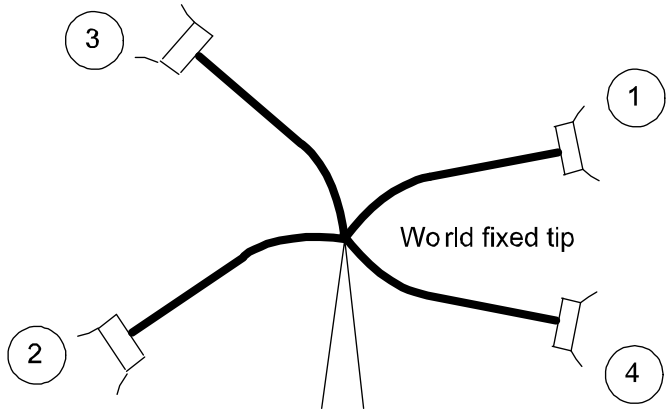
描述

为定义工具的TCP，需要位于机械臂工作场所内的世界固定电极头。

在使用指令`MToolTCPCalib`之前，必须满足一些先决条件：

- 必须将有待校准的工具安装在机械臂上，并通过正确的部件`robhold (TRUE)`进行定义。
- 如果使用具有绝对精度的机械臂，则应当已经确定工具的负载和重心。`LoadIdentify`可用于负载定义。
- 在机械臂点动之前，必须启用`tool0`、`wobj0`和`PDispOff` 。
- 手动控制尽可能靠近世界固定电极头的实际工具的TCP，并定义第一个点`p1`的`jointtarget`。
- 进一步定义具有不同姿态的三个位置（`p2`、 `p3`和`p4`）。

4个接头位置（p1....p4）的定义，请参见下图。



xx0500002191

注意

不建议修改指令`MToolTCPCalib`中的位置`Pos1`至`Pos4`。

4个位置之间的姿态调整应尽可能大，将机械臂置于不同的配置中。在校准后检查TCP的质量亦为一种良好的做法。可通过工具的方位调整来实施上述操作，以检查TCP是否静止不动。

下一页继续

基本示例

以下实例介绍了指令MToolTCPCalib：

例 1

```
! Created with actual TCP pointing at the world fixed tip
CONST jointtarget p1 := [...];
CONST jointtarget p2 := [...];
CONST jointtarget p3 := [...];
CONST jointtarget p4 := [...];

PERS tooldata tool1:= [TRUE, [[0, 0, 0], [1, 0, 0 ,0]], [0.001,
     [0, 0, 0.001], [1, 0, 0, 0], 0, 0, 0]];
VAR num max_err;
VAR num mean_err;
...
! Instructions for createing or ModPos of p1 - p4
MoveAbsJ p1, v10, fine, tool0;
MoveAbsJ p2, v10, fine, tool0;
MoveAbsJ p3, v10, fine, tool0;
MoveAbsJ p4, v10, fine, tool0;
...
MToolTCPCalib p1, p2, p3, p4, tool1, max_err, mean_err;
```

将校准和更新tool1的TCP值（tframe.trans）。max_err和mean_err将分别维持与计算出的TCP的最大误差以及平均误差，以mm计。

变元

```
MToolTCPCalib Pos1 Pos2 Pos3 Pos4 Tool MaxErr MeanErr
```

Pos1

数据类型：jointtarget

第一个进刀点。

Pos2

数据类型：jointtarget

第二个进刀点。

Pos3

数据类型：jointtarget

第三个进刀点。

Pos4

数据类型：jointtarget

第四个进刀点。

Tool

数据类型： tooldata

有待校准的工具的永久变量。

MaxErr

数据类型：num

下一页继续

1.154 MToolTCPCalib - 关于移动工具的TCP的校准

RobotWare - OS

续前页

一个进刀点的最大误差，以mm计。

`MeanErr`

数据类型：`num`

进刀点距离计算出的TCP的平均距离，即如何准确地定位相对于焊嘴的机械臂。

程序执行

本系统计算和更新指定`tooldata`中腕坐标系（`tfame.trans`）中的TCP值。计算基于指定4`jointtarget`。tooldata中的剩余数据，例如，工具方位（`tframe.rot`），不会发生改变。

语法

```
MToolTCPCalib
  [ Pos1 ':=' ] < expression (IN) of jointtarget > ','
  [ Pos2 ':=' ] < expression (IN) of jointtarget > ','
  [ Pos3 ':=' ] < expression (IN) of jointtarget > ','
  [ Pos4 ':=' ] < expression (IN) of jointtarget > ','
  [ Tool ':=' ] < persistent (PERS) of tooldata > ','
  [ MaxErr ':=' ] < variable (VAR) of num > ','
  [ MeanErr ':=' ] < variable (VAR) of num > ';'
```

相关信息

信息，关于	请参阅
关于移动工具的旋转的校准	第*409*页的*MToolRotCalib - 移动工具旋转校准*
关于固定工具的TCP的校准	第*682*页的*SToolTCPCalib - 关于固定工具的TCP的校准*
关于固定工具的TCP和旋转的校准	第*679*页的*SToolRotCalib - 关于固定工具的TCP和旋转的校准*

1.155 Open - 打开文件或串行通道

手册用法

Open用于打开文件或串行通道，以进行读取或写入。

基本示例

以下实例介绍了指令Open：

另请参阅第*417*页的更多示例

例 1

```
VAR iodev logfile;
...
Open "HOME:" \File:= "LOGFILE1.DOC", logfile \Write;
```

打开单元HOME:中的文件LOGFILE1.DOC，以进行写入。随后，当写入文件时，在程序中使用参考名称logfile。

例 2

```
VAR iodev logfile;
...
Open "LOGFILE1.DOC", logfile \Write;
```

结果与例1相同。默认路径为HOME:。

变元

```
Open Object [\File] IODevice [\Read] | [\Write] | [\Append] [\Bin]
```

Object

数据类型：string

有待打开的I/O工件（I/O设备），例如，"HOME:"、"TEMP:"、"com1:"或"pc:"（选项）。

本表描述了机器人控制器上的不同I/O设备。

I/O设备名称	I/O设备的类型
"HOME:"或diskhome[i]	SD卡
"TEMP:"或disktemp[i]	SD卡
"RemovableDisk1:"或usbdisk1[i] "RemovableDisk2:"或usbdisk2[i] "RemovableDisk3:"或usbdisk3[i]	例如，USB记忆棒
"com1:"[ii]	串行通道
"pc:"[iii]	挂载磁盘
"RAMDISK:"或diskram[i, iv]	RAM磁盘存储器

[i] 定义设备名称的RAPID字符串。
[ii] 用户在系统参数中定义的串行通道名称。
[iii] 在系统参数中定义的应用层协议、服务器路径。
[iv] RAM磁盘存储器不适用永久地储存所有数据。容量接近100 Mb，并在每次关闭时清除。

下一页继续

1.155 Open - 打开文件或串行通道
RobotWare - OS
续前页

下表描述了虚拟控制器上的不同I/O设备。

I/O设备名称	I/O设备的类型
"HOME:"或diskhome[i]	
"TEMP:"或disktemp	硬盘驱动器
"RemovableDisk1:"或usbdisk1 "RemovableDisk2:"或usbdisk2 "RemovableDisk3:"或usbdisk3	例如，USB记忆棒
"RAMDISK:"或diskram[i]	硬盘驱动器 ..\TEMP（指向TEMP文件夹，该文件夹位于虚拟系统中的相同文件夹中）

[i] 定义设备名称的RAPID字符串。

[\File]

数据类型：string

待打开的文件夹的名称，例如，"LOGFILE1.DOC"或"LOGDIR/LOGFILE1.DOC"

同时在参数Object、 "HOME:/LOGDIR/LOGFILE.DOC".中指定了完整的路径

IODevice

数据类型：iodev

关于待打开文件或串行通道的引用。随后，该参考用于从文件或串行通道读取或写入。

[\Read]

数据类型：switch

打开文件或串行通道进行读取。当从文件读取时，应当从文件开头开始读取。

[\Write]

数据类型：switch

打开文件或串行通道进行写入。如果选定的文件已经存在，则删除其内容。在文件开头写入后续内容。

[\Append]

数据类型：switch

打开文件或串行通道进行写入。如果选定的文件已经存在，则在文件末尾写入后续内容。

通过\Append且不通过\Bin参数，打开文件或串行通道。为进行写入，本指令打开基于字符的文件或串行通道。

通过\Append和\Bin参数，打开文件或串行通道。为进行读取和写入，本指令打开二进制文件或串行通道。参数\Read、\Write、\Append互相排斥。如果未指定上述任何方面，则本指令与基于字符的文件或串行通道的\Write参数（不含\Bin参数的指令）的作用方式相同，并且与二进制文件或串行通道的\Append参数（含有\Bin参数的指令）的作用方式相同。

[\Bin]

数据类型：switch

以二进制模式打开文件或串行通道。如果未指定参数\Read、\Write或\Append，则本指令打开二进制文件或串行通道以进行读取和写入，并使文件指针位于文件末尾。

下一页继续

如果有需要，Rewind指令可用于将文件指针设置到文件开头。

访问二进制文件或串行通道的指令设置不同于访问基于字符的文件的指令设置。

更多示例

有关于如何使用指令Open的更多例子阐述如下。

例 1

```
VAR iodev printer;
...
Open "com1:", printer \Bin;
WriteStrBin printer, "This is a message to the printer\0D";
Close printer;
```

打开串行通道com1:，以进行二级制读取和写入。随后，当写入和关闭串行通道时，使用参考名称printer。

例 2

```
VAR iodev io_device;
VAR rawbytes raw_data_out;
VAR rawbytes raw_data_in;
VAR num float := 0.2;
VAR string answer;

ClearRawBytes raw_data_out;
PackDNHeader "10", "20 1D 24 01 30 64", raw_data_out;
PackRawBytes float, raw_data_out, (RawBytesLen(raw_data_out)+1)
     \Float4;

Open "/FCI1:/dsqc328_1", io_device \Bin;
WriteRawBytes io_device, raw_data_out;
ReadRawBytes io_device, raw_data_in \Time:=1;
Close io_device;

UnpackRawBytes raw_data_in, 1, answer \ASCII:=10;
```

在该例子中，清除raw_data_out，随后加载DeviceNet标题以及大小为0.2的浮动值。

打开设备"/FCI1/:dsqc328_1"，将当前raw_data_out中的有效数据写入到设备。随后，本程序最多等待1秒，以从设备进行读取，该程序储存在raw_data_in。

在关闭设备“/FCI1/:dsqc328_1”后，打开读取的数据，以作为一连串10字符，并储存在名为answer的字符串中。

程序执行

打开指定文件或串行通道，以便有可能进行读取或写入。

有可能同时多次打开相同的物理文件，但是每次调用Open指令，均将返回不同的引用到文件（数据类型iodev）。例如，有可能使一个写入指针和一个不同的读取指针同时指向相同的文件。

当打开文件或串行通道时，必须自由使用iodev变量。如果先前已经使用了该变量来打开文件，则在发出关于相同iodev变量的新Open指令之前，必须关闭该文件。

下一页继续

1.155 Open - 打开文件或串行通道

RobotWare - OS

续前页

当程序停止且移动PP到Main时，将关闭程序任务中所有打开的文件或串行通道，并将重置 iodev型变量中的I/O描述符。全局VAR或LOCAL VAR型系统中安装、共享的变量可不遵循上述规则。仍将打开属于整个系统的上述文件或串行通道。

上电失败重启时，将关闭系统中所有打开的文件或串行通道，并将重置 iodev型变量中的I/O描述符。

错误处理

如果不能打开文件，则将系统变量ERRNO设置为ERR_FILEOPEN。随后，可用错误处理器对该错误进行处理。

语法

```
Open
  [ Object ':='] <expression (IN) of string>
  [ '\' File ':=' <expression (IN) of string>] ','
  [ IODevice ':='] <variable (VAR) of iodev>
  [ '\' Read] |
  [ '\' Write] |
  [ '\' Append]
  [ '\' Bin] ';'
```

相关信息

信息，关于	请参阅
写入、读取和关闭文件或串行通道	技术参考手册 - *RAPID*语言概览
现场总线命令接口 文件和串行通道处理	应用手册 - 控制器软件*IRC5*

1.156 OpenDir - 打开路径

手册用法

OpenDir用于打开路径，以进行进一步调查。

基本示例

以下实例介绍了指令OpenDir：

例 1

```
PROC lsdir(string dirname)
  VAR dir directory;
  VAR string filename;
  OpenDir directory, dirname;
  WHILE ReadDir(directory, filename) DO
    TPWrite filename;
  ENDWHILE
  CloseDir directory;
ENDPROC
```

此例子打印出指定路径下所有文件或子路径的名称。

变元

```
OpenDir Dev Path
```

Dev

数据类型： dir

关于路径的变量，由OpenDir取得。随后，该变量用于从路径进行读取。

Path

数据类型：string

路径。

限制

读取之后，始终由用户关闭已打开的路径（指令CloseDir）。

错误处理

如果路径点通往不存在的路径，或者同时打开了过多的路径，则将系统变量ERRNO设置为ERR_FILEACC。随后，可用错误处理器来处理该错误。

语法

```
OpenDir
  [ Dev ':=' ] < variable (VAR) of dir> ','
  [ Path ':=' ] < expression (IN) of string> ';'
```

相关信息

信息，关于	请参阅
目录	第*1384*页的*dir - 路径结构*
建立路径	第*322*页的*MakeDir - 创建新路径*

下一页继续

1.156 OpenDir - 打开路径
RobotWare - OS
续前页

信息，关于	请参阅
删除路径	第*503*页的*RemoveDir - 删除路径*
读取路径	第*1197*页的*ReadDir - 读取路径中的下个条目*
关闭路径	第*117*页的*CloseDir - 关闭路径*
删除文件	第*504*页的*RemoveFile - 删除文件*
重命名文件	第*507*页的*RenameFile - 重命名文件*
文件和串行通道处理	应用手册 - 控制器软件*IRC5*

1.157 PackDNHeader - 将DeviceNet标题装入原始数据字节数据中。

手册用法

PackDNHeader用于将DeviceNet显式消息的标题加载到rawbytes型容器中。

随后，可用指令PackRawBytes来设置DeviceNet消息的数据部分。

基本示例

以下实例介绍了指令PackDNHeader：

例 1

```
VAR rawbytes raw_data;

PackDNHeader "0E", "6,20 01 24 01 30 06,9,4", raw_data;
```

通过服务模式"0E"和路径字符串"6,2001 24 01 30 06,9,4"，将DeviceNet显式消息的标题加载到相应的raw_data中，以便从某些I/O单元获取序列号。

在不向消息填充其他数据的情况下，准备发送该消息。

例 2

```
VAR rawbytes raw_data;

PackDNHeader "10", "20 1D 24 01 30 64", raw_data;
```

通过服务模式"10"和路径字符串"201D 24 01 30 64"，将DeviceNet显式消息的标题加载到相应的raw_data中，从而为某些I/O单元insignal 1上的上升沿设置滤波时间。

必须在该消息中增加关于滤波时间的数据。可通过始于索引RawBytesLen(raw_data)+1的指令PackRawBytes，满足上述要求（在PackDNHeader之后完成）。

变元

```
PackDNHeader Service Path RawData
```

Service

数据类型：string

有待完成的服务，例如，获取或设置属性。通过字符串中的十六进制代码进行指定，例如"IF"。

字符串长度	2个字符
格式	'0' -' 9', 'a' -'f', 'A' - 'F'
范围	"00" - "FF"

在EDS文件中发现关于Service的值。欲知更多的描述，请参见已打开的DeviceNet供应商协会*ODVA DeviceNet Specification revision 2.0*文件。

Path

数据类型：string

在EDS文件中发现关于Path的值。欲知更多的描述，请参见已打开的DeviceNet供应商协会*ODVA DeviceNet*规范第*2.0*版。

下一页继续

1.157 PackDNHeader - 将DeviceNet标题装入原始数据字节数据中。
RobotWare - OS
续前页

针对长字符串格式（例如，“6,20 1D 24 01 30 64,8,1”）和短字符串格式（例如，“20 1D 24 01 30 64”）的支持。

RawData

数据类型：rawbytes
用以加载消息标题数据的变量容器，始于RawData中的索引1。

程序执行

程序执行期间，DeviceNet消息RawData容器为：

- 第一次完全清除
- 随后，在标题部分加载数据

格式化DeviceNet标题

指令PackDNHeader将创建以下格式的DeviceNet消息标题：

原始数据标题格式	无字节数	注释
格式	1	关于DeviceNet的内部IRC5代码
运行	1	关于服务的十六进制代码
路径的尺寸	1	以字节计
路径	x	ASCII字符

随后，可通过指令PackRawBytes来设置DeviceNet消息的数据部分，该指令始于通过(RawBytesLen(my_rawdata)+1)而获取的索引。

语法

```
PackDNHeader
  [ Service ':=' ] < expression (IN) of string> ','
  [ Path ':=' ] < expression (IN) of string> ','
  [ RawData ':=' ] < variable (VAR) of rawbytes> ';'
```

相关信息

信息，关于	请参阅
rawbytes数据	第*1455*页的原始数据字节 - 原始数据
获取rawbytes数据的长度	第*1193*页的*RawBytesLen* - 获取原始数据字节数据的长度
清除rawbytes数据的内容	第*110*页的*ClearRawBytes* - 清除原始数据字节数据的内容
复制rawbytes数据的内容	第*128*页的*CopyRawBytes* - 复制原始数据字节数据的内容
将数据装至rawbytes数据	第*424*页的*PackRawBytes* - 将数据装入原始数据字节数据
写入rawbytes数据	第*923*页的*WriteRawBytes* - 写入原始数据字节数据
读取rawbytes数据	第*499*页的*ReadRawBytes* - 读取原始数据字节数据
打开来自rawbytes数据的数据	第*845*页的*UnpackRawBytes* - 打开来自原始数据字节数据的数据
位/字节功能	技术参考手册 - *RAPID*语言概览

下一页继续

续前页

信息，关于	请参阅
字符串功能	技术参考手册 - *RAPID*语言概览
文件和串行通道处理	应用手册 - 控制器软件*IRC5*

1.158 PackRawBytes - 将数据装入原始数据字节数据

手册用法

PackRawBytes用于将num、dnum、byte或string类变量的内容装到rawbytes.类容器中

基本示例

以下实例介绍了指令PackRawBytes：

```
VAR rawbytes raw_data;
VAR num integer := 8;
VAR dnum bigInt := 4294967295;
VAR num float := 13.4;
VAR byte data1 := 122;
VAR byte byte1;
VAR string string1:="abcdefg";
PackDNHeader "10", "20 1D 24 01 30 64", raw_data;
```

将DeviceNet的标题装到raw_data中。

随后，通过PackRawBytes，将要求的现场总线数据装入raw_data中。以下例子显示了如何添加不同的数据。

例 1

```
PackRawBytes integer, raw_data, (RawBytesLen(raw_data)+1) \IntX :=
    DINT;
```

在raw_data中的标题后，下4个字节的内容将为8，小数。

例 2

```
PackRawBytes bigInt, raw_data, (RawBytesLen(raw_data)+1) \IntX :=
    UDINT;
```

在raw_data中的标题后，下4个字节的内容将为4294967295，小数。

例 3

```
PackRawBytes bigInt, raw_data, (RawBytesLen(raw_data)+1) \IntX :=
    LINT;
```

在raw_data中的标题后，下8个字节的内容将为4294967295，小数。

例 4

```
PackRawBytes float, raw_data, RawBytesLen(raw_data)+1) \Float4;
```

raw_data中下4个字节的内容将为13.4，小数。

例 5

```
PackRawBytes data1, raw_data, (RawBytesLen(raw_data)+1) \ASCII;
```

raw_data中下一字节的内容将为122，其ASCII代码为“z”。

例 6

```
PackRawBytes string1, raw_data, (RawBytesLen(raw_data)+1) \ASCII;
```

raw_data中下7个字节的内容将为"abcdefg"，以ASCII代码表示。

例 7

```
byte1 := StrToByte("1F" \Hex);
PackRawBytes byte1, raw_data, (RawBytesLen(raw_data)+1) \Hex1;
```

下一页继续

raw_data中下一字节的内容将为"1F"，十六进制数。

变元

```
PackRawBytes Value RawData [ \Network ] StartIndex [\Hex1]|[\IntX
    ] | [ \Float4 ] | [ \ASCII ]
```

Value

数据类型：anytype

有待装入RawData的数据。

容许的数据类型包括：num、dnum，byte或string。无法使用数组。

RawData

数据类型：rawbytes

将装入数据的变量容器。

[\Network]

数据类型：switch

表明应当将integer和float装入RawData中的大端法（网络顺序）代表中。ProfiBus和InterBus均采用大端法。

在没有该开关的情况下，将integer和float装入RawData中的小端法（非网络顺序）代表中。DeviceNet采用小端法。

仅与选项参数\IntX - UINT、UDINT、INT、DINT和\Float4相关。

StartIndex

数据类型：num

介于1和1024之间的StartIndex表明，应当将Value中包含的首字节置于RawData中。

[\Hex1]

数据类型：switch

待装入的Value具有byte格式，应当转换为十六进制格式，并储存在RawData中的1字节。

[\IntX]

数据类型：inttypes

待装入的Value具有num或dnum格式。根据该指定数据类型inttypes的常量，其为整数，且应当储存在RawData中。

请参阅 第*426*页的预定义数据。

[\Float4]

数据类型：switch

待装入的Value具有num格式，且应当作为4字节浮动值，储存在RawData中。

[\ASCII]

数据类型：switch

待装入的Value具有byte或string格式。

如果待装入的Value具有byte格式，则其将储存在RawData中，并根据字符的ASCII代码，充当翻译Value的1字节。

下一页继续

1.158 PackRawBytes - 将数据装入原始数据字节数据

RobotWare - OS

续前页

如果待装入的Value具有字符串格式（1-80字符），则其将储存在RawData中，以作为与Value所含字符数相同的ASCII字符。字符串数据并非由rawbytes类数据中的系统所终结的NULL。由程序员决定在必要时（DeviceNet所需）添加字符串标题。

必须编程参数\Hex1、\IntX、\Float4或\ASCII 之一。

允许以下组合：

数值的数据类型：	容许的选项参数：
num *)	\IntX
dnum **)	\IntX
num	\Float4
string	\ASCII（1-80个字符）
byte	\Hex1 \ASCIIob

*)必须为选定符号常量USINT、UINT、UDINT、SINT、INT或DINT数值范围内的整数。

**)必须为选定符号常量USINT、UINT、UDINT、ULINT、SINT、INT、DINT或LINT数值范围内的整数。

程序执行

程序执行期间，将来自anytype型变量的数据装入rawbytes型容器中。

将RawData变量中的有效字节当前长度设置为：

- （StartIndex + packed_number_of_bytes - 1）
- 如果在RawData变量中的最初有效字节当前长度内完成完整的装入操作，则不会改变RawData变量中最初的有效字节当前长度。

预定义数据

预定义数据类型inttypes的以下符号常量，并可将其用于确定参数\IntX中的整数。

符号常量	常量值	整数格式	整数值域
USINT	1	无符号1字节整数	0 ... 255
UINT	2	无符号2字节整数	0 ... 65 535
UDINT	4	无符号4字节整数	0 ... 8 388 608 *) 0 ... 4 294 967 295 ****)
ULINT	8	无符号8字节整数	0 ... 4 503 599 627 370 496**)
SINT	- 1	有符号1字节整数	- 128... 127
INT	- 2	有符号2字节整数	- 32 768 ... 32 767
DINT	- 4	有符号4字节整数	- 8 388 607 ... 8 388 608 *) -2 147 483 648 ... 2 147 483 647 ***)
LINT	- 8	有符号8字节整数	- 4 503 599 627 370 496... 4 503 599 627 370 496 **)

*) 关于储存数据类型num中整数的RAPID限制。

**) 关于储存数据类型dnum中整数的RAPID限制。

***)使用双数值变量和inttypeDINT时的范围。

下一页继续

****)使用双数值变量和inttypeUDINT时的范围。

语法

```
PackRawBytes
  [ Value ':=' ] < expression (IN) of anytype> ','
  [ RawData ':=' ] < variable (VAR) of rawbytes>
  [ '\' Network ] ','
  [ StartIndex ':=' ] < expression (IN) of num>
  [ '\' Hex1 ]
  | [ '\' IntX ':=' < expression (IN) of inttypes>]
  | [ '\' Float4 ]
  | [ '\' ASCII] ';'
```

相关信息

信息，关于	请参阅
rawbytes数据	第*1455*页的原始数据字节 - 原始数据
获取rawbytes数据的长度	第*1193*页的*RawBytesLen* - 获取原始数据字节数据的长度
清除rawbytes数据的内容	第*110*页的*ClearRawBytes* - 清除原始数据字节数据的内容
复制rawbytes数据的内容	第*128*页的*CopyRawBytes* - 复制原始数据字节数据的内容
将DeviceNet标题装入rawbytes数据	第*421*页的*PackDNHeader* - 将*DeviceNet*标题装入原始数据字节数据中。
写入rawbytes数据	第*923*页的*WriteRawBytes* - 写入原始数据字节数据
读取rawbytes数据	第*499*页的*ReadRawBytes* - 读取原始数据字节数据
打开来自rawbytes数据的数据	第*845*页的*UnpackRawBytes* - 打开来自原始数据字节数据的数据
位/字节功能	技术参考手册 - *RAPID*语言概览
字符串功能	技术参考手册 - *RAPID*语言概览
文件和串行通道处理	应用手册 - 控制器软件*IRC5*

1.159 PathAccLim - 降低路径沿线的TCP加速度

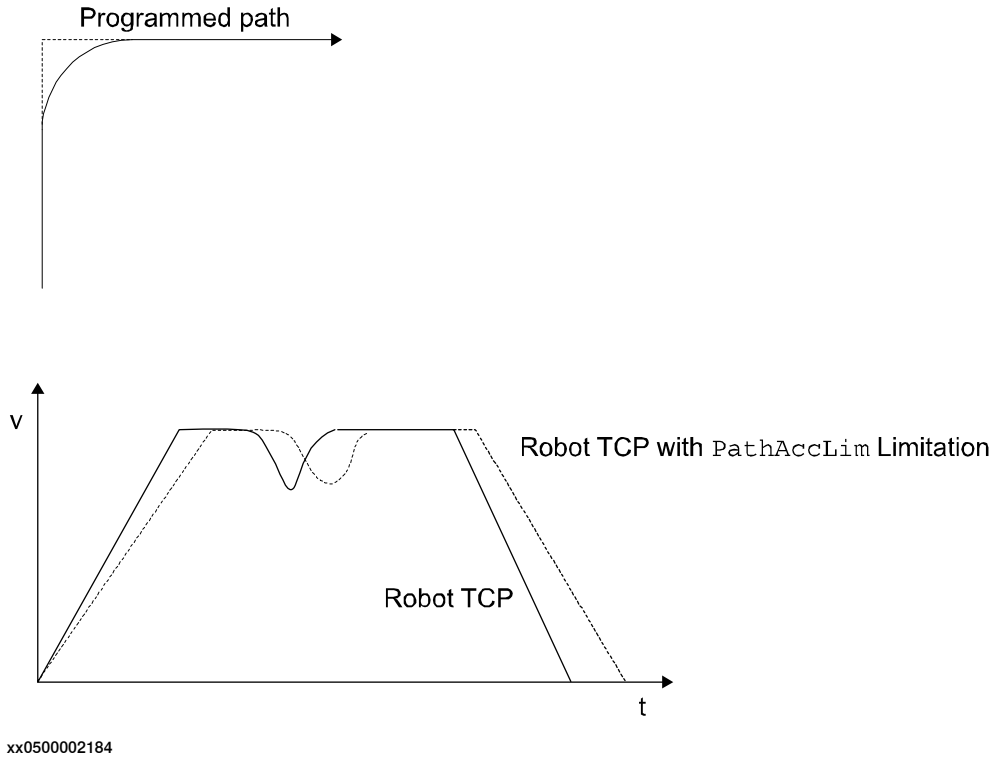
手册用法

PathAccLim（*Path Acceleration Limitation*）用于设置或重置沿运动路径的TCP加速度和/或减速度限值。

将沿运动路径（即路径坐标系中的加速度）实施限制。路径方向中的正切加速度/减速度将会受到限制。

本指令未限制设备的总加速度，即世界坐标系中的加速度，因此，其无法直接用于保护设备，以免出现较大的加速度。

本指令仅可用于主任务T_ROB1，或者如果在*MultiMove*系统中，则可用于运动任务中。



xx0500002184

基本示例

以下实例介绍了指令PathAccLim：

另请参阅第*429*页的更多示例

例 1

```
PathAccLim TRUE \AccMax := 4, TRUE \DecelMax := 4;
```

将TCP加速度和TCP减速度限制在4 m/s^2。

例 2

```
PathAccLim FALSE, FALSE;
```

将TCP加速度和减速度重置为最大（默认）。

变元

```
PathAccLim AccLim [\AccMax] DecelLim [\DecelMax]
```

下一页继续

AccLim

数据类型：bool

如果加速度存在限制，则为TRUE，否则为FALSE。

[\AccMax]

数据类型：num

加速度限制的绝对值，以m/s^{2}计。仅当AccLim为真时使用。

DecelLim

数据类型：bool

如果减速度存在限制，则为TRUE，否则为FALSE。

[\DecelMax]

数据类型：num

减速度限制的绝对值，以m/s^{2}计。仅当DecelLim为真时使用。

程序执行

加速度/减速度限制适用于下一个执行机械臂运动的命令，其始终有效，直至执行新的PathAccLim指令。

自动设置最大加速度/减速度（PathAccLim FALSE, FALSE）

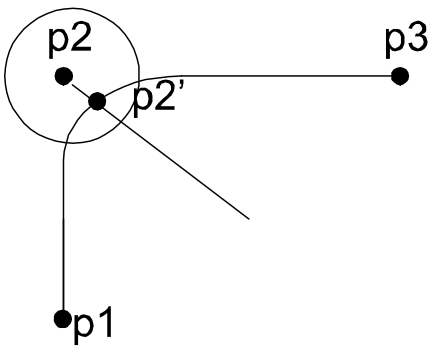
- 当使用重启模式重置RAPID时
- 当加载一则新程序或一个新模块时
- 当从起点开始执行程序时
- 当将程序指针移动到main时
- 当将程序指针移动到子程序时
- 移动程序指针造成执行顺序丢失时

如果存在指令AccSet和PathAccLim的组合，则系统以下列顺序降低加速度/减速度：

- 根据AccSet
- 根据PathAccLim

更多示例

有关于如何使用指令PathAccLim的更多例子阐述如下。



xx0500002183

下一页继续

1.159 PathAccLim - 降低路径沿线的TCP加速度

RobotWare - OS

续前页

例 1

```
MoveL p1, v1000, fine, tool0;
PathAccLim TRUE\AccMax := 4, FALSE;
MoveL p2, v1000, z30, tool0;
MoveL p3, v1000, fine, tool0;
PathAccLim FALSE, FALSE;
```

将TCP加速度限制为4 m/s^2，介于p1和p3之间。

例 2

```
MoveL p1, v1000, fine, tool0;
MoveL p2, v1000, z30, tool0;
PathAccLim TRUE\AccMax :=3, TRUE\DecelMax := 4;
MoveL p3, v1000, fine, tool0;
PathAccLim FALSE, FALSE;
```

将TCP加速度限制为3 m/s^2，介于p2'和p3之间。

将TCP减速度限制为4 m/s^2，介于p2'和p3之间。

错误处理

如果将参数\AccMax或\DecelMax设置为一个过低的值，则将系统变量ERRNO设置为ERR_ACC_TOO_LOW。随后，可用错误处理器对该错误进行处理。

限制

容许的最小加速度/减速度为0.1 m/s^2。建议使加速度和减速度限制对称，也就是使AccMax和DecelMax具有相同的值。

语法

```
PathAccLim
  [ AccLim ':=' ] < expression (IN) of bool >
  [ '\' AccMax ':=' <expression (IN) of num >] ','
  [ DecelLim ':=' ] < expression (IN) of bool>
  [ '\' DecelMax ':=' <expression (IN) of num >] ';'
```

相关信息

信息，关于	请参阅
定位器指令	技术参考手册 - *RAPID*语言概览，*RAPID*概要 - 运动一节
运动设置数据	第*1430*页的*motsetdata - 运动设置数据*
加速度的降低	第*19*页的*AccSet – 降低加速度*

1.160 PathRecMoveBwd - 将路径记录器向后移动

手册用法

PathRecMoveBwd用于使机械臂沿所记录的路径后退。

基本示例

以下实例介绍了指令PathRecMoveBwd：

另请参阅第*432*页的更多示例

例 1

```
VAR pathrecid fixture_id;
PathRecMoveBwd \ID:=fixture_id \ToolOffs:=[0, 0, 10] \Speed:=v500;
```

机械臂向后移动至程序中指令PathRecStart布下fixture_id标识符的位置。TCP沿Z方向的偏移量为10 mm，且速度设置为500 mm/s。

变元

```
PathRecMoveBwd [\ID] [\ToolOffs] [\Speed]
```

[\ID]

Identifier

数据类型：pathrecid

指定ID位置向后移动的变量。数据类型pathrecid为非值类型，仅用作命名和记录位置的标识符。

如果未指定ID位置，则单系统中，最接近所记录ID位置处出现向后移动。但是在MultiMove同步模式中，在最接近以下位置处出现向后移动：

- 返回同步移动开始的位置
- 返回记录的最接近的ID位置

[\ToolOffs]

Tool Offset

数据类型：pos

移动期间，为TCP提供间隙补偿。笛卡尔偏移量坐标适用于TCP坐标。正Z偏移量指明间隙。当机械臂运行添加材料的过程时，其较为有用。如果运行同步运动，则全部机械单元需要或不需要使用该参数。如果某些机械单元不需要任何偏移量，则可应用零偏移量。即使在不同的任务中使用TCP机械臂，也没有TCP机械单元需要使用该参数。

[\Speed]

数据类型： speeddata

取代向前移动期间使用的原始速度。Speeddata确定了工具中心点、工具方位调整和外轴的速率。如存在，则将在向后移动期间使用该速度。如省略，将以原始运动指令中的速度执行向后移动。

下一页继续

1.160 PathRecMoveBwd - 将路径记录器向后移动

Path Recovery

续前页

程序执行

通过PathRecStart指令，启用路径记录器。在起动记录器之后，将记录所有移动指令，并可通过执行PathRecMoveBwd，使机械臂在任意点沿其记录的路径向后移动。

同步运动

以同步运动模式运行路径记录器，由此增加一些注意事项。

- 在任何机械臂开始移动前，同步记录运动中涉及的所有任务，必须下达PathRecMoveBwd指令。
- 记录所有同步处理，并逆向执行。例如，如果从同步块内到独立位置，下达PathRecMoveBwd指令，则在收到SyncMoveOn指令时，路径记录器将自动把状态改变为独立。
- 将SyncMoveOn视为不带路径标识符的断点。即已经通过不含可选参数的PathRecStart和PathRecMoveBwd，起动了路径记录器，在同步运动块内执行\ID，随后，机械臂将向后移动至执行SyncMoveOn时所应在的位置。由于在SyncMoveOn之前停止向后运动，因此，状态将改变为独立。
- 将WaitSyncTask视为不含路径标识符的断点。即如果已通过PathRecStart起动了路径记录器，并执行了PathRecMoveBwd，则机械臂将移动回执行WaitSyncTask时机械臂所在位置以外的位置。

更多示例

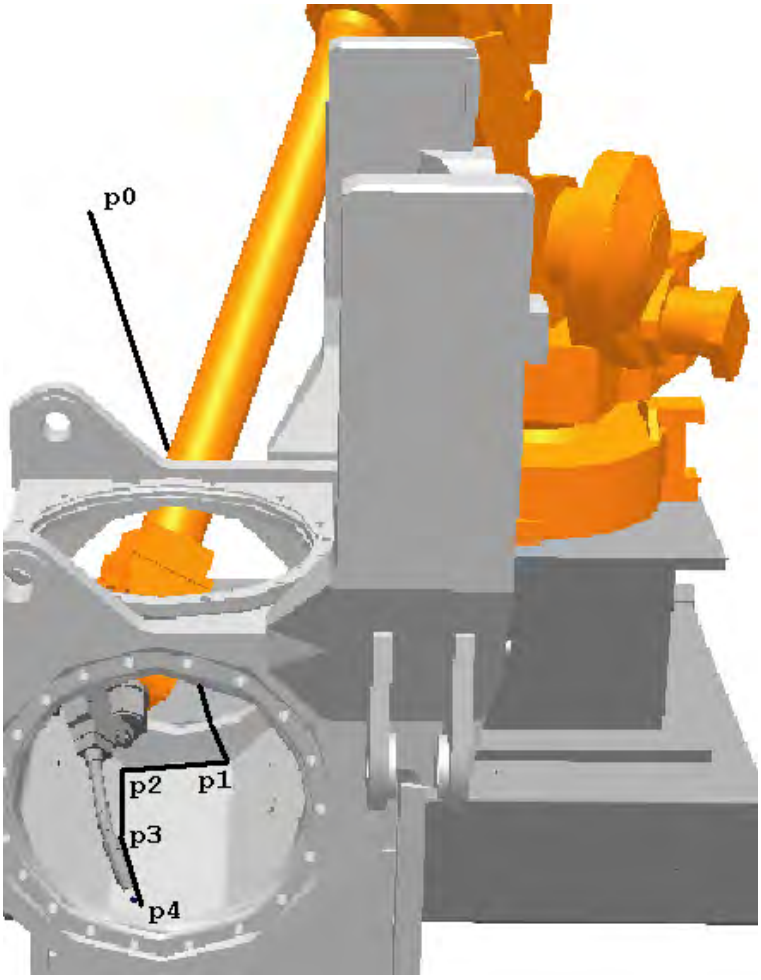
有关于如何使用指令PathRecMoveBwd的更多例子阐述如下。

例1 - 独立运动

```
VAR pathrecid safe_id;
CONST robtarget p0 := [...];
...
CONST robtarget p4 := [...];
VAR num choice;

MoveJ p0, vmax, z50, tool1;
PathRecStart safe_id;
MoveJ p1, vmax, z50, tool1;
MoveL p2, vmax, z50, tool1;
MoveL p3, vmax, z50, tool1;
MoveL p4, vmax, z50, tool1;

ERROR:
TPReadFK choice,"Go to safe?",stEmpty,stEmpty,stEmpty,stEmpty,"Yes";
  IF choice=5 THEN
    IF PathRecValidBwd(\ID:=safe_id) THEN
      StorePath;
      PathRecMoveBwd \ID:=safe_id \ToolOffs:=[0, 0 , 10];
      Stop;
      !Fix problem
      PathRecMoveFwd;
      RestoPath;
      StartMove;
      RETRY;
    ENDIF
```

下一页继续

```
ENDIF
```



xx0500002135

本例子显示了如何利用路径记录器，以从有关错误的狭窄空间提取机械臂，期间不对指定路径进行编程。

零件正在制造中。在进刀点 p0，起动路径记录器，并发出路径记录器标识符 safe_id。假定在机械臂从p3移动至p4时，出现可恢复的错误。在该点，通过执行 StorePath，以储存路径。通过储存路径，错误处理器可起动新的运动，并在随后重启原始运动。当完成路径的储存时，通过执行PathRecMoveBwd，路径记录器用于使机械臂移动至安全位置p0。

注意，应用工具偏移量，从而为新增加的焊缝提供间隙。当机械臂已经移出时，操作员可采取必要的行动，以修复错误（例如，清洁焊炬）。随后，通过 PathRecMoveFwd，使机械臂移动回错误位置。在错误位置，通过RestoPath，将路径等级切换回基础等级，并进行重复尝试。

例2 - 同步运动

T_ROB1

```
VAR pathrecid HomeROB1;
CONST robtarget pR1_10:=[...];
...
CONST robtarget pR1_60:=[...];
```

下一页继续

1.160 PathRecMoveBwd - 将路径记录器向后移动

Path Recovery

续前页

```
PathRecStart HomeROB1;
MoveJ pR1_10, v1000, z50, tGun;
MoveJ pR1_20, v1000, z50, tGun;
MoveJ pR1_30, v1000, z50, tGun;
SyncMoveOn sync1, tasklist;
MoveL pR1_40 \ID:=1, v1000, z50, tGun\wobj:=pos1;
MoveL pR1_50 \ID:=2, v1000, z50, tGun\wobj:=pos1;
MoveL pR1_60 \ID:=3, v1000, z50, tGun\wobj:=pos1;
SyncMoveOff sync2;

ERROR
  StorePath \KeepSync;
  TEST ERRNO
  CASE ERR_PATH_STOP:
    PathRecMoveBwd \ID:= HomeROB1\ToolOffs:=[0,0,10];
  ENDTEST
  !Perform service action
  PathRecMoveFwd \ToolOffs:=[0,0,10];
  RestoPath;
  StartMove;
```

T_ROB2

```
VAR pathrecid HomeROB2;
CONST robtarget pR2_10:=[...];
...
CONST robtarget pR2_50:=[...];

PathRecStart HomeROB2;
MoveJ pR2_10, v1000, z50, tGun;
MoveJ pR2_20, v1000, z50, tGun;
SyncMoveOn sync1, tasklist;
MoveL pR2_30 \ID:=1, v1000, z50, tGun\wobj:=pos1;
MoveL pR2_40 \ID:=2, v1000, z50, tGun\wobj:=pos1;
MoveL pR2_50 \ID:=3, v1000, z50, tGun\wobj:=pos1;
SyncMoveOff sync2;

ERROR
  StorePath \KeepSync;
  TEST ERRNO
  CASE ERR_PATH_STOP:
    PathRecMoveBwd \ToolOffs:=[0,0,10];
  ENDTEST
  !Perform service action
  PathRecMoveFwd \ToolOffs:=[0,0,10];
  RestoPath;
  StartMove;
```

T_ROB3

```
VAR pathrecid HomePOS1;
CONST jointtarget jP1_10:=[...];
...
```

下一页继续

```
CONST jointtarget jP1_40:=[...];

PathRecStart HomePOS1;
MoveExtJ jP1_10, v1000, z50;
SyncMoveOn sync1, tasklist;
MoveExtJ jP1_20 \ID:=1, v1000, z50;
MoveExtJ jP1_30 \ID:=2, v1000, z50;
MoveExtJ jP1_40 \ID:=3, v1000, z50;
SyncMoveOff sync2;

ERROR
  StorePath \KeepSync;
  TEST ERRNO
  CASE ERR_PATH_STOP:
    PathRecMoveBwd \ToolOffs:=[0,0,0];
  DEFAULT:
  PathRecMoveBwd \ID:=HomePOS1\ToolOffs:=[0,0,0];
  ENDTEST
  !Perform service action
  PathRecMoveFwd \ToolOffs:=[0,0,0];
  RestoPath;
  StartMove;
```

系统由在单独任务中运行的三个机械臂组成。假定T_ROB1在同步块sync1内经历了错误ERR_PATH_STOP。出现错误时，期望返回标记有路径记录器标识符HomeROB1的起始位置，以服务机械臂的外部设备。通过使用PathRecMoveBwd，并提供pathrecid标识符，以完成上述操作。

由于在同步运动期间出现错误，因此，第二个TCP机械臂T_ROB2和外轴T_POS1亦有必要下达PathRecMoveBwd指令。在开始同步运动前，此类机械臂无须进一步后退。在带有路径记录器标识符的ERR_PATH_STOP处，未提供PathRecMoveBwd，在利用SyncMoveOn后，路径记录器能够停止。注意，不含TCP的外轴仍然会增加零工具偏移量，以提供TCP机械臂如此做的可能。

在本例子中，ERROR处理器中的DEFAULT行为包括：所有机械臂首先向后同步移动，随后，朝已记录路径起点向后独立移动。通过指定PathRecMoveBwd中关于所有机械臂的\ID，以实施上述行为。

限制

在基础等级上，无法运用路径记录器来实施运动，即在PathRecMoveBwd之前，必须执行StorePath。

通过SynchMoveOff声明，不可能向后运动。

通过WaitSyncTask声明，不可能向后运动。

如果想要向后运动至同步移动开始的位置，则必须在SyncMoveOn之前，至少进行一次独立移动。

如果不需要返回至PathRecMoveBwd执行点（通过执行PathRecMoveFwd），则必须通过PathRecStop来停止PathRecorder。PathRecStop\Clear亦清除了已记录的路径。

无法在与任意下列特殊系统事件关联的RAPID程序中执行PathRecMoveBwd：PowerOn、Stop、QStop、Restart,、Reset或者Step。

下一页继续

1.160 PathRecMoveBwd - 将路径记录器向后移动

Path Recovery

续前页

语法

```
PathRecMoveBwd
  [ '\' ID ':=' < variable (VAR) of pathrecid > ]
  [ '\' ToolOffs ':=' <expression (IN) of pos> ]
  [ '\' Speed ':=' <expression (IN) of speeddata> ]';'
```

相关信息

信息，关于	请参阅
路径记录器标识符	第*1448*页的*pathrecid - 路径记录器标识符*
起动-停止路径记录器	第*440*页的*PathRecStart - 起动路径记录器* 第*442*页的*PathRecStop - 停止路径记录器*
检查有效记录的路径	第*1172*页的*PathRecValidBwd - 是否记录了有效的后退路径* 第*1175*页的*PathRecValidFwd - 是否记录了有效的前进路径*
向前移动路径记录器	第*437*页的*PathRecMoveFwd - 向前移动路径记录器*
储存-恢复路径	第*695*页的*StorePath - 发生中断时，存储路径* 第*512*页的*RestoPath - 中断之后，恢复路径*
其他定位指令	技术参考手册 - *RAPID*语言概览
错误恢复	技术参考手册 - *RAPID*语言概览

1.161 PathRecMoveFwd - 向前移动路径记录器

手册用法

PathRecMoveFwd用于将机械臂移动回执行PathRecMoveBwd的位置。向后移动期间，通过提供合格的标识符，亦可能使机械臂部分向前。

基本示例

以下实例介绍了指令PathRecMoveFwd：

另请参阅第*437*页的更多示例

例 1

```
PathRecMoveFwd;
```

机械臂向后移动至路径记录器开始向后移动的位置。

变元

```
PathRecMoveFwd [\ID] [\ToolOffs] [\Speed]
```

[\ID]

Identifier

数据类型：pathrecid

指定ID位置向前移动的变量。数据类型pathrecid为非值类型，仅用作命名和记录位置的标识符。

如果未指定任何ID位置，则原始路径上的中断位置将始终向前移动。

[\ToolOffs]

Tool Offset

数据类型：pos

运动期间，为TCP提供间隙偏移量。笛卡尔坐标适用于TCP坐标。当机械臂进行工艺添加材料时，上述情况适用。

[\Speed]

数据类型： speeddata

速度覆盖向前移动期间的原始速度。Speeddata确定了工具中心点、工具方位调整和外轴的速率。如存在，则将在向前移动期间使用该速度。如省略，将以原始运动指令中的速度执行向前移动。

程序执行

通过PathRecStart指令，启用路径记录器。在记录器已经起动后，通过执行PathRecMoveBwd，机械臂可沿其执行路径向后移动。因此，通过调用PathRecMoveFwd，可下指令以使达机械臂返回至开始执行向后移动的位置。通过提供向后移动期间通过的标识符，亦可能使机械臂部分向前。

更多示例

有关于如何使用指令PathRecMoveFwd的更多例子阐述如下。

```
VAR pathrecid start_id;
VAR pathrecid mid_id;
CONST robtarget p1 := [...];
```

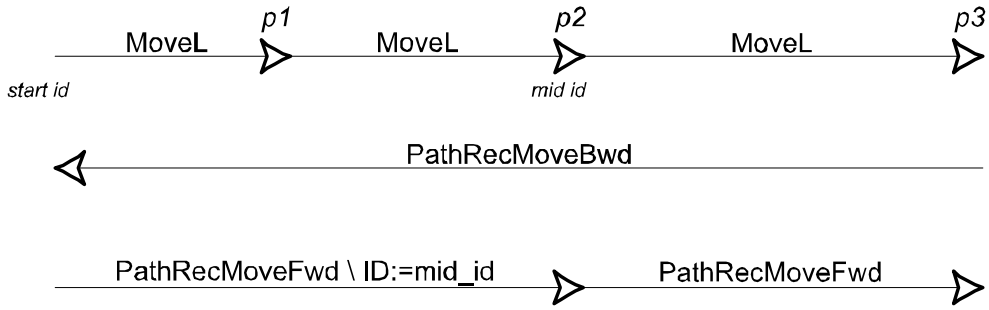
下一页继续

1.161 PathRecMoveFwd - 向前移动路径记录器

PathRecovery

续前页

```
CONST robtarget p2 := [...];
CONST robtarget p3 := [...];

  PathRecStart start_id;
  MoveL p1, vmax, z50, tool1;
  MoveL p2, vmax, z50, tool1;
  PathRecStart mid_id;
  MoveL p3, vmax, z50, tool1;
  StorePath;
  PathRecMoveBwd \ID:=start_id;
  PathRecMoveFwd \ID:=mid_id;
  PathRecMoveFwd;
  RestoPath;
```



xx0500002133

上述例子将启动路径记录器，且起动点将标记以路径标识符start_id。此后，采用传统移动指令，机械臂将向前移动，随后，通过使用已记录的路径，使路径记录器标识符start_id向后移动。最后，通过PathRecMoveFwd，其将分两步再次向前移动。

限制

必须在软中断等级上实施运用路径记录器的移动，即在PathRecMoveFwd之前，必须执行StorePath。

为了能够执行PathRecMoveFwd，必须已经执行过PathRecMoveBwd。

如果不需要返回至PathRecMoveBwd执行点（通过执行PathRecMoveFwd），则必须通过PathRecStop来停止PathRecorder。PathRecStop\Clear亦清除了已记录的路径。

无法在与任意下列特殊系统事件关联的RAPID程序中执行PathRecMoveFwd：PowerOn、Stop、QStop、Restart、Reset或者 Step。

语法

```
PathRecMoveFwd '('
  [ '\' ID ':=' < variable (VAR) of pathid > ]
  [ '\' ToolOffs ':=' <expression (IN) of pos> ]
  [ '\' Speed ':=' <expression (IN) of speeddata> ]';'
```

相关信息

信息，关于	请参阅
路径记录器标识符	第1448页的*pathrecid - 路径记录器标识符*

下一页继续

信息，关于	请参阅
起动-停止路径记录器	第*440*页的*PathRecStart* - 起动路径记录器 第*442*页的*PathRecStop* - 停止路径记录器
检查有效记录的路径	第*1172*页的*PathRecValidBwd* - 是否记录了有效的后退路径 第*1175*页的*PathRecValidFwd* - 是否记录了有效的前进路径
向后移动路径记录器	第*431*页的*PathRecMoveBwd* - 将路径记录器向后移动
储存-恢复路径	第*695*页的*StorePath* - 发生中断时，存储路径 第*512*页的*RestoPath* - 中断之后，恢复路径
其他定位指令	技术参考手册 - *RAPID*语言概览
错误恢复	技术参考手册 - *RAPID*语言概览 技术参考手册 - *RAPID*语言概览

1.162 PathRecStart - 起动路径记录器

手册用法

PathRecStart用于开始记录机械臂路径。在RAPID程序执行期间，路径记录器将储存路径信息。

基本示例

以下实例介绍了指令PathRecStart：

例 1

```
VAR pathrecid fixture_id;

PathRecStart fixture_id;
```

启动路径记录器，并在起动点（RAPID程序中的指令位置）标记标识符 fixture_id。

变元

```
PathRecStart ID
```

ID

Identifier

数据类型：pathrecid

指定记录初始位置名称的变量。数据类型pathrecid为非值类型，仅用作命名和记录位置的标识符。

程序执行

当下达路径记录器的起动指令时，将在机器人控制器内部记录机械臂路径。通过PathRecMoveBwd，已记录的程序位置序列可向后旋转，导致机械臂沿其执行的路径向后移动。

更多示例

有关于如何使用指令PathRecStart的更多例子阐述如下。

例 1

```
VAR pathrecid origin_id;
VAR pathrecid corner_id;
VAR num choice;
MoveJ p1, vmax, z50, tool1;
PathRecStart origin_id;
MoveJ p2, vmax, z50, tool1;
PathRecStart corner_id;
MoveL p3, vmax, z50, tool1;
MoveAbsJ jt4, vmax, fine, tool1;
ERROR
  TPReadFK choice,"Extract
       to:",stEmpty,stEmpty,stEmpty,"Origin","Corner";
  IF choice=4 OR choice=5 THEN
    StorePath;
    IF choice=4 THEN
      PathRecMoveBwd \ID:=origin_id;
```

下一页继续

```
    ELSE
      PathRecMoveBwd \ID:=corner_id;
    ENDIF
    Stop;
    !Fix problem
    PathRecMoveFwd;
    RestoPath;
    StartMove;
    RETRY;
  ENDIF
```

在上述例子中，如果在正常执行期间出现错误，则路径记录器用于将机械臂移动至工作位置。

机械臂正沿路径执行。在位置p1之后，路径记录器起动。在点 p2之后，插入另一个路径标识符。假定在从位置p3移动至位置jt4时，出现可恢复的错误。此时，将调用错误处理器，且用户可选择提取机械臂至位置Origin（点p1）或Corner（点p2）。随后，通过 StorePath来切换路径等级，以便能够在随后的错误位置重启。当机械臂已经从错误位置退出时，由用户决定是否解决错误（常常固定设备周围的机械臂）。

随后，下达机械臂返回错误位置的指令。将路径等级切换回正常，并再次记性尝试。

限制

仅能起动路径记录器，且仅将记录基路径等级的路径，即不予记录StorePath等级下的移动。

语法

```
PathRecStart
  [ ID ':='] < variable (VAR) of pathrecid> ';'
```

相关信息

信息，关于	请参阅
路径记录器标识符	第*1448*页的*pathrecid - 路径记录器标识符*
停止路径记录器	第*442*页的*PathRecStop - 停止路径记录器*
检查有效记录的路径	第*1172*页的*PathRecValidBwd - 是否记录了有效的后退路径* 第*1175*页的*PathRecValidFwd - 是否记录了有效的前进路径*
使路径记录器向后	第*431*页的*PathRecMoveBwd - 将路径记录器向后移动*
使路径记录器向前	第*437*页的*PathRecMoveFwd - 向前移动路径记录器*
一般动作	技术参考手册 - *RAPID*语言概览

1.163 PathRecStop - 停止路径记录器

手册用法

PathRecStop用于停止记录机械臂的路径。

基本示例

以下实例介绍了指令PathRecStop：

同时参见下文更多例子。

例 1

```
PathRecStop \Clear;
```

停止路径记录器，清除已储存路径信息的缓冲。

变元

```
PathRecStop [\Clear]
```

[\Clear]

数据类型： switch

清除已记录的路径。

程序执行

当命令路径记录器停止时，将停止记录路径。可选参数\Clear将清除已储存路径信息的缓冲，以防止错误执行已记录的路径。

在已通过PathRecStop来停止记录器之后，较早前记录的路径无法用于备份运动（PathRecMoveBwd）。如果再次从路径记录器停止的同一位置下达PathRecStart指令，则可能使用较早前记录的路径。参见以下例子。

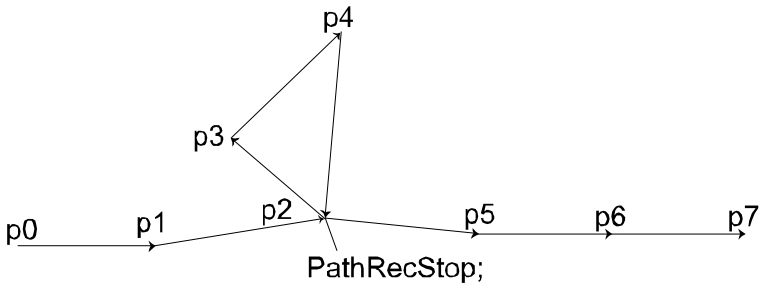
更多示例

有关于如何使用指令PathRecStop的更多例子阐述如下。

```
LOCAL VAR pathrecid id1;
LOCAL VAR pathrecid id2;
LOCAL CONST robtarget p0:= [...];
......
LOCAL CONST robtarget p6 := [...];
PROC example1()
  MoveL p0, vmax, z50, tool1;
  PathRecStart id1;
  MoveL p1, vmax, z50, tool1;
  MoveL p2, vmax, z50, tool1;
  PathRecStop;
  MoveL p3, vmax, z50, tool1;
  MoveL p4, vmax, z50, tool1;
  MoveL p2, vmax, z50, tool1;
  PathRecStart id2;
  MoveL p5, vmax, z50, tool1;
  MoveL p6, vmax, z50, tool1;
  StorePath;
  PathRecMoveBwd \ID:=id1;
```

下一页继续

```
  PathRecMoveFwd;
  RestoPath;
  StartMove;
  MoveL p7, vmax, z50, tool1;
ENDPROC
PROC example2()
  MoveL p0, vmax, z50, tool1;
  PathRecStart id1;
  MoveL p1, vmax, z50, tool1;
  MoveL p2, vmax, z50, tool1;
  PathRecStop;
  MoveL p3, vmax, z50, tool1;
  MoveL p4, vmax, z50, tool1;
  PathRecStart id2;
  MoveL p2, vmax, z50, tool1;
  MoveL p5, vmax, z50, tool1;
  MoveL p6, vmax, z50, tool1;
  StorePath;
  PathRecMoveBwd \ID:=id1;
  PathRecMoveFwd;
  RestoPath;
  StartMove;
  MoveL p7, vmax, z50, tool1;
ENDPROC
```



PathRecStop_

上述例子描述了在序列中间位置停止记录时的机械臂路径记录情形。在example1中，PathRecMoveBwd \ID:=id1;命令有效，且机械臂将执行以下路径：p6 -> p5 -> p2 -> p1 -> p0

该命令有效的原因是记录器在完全相同的机械臂位置停止和起动。如果该行为不令人满意，则停止命令应当包括可选参数 \Clear。那样，将清除已记录的路径，且永远不可能备用至先前的路径记录器标识符。

example2中的唯一区别在于记录器第二次起动。在这种情况下， PathRecMoveBwd \ID:=id1将引起错误。因为p4、p3和p2之间不存在已记录的路径。有可能执行 PathRecMoveBwd \ID:=id2。

语法

```
PathRecStop
  [ '\'switch Clear ] ';'
```

下一页继续

1.163 PathRecStop - 停止路径记录器

Path Recovery

续前页

相关信息

信息，关于	请参阅
路径记录器标识符	第*1448*页的*pathrecid - 路径记录器标识符*
启动路径记录器	第*440*页的*PathRecStart - 起动路径记录器*
检查有效记录的路径	第*1172*页的*PathRecValidBwd - 是否记录了有效的后退路径* 第*1175*页的*PathRecValidFwd - 是否记录了有效的前进路径*
使记录器向后	第*431*页的*PathRecMoveBwd - 将路径记录器向后移动*
使记录器向前	第*437*页的*PathRecMoveFwd - 向前移动路径记录器*
一般动作	技术参考手册 - *RAPID*语言概览

1.164 PathResol - 覆盖路径分辨率

手册用法

针对当前程序任务所控制的机械单元，PathResol（*Path Resolution*）用于覆盖系统参数中确定的配置几何路径样本时间。

本指令仅可用于主任务T_ROB1，或者如果在*MultiMove*系统中，则可用于任何运动任务中。

描述

路径分辨率影响插补路径和程序周期时间的准确性。当参数PathSampleTime降低时，路径准确性得以提高，且周期时间常常减小。参数PathSampleTime的值过小，可能在一些需要的应用中引起CPU负载问题。使用标准的配置路径分辨率（PathSampleTime 100%）将避免CPU负载问题，并在大多数情形下，提供足够的路径精度。

PathResol使用例子：

动态临界运动（靠近工作区域边界的最大负载、高速度、结合接头运动）可能引起CPU负载问题。增加参数PathSampleTime。

协调期间，低性能外轴可能引起CPU负载问题。增加参数PathSampleTime。

具有高频摆动的电弧焊接可能需要高分辨率的插补路径。减小参数PathSampleTime。

小圆或结合方向变化的小运动，可降低路径性能质量，并增加周期时间。减小参数PathSampleTime。

大幅度调整姿态以及较小角区的结合可引起速度变化。减小参数PathSampleTime。

基本示例

以下实例介绍了指令PathResol：

```
MoveJ p1,v1000,fine,tool1;
PathResol 150;
```

通过位于停止点的机械臂，路径样本时间得以增加至配置值的150 %。

变元

```
PathResol PathSampleTime
```

PathSampleTime

数据类型：num

覆盖配置路径样本时间的百分比。100%相当于配置路径样本时间，且介于25-400%范围内。

较低的参数PathSampleTime值会提高路径分辨率（路径精度）。

程序执行

所有后续定位指令的路径分辨率均受到影响，直至新的PathResol指令得以执行。这将会影响所有运动程序执行期间（默认路径等级以及StorePath之后的路径等级）以及点动期间的路径分辨率。

在同步协调模式下的MultiMove系统中，以下各点有效：

- 同步协调模式中涉及的所有机械单元，将匹配关于实际（使用过）的运动规划器的当前路径分辨率。

下一页继续

1.164 PathResol - 覆盖路径分辨率

RobotWare - OS

续前页

- 关于实际运动规划器的新路径分辨率指令，会影响该运动规划器中的同步协调运动和未来独立运动。
- 关于另一个运动规划器的新路径分辨率指令，仅影响该运动规划器中的未来独立运动。

关于程序任务与运动规划器之间的联系，请参见应用手册 - *MultiMove*。

路径样本时间的默认覆盖值为100%。自动设置该值

- 当使用重启模式重置RAPID时
- 当加载一则新程序或一个新模块时
- 当从起点开始执行程序时
- 当将程序指针移动到`main`时
- 当将程序指针移动到子程序时
- 移动程序指针造成执行顺序丢失时

可从部件`pathresol`中的变量`C_MOTSET`（数据类型`motsetdata`）中读取路径样本时间的当前覆盖情形。

限制

如果该指令位于移动指令之后，则必须使用停止点（zonedata `fine`）而非飞越点来编程移动指令，否则，将无法在电源故障后重启。

无法在与任意下列特殊系统事件相连的RAPID程序中执行`PathResol`：PowerOn、Stop、QStop,、Restart或者Step。

语法

```
PathResol
  [PathSampleTime ':=' ] < expression (IN) of num>' ;'
```

相关信息

信息，关于	请参阅
定位器指令	技术参考手册 - *RAPID*语言概览
运动设置	技术参考手册 - *RAPID*语言概览
路径分辨率配置	技术参考手册 - 系统参数
运动设置数据	第*1430*页的*motsetdata* - 运动设置数据

1.165 PDispOff - 停用程序位移

手册用法

PDispOff（*Program Displacement Off*）用于停用程序位移。

通过指令PDispSet或PDispOn，启用程序位移，并应用于所有运动，直至启用一些其他的程序位移，或直至停用程序位移。

本指令仅可用于主任务T_ROB1，或者如果在*MultiMove*系统中，则可用于运动任务中。

基本示例

以下实例介绍了指令PDispOff：

例 1

```
PDispOff;
```

停用程序位移。

例 2

```
MoveL p10, v500, z10, tool1;
PDispOn \ExeP:=p10, p11, tool1;
MoveL p20, v500, z10, tool1;
MoveL p30, v500, z10, tool1;
PDispOff;
MoveL p40, v500, z10, tool1;
```

将程序位移定义为位置p10和p11之间的差异。该位移会对 p20和p30的运动产生影响，但是不会对p40的运动产生影响。

程序执行

重置有效程序位移。这意味着程序位移坐标系与工件坐标系相同，因此，所有编程位置将与后者相关。

语法

```
PDispOff ´;'
```

相关信息

信息，关于	请参阅
定义使用两个位置的程序位移	第*448*页的*PDispOn - 启用程序位移*
定义使用已知坐标系的程序位移	第*452*页的*PDispSet - 启用使用已知坐标系的程序位移*

1.166 PDispOn - 启用程序位移

手册用法

PDispOn（*Program Displacement On*）用于定义和启用程序位移（使用两个机械臂位置）。

例如，在已经实施搜索之后，或在程序中若干不同位置处重复类似的运动模板时，使用程序位移。

本指令仅可用于主任务T_ROB1，或者如果在*MultiMove*系统中，则可用于运动任务中。

基本示例

以下实例介绍了指令PDispOn：

另请参阅第*450*页的更多示例

例 1

```
MoveL p10, v500, z10, tool1;
PDispOn \ExeP:=p10, p20, tool1;
```

启用程序位移（平行位移）。根据位置p10和p20之间的差异，计算程序位移。

例 2

```
MoveL p10, v500, fine \Inpos := inpos50, tool1;
PDispOn *, tool1;
```

启用程序位移（平行位移）。由于已经在先前指令中使用了准确定义的停止点，因此，无须使用参数\ExeP。根据机械臂的实际位置与指令中储存的编程点（*）之间的差异，计算位移。

例 3

```
PDispOn \Rot \ExeP:=p10, p20, tool1;
```

启用涵盖旋转的程序位移。根据位置p10和p20之间的差异，计算程序位移。

变元

```
PDispOn [\Rot] [\ExeP] ProgPoint Tool [\WObj]
```

[\Rot]

Rotation

数据类型：switch

考虑工具方位中的差异，其涉及程序的旋转。

[\ExeP]

Executed Point

数据类型： robtarget

用于计算位移的新机械臂位置。如果省略该参数，则使用程序执行时机械臂的当前位置。

ProgPoint

Programmed Point

数据类型： robtarget

编程时机械臂的原始位置。

下一页继续

Tool

数据类型： tooldata

编程期间使用的工具，即同ProgPoint位置相关的TCP。

[\WObj]

Work Object

数据类型： wobjdata

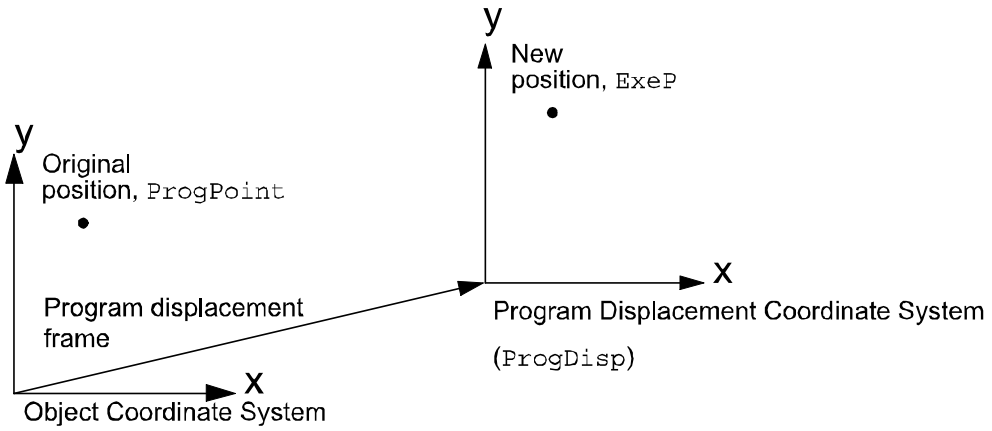
同ProgPoint位置相关的工件（坐标系）。

可省略该参数，随后，位置与世界坐标系相关。如果使用固定式TCP或协调的外轴，则必须指定该参数。

如果未编程\ExeP参数，则参数**Tool**和**\WObj**用于计算编程期间的ProgPoint，并用于计算程序执行期间的当前位置。

程序执行

程序位移意味着ProgDisp坐标系相对于工件坐标系平移。由于所有位置均与ProgDisp坐标系相关，因此，亦将取代所有编程位置。参见下图，其显示了运用程序位移的编程位置的平行位移。



xx0500002186

当执行指令PDispOn时，程序位移得以启用，并保持有效，直至一些其他的程序位移得以启用（指令PDispSet或PDispOn），或者直至程序位移停用（指令PDispOff）。

在同一时间，仅可启用唯一一个程序位移。另一方面，可陆续地编程多个PDispOn指令，在这种情况下，将增加不同的程序位移。

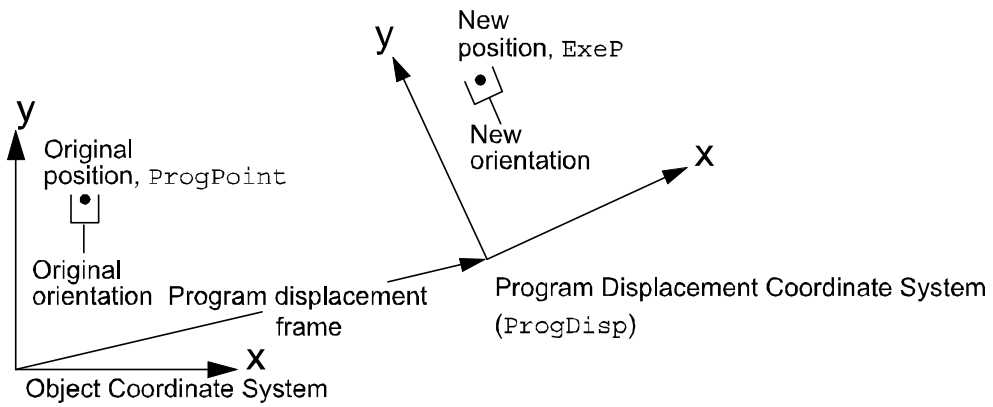
根据ExeP和ProgPoint之间的差异，计算程序位移。如果未指定ExeP，则转而使用机械臂在程序执行时的位置。由于使用机械臂的实际位置，因此，当执行PDispOn，不得移动机械臂。

如果使用参数\Rot，则亦根据两个位置处的工具方位，计算旋转。采用此种方式计算位移，以便新位置（ExeP）将拥有涉及已更换坐标系ProgDisp的相同位置和方位，

下一页继续

1.166 PDispOn - 启用程序位移
RobotWare - OS
续前页

这与涉及原始工件坐标系的旧位置（ProgPoint）相同。参见下图，其显示了编程位置的平移和旋转。



xx0500002187

自动重置程序位移

- 当使用重启模式重置RAPID时
- 当加载一则新程序或一个新模块时
- 当从起点开始执行程序时
- 当将程序指针移动到main时
- 当将程序指针移动到子程序时
- 移动程序指针造成执行顺序丢失时

更多示例

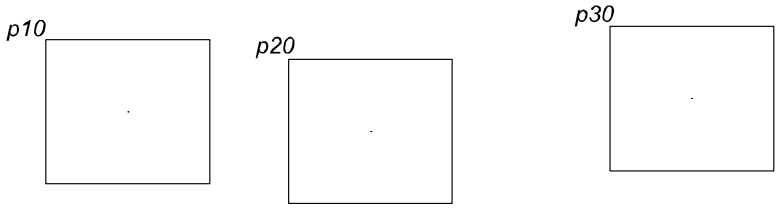
有关于如何使用指令PDispOn的更多例子阐述如下。

例 1

```
PROC draw_square()
  PDispOn *, tool1;
  MoveL *, v500, z10, tool1;
  MoveL *, v500, z10, tool1;
  MoveL *, v500, z10, tool1;
  MoveL *, v500, z10, tool1;
  PDispOff;
ENDPROC
...
MoveL p10, v500, fine \Inpos := inpos50, tool1;
draw_square;
MoveL p20, v500, fine \Inpos := inpos50, tool1;
draw_square;
MoveL p30, v500, fine \Inpos := inpos50, tool1;
draw_square;
```

下一页继续

程序draw_square用于在三个不同的位置执行相同的运动方式，这三个位置基于位置p10、 p20和p30。参见下图，其显示了使用程序位移时所能重复使用的运动方式。



xx0500002185

例 2

```
SearchL sen1, psearch, p10, v100, tool1\WObj:=fixture1;
PDispOn \ExeP:=psearch, *, tool1 \WObj:=fixture1;
```

实施搜索，并将机械臂搜索的位置储存在位置psearch中。所有于此后实施的运动均始于该位置，其使用了程序位移（平行位移）。后者基于指令中储存的搜索位置与编程点（*）之间的差异进行计算。所有位置均基于fixture1工件坐标系。

语法

```
PDispOn
  [ [ ’\’ Rot ]
    [’\’ ExeP ’:=’ < expression (IN) of robtarget>]’,’]
  [ ProgPoint’ :=’ ] < expression (IN) of robtarget> ’,’
  [ Tool ’:=’ ] < persistent (PERS) of tooldata>
  [ ´\’WObj’ :=’ < persistent (PERS) of wobjdata> ] ´;’
```

相关信息

信息，关于	请参阅
停用程序位移	第*447*页的*PDispOff - 停用程序位移*
定义使用已知坐标系的程序位移	第*452*页的*PDispSet - 启用使用已知坐标系的程序位移*
坐标系	技术参考手册 - 系统参数
工具的定义	第*1502*页的*tooldata - 工具数据*
工件的定义	第*1523*页的*wobjdata - 工件数据*

1.167 PDispSet - 启用使用已知坐标系的程序位移

手册用法

PDispSet（*Program Displacement Set*）用于定义和启用程序位移（使用已知坐标系）。

例如，在程序中若干不同位置处重复类似的运动模板时，使用程序位移。

本指令仅可用于主任务T_ROB1，或者如果在*MultiMove*系统中，则可用于运动任务中。

基本示例

以下实例介绍了指令PDispSet：

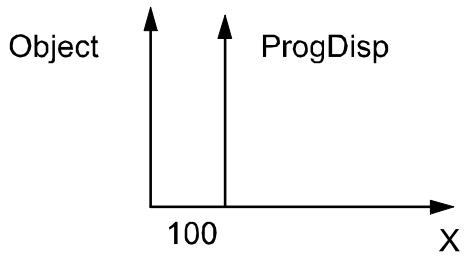
例 1

```
VAR pose xp100 := [ [100, 0, 0], [1, 0, 0, 0] ];
...
PDispSet xp100;
```

启用xp100程序位移意味着：

- ProgDisp坐标系从正x轴方向的工件坐标系移动了100 mm（参见下图）。
- 只要程序位移有效，则在x轴方向上的所有位置均将转移100 mm。

本图显示了沿x轴的100 mm程序位移。



xx0500002199

变元

```
PDispSet DispFrame
```

DispFrame

Displacement Frame

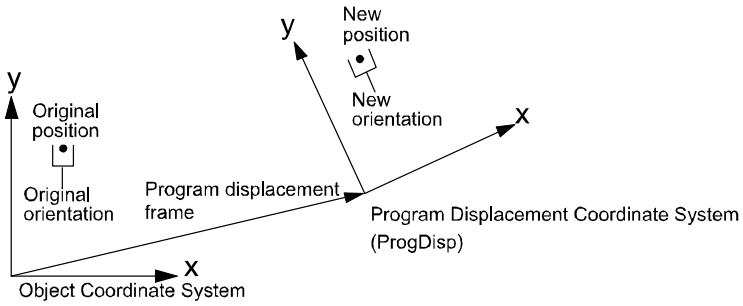
数据类型：pose

将程序位移定义为pose型数据。

下一页继续

程序执行

程序位移涉及与工件坐标系相关的ProgDisp坐标系的平移和/或旋转。由于所有位于与ProgDisp坐标系相关，同时将取代所有编程位置。参见下图，其显示了编程位置的平移和旋转。



xx0500002204

当执行指令PDispSet时，程序位移得以启用，并保持有效，直至一些其他的程序位移得以启用（指令PDispSet或PDispOn），或者直至程序位移停用（指令PDispOff）。

在同一时间，仅能启用一个程序位移。运用PDispSet，无法相互叠加程序位移。

自动重置程序位移

- 当使用重启模式重置RAPID时
- 当加载一则新程序或一个新模块时
- 当从起点开始执行程序时
- 当将程序指针移动到main时
- 当将程序指针移动到子程序时
- 移动程序指针造成执行顺序丢失时

语法

```
PDispSet
  [ DispFrame ':=' ] < expression (IN) of pose> ';'
```

相关信息

信息，关于	请参阅
停用程序位移	第*447*页的*PDispOff - 停用程序位移*
定义使用两个位置的程序位移	第*448*页的*PDispOn - 启用程序位移*
定义pose型数据	第*1451*页的*pose - 坐标变换*
坐标系	技术参考手册 - *RAPID*语言概览
关于如何使用程序位移的例子	第*448*页的*PDispOn - 启用程序位移*

1.168 ProcCall - 调用新无返回值程序

手册用法

过程调用用于将程序执行转移至另一个无返回值程序。当充分执行本无返回值程序时，程序执行将继续过程调用后的指令。

通常有可能将一系列参数发送至新的无返回值程序。其控制无返回值程序的行为，并使相同无返回值程序可能用于不同的事宜。

基本示例

以下实例介绍了指令ProcCall：

例 1

```
weldpipe1;
```

调用weldpipe1 无返回值程序。

例 2

```
errormessage;
Set do1;
...
PROC errormessage()
  TPWrite "ERROR";
ENDPROC
```

调用errormessage 无返回值程序。当该无返回值程序就绪时，程序执行返回过程调用后的指令Set do1。

变元

```
Procedure { Argument }
```

Procedure

Identifier

待调用无返回值程序的名称。

Argument

数据类型：符合无返回值程序声明。

无返回值程序参数（符合无返回值程序的参数）。

基本示例

有关于指令ProcCall的基本例子阐述如下。

例 1

```
weldpipe2 10, lowspeed;
```

调用包含两个参数的weldpipe2无返回值程序。

例 2

```
weldpipe3 10 \speed:=20;
```

调用包含一个强制参数和一个可选参数的weldpipe3无返回值程序。

下一页继续

限制

无返回值程序的参数必须符合其参数：

- 必须包括所有的强制参数。
- 必须以相同的顺序进行放置。
- 必须采用相同的数据类型。
- 必须采用有关于访问模式（输入、变量或永久数据对象）的正确类型。

程序可相互调用，并反过来调用另一个程序。程序亦可自我调用，即递归调用。允许的程序等级取决于参数数量。通常允许10级以上。

语法

```
<procedure> [ <argument list> ] ';'
```

相关信息

信息，关于	请参阅
参数	技术参考手册 - *RAPID*语言概览

1.169 ProcerrRecovery - 由过程运动错误产生和恢复

手册用法

ProcerrRecovery可用于在机械臂运动期间产生过程错误，并有可能处理该错误，重启ERROR处理器的过程和运动。

基本示例

以下实例介绍了指令ProcerrRecovery：

另请参阅第*457*页的更多示例

以下例子不现实，但是出于教授原因而显示。

例 1

```
MoveL p1, v50, z30, tool2;
ProcerrRecovery \SyncOrgMoveInst;
MoveL p2, v50, z30, tool2;
ERROR
  IF ERRNO = ERR_PATH_STOP THEN
    StartMove;
    RETRY;
  ENDIF
```

机械臂停止了朝p1的运动，且程序执行转移至程序中的ERROR处理器，该程序创建了产生错误的实际路径，在这种情况下系至MoveL p1的路径。通过StartMove重启运动，且通过RETRY继续执行。

例 2

```
MoveL p1, v50, fine, tool2;
ProcerrRecovery \SyncLastMoveInst;
MoveL p2, v50, z30, tool2;
ERROR
  IF ERRNO = ERR_PATH_STOP THEN
    StartMove;
    RETRY;
  ENDIF
```

在通往p2的道路上，机械臂立即停止运动。当出现错误时，通过当前执行的程序或将要执行移动指令的程序（在该情况下为MoveL p2），将程序执行转移至ERROR处理器。通过StartMove来重启运动，并通过RETRY来继续执行。

变元

```
ProcerrRecovery[\SyncOrgMoveInst] | [\SyncLastMoveInst]
     [\ProcSignal]
```

[\SyncOrgMoveInst]

数据类型：switch

可通过创建实际路径（由此出现错误）的程序，处理错误。

[\SyncLastMoveInst]

数据类型：switch

当出现错误时，可通过当前执行移动指令的程序来处理错误。

下一页继续

当出现错误时，如果未执行移动指令，则将延迟ERROR处理器的执行，直至程序执行下一条运动指令。这意味着，如果机械臂位于停止点或介于预取点与角路径中部之间，则至ERROR处理器的转移将会延迟。可通过该程序来处理错误。

[\ProcSignal]

数据类型：signaldo

使用户开始/停止使用指令的可选参数。如果使用该参数，且信号值为0，则将舍弃可恢复错误，且将不会产生任何过程错误。

程序执行

以连续模式执行ProcerrRecovery，会产生以下后果：

- 机械臂立即在其路径中停止。
- 将变量ERRNO设置为ERR_PATH_STOP。
- 根据关于异步引起的错误的规则，将执行转移至一些ERROR处理器。

该指令并未以步进模式做出任何事情。

关于通过ProcerrRecovery而产生的异步引起的错误的说明，请参见RAPID核心参考/错误恢复/异步引起的错误。

如果以同步模式运行，则ProcerrRecovery亦可用于MultiMove系统，以转移执行至若干程序任务中的ERROR处理器。

更多示例

有关于如何使用指令ProcerrRecovery的更多例子阐述如下。

关于ProcerrRecovery\SyncOrgMoveInst的例子

```
MODULE user_module
  VAR intnum proc_sup_int;

  PROC main()
    ...
    MoveL p1, v1000, fine, tool1;
    do_process;
    ...
  ENDPROC
  PROC do_process()
    my_proc_on;
    MoveL p2, v200, z10, tool1;
    MoveL p3, v200, fine, tool1;
    my_proc_off;
  ERROR
    IF ERRNO = ERR_PATH_STOP THEN
      my_proc_on;
      StartMove;
      RETRY;
    ENDIF
  ENDPROC

  TRAP iprocfail
    my_proc_off;
```

下一页继续

1.169 ProcerrRecovery - 由过程运动错误产生和恢复
RobotWare - OS
续前页

```
    ProcerrRecovery \SyncOrgMoveInst;
  ENDTRAP

  PROC my_proc_on()
    SetDO do_myproc, 1;
    CONNECT proc_sup_int WITH iprocfail;
    ISignalDI di_proc_sup, 1, proc_sup_int;
  ENDPROC

  PROC my_proc_off()
    SetDO do_myproc, 0;
    IDelete proc_sup_int;
  ENDPROC
ENDMODULE
```

在该例子中，由ProcerrRecovery以及开关\SyncOrgMoveInst产生的异步引起的错误，可通过程序do_process进行处理，因为始终通过程序do_process来创建路径（由此出现错误）。

通过将信号do_myproc设置为1，起动工艺流程。如果di_proc_sup成为1，则信号di_proc_sup对该过程进行监督，并会引起异步引起的错误。在该简单例子中，通过在恢复运动之前，再次将do_myproc设置为1，以此解决错误。

关于ProcerrRecovery\SyncLastMoveInst的例子

```
MODULE user_module
  PROC main()
    ...
    MoveL p1, v1000, fine, tool1;
    do_process;
    ...
  ENDPROC
  PROC do_process()
    proc_on;
    proc_move p2, v200, z10, tool1;
    proc_move p3, v200, fine, tool1;
    proc_off;
  ERROR
    IF ERRNO = ERR_PATH_STOP THEN
      StorePath;
      p4 := CRobT(\Tool:=tool1);
      ! Move to service station and fix the problem
      MoveL p4, v200, fine, tool1;
      RestoPath;
      proc_on;
      StartMoveRetry;
    ENDIF
  ENDPROC
ENDMODULE

MODULE proc_module (SYSMODULE, NOSTEPIN)
  VAR intnum proc_sup_int;
  VAR num try_no := 0;
```

下一页继续

```
TRAP iprocfail
    proc_off;
    ProcerrRecovery \SyncLastMoveInst;
  ENDTRAP

  PROC proc_on()
    SetDO do_proc, 1;
    CONNECT proc_sup_int WITH iprocfail;
    ISignalDI di_proc_sup, 1, proc_sup_int;
  ENDPROC

  PROC proc_off()
    SetDO do_proc, 0;
    IDelete proc_sup_int;
  ENDPROC

PROC proc_move (robtarget ToPoint, speeddata Speed, zonedata Zone,
     PERS tooldata Tool)
    MoveL ToPoint, Speed, Zone, Tool;
    ERROR
      IF ERRNO = ERR_PATH_STOP THEN
        try_no := try_no + 1;
        IF try_no < 4 THEN
          proc_on;
          StartMoveRetry;
        ELSE
          RaiseToUser \Continue;
      ENDIF
    ENDPROC
ENDMODULE
```

在该例子中，由ProcerrRecovery以及开关\SyncLastMoveInst 产生的异步引起的错误，可通过程序proc_move进行处理，因为始终用程序proc_move来创建所有移动指令。当程序指针位于程序do_process中时，至ERROR处理器的转移将会有所延迟，直至通过程序proc_move运行下一个MoveL。注意，运动始终立即停止。

通过将信号do_myproc 设置为1，起动工艺流程。如果di_proc_sup成为1，则信号di_proc_sup对该过程进行监督，并会引起异步引起的错误。在该简单例子中，通过在恢复运动之前，再次将do_myproc设置为1，以此解决错误。

当使用预定义NOSTEPIN程序时，我们建议使用选项开关参数\SyncLastMoveInst，因为预定义程序可决定处理程序内的一些错误情况，而其他错误情况则必须由最终用户进行处理。

错误处理

系统会生成下列可恢复错误，并在错误处理器中处理这些错误。系统变量ERRNO将被设置成：

如果信号变量是RAPID中声明的变量，则ERR_NO_ALIASIO_DEF。尚未同I/O配置以及指令AliasIO中确定的I/O信号相连。

如果与I/O单元无接触，则ERR_NORUNUNIT。

下一页继续

1.169 ProcerrRecovery - 由过程运动错误产生和恢复

RobotWare - OS

续前页

如果使用可选参数\ProcSignal，且如果执行指令时信号断开，则ERR_PROCSIGNAL_OFF。

如果无法访问I/O信号（仅对ICI现场总线有效），则ERR_SIG_NOT_VALID。

限制

出现过程错误时，仅当在基础等级上执行运动任务以及过程移动指令，方可从异步过程错误中恢复。因此，如果通过以下形式执行程序任务以及过程指令，则无法恢复错误：

- 任意事件程序
- 任意程序处理器（ERROR、BACKWARD或UNDO）
- 用户执行等级（服务程序）

参见*RAPID*参考手册 - *RAPID*内核、错误恢复、异步引起的错误。

如果未使用带有StartMove和RETRY或StartMoveRetry的错误处理器，则将暂停程序执行。唯一的重置方式是实施PP到Main。

语法

```
ProcerrRecovery
  [ ’\’ SyncOrgMoveInst ] | [’ \’ SyncLastMoveInst ]
  [ ’\’ ProcSignal’ :=’ ] < variable (VAR) of signaldo > ’;’
```

相关信息

信息，关于	请参阅
错误处理器	技术参考手册 - *RAPID*语言概览
异步引起的错误	*RAPID*参考手册-*RAPID*内核-错误恢复
将错误传播至用户等级	第*484*页的*RaiseToUser* - 将错误传播至用户等级
恢复运动和程序执行	第*665*页的*StartMoveRetry* - 重启机械臂移动和执行

1.170 PrxActivAndStoreRecord - 启用和储存已记录的配置文件数据

手册用法

PrxActivAndStoreRecord用于启用已记录的配置文件数据，并将其储存在文件中。

可代替调用PrxActivRecord和PrxStoreRecord而使用。

基本示例

```
PrxActivAndStoreRecord SSYNC1, 1, "profile.log";
```

启用传感器运动配置文件，并将其储存在文件*profile.log*中。

变元

```
PrxActivAndStoreRecord MechUnit Delay File_name
```

MechUnit

数据类型：mechunit

同步机械臂运动的移动机械单元工件。

Delay

数据类型：num

以秒计的延迟可用于转变时间记录。其必须介于0.01与0.1之间。如果给定该值为0，则不会增加延迟。延迟不保存于配置文件中，其仅用于启用。如果延迟应当与保存的配置文件一同使用，则必须再次通过指令PrxUseFileRecord来规定该延迟。

File_name

数据类型：string

储存配置文件的文件名称。

程序执行

如果记录用以同步，则必须在传感器开始运动前至少0.2秒，执行PrxActivAndStoreRecord。

错误处理

可能会产生下列可恢复错误。错误可以用错误处理器进行处理。将系统变量ERRNO设置为：

ERR_ACTIV_PROF	已启用配置文件中的错误
ERR_STORE_PROF	已储存配置文件中的错误
ERR_USE_PROF	已使用配置文件中的错误

语法

```
PrxActivAndStoreRecord
  [ MechUnit ':=' ] < expression (IN) of mechunit> ','
  [ Delay ':=' ] < expression (IN) of num > ','
  [ File_name ':=' ] < expression (IN) of string > ';'
```

下一页继续

1.170 PrxActivAndStoreRecord - 启用和储存已记录的配置文件数据
续前页

相关信息

信息，关于	请参阅
激活已记录的曲线数据	第*463*页的*PrxActivRecord - 启用已记录的配置文件数据*
停用一项记录	第*466*页的*PrxDeactRecord - 停用记录*
保存已记录的曲线数据	第*475*页的*PrxStoreRecord - 储存已记录的配置文件数据*
Machine Synchronization	应用手册 - 控制器软件*IRC5*

1.171 PrxActivRecord - 启用已记录的配置文件数据

手册用法

PrxActivRecord用于启用刚刚进行的记录，从而在无须事先保存的情况下予以使用。

基本示例

```
PrxActivRecord SSYNC1, 0;
WaitTime 0.2;
SetDO do_startstop_machine, 1;
!Work synchronized with sensor
...
SetDO do_startstop_machine, 0;
```

一旦记录就绪，随即启用传感器的记录，并用于预测传感器的运动。

变元

```
PrxActivRecord MechUnit Delay
```

MechUnit

数据类型：mechunit

同步机械臂运动的移动机械单元工件。

Delay

数据类型：num

延迟以秒计，可用于转变时间记录。其必须介于0.01与0.1之间。如果给定的值为0，则不会增加任何延迟。

程序执行

必须至少在传送带开始运动前的0.2秒，执行PrxActivRecord。

错误处理

可能会产生下列可恢复错误。错误可以用错误处理器进行处理。将系统变量ERRNO设置为：

ERR_ACTIV_PROF	已启用配置文件中的错误
ERR_STORE_PROF	已储存配置文件中的错误
ERR_USE_PROF	已使用配置文件中的错误

语法

```
PrxActivRecord
  [ MechUnit ':=' ] < expression (IN) of mechunit> ','
  [ Delay ':=' ] < expression (IN) of num > ';'
```

相关信息

信息，关于	请参阅
激活和保存已记录的曲线数据	第461页的*PrxActivAndStoreRecord - 启用和储存已记录的配置文件数据*

下一页继续

1.171 PrxActivRecord - 启用已记录的配置文件数据

续前页

信息，关于	请参阅
停用一项记录	第*466*页的*PrxDeactRecord - 停用记录*
保存已记录的曲线数据	第*475*页的*PrxStoreRecord - 储存已记录的配置文件数据*
Machine Synchronization	应用手册 - 控制器软件*IRC5*

1.172 PrxDbgStoreRecord - 储存和调试已记录的配置文件数据

手册用法

PrxDbgStoreRecord用于储存未启用的记录，以进行调试。

可用于比较记录，并检查可重现性。

基本示例

```
PrxDbgStoreRecord SSYNC1, "debug_profile.log";
```

将记录保存在文件*debug_profile.log*中。

变元

```
PrxDbgStoreRecord MechUnit Filename
```

MechUnit

数据类型：mechunit

同步机械臂运动的移动机械单元工件。

File_name

数据类型：string

储存记录的文件名称。

语法

```
PrxDbgStoreRecord
  [ MechUnit ':=' ] < expression (IN) of mechunit> ','
  [ File_name ':=' ] < expression (IN) of string > ';'
```

相关信息

信息，关于	请参阅
激活和保存已记录的曲线数据	第*461*页的*PrxActivAndStoreRecord* - 启用和储存已记录的配置文件数据
激活已记录的曲线数据	第*463*页的*PrxActivRecord* - 启用已记录的配置文件数据
保存已记录的曲线数据	第*475*页的*PrxStoreRecord* - 储存已记录的配置文件数据
Machine Synchronization	应用手册 - 控制器软件*IRC5*

1.173 PrxDeactRecord - 停用记录

手册用法

PrxDeactRecord用于停用记录。

基本示例

```
PrxDeactRecord SSYNC1;
```

停用传感器运动记录，且不再将其用于预测传感器运动。可再次启用记录。

变元

```
PrxDeactRecord MechUnit
```

MechUnit

数据类型：mechunit

同步机械臂运动的移动机械单元工件。

限制

同步期间，不得调用PrxDeactRecord。

语法

```
PrxDeactRecord
  [ MechUnit ':=' ] < expression (IN) of mechunit> ';'
```

相关信息

信息，关于	请参阅
激活已记录的曲线数据	第463页的*PrxActivRecord - 启用已记录的配置文件数据*
Machine Synchronization	应用手册 - 控制器软件*IRC5*

1.174 PrxResetPos - 重置传感器零位置

手册用法

PrxResetPos用于重置传感器零位置。

针对同步功能和记录的文件重置传感器位置，但是未重置I/O信号值。该指令用于软件重置传感器输入，同步开关由此不可用于重置I/O信号。

基本示例

```
PrxResetPos SSYNC1;
```

将传感器设置为零。

变元

```
PrxResetPos MechUnit
```

MechUnit

数据类型：mechunit

同步机械臂运动的移动机械单元工件。

程序执行

调用PrxResetPos之前，必须停止传感器单元（位于所需零位置）。

限制

不与DSQC 377A板一同使用。

该指令相当于同步开关。点动窗口应当显示0.0，以作为该指令后的附加轴位置。

语法

```
PrxResetPos
  [ MechUnit ':=' ] < expression (IN) of mechunit> ';'
```

相关信息

信息，关于	请参阅
为相关传感器设置一个参考位置	第469页的*PrxSetPosOffset - 设置传感器的参考位置*
Machine Synchronization	应用手册 - 控制器软件*IRC5*

1.175 PrxResetRecords - 重置和停用所有记录

手册用法

PrxResetRecords用于重置和停用所有记录。

基本示例

```
PrxResetRecords SSYNC1;
```

停用传感器运动记录，且不再将其用于预测传感器运动，并移除记录数据。

变元

```
PrxResetRecords MechUnit
```

MechUnit

数据类型：mechunit

同步机械臂运动的移动机械单元工件。

程序执行

必须至少在传送带开始运动前的0.2秒，执行PrxResetRecords。

语法

```
PrxResetRecords
  [ MechUnit ':=' ] < expression (IN) of mechunit> ';'
```

相关信息

信息，关于	请参阅
Machine Synchronization	应用手册 - 控制器软件*IRC5*

1.176 PrxSetPosOffset - 设置传感器的参考位置

手册用法

PrxSetPosOffset用于设置传感器的参考位置。

设置传感器位置，以供同步功能和记录的文件参考。当同步开关不适用于重置I/O信号时，该功能用于传感器参考的软件设置。

基本示例

```
PrxSetPosOffset SSYNC1, reference;
```

将传感器位置设置为参考值。

变元

```
PrxSetPosOffset MechUnit Reference
```

MechUnit

数据类型：mechunit

同步机械臂运动的移动机械单元工件。

Reference

数据类型：num

参考以米（或传感器单元）计。其必须介于-5000到5000之间。

程序执行

调用PrxSetPosOffset之前，必须停止传感器单元。

限制

不与DSQC 377A板一同使用。

语法

```
PrxSetPosOffset
  [ MechUnit ':=' ] < expression (IN) of mechunit> ','
  [ Reference ':=' ] < expression (IN) of num > ';'
```

相关信息

信息，关于	请参阅
重置相关传感器的零位	第467页的*PrxResetPos - 重置传感器零位置*
Machine Synchronization	应用手册 - 控制器软件*IRC5*

1.177 PrxSetRecordSampleTime - 设置有关记录配置文件的样本时间

手册用法

PrxSetRecordSampleTime用于设置关于记录配置文件的样本时间（以秒计）。

从系统参数*Pos Update time*获得默认样本时间，其属于题为*Process*中的*CAN interface*型参数。注意，*Pos Update time*规定了样本时间（以毫秒计），而PrxSetRecordSampleTime则规定了样本时间（以秒计）。

记录的配置文件中的最大样本数量为300。如果记录超过300 * *Pos Update time*，则必须增加样本时间。

基本示例

给予12秒的时间来进行记录。样本时间不得小于12/300 = 0.04。因此，将样本时间设置为0.04秒。

```
PrxSetRecordSampleTime SSYNC1, 0.04;
```

变元

```
PrxSetRecordSampleTime MechUnit SampleTime
```

MechUnit

数据类型：mechunit

同步机械臂运动的移动机械单元工件。

SampleTime

数据类型：num

样本时间以秒计。样本时间必须介于0.01和0.1之间。

语法

```
PrxSetRecordSampleTime
  [ MechUnit ':=' ] < expression (IN) of mechunit> ','
  [ SampleTime ':=' ] < expression (IN) of num> ';'
```

相关信息

信息，关于	请参阅
Machine Synchronization	应用手册 - 控制器软件*IRC5*

1.178 PrxSetSyncalarm - 设置同步报警行为

手册用法

PrxSetSyncalarm用于在指定时间期间，将*sync_alarm_signal*行为设置为脉冲。

如果触发同步警报，则在指令PrxSetSyncalarm所指定的时间内，用脉冲输送*Sync_alarm_signal*。其亦可设置为无脉冲，即信号持续处于高水平。

默认脉冲长度为1妙。

基本示例

例 1

```
PrxSetSyncalarm SSYNC1 \time:=2;
```

将*sync_alarm_signal*上的脉冲长度设置为2秒。

例 2

```
PrxSetSyncalarm SSYNC1 \NoPulse;
```

如果触发同步警报，则设置*sync_alarm_signal*（非脉冲）。

变元

```
PrxSetSyncalarm MechUnit [\Time] | [\NoPulse]
```

MechUnit

数据类型：mechunit

同步机械臂运动的移动机械单元工件。

[\Time]

数据类型：num

脉冲长度以秒计。脉冲长度必须介于0.1和60之间。

如果将\Time设置为60秒以上，则不会使用脉冲（与使用\NoPulse的影响相同）。

[\NoPulse]

数据类型：switch

未使用脉冲。设置信号，直至执行SupSyncSensorOff指令：

语法

```
PrxSetSyncalarm
  [ MechUnit ':=' ] < expression (IN) of mechunit>
  [ '\' Time ':=' < expression (IN) of num > ]
  | [ '\' NoPulse ] ';'
```

相关信息

信息，关于	请参阅
SupSyncSensorOff - 停止同步传感器监督	第701页的*SupSyncSensorOff - 停止同步传感器监督*
Machine Synchronization	应用手册 - 控制器软件*IRC5*

1.179 PrxStartRecord - 记录新的配置文件

手册用法

一旦设置*sensor_start_signal*，PrxStartRecord随即用于重置所有配置文件数据，并记录关于传感器运动的新配置文件。

为了能够进行记录，重要的是首先连接传感器（速度会影响机械臂速度的机械单元）。这意味着，在开始记录之前，必须执行WaitSensor指令。

基本示例

```
ActUnit SSYNC1;
WaitSensor SSYNC1;
PrxStartRecord SSYNC1, 1, PRX_PROFILE_T1;
WaitTime 0.2;
SetDO do_startstop_machine 1;
```

在该例子中，信号*do_startstop_machine*使传感器开始运动。一旦机器设置信号*sensor_start_signal*，随即记录传感器的配置文件。

变元

```
PrxStartRecord MechUnit, Record_duration, Profile_type
```

MechUnit

数据类型：mechunit

同步机械臂运动的移动机械单元工件。

Record_duration

数据类型：num

规定记录的持续时间（以秒计）。其必须介于0.1与*Pos Update time* * 300之间。如果该值为0，则必须使用指令PrxStopRecord来停止记录。

Profile_type

数据类型：num

下文列出了可能的值及其说明：

值	描述
PRX_INDEX_PROF	通过*sensor_start_signal*开始记录。
PRX_START_ST_PR	可记录开始和停止运动。*sensor_start_signal*用于记录开始运动，且*sensor_stop_signal*用于记录停止运动。
PRX_STOP_ST_PROF	与PRX_START_ST_PR相同，仅信号顺序有所不同。首先使用*sensor_stop_signal*。
PRX_STOP_M_PROF	通过*sensor_stop_signal*开始记录。
PRX_HPRESS_PROF	为记录液压机（传感器位置零相当于打开的液压机）。
PRX_PROFILE_T1	为记录IMM或其他机器（传感器位置零相当于关闭的液压机）。

程序执行

必须至少在传感器开始运动前的0.2秒，执行PrxStartRecord。

下一页继续

语法

```
PrxStartRecord
  [ MechUnit ':=' ] < expression (IN) of mechunit> ','
  [ Record_duration ':=' ] < expression (IN) of num > ','
  [ Profile_type ':=' ] < expression (IN) of num > ';'
```

相关信息

信息，关于	请参阅
停止对曲线的记录	第*474*页的*PrxStopRecord - 停止记录配置文件*
Machine Synchronization	应用手册 - 控制器软件*IRC5*

1.180 PrxStopRecord - 停止记录配置文件

手册用法

PrxStopRecord用于停止记录配置文件。

当PrxStartRecord将Record_duration设置为0时，应当始终进行使用。

基本示例

```
ActUnit SSYNC1;
WaitSensor SSYNC1;
PrxStartRecord SSYNC1, 0, PRX_PROFILE_T1;
WaitTime 0.2;
SetDo do_startstop_machine 1;
WaitTime 2;
PrxStopRecord SSYNC1;
```

在该例子中，信号*do_startstop_machine*使传感器开始运动。一旦设置*sensor_start_signal*，随即记录传感器运动的配置文件，且在两秒之后，通过指令PrxStopRecord来停止记录。

变元

```
PrxStopRecord MechUnit
```

MechUnit

数据类型：mechunit

同步机械臂运动的移动机械单元工件。

语法

```
PrxStopRecord
  [ MechUnit ':=' ] < expression (IN) of mechunit> ';'
```

相关信息

信息，关于	请参阅
记录一条新曲线	第*472*页的*PrxStartRecord - 记录新的配置文件*
Machine Synchronization	应用手册 - 控制器软件*IRC5*

1.181 PrxStoreRecord - 储存已记录的配置文件数据

手册用法

PrxStoreRecord用于在文件中保存已启用的记录。

基本示例

```
ActUnit SSYNC1;
WaitSensor SSYNC1;
PrxStartRecord SSYNC1, 0, PRX_PROFILE_T1;
WaitTime 0.2;
SetDo do_startstop_machine 1;
WaitTime 2;
PrxStopRecord SSYNC1;
PrxActivRecord SSYNC1;
SetDo do_startstop_machine 0;
PrxStoreRecord SSYNC1, 0, "profile.log";
```

一旦设置*sensor_start_signal*，并将其储存在文件*profile.log*中，随即记录传感器运动的配置文件。

变元

```
PrxStoreRecord MechUnit Delay Filename
```

MechUnit

数据类型：mechunit

同步机械臂运动的移动机械单元工件。

Delay

数据类型：num

以秒计的延迟可用于转变时间记录。其必须介于0.01与0.1之间。如果给定该值为0，则不会增加延迟。延迟不保存于配置文件中，其仅用于启用。如果延迟应当与保存的配置文件一同使用，则必须再次通过指令PrxUseFileRecord来规定该延迟。

File_name

数据类型：string

储存配置文件的文件名称。

限制

必须在调用PrxStoreRecord之前，启用记录。

语法

```
PrxStoreRecord
  [ MechUnit ':=' ] < expression (IN) of mechunit> ','
  [ Delay ':=' ] < expression (IN) of num > ','
  [ File_name ':=' ] < expression (IN) of string > ';'
```

相关信息

信息，关于	请参阅
激活已记录的曲线数据	第463页的*PrxActivRecord - 启用已记录的配置文件数据*

下一页继续

1.181 PrxStoreRecord - 储存已记录的配置文件数据

续前页

信息，关于	请参阅
停用一项记录	第*466*页的*PrxDeactRecord - 停用记录*
使用已记录的曲线数据	第*477*页的*PrxUseFileRecord - 使用已记录的配置文件数据*
Machine Synchronization	应用手册 - 控制器软件*IRC5*

1.182 PrxUseFileRecord - 使用已记录的配置文件数据

手册用法

PrxUseFileRecord用于加载和启用传感器同步文件的记录。

基本示例

```
PrxUseFileRecord SSYNC1, 0, "profile.log";
WaitTime 0.2;
SetDo do_startstop_machine 1;
!Work synchronized with sensorWork synchronized with sensor
...
SetDo do_startstop_machine 0;
```

变元

```
PrxUseFileRecord MechUnit Delay Filename
```

MechUnit

数据类型：mechunit

同步机械臂运动的移动机械单元工件。

Delay

数据类型：num

延迟以秒计，可用于转变时间记录。其必须介于0.01与0.1之间。如果给定的值为0，则不会增加任何延迟。

File_name

数据类型：string

储存配置文件的文件名称。

程序执行

必须至少在传送带开始运动前的0.2秒，执行PrxUseFileRecord。

语法

```
PrxUseFileRecord
  [ MechUnit ':=' ] < expression (IN) of mechunit> ','
  [ Delay ':=' ] < expression (IN) of num > ','
  [ File_name ':=' ] < expression (IN) of string > ';'
```

相关信息

信息，关于	请参阅
激活已记录的曲线数据	第463页的*PrxActivRecord - 启用已记录的配置文件数据*
停用一项记录	第466页的*PrxDeactRecord - 停用记录*
保存已记录的曲线数据	第475页的*PrxStoreRecord - 储存已记录的配置文件数据*
Machine Synchronization	应用手册 - 控制器软件*IRC5*

1.183 PulseDO - 产生关于数字信号输出信号的脉冲

手册用法

PulseDO为，用于产生关于数字信号输出信号的脉冲。

基本示例

以下实例介绍了指令PulseDO：

例 1

```
PulseDO do15;
```

输出信号do15产生脉冲长度为0.2 s的脉冲。

例 2

```
PulseDO \PLength:=1.0, ignition;
```

信号ignition产生的脉冲长度为1.0 s。

例 3

```
! Program task MAIN
PulseDO \High, do3;
! At almost the same time in program task BCK1
PulseDO \High, do3;
```

两个程序任务的信号do3几乎同时产生正脉冲（值1）。其将产生脉冲长度长于默认值0.2 s的正脉冲，或者在脉冲长度为0.2 s的各脉冲之后产生两个正脉冲。

变元

```
PulseDO [ \High ] [ \PLength ] Signal
```

[\High]

High level

数据类型：switch

当独立于其当前状态而执行指令时，规定始终将信号值设置为高（值1）。

[\PLength]

Pulse Length

数据类型：num

脉冲长度以秒计（0.001 - 2000 s）。如果省略参数，则产生0.2秒的脉冲。

Signal

数据类型：signaldo

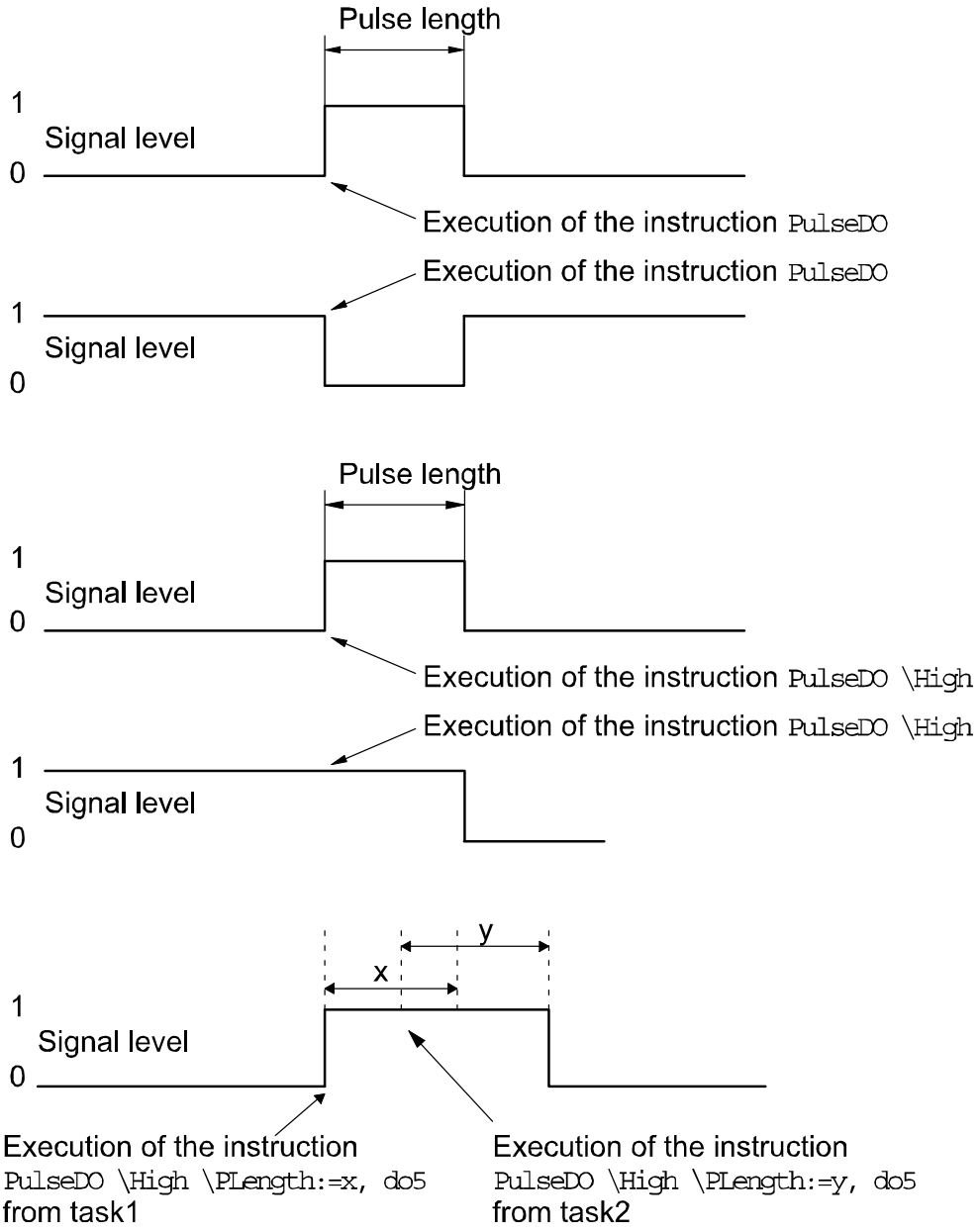
将产生脉冲的信号名称。

下一页继续

程序执行

在脉冲起动后，直接执行PulseDO后的下一指令。可在不影响程序执行的情况下，设置/重置脉冲。

下图显示了关于数字信号输出信号产生脉冲的实例。



xx0500002217

在脉冲起动后，直接执行下一指令。可在不影响程序执行的情况下，设置/重置脉冲。

限制

脉冲长度的分辨率为0.001秒。与此不同的编程值应予以舍入。

错误处理

系统会生成下列可恢复错误，并在错误处理器中处理这些错误。系统变量ERRNO将被设置成：

下一页继续

1.183 PulseDO - 产生关于数字信号输出信号的脉冲

RobotWare - OS

续前页

如果信号变量是RAPID中声明的变量，则ERR_NO_ALIASIO_DEF。尚未同I/O配置以及指令AliasIO中确定的I/O信号相连。

如果与I/O单元无接触，则ERR_NORUNUNIT。

如果无法访问I/O信号（仅对ICI现场总线有效），则ERR_SIG_NOT_VALID。

语法

```
PulseDO
  [ '\' High]
  [ '\' PLength ':=' < expression (IN) of num >] ','
  [ Signal ':=' ] < variable (VAR) of signaldo > ';'
```

相关信息

信息，关于	请参阅
输入/输出指令	技术参考手册 - *RAPID*语言概览
输入/输出功能性概述	技术参考手册 - *RAPID*语言概览
I/O配置	技术参考手册 - 系统参数

1.184 RAISE - 调用错误处理器

手册用法

RAISE用于在程序中产生错误，随后调用程序的错误处理器。

RAISE亦可用于错误处理器中，以传播当前错误至调用程序的错误处理器。

该指令可用于跳转回程序结构中的更高等级，例如，如果在较低等级出现错误，则跳转回主程序中的错误处理器。

基本示例

以下实例介绍了指令RAISE：

另请参阅第*482*页的更多示例

例 1

```
MODULE MainModule .
VAR errnum ERR_MY_ERR := -1;

PROC main()
  BookErrNo ERR_MY_ERR;

  IF di1 = 0 THEN
    RAISE ERR_MY_ERR;
  ENDIF

ERROR
  IF ERRNO = ERR_MY_ERR THEN
    TPWrite "di1 equals 0";
  ENDIF

ENDPROC

ENDMODULE
```

为此，di1的实施等于0，并被视为错误。RAISE将强制执行至错误处理器。在该例子中，用户已经创建了用以处理具体错误的错误编号。

变元

```
RAISE [ Error no. ]
```

Error no.

数据类型：errnum

错误编号：任意编号介于1到90之间的错误处理器，可用于定位已出现的错误（ERRNO系统变量）。

通过指令BookErrNo.，亦可能预定1-90范围外的错误编号。

必须通过RAISE指令，指定错误处理器外的错误编号，以将执行转移至该程序的错误处理器。

如果在程序的错误处理器中存在该指令，则将错误传播至调用程序的错误处理器。在该情形下，无须指定错误编号。

下一页继续

1.184 RAISE - 调用错误处理器
RobotWare-OS
续前页

更多示例

有关于指令RAISE的更多例子阐述如下。

例 1

```
MODULE MainModule
VAR num value1 := 10;
VAR num value2 := 0;

PROC main()
  routine1;

ERROR
  IF ERRNO = ERR_DIVZERO THEN
    value2 := 1;
    RETRY;
  ENDIF
ENDPROC

PROC routine1()
  value1 := 5/value2;!This will lead to an error when value2 is
       equal to 0.
ERROR
  RAISE;
ENDPROC

ENDMODULE
```

在该例子中，关于零的除法将产生错误。在ERROR中，处理器RAISE将错误传播至调用程序“main”中的错误处理器。相同的错误编号始终有效。RETRY将再次运行整个程序“routine1”。

程序执行

在程序的错误处理器中继续程序执行。在已经执行错误处理器后，通过以下程序可继续程序执行：

- 用以调用讨论中程序的程序（RETURN）。
- 用以调用讨论中程序的程序错误处理器（RAISE）。

程序错误处理器中的RAISE指令同时拥有另一个特征。其可以用于长跳转（参见“关于长跳转的错误恢复”）。通过长跳转，可能将错误处理器的错误由深度嵌入的调用链一步传播至更高的等级。

如果软中断程序中存在RAISE指令，则由系统的错误处理器来处理错误。

错误处理

如果错误编号超出范围，则将系统变量ERRNO设置为ERR_ILLRAISE（参见“数据类型-errnum”）。可用错误处理器来处理该错误。

语法

```
RAISE [<error number>] ';'
```

下一页继续

相关信息

信息，关于	请参阅
错误处理	技术参考手册 - 系统参数
关于长跳转的错误恢复	技术参考手册 - *RAPID*语言内核
登记错误编号	第*40*页的*BookErrNo - 登记RAPID*系统错误编号

1.185 RaiseToUser - 将错误传播至用户等级

手册用法

在nostepin程序中的错误处理器中使用RaiseToUser，以将当前错误或其他规定错误传播至处于用户等级的错误处理器。在这种情况下，用户等级是调用链中高于nostepin程序的第一个程序。

基本示例

以下实例介绍了指令RaiseToUser：

例 1

```
MODULE MyModule
  VAR errnum ERR_MYDIVZERO:= -1;
  PROC main()
      BookErrNo ERR_MYDIVZERO;
      ...
      routine1;
      ...
      ERROR
      IF ERRNO = ERR_MYDIVZERO THEN
        TRYNEXT;
      ELSE
        RETRY;
      ENDIF
  ENDPROC
ENDMODULE
MODULE MySysModule (SYSMODULE, NOSTEPIN, VIEWONLY)
  PROC routine1()
    ...
    routine2;
    ...
  UNDO
  ! Free allocated resources
  ENDPROC
  PROC routine2()
    VAR num n:=0;
    ...
    reg1:=reg2/n;
    ...
    ERROR
    IF ERRNO = ERR_DIVZERO THEN
      RaiseToUser \Continue \ErrorNumber:=ERR_MYDIVZERO;
    ELSE
      RaiseToUser \BreakOff;
    ENDIF
  ENDPROC
ENDMODULE
```

下一页继续

在routine2中除以零，将传播至主程序中的错误处理器，并将errno设置为ERR_MYDIVZERO。随后，主错误处理器中的TRYNEXT指令将使得程序执行，从而在routine2中除以零后，继续该指令。\Continue开关会控制该行为。

如果routine2中出现任何其他错误，则\BreakOff开关强制从主程序中的错误处理器继续执行。在这种情况下，将执行程序1中的撤销处理器，并同时将其升至用户等级。主程序中错误处理器中的RETRY指令，将再次从开始执行程序1.

如果在用户等级上，完成以下RAISE或RETURN，则亦将在\Continue情形下执行程序1中的撤销处理器。

变元

```
RaiseToUser[ \Continue] | [ \BreakOff][ \ErrorNumber]
```

[\Continue]

数据类型：switch

继续在引起错误的程序中执行。

[\BreakOff]

数据类型：switch

中断调用链，并继续以用户等级执行。将执行调用链中的撤销处理器，引起错误的程序中的撤销处理器除外。

必须编程参数\Continue或\BreakOff之一，以避免执行错误。

[\ErrorNumber]

数据类型：errnum

任意编号介于1到90之间的错误处理器，可用于定位出现的错误（ERRNO 系统变量）。

通过指令BookErrNo，亦可能预定1-90范围外的错误编号。

如果未指定参数\ErrorNumber，则原始错误编号会传播至用户等级程序中的错误处理器。

程序执行

RaiseToUser can仅可用于nostepin程序中的错误处理器。

在用户等级程序的错误处理器中，继续程序执行。如果不存在可选参数\ErrorNumber，则错误编号保持有效。如果不存在用户等级的错误处理器，则系统的错误处理器会处理错误。如果未指定参数\Continue或\BreakOff，则调用系统的错误处理器。

在执行错误处理器之后，存在两种不同的行为。如果\Continue开关开启，则通过程序以及 RaiseToUser，继续程序执行。如果\BreakOff开关开启，则在用户等级继续程序执行。

程序执行可继续：

- 引起错误的指令（RETRY）
- 以下指令（TRYNEXT）
- 以用户等级调用程序的程序错误处理器（RAISE）
- 以用户等级调用程序的程序（RETURN）

下一页继续

1.185 RaiseToUser - 将错误传播至用户等级

RobotWare - OS

续前页

错误处理

如果错误编号超出范围，则将系统变量ERRNO设置为ERR_ILLRAISE（参见“数据类型-errnum”）。系统的错误处理器会处理该错误。

语法

```
RaiseToUser
  [ '\' Continue ]
  '|' [ '\' BreakOff ]
  [ '\' ErrorNumber ':=' ] < expression (IN) of errnum> ';'
```

相关信息

信息，关于	请参阅
错误处理	技术参考手册 - *RAPID*语言概览
撤销处理	技术参考手册 - *RAPID*语言概览
登记错误编号	第*40*页的*BookErrNo - 登记RAPID*系统错误编号

1.186 ReadAnyBin - 读取二进制串行通道或文件的数据

手册用法

ReadAnyBin（读取所有二进制）用于从二进制串行通道或文件读取所有类型的数据。

基本示例

以下实例介绍了指令ReadAnyBin：

另请参阅第*487*页的更多示例

例 1

```
VAR iodev channel1;
VAR robtarget next_target;
...
Open "com1:", channel1 \Bin;
ReadAnyBin channel1, next_target;
```

从channe12参考的通道，读取有待执行的下一个机械臂目标next_target。

变元

```
ReadAnyBin IODevice Data [\Time]
```

IODevice

数据类型：iodev

有待读取的二进制串行通道或文件的名称（参考）。

Data

数据类型：ANYTYPE

将用于储存读取数据的VAR 或PERS。

[\Time]

数据类型：num

读取操作（超时）的最长时间以秒计。如果未规定该参数，则将最长时间设置为60秒。为了永远等待，使用预定义常量WAIT_MAX。

如果在完成读取操作之前耗尽时间，则将通过错误代码ERR_DEV_MAXTIME来调用错误处理器。如果不存在错误处理器，则将停止执行。

程序停止期间，亦使用超时功能，并由程序开始时的RAPID程序进行通知。

程序执行

从指定二进制串行通道或文件读取的指定数据，需要尽可能多的字节。

上电失败重启时，将关闭系统中所有打开的文件或串行通道，并将重置iodev型变量中的I/O描述符。

更多示例

有关于指令ReadAnyBin的更多例子阐述如下。

例 1

```
CONST num NEW_ROBT:=12;
CONST num NEW_WOBJ:=20;
VAR iodev channel;
```

下一页继续

1.186 ReadAnyBin - 读取二进制串行通道或文件的数据

RobotWare - OS

续前页

```
VAR num input;
VAR robtarget cur_robt;
VAR wobjdata cur_wobj;

Open "com1:", channel\Bin;

! Wait for the opcode character
input := ReadBin (channel \Time:= 0.1);
TEST input
CASE NEW_ROBT:
  ReadAnyBin channel, cur_robt;
CASE NEW_WOBJ:
  ReadAnyBin channel, cur_wobj;
ENDTEST
Close channel;
```

第一步是从串行通道读取消息的opcode。根据该opcode，从串行通道读取robtarget或wobjdata。

错误处理

如果在读取期间出现错误，则将系统变量ERRNO设置为ERR_FILEACC。

如果在完成读取操作之前超时，则将系统变量ERRNO设置为ERR_DEV_MAXTIME.

如果在读取过程中出现校验和错误，则将系统变量ERRNO设置为ERR_RANYBIN_CHK.

如果在读取所有字节之前，探测到文件结束，则将系统变量ERRNO设置为ERR_RANYBIN_EOF.

随后，可通过错误处理器来处理此类错误。

限制

该指令仅可用于为二进制读取而打开的串行通道或文件。

通过该指令ReadAnyBin而读取的数据，必须为数值数据类型，例如num、bool或string。亦可使用此类数值数据类型的记录、记录成分、数组或数组元素。无法使用半值或非值数据类型的整体数据集或部分数据集。

注意

为储存读取的数据，可通过若干步骤来更新VAR或PERS变量。因此，在使用从TRAP或另一程序任务读取的数据之前，始终等待，直至整个数据结构得以更新。

因为WriteAnyBin-ReadAnyBin仅设计用以处理IRC5控制系统之间或内部的二进制控制器数据以及串行通道或文件，因此，未发布数据协议，且无法在任何PC上翻译数据。

控制软件开发可打破兼容性，因此，可能无法使用不同RobotWare软件版本之间的WriteAnyBin-ReadAnyBin。

语法

```
ReadAnyBin
  [ IODevice ':='] <variable (VAR) of iodev> ','
  [ Data ':='] <var or pers (INOUT) of ANYTYPE>
  [ '\' Time ':=' <expression (IN) of num>] ';'
```

下一页继续

相关信息

信息，关于	请参阅
打开串行通道或文件	技术参考手册 - *RAPID*语言概览
将数据写入二进制串行通道或文件	第*913*页的*WriteAnyBin* - 将数据写入二进制串行通道或文件
文件和串行通道处理	应用手册 - 控制器软件*IRC5*

1.187 ReadBlock - 读取设备的数据块

手册用法

ReadBlock用于读取与串行传感器接口. *The data is stored in a file.*相连的设备的数据块。

传感器接口与两个位于串行通道（使用RTP1传输层协议）上方的传感器进行通信。

此为关于传感器通道配置的实例。

COM_PHY_CHANNEL:

- Name “COM1:”
- Connector “COM1”
- Baudrate 19200

COM_TRP:

- Name “sen1:”
- Type “RTP1”
- PhyChannel “COM1”

基本示例

以下实例介绍了指令ReadBlock：

例 1

```
CONST string SensorPar := "flp1:senpar.cfg";
CONST num ParBlock:= 1;

! Connect to the sensor device "sen1:" (defined in sio.cfg).
SenDevice "sen1:";

! Read sensor parameters from sensor datablock 1
! and store on flp1:senpar.cfg

ReadBlock "sen1:", ParBlock, SensorPar;
```

变元

```
ReadBlock device BlockNo FileName [ \TaskName ]
```

device

数据类型：string

在sio.cfg中配置了I/O设备名称，以供传感器使用。

BlockNo

数据类型：num

参数BlockNo用于在传感器中选择有待读取的数据块。

FileName

数据类型：string

参数FileName用于定义文件，此类文件的数据根据BlockNo参数所选择的传感器中的数据块写入。

下一页继续

[\TaskName]

数据类型：string

参数TaskName使其可能访问其他RAPID任务中的设备。

错误处理

错误常量 (ERRNO值)	描述
SEN_NO_MEAS	测量失败
SEN_NOREADY	传感器不能处理命令
SEN_GENERRO	一般性传感器错误
SEN_BUSY	传感器繁忙
SEN_UNKNOWN	未知的传感器
SEN_EXALARM	外部传感器错误
SEN_CAALARM	内部传感器错误
SEN_TEMP	传感器温度错误
SEN_VALUE	非法通信值
SEN_CAMCHECK	传感器检查失败
SEN_TIMEOUT	通信错误

语法

```
ReadBlock
  [ device ':=' ] < expression(IN) of string> ','
  [ BlockNo ':=' ] < expression (IN) of num > ','
  [ FileName ':=’ ] < expression (IN) of string > ','
  [ '\' TaskName ':=' < expression (IN) of string > ] ';'
```

相关信息

信息，关于	请参阅
与传感器设备相连	第*574*页的*SenDevice - 与传感器设备相连*
写入传感器变量	第*927*页的*WriteVar - 写入变量*
读取传感器变量	第*1210*页的*ReadVar - 从设备读取变量*
写入传感器数据块	第*917*页的*WriteBlock - 将数据块写入设备*
传感器通信配置	技术参考手册 - 系统参数

1.188 ReadCfgData - 读取系统参数的属性

手册用法

ReadCfgData的作用是读取一个系统参数（配置数据）的一项属性。

除了读取已命名的参数外，同时有可能搜索未命名的参数。

基本示例

以下例子阐明了指令ReadCfgData。此类例子均表明了如何读取已命名的参数。

例 1

```
VAR num offset1;
...
ReadCfgData "/MOC/MOTOR_CALIB/rob1_1","cal_offset",offset1;
```

针对num变量offset1中的rob_1，读取轴1校准偏移量的值。

例 2

```
VAR string io_device;
...
ReadCfgData "/EIO/EIO_SIGNAL/process_error","Device",io_device;
```

读取I/O设备的名称，由此将信号process_error定义到string变量io_device中。

变元

```
ReadCfgData InstancePath Attribute CfgData [\ListNo]
```

InstancePath

数据类型： string

指定一条通往待访问参数的路径。

针对已命名的参数，该字符串的格式为/DOMAIN/TYPE/ParameterName.

针对未命名的参数，该字符串的格式为/DOMAIN/TYPE/Attribute/AttributeValue.

Attribute

数据类型：string

待读取参数的属性名称。

CfgData

数据类型：any type

将储存属性值变量。根据属性类型，有效类型为bool、num或string。

[\ListNo]

数据类型：num

包含待发现Attribute +AttributeValue实例编号的变量。

首次出现的 Attribute+AttributeValue的实例编号为0。如果搜索更多的实例，则\ListNo中的返回值将递增1。否则，如果不再存在实例，则返回值将为-1。预定义常量END_OF_LIST可用于检查是否搜索更多的实例。

下一页继续

程序执行

由Attribute参数规定的属性值，储存在由CfgData参数规定的变量中。

如果使用InstancePath中的格式/DOMAIN/TYPE/ParameterName，则仅可访问已命名的参数，即第一属性为name、Name或NAME的参数。

针对未命名的参数，使用可选参数\ListNo，以选择读取属性值的实例。在每次成功读取后，将其更新为下一个可用的实例。

更多示例

关于指令ReadCfgdata的更多例子，如下所示。此类例子均表明如何读取未命名的参数。

例 1

```
VAR num list_index;
VAR string read_str;
...
list_index:=0;
ReadCfgData "/EIO/EIO_CROSS/Act1/do_13", "Res", read_str,
     \ListNo:=list_index;
IF read_str <> "" THEN
  TPWrite "Resultant signal for signal do_13 is: " + read_str;
ENDIF
```

读取关于未命名数字作用器信号di_13的总信号，并将名称置于string变量read_str中。

在该例子中，域EIO拥有以下cfg代码：

EIO_CROSS:

-名称“Cross_di_1_do_2” -Res “di_1” -Act1 “do_2”

-名称“Cross_di_2_do_2” -Res “di_2” -Act1 “do_2”

-名称“Cross_di_13_do_13” -Res “di_13” -Act1 “do_13”

例 2

```
VAR num list_index;
VAR string read_str;
...
list_index:=0;
WHILE list_index <> END_OF_LIST DO
  read_str:="";
  ReadCfgData "/EIO/EIO_SIGNAL/Device/USERIO", "Name", read_str,
       \ListNo:=list_index;
  IF read_str <> "" THEN
    TPWrite "Signal: " + read_str;
  ENDIF
ENDWHILE
..
ERROR
  TRYNEXT;
```

读取针对I/O设备USERIO而定义的所有信号的名称。

下一页继续

1.188 ReadCfgData - 读取系统参数的属性
RobotWare - OS
续前页

在该例子中，域EIO拥有以下cfg代码：

```
EIO_SIGNAL:
  -Name "USERDO1" -SignalType "DO" -Device "USERIO" -DeviceMap "0"
  -Name "USERDO2" -SignalType "DO" -Device "USERIO" -DeviceMap "1"
  -Name "USERDO3" -SignalType "DO" -Device "USERIO" -DeviceMap "2"
```

例 3

```
VAR num list_index;
VAR string read_str;
...
list_index:=0;
WHILE list_index <> END_OF_LIST DO
  read_str:="";
  ReadCfgData "/EIO/DEVICENET_DEVICE/Network/DeviceNet", "Name",
       read_str, \ListNo:=list_index;
  IF read_str <> "" THEN
    TPWrite read_str;
  ENDIF
ENDWHILE
..
ERROR
  TRYNEXT;
```

读取所有DeviceNet设备的名称。

在该例子中，域EIO拥有以下cfg代码：

```
DEVICENET_DEVICE:
  -Name PANEL -Network "DeviceNet" -Simulated
  -Name DRV_1 -Network "DeviceNet" -Simulated
  -Name DEVICE1 -Network "DeviceNet" -Simulated
  -Name DEVICE2 -Network "DeviceNet" -Simulated
```

错误处理

如果不可能发现由配置数据库中的“InstancePath + Attribute”所指定的数据，则将系统变量ERRNO设置为ERR_CFG_NOTFND。

如果参数CfgData的数据类型不等于配置数据库中“InstancePath + Attribute”所指定的已发现数据的实际数据类型，则将系统变量ERRNO设置为ERR_CFG_ILLTYPE。

如果试图读取内部数据，则将系统变量ERRNO设置为ERR_CFG_INTERNAL。

如果在执行指令时，参数\ListNo中的变量具有超出可用实例（0 ... n）范围的值，则将ERRNO设置为ERR_CFG_OUTOFBOUNDS。

随后，可用错误处理器来处理此类错误。

限制

针对数据类型num的CfgData，从系统参数单元（米、弧度、秒等等）到RAPID程序单元（毫米、度、秒等等）的转换，必须由RAPID程序中的用户来完成。

如果使用InstancePath中的格式/DOMAIN/TYPE/ParameterName，则仅可访问已命名的参数，即第一属性为name、Name或NAME的参数。

将RAPID字符串限制在80个字符。在某些情况下，理论上，该限制对于定义InstancePath、Attribute或CfgData而言过小。

下一页继续

预定义数据

当无法发现更多的实例时，值为-1的预定义常量END_OF_LIST可用于停止读取。

语法

```
ReadCfgData
  [ InstancePath ':=' ] < expression (IN) of string > ','
  [ Attribute ':=' ] < expression (IN) of string > ','
  [ CfgData ':=' ] < variable (VAR) of anytype >
  [ '\' ListNo ':=' < variable (VAR) of num >] ';'
```

相关信息

信息，关于	请参阅
字符串的定义	第*1489*页的*string - 字符串*
写入系统参数的属性	第*919*页的*WriteCfgData - 写入系统参数的属性*
获取当前任务中的机械臂名称	第*1217*页的*RobName - 获取TCP机械臂名称*
配置	技术参考手册 - 系统参数
Advanced RAPID	应用手册 - 控制器软件*IRC5*

1.189 ReadErrData - 获取关于错误的信息

手册用法

ReadErrData用于软中断程序，以获取导致软中断程序被执行的错误、状态变化或警告的信息（域、类型、编号和混合字符串%s）。

基本示例

以下实例介绍了指令ReadErrData：

参阅第*497*页的更多示例一章。

例 1

```
VAR errdomain err_domain;
VAR num err_number;
VAR errtype err_type;
VAR trapdata err_data;
VAR string titlestr;
VAR string string1;
VAR string string2;
...
TRAP trap_err
  GetTrapData err_data;
  ReadErrData err_data, err_domain, err_number,
  err_type \Title:=titlestr \Str1:=string1 \Str2:=string2;
ENDTRAP
```

当错误陷于软中断程序trap_err时，将错误消息中的错误域、错误编号、错误类型和两个混合字符串保存在适当的变量中。

变元

```
ReadErrData TrapEvent ErrorDomain ErrorId ErrorType [\Title] [\Str1]
     [\Str2] [\Str3] [\Str4] [\Str5]
```

TrapEvent

数据类型：trapdata

包含关于促使执行软中断的信息的变量。

ErrorDomain

数据类型：errdomain

将错误域储存在随之出现的错误、状态变化或警告的变量。参考errdomain型预定义数据。

ErrorId

数据类型：num

储存出现的错误编号的变量。在没有完整错误编号的首位（错误域）和起始零位的情况下，错误编号返回。

例如，重启的10008程序作为8返回。

ErrorType

数据类型：errtype

下一页继续

用以储存诸如出现的错误、状态变化或警报等事件类型的变量。参考errtype型预定义数据。

[\Title]

数据类型：string

用于在错误消息中保存标题。标题采用 UTF8 格式，所有语言的所有字符在 FlexPendant 上都不会正确显示。

[\Str1] ... [\Str5]

数据类型：string

通过在错误消息中混合的参数，更新指定的字符串变量。在%s、%f、%d或%ld型消息中，可能存在多达五个参数，在执行该指令期间，其将始终转换为字符串。Str1将保存第一个参数，Str2将保存第二个参数，以此类推。操作手册 - 故障排除对关于消息中参数数量的信息进行了描述。将混合参数标记为该文件中的arg。

程序执行

根据TrapEvent的内容，更新ErrorDomain、ErrorId、ErrorType、 Title和Str1 ... Str5变量。

如果不同的事件与相同的软中断程序相关，则程序必须确保事件与错误监测相关。通过测试INTNO匹配指令IError中使用的中断编号，完成上述操作；

更多示例

有关于指令ReadErrData的更多例子阐述如下。

例 1

```
VAR intnum err_interrupt;
VAR trapdata err_data;
VAR errdomain err_domain;
VAR num err_number;
VAR errtype err_type;
...
PROC main()
  CONNECT err_interrupt WITH trap_err;
  IError COMMON_ERR, TYPE_ERR, err_interrupt;
  ...
  IDelete err_interrupt;
  ...
TRAP trap_err
  GetTrapData err_data;
  ReadErrData err_data, err_domain, err_number, err_type;
  ! Set domain no 1 ... 11
  SetGO go_err1, err_domain;
  ! Set error no 1 ...9999
  SetGO go_err2, err_number;
ENDTRAP
```

出现错误（仅限错误，而非警告或状态变更）时，在软中断程序中检索错误编号，并使用其值来设置2组数字信号输出信号。

下一页继续

1.189 ReadErrData - 获取关于错误的信息

RobotWare - OS

续前页

限制

不可能获取关于内部错误的信息。

语法

```
ReadErrData
  [TrapEvent ':='] <variable (VAR) of trapdata>','
  [ErrorDomain' :='] <variable (VAR) of errdomain>','
  [ErrorId':='] <variable (VAR) of num>','
  [ErrorType' :='] <variable (VAR) of errtype>
  ['\'Title ':='<variable (VAR) of string>]
  ['\'Str1 ':='<variable (VAR) of string>]
  ['\'Str2 ':='<variable (VAR) of string>]
  ['\'Str3 ':='<variable (VAR) of string>]
  ['\'Str4 ':='<variable (VAR) of string>]
  ['\'Str5 ':='<variable (VAR) of string>]';'
```

相关信息

信息，关于	请参阅
中断概述	技术参考手册 - *RAPID*语言概览
更多关于中断管理的信息	技术参考手册 - *RAPID*语言概览
错误域、预定义常量	第*1393*页的*errdomain* - 错误域
错误类型、预定义常量	第*1402*页的*errtype* - 错误类型
调整关于错误的中断	第*234*页的*IError* - 调整关于错误的中断
获取当前TRAP的中断数据	第*219*页的*GetTrapData* - 获取当前*TRAP*的中断数据
Advanced RAPID	应用手册 - 控制器软件*IRC5*

1.190 ReadRawBytes - 读取原始数据字节数据

手册用法

ReadRawBytes用于从通过Open\Bin而打开的设备中，读取rawbytes型数据。

基本示例

以下实例介绍了指令ReadRawBytes：

例 1

```
VAR iodev io_device;
VAR rawbytes raw_data_out;
VAR rawbytes raw_data_in;
VAR num float := 0.2;
VAR string answer;

ClearRawBytes raw_data_out;
PackDNHeader "10", "20 1D 24 01 30 64", raw_data_out;
PackRawBytes float, raw_data_out, (RawBytesLen(raw_data_out)+1)
     \Float4;

Open "/FC1:/dsqc328_1", io_device \Bin;
WriteRawBytes io_device, raw_data_out;
ReadRawBytes io_device, raw_data_in \Time:=1;
Close io_device;
UnpackRawBytes raw_data_in, 1, answer \ASCII:=10;
```

在该例子中，清除raw_data_out，随后加载DeviceNet标题以及大小为0.2的浮动值。

打开设备"/FC1:/dsqc328_1"，将当前raw_data_out中的有效数据写入到设备。随后，本程序最多等待1秒，以从设备进行读取，该程序储存在raw_data_in。

在关闭设备"/FC1:/dsqc328_1"后，将读取的数据作为字符串打开，并储存在中answer。

变元

```
ReadRawBytes IODevice RawData [\Time]
```

IODevice

数据类型：iodev

IODevice为设备的标示符，应当由此读取数据。

RawData

数据类型：rawbytes

RawData为数据容器，其储存从始于索引1的IODevice所读取的数据。

[\Time]

数据类型：num

读取操作（超时）的最长时间以秒计（分辨率为0,001s）。如果未规定该参数，则将最长时间设置为60秒。为了永远等待，使用预定义常量WAIT_MAX。

下一页继续

1.190 ReadRawBytes - 读取原始数据字节数据

RobotWare - OS

续前页

如果在完成读取操作之前耗尽时间，则将通过错误代码ERR_DEV_MAXTIME来调用错误处理器。如果不存在错误处理器，则将停止执行。

程序停止期间，亦使用超时功能，并由程序开始时的RAPID程序进行通知。

程序执行

程序执行期间，从IODevice指明的设备读取数据。

如果将WriteRawBytes用于现场总线命令，例如DeviceNet，则现场总线将始终发送答案。必须通过ReadRawBytes指令，用RAPID来处理答案。

设置RawData变量中有效字节的当前长度，以读取字节数。数据始于RawData中的索引1。

上电失败重启时，将关闭系统中所有打开的文件或串行通道，并将重置iodev型变量中的I/O描述符。

错误处理

如果在读取期间出现错误，则将系统变量ERRNO设置为ERR_FILEACC。

如果在完成读取操作之前超时，则变量RawData中的任何方面均不受影响，并将系统变量ERRNO设置为ERR_DEV_MAXTIME。

随后，可通过错误处理器来处理此类错误。

语法

```
ReadRawBytes
  [IODevice ':=' ] < variable (VAR) of iodev>' ,'
  [RawData ':=' ] < variable (VAR) of rawbytes> ',''
  [ '\' Time ´:=' < expression (IN) of num>] ';'
```

相关信息

信息，关于	请参阅
rawbytes数据	第*1455*页的原始数据字节 - 原始数据
获取rawbytes数据的长度	第*1193*页的*RawBytesLen* - 获取原始数据字节数据的长度
清除rawbytes数据的内容	第*110*页的*ClearRawBytes* - 清除原始数据字节数据的内容
复制rawbytes数据的内容	第*128*页的*CopyRawBytes* - 复制原始数据字节数据的内容
将DeviceNet标题装入rawbytes数据	第*421*页的*PackDNHeader* - 将*DeviceNet*标题装入原始数据字节数据中。
将数据装入rawbytes数据	第*424*页的*PackRawBytes* - 将数据装入原始数据字节数据
写入rawbytes数据	第*923*页的*WriteRawBytes* - 写入原始数据字节数据
打开来自rawbytes数据的数据	第*845*页的*UnpackRawBytes* - 打开来自原始数据字节数据的数据
文件和串行通道处理	应用手册 - 控制器软件*IRC5*

1.191 RemoveAllCyclicBool - 撤除所有进行循环求值的逻辑条件

手册用法

RemoveAllCyclicBool将用于撤除所有逻辑条件的循环求值。

基本示例

下列示例说明了指令RemoveAllCyclicBool。

例 1

```
PERS bool cyclicflag1;
TASK PERS bool cyclicflag2;
PERS bool mypersbool:=FALSE;

PROC main()
  SetupCyclicBool cyclicflag1, di1=1 AND do2=1;
  SetupCyclicBool cyclicflag2, di3 AND di4 AND mypersbool=FALSE;
  ...
  RemoveAllCyclicBool;
  ...
```

首先，设置两个循环求值，然后，随后撤销所有逻辑条件的循环求值。

语法

```
RemoveAllCyclicBool ';'
```

相关信息

信息，关于	请参阅
设置进行循环求值的逻辑条件	第596页的*SetupCyclicBool - 设置进行循环求值的逻辑条件*
撤销进行循环求值的逻辑条件	第502页的*RemoveCyclicBool - 撤销进行循环求值的逻辑条件*
进行循环求值的逻辑条件，*Cyclic bool*。	应用手册 - 控制器软件*IRC5*

1.192 RemoveCyclicBool - 撤销进行循环求值的逻辑条件

手册用法

RemoveCyclicBool将用于撤除特定逻辑条件的循环求值。

基本示例

下列示例说明了指令RemoveCyclicBool。

例 1

```
PERS bool cyclicflag1;

PROC main()
SetupCyclicBool cyclicflag1, di1=1 AND do2=1;
...
RemoveCyclicBool cyclicflag1;
...
```

首先，设置一个循环求值，然后，随后撤销所有逻辑条件的循环求值。

变元

```
RemoveCyclicBool Flag
```

Flag

数据类型：bool

保存逻辑条件值的永久布尔变量。

语法

```
RemoveCyclicBool
  [ Flag ':=' ] <persistent (PERS) of bool> ';'
```

相关信息

信息，关于	请参阅
设置进行循环求值的逻辑条件	第596页的*SetupCyclicBool - 设置进行循环求值的逻辑条件*
撤除所有进行循环求值的逻辑条件	第501页的*RemoveAllCyclicBool - 撤除所有进行循环求值的逻辑条件*
进行循环求值的逻辑条件，*Cyclic bool*。	应用手册 - 控制器软件*IRC5*

1.193 RemoveDir - 删除路径

手册用法

RemoveDir用于移除路径。

用户必须写入并执行关于路径的许可，且该路径必须为空白。

基本示例

以下实例介绍了指令RemoveDir：

例 1

```
RemoveDir "HOME:/mydir";
```

在该例子中，删除HOME:下的mydir路径。

变元

```
RemoveDir Path
```

Path

数据类型：string

通过完整或相关的路径，指定待移除路径的名称。

错误处理

如果不存在路径，或路径并非空白，或用户并未写入和执行程序库的许可，则将系统变量ERRNO设置为ERR_FILEACC。随后，可通过错误处理器来处理该错误。

语法

```
RemoveDir
[ Path':=' ] < expression (IN) of string>';'
```

相关信息

信息，关于	请参阅
目录	第*1384*页的*dir - 路径结构*
打开路径	第*419*页的*OpenDir - 打开路径*
读取路径	第*1197*页的*ReadDir - 读取路径中的下个条目*
关闭路径	第*117*页的*CloseDir - 关闭路径*
建立路径	第*322*页的*MakeDir - 创建新路径*
重命名文件	第*507*页的*RenameFile - 重命名文件*
删除文件	第*504*页的*RemoveFile - 删除文件*
复制文件	第*126*页的*CopyFile - 复制文件*
检查文件类型	第*1126*页的*IsFile - 检查文件的类型*
检查文件大小	第*1077*页的*FileSize - 检索文件的大小*
检查文件系统大小	第*1082*页的*FSSize - 检索文件系统的大小*
文件和串行通道处理	应用手册 - 控制器软件*IRC5*

1.194 RemoveFile - 删除文件

手册用法

RemoveFile用于移除文件。用户必须写入和执行关于文件保存路径的许可，且用户必须针对文件自身写入许可。

基本示例

以下实例介绍了指令RemoveFile：

例 1

```
RemoveFile "HOME:/mydir/myfile.log";
```

在该例子中，删除磁盘HOME:上路径mydir中的文件myfile.log。

变元

```
RemoveFile Path
```

Path

数据类型：字符串

通过完整或相关的路径，指定待删除文件的名称。

错误处理

如果不存在该文件，则将系统变量ERRNO设置为ERR_FILEACC。随后，可用错误处理器对该错误进行处理。

语法

```
RemoveFile
[ Path':=' ] < expression (IN) of string>';'
```

相关信息

信息，关于	请参阅
建立路径	第*322*页的*MakeDir - 创建新路径*
删除路径	第*503*页的*RemoveDir - 删除路径*
重命名文件	第*507*页的*RenameFile - 重命名文件*
复制文件	第*126*页的*CopyFile - 复制文件*
检查文件类型	第*1126*页的*IsFile - 检查文件的类型*
检查文件大小	第*1077*页的*FileSize - 检索文件的大小*
检查文件系统大小	第*1082*页的*FSSize - 检索文件系统的大小*
文件和串行通道处理	应用手册 - 控制器软件*IRC5*

1.195 RemoveSuperv - 撤除一个信号的条件

手册用法

RemoveSuperv将用于撤销由SetupSuperv从监控添加的条件。

基本示例

```
PROC main()
  InitSuperv;
  SetupSuperv diWR_EST, ACT, SUPERV_MAIN \ErrIndSig:= do_WR_Sup;
  SetupSuperv diGA_EST, ACT, SUPERV_MAIN;
  CapL p2, v100, cdata1, weavestart, weave,fine, tWeldGun;
  RemoveSuperv di_Arc_Sup, ACT, SUPERV_START;
ENDPROC
```

撤销START列表中的信号*di_Arc_Sup*。

变元

```
RemoveSuperv Signal Condition Listtype
```

Signal

数据类型：signaldi

从监控列表撤销的数字信号

Condition

数据类型：num

代表下列可用条件之一的名称：

ACT:	用于状态监控。监控期间的预计信号状态：有源。如果信号变为无源，那么监控将启动。
PAS:	用于状态监控。监控期间的预计信号状态：无源。如果信号变为有源，那么监控将启动。
POS_EDGE:	用于握手监控。监控结束时的预计信号状态：有源。如果信号在选定超时范围内未变为有源，那么监控将启动。
NEG_EDGE:	用于握手监控。监控结束时的预计信号状态：无源。如果信号在选定超时范围内未变为无源，那么监控将启动。

Listtype

数据类型：num

代表不同列表（比如，进程中的各阶段）的编号的名称：

- SUPERV_PRE
- SUPERV_PRE_START
- SUPERV_END_PRE
- SUPERV_START
- SUPERV_MAIN
- SUPERV_END_MAIN
- SUPERV_START_POST1
- SUPERV_POST1
- SUPERV_END_POST1

下一页继续

1.195 RemoveSuperv - 撤除一个信号的条件

Continuous Application Platform (CAP)

续前页

- SUPERV_START_POST2
- SUPERV_POST2
- SUPERV_END_POST2

语法

```
RemoveSuperv
  [Signal ':='] < variable (VAR) of signaldi > ','
  [Condition ':='] < variable (IN) of num > ','
  [Listtype ':='] < variable (IN) of num >';'
```

相关信息

信息，关于	请参阅
Continuous Application Platform	应用手册 - *Continuous Application Platform*
`InitSuperv`指令	第*258*页的*InitSuperv - 重置CAP*的所有监控
`SetupSuperv`指令	第*599*页的*SetupSuperv - 设置CAP*信号监控条件

1.196 RenameFile - 重命名文件

手册用法

RenameFile用于为现有文件命名。其亦可用于将文件从路径结构中的一处移至另一处。

基本示例

以下实例介绍了指令RenameFile：

例 1

```
RenameFile "HOME:/myfile", "HOME:/yourfile;
```

将文件myfile命名为yourfile。

```
RenameFile "HOME:/myfile", "HOME:/mydir/yourfile";
```

将文件myfile命名为yourfile，并移动至路径mydir。

变元

```
RenameFile OldPath NewPath
```

OldPath

数据类型：string

有待重命名的文件的完整路径。

NewPath

数据类型：string

重命名文件的完整路径。

程序执行

OldPath中指定的文件，将以NewPath中指定的名称进行命名。如果NewPath中的路径与OldPath中的路径不同，则文件亦将移动至新位置。

错误处理

如果NewPath中指定的文件已经存在，则将系统变量ERRNO设置为ERR_FILEEXIST。随后，可以用错误处理器来处理该错误。

语法

```
RenameFile
[ OldPath' :=' ] < expression (IN) of string > ','
[ NewPath' :=' ] < expression (IN) of string >';'
```

相关信息

信息，关于	请参阅
建立路径	第*322*页的*MakeDir - 创建新路径*
删除路径	第*503*页的*RemoveDir - 删除路径*
删除文件	第*504*页的*RemoveFile - 删除文件*
复制文件	第*126*页的*CopyFile - 复制文件*
检查文件类型	第*1126*页的*IsFile - 检查文件的类型*

下一页继续

1.196 RenameFile - 重命名文件

RobotWare - OS

续前页

信息，关于	请参阅
检查文件大小	第*1077*页的*FileSize - 检索文件的大小*
检查文件系统大小	第*1082*页的*FSSize - 检索文件系统的大小*
文件和串行通道处理	应用手册 - 控制器软件*IRC5*

1.197 Reset - 重置数字信号输出信号

手册用法

Reset为，用于将数字信号输出信号的值重置为零。

基本示例

以下实例介绍了指令Reset：

例 1

```
Reset do15;
```

将信号do15设置为0。

例 2

```
Reset weld;
```

将信号weld设置为0。

变元

```
Reset Signal
```

Signal

数据类型：signaldo

有待重置为零的信号的名称。

程序执行

真实值取决于信号的配置。如果在系统参数中反转信号，则该指令将物理通道设置为1。

错误处理

可能会产生下列可恢复错误。错误可以用错误处理器进行处理。将系统变量ERRNO设置为：

如果信号变量是RAPID中声明的变量，则ERR_NO_ALIASIO_DEF。尚未同I/O配置以及指令AliasIO中确定的I/O信号相连。

如果与I/O单元无接触，则ERR_NORUNUNIT。

如果无法访问I/O信号（仅对ICI现场总线有效），则ERR_SIG_NOT_VALID。

语法

```
Reset
  [ Signal ':=' ] < variable (VAR) of signaldo > ';'
```

相关信息

信息，关于	请参阅
设置数字信号输出信号	第576页的*Set - 设置数字信号输出信号*
输入/输出指令	技术参考手册 - *RAPID*语言概览
输入/输出功能性概述	技术参考手册 - *RAPID*语言概览
I/O配置	技术参考手册 - 系统参数

1.198 ResetPPMoved - 重置以手动模式移动的程序指针的状态

手册用法

ResetPPMoved重置以手动模式移动的程序指针的状态。当控制器处于手动模式，即运算符键处于手动降速度或手动全速时，如果用户移动了程序指针，则PPMovedInManMode返回TRUE。当该键由自动切换为手动时，或当使用指令ResetPPMoved时，重置程序指针移动状态。

基本示例

以下实例介绍了指令ResetPPMoved：

例 1

```
IF PPMovedInManMode() THEN
  WarnUserOfPPMovement;
  ! DO THIS ONLY ONCE
  ResetPPMoved;
  DoJob;
ELSE
  DoJob;
ENDIF
```

程序执行

针对当前的程序任务，以手动模式重置程序指针的移动状态。

语法

```
ResetPPMoved';'
```

相关信息

信息，关于	请参阅
测试是否以手动模式移动了程序指针。	第*1188*页的*PPMovedInManMode - 测试是否以手动模式移动程序指针。*

1.199 ResetRetryCount - 重置重试次数

手册用法

ResetRetryCount用于重置错误处理器的重试次数。在配置中定义可进行的最多重试次数。

基本示例

以下实例介绍了指令ResetRetryCount：

例 1

```
VAR num myretries := 0;
...
ERROR
  IF myretries > 2 THEN
    ResetRetryCount;
    myretries := 0;
    TRYNEXT;
  ENDIF
  myretries:= myretries + 1;
  RETRY;
...
```

本程序将重试有错误的指令3次，随后，尝试下一指令。在尝试下一指令之前，重置内部系统重置计数（即使由位于TRYNEXT的系统完成上述操作）。

程序执行

针对错误处理器所进行的每次尝试，内部系统计数器将检查是否未超出配置中所指定的最多重试次数。指令ResetRetryCount的执行，将重置计数器，并使其有可能再次撤销最多重试次数。

语法

```
ResetRetryCount ';'
```

相关信息

信息，关于	请参阅
错误处理器	技术参考手册 - *RAPID*语言概览
在错误后恢复执行	第*514*页的*RETRY - 在错误后恢复执行*
配置最多重试次数	技术参考手册 - 系统参数
剩余的重试次数	第*1214*页的*RemainingRetries - 剩余的重试*

1.200 RestoPath - 中断之后，恢复路径

手册用法

RestoPath用于恢复在使用指令StorePath的前一阶段所储存的路径。

本指令仅可用于主任务T_ROB1，或者如果在MultiMove系统中，则可用于运动任务中。

基本示例

以下实例介绍了指令RestoPath：

同时参见下文更多例子。

例 1

```
RestoPath;
```

恢复早期使用StorePath所储存的路径。

程序执行

删除机械臂和外轴的当前移动路径，并恢复早期使用StorePath所储存的路径。注意，不得移动，直至执行指令StartMove，或通过使用来自错误处理器的RETRY而返回。

更多示例

有关于如何使用指令RestoPath的更多例子阐述如下。

例 1

```
ArcL p100, v100, seam1, weld5 \Weave:=weave1, z10, gun1;
...
ERROR
  IF ERRNO=AW_WELD_ERR THEN
    gun_cleaning;
    StartMoveRetry;
  ENDIF
...
PROC gun_cleaning()
  VAR robtarget p1;
  StorePath;
  p1 := CRobT();
  MoveL pclean, v100, fine, gun1;
  ...
  MoveL p1, v100, fine, gun1;
  RestoPath;
ENDPROC
```

如果出现焊接错误，将在程序错误处理器中继续程序执行，其反过来调用gun_cleaning。随后，储存此时正在执行的移动路径，且机械臂移动至位置pclean，并于此处纠正错误。当完成上述操作时，机械臂返回出现错误的位置p1，并再次储存原始移动。随后自动重新开始焊接，这意味着，在焊接开始以及继续普通程序执行之前，首次沿路径反转机械臂。

下一页继续

限制

通过指令StorePath，仅可储存移动路径数据。如果用户想要下达在新路径等级上移动的指令，则必须在StorePath朝路径上的储存停止位置移动之后以及RestoPath朝路径上的储存停止位置移动之前，直接储存实际停止位置。

如果该指令位于移动指令之后，则必须使用停止点（zonedata fine）而非飞越点来编程移动指令，否则，将无法在电源故障后重启。

无法在与任意下列特殊系统事件相连的RAPID程序中执行RestoPath：PowerOn、Stop、QStop、Restart或者Step。

语法

```
RestoPath´;´
```

相关信息

信息，关于	请参阅
储存路径	第*695*页的*StorePath - 发生中断时，存储路径*
更多示例	第*695*页的*StorePath - 发生中断时，存储路径* 第*440*页的*PathRecStart - 起动路径记录器* 第*717*页的*SyncMoveSuspend - 设置独立-半协调移动*

1.201 RETRY - 在错误后恢复执行

手册用法

RETRY指令用于在通过（重新执行）引起错误的指令来起动错误之后，恢复程序执行。

基本示例

以下实例介绍了指令RETRY：

例 1

```
reg2 := reg3/reg4;
...
ERROR
  IF ERRNO = ERR_DIVZERO THEN
    reg4 :=1;
    RETRY;
  ENDIF
```

尝试用reg3除以reg4。如果reg4等于0（除以0），则跳转到错误处理器，其初始化reg4。随后，RETRY指令用于从错误处理器跳转，并再次尝试完成相除。

程序执行

通过（再次执行）引起错误的指令，继续程序执行。

错误处理

如果超出最多重试次数（4次重试），则通过错误消息，停止程序执行。可在系统参数（*System Misc*型）中，配置最多重试次数。

限制

本指令仅能存在于程序的错误处理器中。如果通过使用RAISE指令而产生错误，则通过RETRY指令，无法重新开始程序执行。随后，应当使用指令TRYNEXT。

语法

```
RETRY ';'
```

相关信息

信息，关于	请参阅
错误处理器	技术参考手册 - *RAPID*语言概览
配置最多重试次数	技术参考手册 - 系统参数
继续下一指令	第*824*页的*TRYNEXT* - 跳转至引起错误的指令

1.202 RETURN - 完成程序的执行

手册用法

RETURN 用于完成程序的执行。如果程序是一个函数，则同时返回函数值。

基本示例

以下实例介绍了指令RETURN：

例 1

```
errormessage;
Set do1;
...
PROC errormessage()
  IF di1=1 THEN
    RETURN;
  ENDIF
  TPWrite "Error";
ENDPROC
```

调用errormessage 无返回值程序。如果无返回值程序到达RETURN指令，则在Set do 1过程调用后，程序执行返回指令。

例 2

```
FUNC num abs_value(num value)
  IF value<0 THEN
    RETURN -value;
  ELSE
    RETURN value;
  ENDIF
ENDFUNC
```

函数返回某一数字的绝对值。

变元

```
RETURN [ Return value ]
```

Return value

数据类型：符合函数声明。

函数的返回值。

必须通过函数中存在的RETURN指令，指定返回值。

如果指令存在于无返回值程序或软中断程序中，则不得指定返回值。

程序执行

根据在以下方面使用的程序的类型，RETURN指令的结果可能有所不同：

- 主程序：如果程序拥有执行模式单循环，则停止程序。否则，通过主程序的第一个指令，继续程序执行。
- 无返回值程序：通过过程调用后的指令，继续程序执行。
- 函数：返回函数的值。
- 软中断程序：从出现中断的位置，继续程序执行。

下一页继续

1.202 RETURN - 完成程序的执行
RobotWare - OS
续前页

- 无返回值程序中的错误处理器：通过调用程序以及错误处理器的程序（通过过程调用后的指令），继续程序执行。
- 函数中的错误处理器：返回函数值。

语法

```
RETURN [ <expression> ]';'
```

相关信息

信息，关于	请参阅
函数和无返回值程序	技术参考手册 - *RAPID*语言概览
软中断程序	技术参考手册 - *RAPID*语言概览
错误处理器	技术参考手册 - *RAPID*语言概览

1.203 Rewind - 重绕文件位置

手册用法

Rewind将文件位置设置为文件开头。

基本示例

以下实例介绍了指令Rewind：

另请参阅第*517*页的更多示例

例 1

```
Rewind iodev1;
```

通过iodev1参考的文件，将其文件位置设置为文件开头。

变元

```
Rewind IODevice
```

IODevice

数据类型：iodev

待重绕文件的名称（参考）。

程序执行

将指定文件重绕至开头。

上电失败重启时，将关闭系统中所有打开的文件或串行通道，并将重置iodev型变量中的I/O描述符。

更多示例

有关于指令Rewind的更多例子阐述如下。

例 1

```
! IO device and numeric variable for use together with a binary
! file
VAR iodev dev;
VAR num bindata;

! Open the binary file with \Write switch to erase old contents
Open "HOME:"\File := "bin_file",dev \Write;
Close dev;

! Open the binary file with \Bin switch for binary read and write
! acess
Open "HOME:"\File := "bin_file",dev \Bin;
WriteStrBin dev,"Hello world";

! Rewind the file pointer to the beginning of the binary file
! Read contents of the file and write the binary result on TP
! (gives 72 101 108 108 111 32 119 111 114 108 100 )
Rewind dev;
bindata := ReadBin(dev);
```

下一页继续

1.203 Rewind - 重绕文件位置

RobotWare - OS

续前页

```
WHILE bindata <> EOF_BIN DO
  TPWrite " " \Num:=bindata; bindata := ReadBin(dev);
ENDWHILE

! Close the binary file
Close dev;
```

指令Rewind用于将二进制文件重绕至开头，以便可通过ReadBin来复述文件内容

限制

针对Virtual Controller，存在某种限制，如果已通过\Bin或\Bin \Append开关打开了使用过的文件，则任意Write型指令之前的Rewind将变得无效。将在文件结尾进行写入。

错误处理

如果在重绕期间出现错误，则将系统变量ERRNO设置为ERR_FILEACC。随后，可通过错误处理器来处理该错误。

语法

```
Rewind [IODevice ':='] <variable (VAR) of iodev>';'
```

相关信息

信息，关于	请参阅
文件的打开等	技术参考手册 - *RAPID*语言概览
文件和串行通道处理	应用手册 - 控制器软件*IRC5*

1.204 RMQEmptyQueue - 空白RAPID消息队列

手册用法

在正在执行指令的任务中，RMQEmptyQueue会清空RAPID消息队列（RMQ）。

基本示例

以下实例介绍了指令RMQEmptyQueue：

示例

```
RMQEmptyQueue;
```

RMQEmptyQueue指令从执行任务中的RMQ移除了所有消息。

程序执行

清空执行任务所拥有的RAPID消息序列。可使用所有执行等级的指令。

限制

RMQEmptyQueue仅清空任务（执行指令）中的RAPID消息序列。保留所有其他的RAPID消息序列。

语法

```
RMQEmptyQueue ';'
```

相关信息

信息，关于	请参阅
RAPID消息队列功能描述	应用手册 - 控制器软件*IRC5*，*RAPID*消息序列一节。
rmqmessage数据类型	第*1464*页的*rmqmessage - RAPID*消息序列消息。
将数据发送至RAPID任务序列	第*534*页的*RMQSendMessage - 发送RMQ*数据消息。
将数据发送至RAPID任务序列，并等待客户端的回答	第*537*页的*RMQSendWait - 发送RMQ*数据消息，并等待响应。
寻找RAPID消息序列任务的识别编号	第*520*页的*RMQFindSlot - 从*槽名中寻找槽识别号。
从rmqmessage提取标题数据	第*528*页的*RMQGetMsgHeader - 从RMQ*消息获取标题信息。
从rmqmessage提取数据	第*525*页的*RMQGetMsgData - 从RMQ*消息获取数据部分。
命令和启用特定数据类型的中断	第*274*页的*IRMQMessage - 下达数据类型的RMQ*中断指令。
从指定槽识别号中获取槽名	第*1215*页的*RMQGetSlotName - 获取RMQ*客户端的名称。
从RMQ接收消息	第*531*页的*RMQReadWait - 从RMQ*返回消息。
从RAPID消息序列获取第一条消息	第*522*页的*RMQGetMessage - 获取RMQ*消息。

1.205 RMQFindSlot - 从槽名中寻找槽识别号

手册用法

RMQFindSlot（*RAPID Message Queue Find Slot*）用于寻找针对RAPID任务而配置的RMQ的槽识别号，或者寻找机械臂应用开发器客户端的槽识别号。

基本示例

以下实例介绍了指令RMQFindSlot：

例 1

```
VAR rmqslot myrmqslot;
RMQFindSlot myrmqslot, "RMQ_T_ROB2";
```

获取针对RAPID任务"T_ROB2"而配置的RMQ"RMQ_T_ROB2"的识别号。

变元

```
RMQFindSlot Slot Name
```

Slot

数据类型：rmqslot

返回数字标识符的变量。

Name

数据类型：string

发现识别号的客户端的名称。名称必须正好关于大小字母。如果RAPID任务名为T_ROB1，并使用关于RMQ的名称RMQ_t_rob1，则将会以错误告终（参见下文错误处理章节）。

程序执行

RMQFindSlot指令用于寻找关于已命名RMQ或机械臂应用开发器客户端的槽识别号。

错误处理

系统会生成下列可恢复错误，并在错误处理器中处理这些错误。系统变量ERRNO将被设置成：

ERR_RMQ_NAME	给定槽名无效或未能发现。

语法

```
RMQFindSlot
  [ Slot ':=' ] < variable (VAR) of rmqslot > ','
  [ Name ':=' ] < expression (IN) of string > ';'
```

相关信息

信息，关于	请参阅
RAPID消息队列功能描述	应用手册 - 控制器软件*IRC5*，*RAPID*消息序列一节。
将数据发送至RAPID任务序列	第*534*页的*RMQSendMessage - 发送RMQ数据消息*
从RAPID消息队列中获取第一个消息。	第*522*页的*RMQGetMessage - 获取RMQ消息*

下一页继续

信息，关于	请参阅
将数据发送至RAPID任务序列，并等待客户端的回答	第*537*页的*RMQSendWait - 发送RMQ数据消息，并等待响应*
从rmqmessage提取标题数据	第*528*页的*RMQGetMsgHeader - 从RMQ消息获取标题信息*
命令和启用特定数据类型的中断	第*274*页的*IRMQMessage - 下达数据类型的RMQ中断指令*
从rmqmessage提取数据	第*525*页的*RMQGetMsgData - 从RMQ消息获取数据部分*
从指定槽识别号中获取槽名	第*1215*页的*RMQGetSlotName - 获取RMQ客户端的名称*
RMQ Slot	第*1465*页的*rmqslot - RMQ客户端的识别号*

1.206 RMQGetMessage - 获取RMQ消息

手册用法

RMQGetMessage（*RAPID Message Queue Get Message*）用于从实际程序任务的序列，获取第一条RMQ消息。

基本示例

以下实例介绍了指令RMQGetMessage：

另请参阅第*522*页的更多示例

例 1

```
TRAP msghandler
  VAR rmqmessage myrmqmsg;
  RMQGetMessage myrmqmsg;
  ...
ENDTRAP
```

在软中断程序msghandler中，从RMQ获取rmqmessage，并复制到变量myrmqmsg。

变元

```
RMQGetMessage Message
```

Message

数据类型：rmqmessage

用于储存RMQ消息的变量。

可在rmqmessage中接收到的最大数据尺寸大约为3000字节。

程序执行

指令RMQGetMessage用于从执行指令的任务序列获取第一条消息。如果存在消息，则将其复制到Message变量，随后从序列移除，以便为新消息腾出空间。仅在TRAP等级上支持指令。

更多示例

有关于如何使用指令RMQGetMessage的更多例子阐述如下。

例 1

```
RECORD mydatatype
  int x;
  int y;
ENDRECORD

VAR intnum msgreceive;
VAR mydatatype mydata;

PROC main()
  ! Setup interrupt
  CONNECT msgreceive WITH msghandler;
  ! Order cyclic interrupt to occur for data type mydatatype
  IRMQMessage mydata, msgreceive;
```

下一页继续

```
  WHILE TRUE DO
    ! Performing cycle
    ...
  ENDWHILE
ENDPROC

TRAP msghandler
  VAR rmqmessage message;
  VAR rmqheader header;

  ! Get the RMQ message
  RMQGetMessage message;
  ! Copy RMQ header information
  RMQGetMsgHeader message \Header:=header;

  IF header.datatype = "mydatatype" AND header.ndim = 0 THEN
    ! Copy the data from the message
    RMQGetMsgData message, mydata;
  ELSE
    TPWrite "Received a type not handled or with wrong dimension";
  ENDIF
ENDTRAP
```

当接收到新消息时，执行软中断程序msghandler，并将新消息复制到变量message（指令RMQGetMessage）。随后，复制RMQ标题数据（指令RMQGetMsgHeader）。如果消息为预期数据类型，且拥有适当尺寸，则将数据复制到变量mydata（指令RMQGetMsgData）。

错误处理

系统会生成下列可恢复错误，并在错误处理器中处理这些错误。系统变量ERRNO将被设置成：

ERR_RMQ_NOMSG	目前，序列中并无消息。如果在软中断程序中执行RMQGetMessage两次，则会出现上述情况。如果在命令的TRAP与执行中的指令RMQGetMessage之间存在电源故障，则亦会产生上述错误。上电失败时，RMQ中的消息将会丢失。
ERR_RMQ_INVMSG	如果消息无效，则将舍弃该错误。如果PC应用发出损坏消息，则可能出现上述情况。

限制

在用户执行等级（即使用服务程序）或普通执行等级上，不支持RMQGetMessage。

可在rmqmessage中接收到的最大数据尺寸大约为3000字节。

建议尽可能地重新使用数据类型rmqmessage的变量，以保存RAPID内存。

语法

```
RMQGetMessage
  [ Message ':=' ] < variable (VAR) of rmqmessage > ';'
```

下一页继续

1.206 RMQGetMessage - 获取RMQ消息
FlexPendant Interface, PC Interface, or Multitasking
续前页

相关信息

信息，关于	请参阅
RAPID消息队列功能描述	应用手册 - 控制器软件*IRC5*，*RAPID*消息序列一节。
寻找RAPID消息序列任务的识别编号	第*520*页的*RMQFindSlot - 从槽名中寻找槽识别号*
将数据发送至RAPID任务序列	第*534*页的*RMQSendMessage - 发送RMQ数据消息*
将数据发送至RAPID任务序列，并等待客户端的回答	第*537*页的*RMQSendWait - 发送RMQ数据消息，并等待响应*
从rmqmessage提取标题数据	第*528*页的*RMQGetMsgHeader - 从RMQ消息获取标题信息*
从rmqmessage提取数据	第*525*页的*RMQGetMsgData - 从RMQ消息获取数据部分*
命令和启用特定数据类型的中断	第*274*页的*IRMQMessage - 下达数据类型的RMQ中断指令*
从指定槽识别号中获取槽名	第*1215*页的*RMQGetSlotName - 获取RMQ客户端的名称*
RMQ Message	第*1464*页的*rmqmessage - RAPID消息序列消息*

1.207 RMQGetMsgData - 从RMQ消息获取数据部分

手册用法

RMQGetMsgData（*RAPID Message Queue Get Message Data*）用于获取RMQ消息内的实际数据。

基本示例

以下实例介绍了指令RMQGetMsgData：

另请参阅第*525*页的*RMQGetMsgData - 从RMQ消息获取数据部分*

例 1

```
VAR rmqmessage myrmqmsg;
VAR num data;
...
RMQGetMsgData myrmqmsg, data;
! Handle data
```

从变量myrmqmsg获取数据类型num的数据，并储存在变量data中。

变元

```
RMQGetMsgData Message Data
```

Message

数据类型：rmqmessage

包含接收到的RMQ消息的变量。

Data

数据类型：anytype

预期数据类型的变量，用于储存接收到的数据。

程序执行

指令RMQGetMsgData用于获取RMQ消息内的实际数据，将其从ASCII字符格式转换为二进制数据，编制数据，以查看是否可能将其储存在指令所指定的变量中，随后，将其复制到变量。

更多示例

有关于如何使用指令RMQGetMsgData的更多例子阐述如下。

例 1

```
RECORD mydatatype
  int x;
  int y;
ENDRECORD

VAR intnum msgreceive;
VAR mydatatype mydata;

PROC main()
  ! Setup interrupt
  CONNECT msgreceive WITH msghandler;
```

下一页继续

1.207 RMQGetMsgData - 从RMQ消息获取数据部分

FlexPendant Interface, PC Interface, or Multitasking

续前页

```
    ! Order cyclic interrupt to occur for data type mydatatype
    IRMQMessage mydata, msgreceive;
    WHILE TRUE DO
      ! Performing cycle
      ...
    ENDWHILE
  ENDPROC

  TRAP msghandler
    VAR rmqmessage message;
    VAR rmqheader header;

    ! Get the RMQ message
    RMQGetMessage message;
    ! Copy RMQ header information
    RMQGetMsgHeader message \Header:=header;

    IF header.datatype = "mydatatype" AND header.ndim = 0 THEN
      ! Copy the data from the message
      RMQGetMsgData message, mydata;
    ELSE
      TPWrite "Received a type not handled or with wrong dimension";
    ENDIF
  ENDTRAP
```

当接收到新消息时，执行TRAP程序msghandler，并将新消息复制到变量message（指令RMQGetMessage）。随后，复制RMQ标题数据（指令RMQGetMsgHeader）。如果消息为预期数据类型，且拥有适当尺寸，则将数据复制到变量mydata（指令RMQGetMsgData）。

错误处理

系统会生成下列可恢复错误，并在错误处理器中处理这些错误。系统变量ERRNO将被设置成：

ERR_RMQ_VALUE	接收到的消息与用于参数Data的数据类型不同。
ERR_RMQ_DIM	数据类型相同，但是消息中的数据与参数Data所用变量中的数据的规模有所不同。
ERR_RMQ_MSGSIZE	接收到数据的规模大于有关接收任务的RMQ的最大配置规模。
ERR_RMQ_INVMSG	如果消息无效，则将舍弃该错误。如果PC应用发出损坏消息，则可能出现上述情况。

语法

```
RMQGetMsgData
  [ Message ':=' ] < variable (VAR) of rmqmessage > ','
  [ Data ':=' ] < reference (VAR) of anytype > ';'
```

相关信息

信息，关于	请参阅
RAPID消息队列功能描述	应用手册 - 控制器软件*IRC5*，*RAPID*消息序列一节。

下一页继续

信息，关于	请参阅
寻找RAPID消息序列任务的识别编号	第*520*页的*RMQFindSlot - 从槽名中寻找槽识别号*
将数据发送至RAPID任务序列	第*534*页的*RMQSendMessage - 发送RMQ数据消息*
从RAPID消息队列中获取第一个消息。	第*522*页的*RMQGetMessage - 获取RMQ消息*
将数据发送至RAPID任务序列，并等待客户端的回答	第*537*页的*RMQSendWait - 发送RMQ数据消息，并等待响应*
从`rmqmessage`提取标题数据	第*528*页的*RMQGetMsgHeader - 从RMQ消息获取标题信息*
命令和启用特定数据类型的中断	第*274*页的*IRMQMessage - 下达数据类型的RMQ中断指令*
从指定槽识别号中获取槽名	第*1215*页的*RMQGetSlotName - 获取RMQ客户端的名称*
`RMQ Message`	第*1464*页的*rmqmessage - RAPID消息序列消息*

1.208 RMQGetMsgHeader - 从RMQ消息获取标题信息

手册用法

RMQGetMsgHeader（*RAPID Message Queue Get Message Header*）获取已接收RMQ消息内的标题信息，并将其储存在rmqheader、rmqslot或num类变量中。

基本示例

以下实例介绍了指令RMQGetMsgHeader：

另请参阅第*529*页的更多示例

例 1

```
VAR rmqmessage myrmqmsg;
VAR rmqheader myrmqheader;
...
RMQGetMsgHeader myrmqmsg, \Header:=myrmqheader;
```

在该例子中，使用从变量myrmqmsg的rmqheader部分复制的数据，填充变量myrmqheader。

例 2

```
VAR rmqmessage rmqmessage1;
VAR rmqheader rmqheader1;
VAR rmqslot rmqslot1;
VAR num userdef := 0;
...
RRMQGetMsgHeader rmqmessage1 \Header:=rmqheader1 \SenderId:=rmqslot1
     \UserDef:=userdef;
```

在该例子中，使用从变量rmqmessage1复制的数据，填充变量rmqheader1、rmqslot1和userdef。

变元

```
RMQGetMsgHeader Message [\Header] [\SenderId] [\UserDef]
```

Message

数据类型：rmqmessage

包含接收到的RMQ消息的变量，应当由此复制关于消息的信息。

[\Header]

数据类型：rmqheader

用于储存从指定为参数Message的变量复制而来的RMQ标题信息的变量。

[\SenderId]

数据类型：rmqslot

用于储存发送方身份信息的变量，此类信息从指定为参数Message的变量复制而来。

[\UserDef]

User Defined data

数据类型：num

下一页继续

用于储存用户定义数据的变量，该数据从指定为参数Message的变量复制而来。为获取该变量中的所有有效数据，发送方需要详细说明，当发送RMQ消息时，应当囊括该变量。如果未予以使用，则将数值设置为-1。

程序执行

指令RMQGetMsgHeader获取在接收到RMQ消息内的标题信息，并将其复制到rmqheader、rmqslot或num类变量，具体取决于使用的参数。

更多示例

有关于如何使用指令RMQGetMsgHeader的更多例子阐述如下。

例 1

```
RECORD mydatatype
  int x;
  int y;
ENDRECORD

VAR intnum msgreceive;
VAR mydatatype mydata;

PROC main()
  ! Setup interrupt
  CONNECT msgreceive WITH msghandler;
  ! Order cyclic interrupt to occur for data type mydatatype
  IRMQMessage mydata, msgreceive;
  WHILE TRUE DO
    ! Performing cycle
    ...
  ENDWHILE
ENDPROC

TRAP msghandler
  VAR rmqmessage message;
  VAR rmqheader header;

  ! Get the RMQ message
  RMQGetMessage message;
  ! Copy RMQ header information
  RMQGetMsgHeader message \Header:=header;

  IF header.datatype = "mydatatype" AND header.ndim = 0 THEN
    ! Copy the data from the message
    RMQGetMsgData message, mydata;
  ELSE
    TPWrite "Received a type not handled or with wrong dimension";
  ENDIF
ENDTRAP
```

当接收到新消息时，执行TRAP程序msghandler，并将新消息复制到变量message（指令RMQGetMessage）。随后，复制RMQ标题数据（指令RMQGetMsgHeader）。

下一页继续

1.208 RMQGetMsgHeader - 从RMQ消息获取标题信息

FlexPendant Interface, PC Interface, or Multitasking

续前页

如果消息为预期数据类型，且拥有适当尺寸，则将数据复制到变量mydata（指令RMQGetMsgData）。

语法

```
RMQGetMsgHeader
  [ Message ':=' ] < variable (VAR) of rmqmessage > ','
  [ '\' Header ':=' < variable (VAR) of rmqheader >
  [ '\' SenderId ':=' < variable (VAR) of rmqslot >
  [ '\' UserDef ':=' < variable (VAR) of num > ';'
```

相关信息

信息，关于	请参阅
RAPID消息队列功能描述	应用手册 - 控制器软件*IRC5*，*RAPID*消息序列一节。
寻找RAPID消息序列任务的识别编号	第*520*页的*RMQFindSlot - 从槽名中寻找槽识别号*
将数据发送至RAPID任务序列	第*534*页的*RMQSendMessage - 发送RMQ数据消息*
从RAPID消息队列中获取第一个消息。	第*522*页的*RMQGetMessage - 获取RMQ消息*
将数据发送至RAPID任务序列，并等待客户端的回答	第*537*页的*RMQSendWait - 发送RMQ数据消息，并等待响应*
从rmqmessage提取数据	第*525*页的*RMQGetMsgData - 从RMQ消息获取数据部分*
命令和启用特定数据类型的中断	第*274*页的*IRMQMessage - 下达数据类型的RMQ中断指令*
从指定槽识别号中获取槽名	第*1215*页的*RMQGetSlotName - 获取RMQ客户端的名称*
RMQ Slot	第*1465*页的*rmqslot - RMQ客户端的识别号*
RMQ Header	第*1464*页的*rmqmessage - RAPID消息序列消息*
RMQ Message	第*1462*页的*rmqheader - RAPID消息序列消息标题*

1.209 RMQReadWait - 从RMQ返回消息

手册用法

以同步模式使用RMQReadWait，从而接收任意类型的消息。

基本示例

以下实例介绍了指令RMQReadWait：

另请参阅第*531*页的更多示例

示例

```
VAR rmqmessage myrmqmsg;
RMQReadWait myrmqmsg;
```

在变量myrmqmsg中接收到的位于序列中的第一条消息。

变元

```
RMQReadWait Message [\TimeOut]
```

Message

数据类型：rmqmessage

用以放置接收到消息的变量。

[\Timeout]

数据类型：num

程序执行等待消息的最长时间量。如果在满足条件之前耗尽时间，则将调用错误处理器，如果存在这样的情况，则采用错误代码ERR_RMQ_TIMEOUT。如果不存在错误处理器，则将停止执行。有可能将超时设置为0（零）秒，以便不存在任何等待。

如果未使用参数\Timeout，则等待时间为60秒。为了永久等待，则使用预定义常量WAIT_MAX。

程序执行

排列所有的进入消息，且RMQReadWait以FIFO顺序处理消息，即每次处理一条消息。用户有责任避免完整的序列，并准备处理由RAPID消息序列支持的任意消息类型。

更多示例

有关于如何使用指令RMQReadWait的更多例子阐述如下。

例 1

```
VAR rmqmessage myrmqmsg;
RMQReadWait myrmqmsg \TimeOut:=30;
```

在变量myrmqmsg中接收到的位于序列中的第一条消息。如果在30秒内，未接收到任何消息，则停止程序执行。

例 2

```
PROC main()
  VAR rmqmessage myrmqmsg;
  FOR i FROM 1 TO 25 DO
    RMQReadWait myrmqmsg \TimeOut:=30;
    ...
```

下一页继续

1.209 RMQReadWait - 从RMQ返回消息

RobotWare - OS

续前页

```
    ENDFOR

  ERROR
    IF ERRNO = ERR_RMQ_TIMEOUT THEN
      TPWrite "ERR_RMQ_TIMEOUT error reported";
      ...
    ENDIF
  ENDPROC
```

接收来自序列的消息，并将其储存在变量myrmqmsg中。如果接收消息将耗时30秒以上，则调用错误处理器。

错误处理

系统会生成下列可恢复错误，并在错误处理器中处理这些错误。系统变量ERRNO将被设置成：

错误代码	描述
ERR_RMQ_TIMEOUT	在超时时间内未能收到任何答复
ERR_RMQ_INVMSG	如果消息无效，则将舍弃该错误。如果PC应用发出损坏消息，则会出现上述情况

限制

仅以同步模式来支持RMQReadWait。采用基于中断的模式来执行该指令，这将引起致命的运行时间错误。

在软中断执行等级或用户执行等级中，未支持RMQReadWait。以上述任意等级执行该指令，均将引起致命的运行时间错误。

语法

```
RMQReadWait
  [ Message ':=' ] < variable (VAR) of rmqmessage>
  [ '\' TimeOut':=' < expression (IN) of num > ] ';'
```

相关信息

信息，关于	请参阅
RAPID消息队列功能描述	应用手册 - 控制器软件*IRC5*，*RAPID*消息序列一节。
任务执行模式说明	技术参考手册 - 系统参数
rmqmessage数据类型	第*1464*页的*rmqmessage - RAPID*消息序列消息。
将数据发送至RAPID任务序列	第*534*页的*RMQSendMessage - 发送RMQ*数据消息。
将数据发送至RAPID任务序列，并等待客户端的回答	第*537*页的*RMQSendWait - 发送RMQ*数据消息，并等待响应。
寻找RAPID消息序列任务的识别编号	第*520*页的*RMQFindSlot - 从*槽名中寻找槽识别号。
从rmqmessage提取标题数据	第*528*页的*RMQGetMsgHeader - 从RMQ*消息获取标题信息。
从rmqmessage提取数据	第*525*页的*RMQGetMsgData - 从RMQ*消息获取数据部分。

下一页继续

信息，关于	请参阅
命令和启用特定数据类型的中断	第*274*页的*IRMQMessage - 下达数据类型的RMQ中断指令*。
从指定槽识别号中获取槽名	第*1215*页的*RMQGetSlotName - 获取RMQ客户端的名称*。
空白RAPID消息队列	第*519*页的*RMQEmptyQueue - 空白RAPID消息队列*。
从RAPID消息序列获取第一条消息	第*522*页的*RMQGetMessage - 获取RMQ消息*。

1.210 RMQSendMessage - 发送RMQ数据消息

手册用法

RMQSendMessage（*RAPID Message Queue Send Message*）用于将数据发送至针对RAPID任务而配置的RMQ，或者发送至机械臂应用开发器客户端。

基本示例

以下实例介绍了指令RMQSendMessage：

另请参阅第*535*页的更多示例

例 1

```
VAR rmqslot destination_slot;
VAR string data:="Hello world";
..
RMQFindSlot destination_slot,"RMQ_Task2";
RMQSendMessage destination_slot,data;
```

本例子表明了如何通过配置的RMQ“RMQ_Task2”，将变量data中的数值发送到RAPID任务"Task2" 。

例 2

```
VAR rmqslot destination_slot;
CONST robtarget p5:=[ [600, 500, 225.3], [1, 0, 0, 0], [1, 1, 0,
     0], [ 11, 12.3, 9E9, 9E9, 9E9, 9E9] ];
VAR num my_id:=1;
..
RMQFindSlot destination_slot,"RMQ_Task2";
RMQSendMessage destination_slot, p5 \UserDef:=my_id;
my_id:=my_id + 1;
```

本例子表明如何通过配置的RMQ“RMQ_Task2”，将常量p5中的数值发送到RAPID任务"Task2"。同时发送用户定义的编号。接收器可将该编号用作标识符。

变元

```
RMQSendMessage Slot SendData [\UserDef]
```

Slot

数据类型：rmqslot

应当接收消息的客户端的槽识别号。

SendData

数据类型：anytype

包含有待发送给客户端的数据以及作为参数Slot中识别号的变量、永久数据对象或常量的引用。

[\UserDef]

User Defined data

数据类型：num

用以说明SendData接收方的用户定义信息的数据，即正如变量Slot中的客户端以及识别号。数值必须为介于0到32767之间的整数。

下一页继续

程序执行

指令RMQSendMessage用于将数据发送至指定客户端。本指令包装储存容器中的indata，并将其发送。

如果接收客户端对接收消息不感兴趣，即尚未设置任何针对RMQSendMessage指令中指定的数据类型而出现的中断，或并未在RMQSendWait指令中等待，则将舍弃该消息，并将发出警告。

并非所有数据类型都能通过指令来发送（参见限制）。

更多示例

有关于如何使用指令RMQSendMessage的更多例子阐述如下。

例 1

```
MODULE SenderMod
  RECORD msgrec
    num x;
    num y;
  ENDRECORD

  PROC main()
    VAR rmqslot destinationSlot;
    VAR msgrec msg :=[0, 0, 0];

    ! Connect to a Robot Application Builder client
    RMQFindSlot destinationSlot ”My_RAB_client”;

    ! Perform cycle
    WHILE TRUE DO
      ! Update msg with valid data
      ...
      ! Send message
      RMQSendMessage destinationSlot, msg;
      ...
    ENDWHILE
  ERROR
    IF ERRNO = ERR_RMQ_INVALID THEN
      ! Handle destination client lost
      WaitTime 1;
      ! Reconnect to Robot Application Builder client
      RMQFindSlot destinationSlot ”My_RAB_client”;
      ! Avoid execution stop due to retry count exceed
      ResetRetryCount;
      RETRY;
    ELSIF ERRNO = ERR_RMQ_FULL THEN
      ! Handle destination queue full
      WaitTime 1;
      ! Avoid execution stop due to retry count exceed
      ResetRetryCount;
      RETRY;
    ENDIF
  ENDPROC
```

下一页继续

1.210 RMQSendMessage - 发送RMQ数据消息

FlexPendant Interface, PC Interface, or Multitasking

续前页

```
ENDMODULE
```

本例子表明了如何使用指令RMQSendMessage，以及如何处理出现的运行时间错误。本程序将用户定义的msgrec型数据发送至调用“My_RAB_client”的机械臂应用开发器客户端。

错误处理

系统会生成下列可恢复错误，并在错误处理器中处理这些错误。系统变量ERRNO将被设置成：

ERR_RMQ_MSGSIZE	消息规模过大。数据超出容许的最大消息规模，或未配置接收客户端来接收已发送数据的规模。
ERR_RMQ_FULL	目的消息序列已满
ERR_RMQ_INVALID	尚未连接目的槽，或目的槽不再适用。如果未连接，则必须调用RMQFindSlot。如果不可用，则原因在于远程客户端已经同控制器断开。

限制

不可能设置中断，发送或接收非值、半值数据类型或数据类型motsetdata等数据类型的实例。

可发送至机械臂应用开发器客户端的最大数据规模大约为5000字节。可通过RMQ接收，并储存在rmqmessage数据类型中的最大数据规模大约为3000字节。可配置通过RMQ接收的数据规模（默认规模400，最大规模3000）。

语法

```
RMQSendMessage
  [ Slot ':=' ] < variable (VAR) of rmqslot > ','
  [ SendData ':=' ] < reference (REF) of anytype >
  [ '\' UserDef ':=' < expression (IN) of num > ] ';'
```

相关信息

信息，关于	请参阅
RAPID消息队列功能描述	应用手册 - 控制器软件*IRC5*，*RAPID*消息序列一节。
寻找RAPID消息序列任务的识别编号	第*520*页的*RMQFindSlot - 从槽名中寻找槽识别号*
从RAPID消息队列中获取第一个消息。	第*522*页的*RMQGetMessage - 获取RMQ消息*
将数据发送至RAPID任务序列，并等待客户端的回答	第*537*页的*RMQSendWait - 发送RMQ数据消息，并等待响应*
从rmqmessage提取标题数据	第*528*页的*RMQGetMsgHeader - 从RMQ消息获取标题信息*
从rmqmessage提取数据	第*525*页的*RMQGetMsgData - 从RMQ消息获取数据部分*
命令和启用特定数据类型的中断	第*274*页的*IRMQMessage - 下达数据类型的RMQ中断指令*
从指定槽识别号中获取槽名	第*1215*页的*RMQGetSlotName - 获取RMQ客户端的名称*
RMQ Slot	第*1465*页的*rmqslot - RMQ客户端的识别号*

1.211 RMQSendWait - 发送RMQ数据消息，并等待响应

手册用法

通过RMQSendWait（*RAPID Message Queue Send Wait*）指令，有可能将数据发送到RMQ或机械臂应用开发器客户端，并等待指定客户端回复。如果使用该指令，则用户需要知晓客户端回复中所发送的数据类型。

基本示例

以下实例介绍了指令RMQSendWait：

另请参阅第*539*页的更多示例

例 1

```
VAR rmqslot destination_slot;
VAR string sendstr:="This string is from T_ROB1";
VAR rmqmessage receivemsg;
VAR num mynum;
..
RMQFindSlot destination_slot, "RMQ_T_ROB2";
RMQSendWait destination_slot, sendstr, receivemsg, mynum;
RMQGetMsgData receivemsg, mynum;
```

本例子表明了如何通过配置的RMQ“RMQ_T_ROB2”，将变量sendstr中的数据发送到RAPID任务“T_ROB2”。目前，指令RMQSendWait等待任务“T_ROB2”的回复。“T_ROB2”中的指令需要发送储存在num数据类型中的数据，以结束等待指令RMQSendWait。当已经收到消息时，通过指令RMQGetMsgData，将数据从变量receivemsg复制到变量mynum。

例 2

```
VAR rmqslot rmqslot1;
VAR string mysendstr;
VAR rmqmessage rmqmessage1;
VAR string receivestr;
VAR num mysendid:=1;
..
mysendstr:="Message from Task1";
RMQFindSlot rmqslot1, "RMQ_Task2";
RMQSendWait rmqslot1, mysendstr \UserDef:=mysendid, rmqmessage1,
     receivestr \TimeOut:=20;
RMQGetMsgData rmqmessage1, receivestr;
mysendid:=mysendid + 1;
```

本例子表明了如何通过配置的RMQ"RMQ_Task2"，将变量mysendstr中的数据发送至RAPID任务“Task2”。同时发送用户定义的编号。该编号可由接收方用作标识符，且必须返回发送方，以结束等待RMQSendWait指令。另一个结束等待指令的要求是从客户端发送正确的数据类型。通过RMQSendWait指令中的变量receivestr来规定该数据类型。在收到消息后，通过指令RMQGetMsgData，将实际数据复制到变量receivestr。

变元

```
RMQSendWait Slot SendData [\UserDef] Message ReceiveDataType
     [\TimeOut]
```

下一页继续

1.211 RMQSendWait - 发送RMQ数据消息，并等待响应
FlexPendant Interface, PC Interface, or Multitasking
续前页

Slot

数据类型：rmqslot

应当接收消息的客户端识别号。

SendData

数据类型：anytype

包含有待发送给客户端的数据以及作为变量Slot中识别号的变量、永久数据对象或常量的引用。

[\UserDef]

User Defined data

数据类型：num

用以说明SendData接收方的用户定义信息的数据，即客户端以及变量Slot中识别号。如果使用该可选参数，则仅当ReceiveDataType且指定的UserDef符合消息回复时，将结束RMQSendWait指令。数值必须为介于0到32767之间的整数。

Message

数据类型：rmqmessage

用以放置接收到消息的变量。

ReceiveDataType

数据类型：anytype

本指令等待的永久数据对象、变量或常量数据类型的引用。当执行RMQSendWait时，未将实际数据复制到该变量。该参数仅用于指定RMQSendWait指令所等待的实际数据类型。

[\Timeout]

数据类型：num

程序执行等待回复的最长时间量。如果在满足条件之前耗尽时间，则将调用错误处理器，如果存在这样的情况，则采用错误代码ERR_RMQ_TIMEOUT。如果不存在错误处理器，则将停止执行。

如果未使用参数\Timeout，则等待时间为60秒。为了永久等待，则使用预定义常量WAIT_MAX。

程序执行

指令RMQSendWait发送数据，并等待客户端以及指定槽识别号的回复。回复必须为来自客户端的rmqmessage，获取的消息和回复必须与参数ReceiveDataType中指定的数据类型相同。当使用RMQSendMessage时，将以相同的方式来发送消息，即接收方将获得普通的RAPID消息序列消息。发送方负责让接收方知晓所需回复。如果在RMQSendWait中使用可选参数UserDef，则要求接收客户端在回复中使用相同的UserDef。

如果接收客户端不对接收消息感兴趣，即尚未针对RMQSendWait指令中指定的数据类型设置任何中断，则将舍弃该消息，并将发出警告。在参数TimeOut中使用或在长为60 s的默认超时时间之后，本指令返回错误。可通过错误处理器来处理该错误。

如果收到消息，且该消息符合对预期答案以及同TRAP程序相关的消息的描述，则RMQSendWait指令具有最高的优先权（参见指令IRMQMessage）。

下一页继续

如果在等待客户端答案时出现电源故障，则将参数Slot中使用的变量设置为0，并再次执行指令。随后，本指令将失效，因为将调用无效的槽识别号和错误处理器，如果存在这样的情况，则错误代码为ERR_RMQ_INVALID。可由此重新初始化槽识别号。

并非所有数据类型都能通过指令来发送（参见限制）。

更多示例

有关于如何使用指令RMQSendWait的更多例子阐述如下。

例 1

```
MODULE RMQ_Task1_mod
  PROC main()
    VAR rmqslot destination_slot;
    VAR string mysendstr:="String sent from RMQ_Task1_mod";
    VAR string myrecstr;
    VAR rmqmessage recmsg;
    VAR rmqheader header;

    !Get slot identity to client called RMQ_Task2
    RMQFindSlot destination_slot, "RMQ_Task2";

    WHILE TRUE DO
      ! Do something
      ...
      !Send data in mysendstr, wait for an answer of type string
      RMQSendWait destination_slot, mysendstr, recmsg, myrecstr;
      !Get information about the received message
      RMQGetMsgHeader recmsg \Header:=header;
      IF header.datatype = "string" AND header.ndim = 0 THEN
        ! Copy the data in recmsg
        RMQGetMsgData recmsg, myrecstr;
        TPWrite "Received string: " + myrecstr;
      ELSE
        TPWrite "Not a string that was received";
      ENDIF
    ENDWHILE
  ENDPROC
ENDMODULE
```

通过指令RMQSendWait以及配置的RAPID消息序列“RMQ_Task2”，将变量mysendstr中的数据发送至RAPID任务“Task2”。来自RAPID任务“Task2”的回答应当为一个字符串（由变量myrecstr的数据类型指定）。将收到的RMQ消息作为变量recmsg中收到的回答。在调用RMQSendWait的过程中，使用的变量myrecstr符合发送方期望作为回答的数据类型。在RMQSendWait调用过程中，未在变量中放置任何有效数据。

错误处理

系统会生成下列可恢复错误，并在错误处理器中处理这些错误。系统变量ERRNO将被设置成：

ERR_RMQ_MSGSIZE	消息规模过大。数据超出容许的最大消息规模，或未配置接收客户端来接收已发送数据的规模。
ERR_RMQ_FULL	目的消息序列已满。

下一页继续

1.211 RMQSendWait - 发送RMQ数据消息，并等待响应

FlexPendant Interface, PC Interface, or Multitasking

续前页

ERR_RMQ_INVALID	尚未初始化rmqslot，或目的槽不再有效。如果目的槽为远程客户端，且远程客户端已经同控制器断开时，会出现上述情况。电源故障使RMQSendWait中断，并在重启时，将rmqslot设置为0。
ERR_RMQ_TIMEOUT	在超时时间内未能收到任何答复。
ERR_RMQ_INVMSG	如果消息无效，则将舍弃该错误。如果PC应用发出损坏消息，则可能出现上述情况。

限制

不允许以同步模式执行RMQSendWait。这将会引起致命的运行时间错误。

不可能设置中断，发送或接收非值、半值数据类型或数据类型motsetdata等数据类型的实例。

可发送至机械臂应用开发器客户端的最大数据规模大约为5000字节。可通过RMQ接收，并储存在rmqmessage数据类型中的最大数据规模大约为3000字节。可配置通过RMQ接收的数据规模（默认规模400，最大规模3000）。

语法

```
RMQSendWait
  [ Slot ':=' ] < variable (VAR) of rmqslot > ','
  [ SendData ':=' ] < reference (REF) of anytype >
  [ '\' UserDef ':=' < expression (IN) of num > ] ','
  [ Message' :=' ] < variable (VAR) of rmqmessage > ','
  [ ReceiveDataType ':=' ] < reference (REF) of anytype > ','
  [ '\' Timeout ':=' < expression (IN) of num > ] ';'
```

相关信息

信息，关于	请参阅
RAPID消息队列功能描述	应用手册 - 控制器软件*IRC5*，*RAPID*消息序列一节。
寻找RAPID消息序列任务的识别编号	第*520*页的*RMQFindSlot* - 从槽名中寻找槽识别号
将数据发送至RAPID任务序列	第*534*页的*RMQSendMessage* - 发送*RMQ*数据消息
从RAPID消息队列中获取第一个消息。	第*522*页的*RMQGetMessage* - 获取*RMQ*消息
从rmqmessage提取标题数据	第*528*页的*RMQGetMsgHeader* - 从*RMQ*消息获取标题信息
从rmqmessage提取数据	第*525*页的*RMQGetMsgData* - 从*RMQ*消息获取数据部分
命令和启用特定数据类型的中断	第*274*页的*IRMQMessage* - 下达数据类型的*RMQ*中断指令
从指定槽识别号中获取槽名	第*1215*页的*RMQGetSlotName* - 获取*RMQ*客户端的名称
RMQ Slot	第*1465*页的*rmqslot* - *RMQ*客户端的识别号
RMQ Message	第*1464*页的*rmqmessage* - *RAPID*消息序列消息

1.212 SafetyControllerSyncRequest - 硬件同步程序的初始化

手册用法

将采用SafetyControllerSyncRequest来对硬件同步程序进行初始化。

基本示例

下列示例说明了指令SafetyControllerSyncRequest。

例 1

```
SafetyControllerSyncRequest;
```

对硬件同步程序进行初始化

程序执行

此指令必须在同步信号激活前进行调用。

语法

```
SafetyControllerSyncRequest ';'
```

相关信息

信息，关于	请参阅
SafetyControllerGetChecksum	第*1225*页的*SafetyControllerGetChecksum - 获取用户配置文件的检查和。*
SafetyControllerGetSWVersion	第*1226*页的*SafetyControllerGetSWVersion - 获取安全控制柜固件版本*
SafetyControllerGetUserChecksum	第*1227*页的*SafetyControllerGetUserChecksum - 获取受保护参数的检查和*
SafeMove安全配置	应用手册 - *Functional safety and SafeMove*

1.213 Save - 保存普通程序模块

手册用法

Save用于保存普通程序模块。

通过原始（在Load或StartLoad中指定）或指定文件路径，将保存程序内存中的指定普通程序模块。

同时可能在指定文件路径保存系统程序模块。

基本示例

以下实例介绍了指令Save：

另请参阅第*543*页的更多示例

例 1

```
Load "HOME:/PART_B.MOD";
...
Save "PART_B";
```

从程序内存中的HOME:，加载文件名为PART_B.MOD的普通程序模块。

通过原始文件路径HOME: 以及原始文件名称PART_B.MOD，保存普通程序模块PART_B。

变元

```
Save [\TaskRef]|[\TaskName] ModuleName [\FilePath] [\File]
```

[\TaskRef]

Task Reference

数据类型：taskid

应当保存普通程序模块的的程序任务识别号。

对于系统中的所有程序任务，数据类型taskid的预定义变量将有效。可变识别号将为“任务名”+“Id”，例如，针对T_ROB1任务，可变识别号将为T_ROB1Id。

[\TaskName]

数据类型：string

应当保存普通程序模块的的程序任务名称。

如果未指定参数\TaskRef或\TaskName，则将指定普通程序模块保存在当前（执行中）程序任务中。

ModuleName

数据类型：string

待保存的普通程序模块。

[\FilePath]

数据类型：string

普通程序模块保存位置的文件路径和文件名称。当使用参数\File时，应当排除文件名称。

[\File]

数据类型：string

下一页继续

当在参数\FilePath中排除文件名称时，则必须通过该参数来进行指定。

通过Load 或StartLoad-WaitLoad，仅可省略有关普通程序模块的参数\FilePath\File，并将普通程序模块储存在此类指令中指定的同一目的地。为将普通程序模块储存在另一目的地，则亦可能使用参数\FilePath \File。

必须使用参数\FilePath \File ，从而能够保存先前从FlexPendant示教器、外部计算机或系统配置所加载的普通程序模块。

程序执行

程序执行等待普通程序模块在继续下一个指令之前，完成保存。

更多示例

有关于如何使用指令Save的更多例子阐述如下。

例 1

```
Save "PART_A" \FilePath:="HOME:/DOORDIR/PART_A.MOD";
```

将普通程序模块PART_A保存至文件PART_A.MOD以及路径DOORDIR中的HOME:。

例 2

```
Save "PART_A" \FilePath:="HOME:" \File:="DOORDIR/PART_A.MOD";
```

与上述例1相同，但是采用另一种语法。

例 3

```
Save \TaskRef:=TSK1Id, "PART_A"
     \FilePath:="HOME:/DOORDIR/PART_A.MOD";
```

将程序任务 TSK1中的普通程序模块 PART_A保存至指定目的地。这便是有关在一个程序任务中执行指令Save ，并在另一个程序任务中实施保存的例子。

例 4

```
Save \TaskName:="TSK1", "PART_A"
     \FilePath:="HOME:/DOORDIR/PART_A.MOD";
```

将程序任务TSK1中的普通程序模块PART_A保存至指定目的地。这便是有关在一个程序任务中执行指令Save，并在另一个程序任务中实施保存的例子。

限制

保存操作期间，无法执行软中断程序、系统I/O事件和其他程序任务。因此，所有此类操作均将延迟。

保存操作可通过其他程序任务来逐步中断PERS数据的更新。这将产生不一致的PERS数据。

执行保存指令期间，程序停止可导致保护停止以及电机关闭。错误消息“20025停止命令超时”将在FlexPendant示教器上有所延迟。

保存期间，避免机械臂不间断地移动。

错误处理

如果无法在系统中发现参数\TaskName中的程序任务名称，则将系统变量ERRNO设置为ERR_TASKNAME。

如果由于无模块名称、存在未知或歧义的模块名称而无法保存普通程序模块，则将系统变量ERRNO设置为ERR_MODULE。

下一页继续

1.213 Save - 保存普通程序模块

RobotWare - OS

续前页

如果由于拒绝批准、无此类路径或设备上无剩余空间而无法打开保存文件，则将系统变量ERRNO设置为ERR_IOERROR。

如果未针对从FlexPendant示教器、系统参数或外部计算机加载的普通程序模块而指定参数\FilePath，则将系统变量ERRNO设置为ERR_PATH。

上述错误可以用错误处理器来处理。

语法

```
Save
  [[ '\' TaskRef ':=' <variable (VAR) of taskid>]
  |[ '\' TaskName' :=' <expression (IN) of string>] ',']
  [ ModuleName' :=' ] <expression (IN) of string>
  [ '\' FilePath' :='<expression (IN) of string> ]
  [ '\' File' :=' <expression (IN) of string>] ';'
```

相关信息

信息，关于	请参阅
程序任务	第*1499*页的*taskid* - 任务标识

1.214 SaveCfgData - 将系统参数保存至文件

手册用法

SaveCfgData用于将系统参数保存至文件。在通过指令WriteCfgData更新系统参数之后，可能有用。

基本示例

以下实例介绍了指令SaveCfgData。

例 1

```
SaveCfgData "SYSPAR" \File:="MYEIO.cfg", EIO_DOMAIN;
```

将I/O配置域保存至路径SYSPAR中的文件MYEIO.cfg。

例 2

```
SaveCfgData "SYSPAR", ALL_DOMAINS;
```

将所有现有配置域保存至路径SYSPAR。文件将获得EIO.cfg、MMC.cfg、PROC.cfg、SIO.cfg、SYS.cfg和MOC.cfg名称。

变元

```
SaveCfgData FilePath [\File] Domain
```

FilePath

数据类型：string

文件保存位置的文件路径和文件名称。当使用参数\File时，应当排除文件名称。

[\File]

数据类型：string

当在参数\FilePath中排除文件名称时，则必须通过该参数来进行指定。

Domain

数据类型：cfgdomain

有待保存的系统参数域。

程序执行

将系统参数保存至文件。

错误处理

系统会生成下列可恢复错误，并在错误处理器中处理这些错误。系统变量ERRNO将被设置成：

ERR_CFG_ILL_DOMAIN	使用的cfgdomain无效或并未使用。
ERR_CFG_WRITEFILE	路径不存在，或使用的FilePath和File是一个路径，或关于保存文件的一些其他问题。

语法

```
SaveCfgData
  [FilePath ':=' ] <expression (IN) of string>
  ['\' File ':=' <expression (IN) of string>]
  [Domain ':=' ] <expression (IN) of cfgdomain> ';'
```

下一页继续

1.214 SaveCfgData - 将系统参数保存至文件
续前页

相关信息

信息，关于	请参阅
cfgdomain数据	第1373页的*cfgdomain - 配置域*
系统参数	技术参考手册 - 系统参数

1.215 SCWrite - 将变量数据发送到客户端应用

手册用法

SCWrite（*Superior Computer Write*）用于将永久变量的名称、类型和数值发送至客户端应用。可能发送单一变量和变量数组。

基本示例

以下实例介绍了指令SCWrite：

例 1

```
PERS num cycle_done;

PERS num numarr{2}:=[1,2];

SCWrite cycle_done;
```

将永久变量cycle_done的名称、类型和数值发送至所有客户端应用。

例 2

```
SCWrite \ToNode := "138.221.228.4", cycle_done;
```

将永久变量cycle_done的名称、类型和数值发送至所有客户端应用。将忽略参数\ToNode。

例 3

```
SCWrite numarr;
```

将永久变量numarr的名称、类型、尺寸和数值发送至所有客户端应用。

例 4

```
SCWrite \ToNode := "138.221.228.4", numarr;
```

将永久变量numarr的名称、类型、尺寸和数值发送至所有客户端应用。将忽略参数\ToNode。

变元

```
SCWrite [ \ToNode ] Variable
```

[\ToNode]

数据类型：datatype

将忽略参数。

Variable

数据类型：anytype

永久变量的名称。

程序执行

将永久变量的名称、类型、尺寸和数值发送至所有客户端应用。‘尺寸’系指变量的尺寸，且仅当变量为数组时，方才发送。

错误处理

在下列情况下，SCWrite指令将返回错误：

下一页继续

1.215 SCWrite - 将变量数据发送到客户端应用
PC interface/backup
续前页

不能将变量发送至客户端。这可能具有以下原因：

- SCWrite消息过于接近，以致无法将其发送至客户端。解决方案：在SCWrite指令之间增加WaitTime指令。
- 变量值过大，以致减小了ARRAY或RECORD的规模。
- 错误消息将为41473 System access error, Failed to send variable arg1，其中，arg1为变量的名称。

出现错误时，程序停止，且必须重启。ERRNO系统变量将包含数值ERR_SC_WRITE。

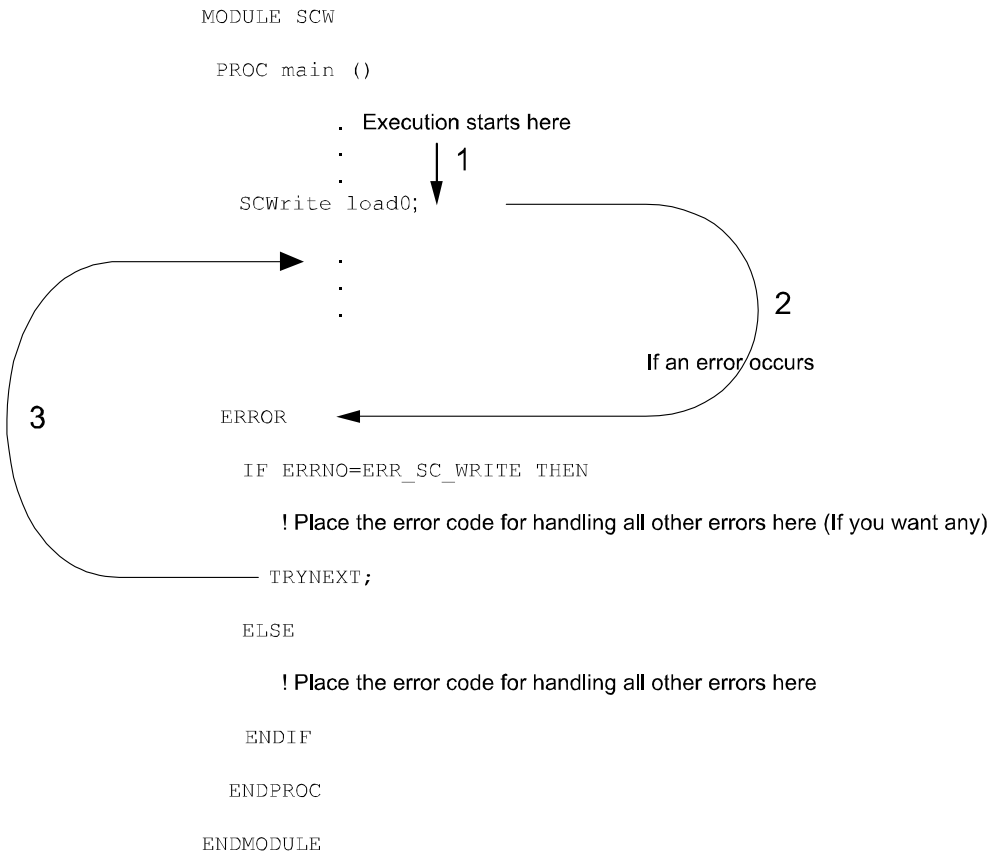
如果客户端应用可能关闭，或通信失效，则SCWrite指令将不会返回错误。程序将继续执行。

SCWrite错误恢复

为避免程序停止，当SCWrite指令中出现错误时，必须通过错误处理器进行处理。仅将错误报告至日志，且程序将继续运行。

由于从未将错误报告给FlexPendant示教器上的显示器（但是可以在日志中发现），切记错误处理将使得发现客户端通信中的错误变得更为困难。

RAPID程序看起来如下所示：



xx0500002339

1.216 SearchC - 使用机械臂沿圆周进行搜索

手册用法

当沿圆周移动工具中心点时，SearchC（*Search Circular*）用于搜索位置。

在移动时，机器人会监控一个数字输入信号或持续变量。当信号持续变量的值变为所需值时，机器人立即读取当前位置。

当由机械臂固定的工具为用于表面探测的探针时，通常可使用该指令。使用SearchC指令，可获得工件的概略坐标。

本指令仅可用于主任务T_ROB1，或者如果在*MultiMove*系统中，则可用于运动任务中。

使用搜索指令时，重要的是配置I/O系统，以便为设置系统物理信号留以极短的时间，从而获得关于设置的信息（使用I/O单元以及中断控制，而非查询控制）。如何进行上述操作，不同的现场总线可能会采用不同的方法。如果使用DeviceNet，则ABB单元DSQC 651（AD Combi I/O）和DSQC 652（数字I/O）将留以短暂的时间，因为他们正在使用状态变化型连接。如果使用其他现场总线，则确保以正确的方式来配置网络，从而获得正确的条件。

基本示例

以下实例介绍了指令SearchC：

另请参阅第*554*页的更多示例

例 1

```
SearchC di1, sp, cirpoint, p10, v100, probe;
```

以v100的速度，使probe的TCP沿圆周朝位置p10 移动。当信号di1 的值改变为有效时，将位置储存在sp中。

例 2

```
SearchC \Stop, di2, sp, cirpoint, p10, v100, probe;
```

将probe的TCP沿圆周朝位置p10移动。当信号di2 的数值改变为有效时，将位置储存在sp中，且机械臂立即停止。

例 3

```
PERS bool mypers:=FALSE;
...
SearchC \Stop, mypers, sp, cirpoint, p10, v100, probe;
```

probe 的 TCP 朝位置 p10 以圆周方式移动。当持续变量 mypers 变为 TRUE 时，位置存储到 sp 且机器人立即停止。

变元

```
SearchC [\Stop] | [\PStop] | [\SStop] | [\Sup] Signal | PersBool
     [\Flanks] | [\PosFlank] | [\NegFlank] | [\HighLevel] |
     [\LowLevel] SearchPoint CirPoint ToPoint [\ID] Speed [\V] |
     [\T] Tool [\WObj] [\Corr] [\TLoad]
```

[\Stop]

Stiff Stop

数据类型：switch

下一页继续

1.216 SearchC - 使用机械臂沿圆周进行搜索
RobotWare - OS
续前页

当搜索信号值变为有效或持续变量值变为 TRUE 时，机器人会尽快停止动作，不会保持 TCP 在路径上（硬停止）。机器人在停止前会移动一小段距离，且不会移回搜索位置，即信号或持续变量改变时的位置。

> **警告**
>
> 如果TCP速度低于100 mm/s，为停止搜索，仅允许硬停止（开关 \Stop）。当以更高的速度硬停止时，一些轴可朝不可预知的方向移动。

[\PStop]

Path Stop

数据类型：switch

当搜索信号值变为有效或持续变量值变为 TRUE 时，机器人会尽快停止动作，同时保持 TCP 在路径上（软停止）。机器人在停止前会移动一小段距离，且不会移回搜索位置，即信号或持续变量改变时的位置。机器人在停止前会移动一小段距离，且不会移回搜索位置，即信号或持续变量改变时的位置。

[\SStop]

Soft Stop

数据类型：switch

当搜索信号值变为有效或持续变量值变为 TRUE 时，机器人会尽快停止移动同时保持 TCP 靠近或在路径上（软停止）。机器人在停止前只会移动一小段距离，且不会移回搜索位置，即信号改变时的位置。SStop 比 PStop 速度更快，但是当机器人运行速度大于 100 mm/s 时，会停在前进方向的切线上，这会导致其稍稍滑出路径。

[\Sup]

Supervision

数据类型：switch

在整个移动过程中，搜索指令对信号激活或持续变量值变化都很敏感（飞速搜索），也就是说，即使在报告了第一次次信号变化或持续变量更改后，也是如此。如果在搜索期间发生了一次以上的匹配，则会生成一个可恢复错误，机器人处于 ToPoint。

如果省略（完全未使用任何开关）参数\Stop、\PStop、\SStop或 \Sup：

- 继续移动（飞焊搜索）至ToPoint参数（与参数\Sup相同）中指定的位置
- 因无搜索匹配而报错，但是不会因多个搜索匹配而报错（第一个搜索匹配作为SearchPoint返回）

Signal

数据类型：signaldi

待监督信号的名称。

PersBool

数据类型：bool

要监控的持续变量。

[\Flanks]

数据类型：switch

下一页继续

信号的正负边缘对搜索匹配有效。如果使用变元 PersBool，则变量的值变化对搜索匹配有效。

对于信号：如果变元 \Flanks 缺失，则仅信号的正边缘对搜索匹配有效，信号监控将在搜索过程开始时激活。这意味着如果信号在搜索过程开始时已经为正值，或与信号的通信中断，则机器人移动将会尽快停止，同时保持TCP在路径上（软停止）。此时会生成用户可恢复错误 ERR_SIGSUPSEARCH，此错误可由错误处理程序处理。

对于持续变量：如果变元 \Flanks 缺失，则仅当其值变为 TRUE 时属于有效搜索匹配，变量监控将在搜索过程开始时激活。这意味着如果持续变量在搜索过程开始时已经为正值，则机器人移动将会尽快停止，同时保持TCP在路径上（软停止）。此时会生成用户可恢复错误 ERR_PERSSUPSEARCH，此错误可由错误处理程序处理。

[\PosFlank]

数据类型：switch

信号的正边缘对搜索匹配有效（在使用持续变量时则值变为 TRUE 时有效）。

[\NegFlank]

数据类型：switch

信号的负边缘对搜索匹配有效（在使用持续变量时则值变为 FALSE 时有效）。

[\HighLevel]

数据类型：switch

与不使用 \Flanks 开关时功能相同。

对于信号：信号的正边缘对搜索匹配有效，信号监控将在搜索过程开始时激活。这意味着如果信号在搜索过程开始时已经为正值，或与信号的通信中断，则机器人移动将会尽快停止，同时保持 TCP 在路径上（软停止）。此时会生成用户可恢复错误 ERR_SIGSUPSEARCH，此错误可由错误处理程序处理。

对于持续变量：仅当其值变为 TRUE 时属于有效搜索匹配，变量监控将在搜索过程开始时激活。这意味着如果持续变量在搜索过程开始时已经为正值，则机器人移动将会尽快停止，同时保持 TCP 在路径上（软停止）。此时会生成用户可恢复错误 ERR_PERSSUPSEARCH，此错误可由错误处理程序处理。

[\LowLevel]

数据类型：switch

对于信号：信号的负边缘对搜索匹配有效，信号监控将在搜索过程开始时激活。这意味着如果信号在搜索过程开始时已经为 0 值，或与信号的通信中断，则机器人移动将会尽快停止，同时保持 TCP 在路径上（软停止）。此时会生成用户可恢复错误 ERR_SIGSUPSEARCH，此错误可由错误处理程序处理。

对于持续变量：仅当其值变为 TRUE 时属于有效搜索匹配，变量监控将在搜索过程开始时激活。这意味着如果持续变量在搜索过程开始时已经为 FALSE 值，则机器人移动将会尽快停止，同时保持 TCP 在路径上（软停止）。此时会生成用户可恢复错误 ERR_PERSSUPSEARCH，此错误可由错误处理程序处理。

SearchPoint

数据类型： robtarget

TCP和外轴在已经触发搜索信号时的位置。通过指定的工具、工件和有效 ProgDisp/ExtOffs坐标系into consideration.，在最外面的坐标系中指定位置。

下一页继续

1.216 SearchC - 使用机械臂沿圆周进行搜索

RobotWare - OS

续前页

`CirPoint`

数据类型：`robtarget`

机械臂的圆周点。有关对圆周运动的更为详细的描述，请参见指令`MoveC`。将圆周点定义为已命名的位置，或直接储存在指令中（在指令中标记有*）。

`ToPoint`

数据类型：`robtarget`

机械臂和外轴的目的点。其定义为已命名的位置，或直接储存在指令中（在指令中标记有*）。`SearchC`始终将停止点作为目的地的区域数据。

`[ \ID ]`

Synchronization id

数据类型：`identno`

如果移动是同步或协调同步的，变元 `[ \ID ]` 在 MultiMove 系统中是强制的。这个变元在任何其他情况下都不允许使用。指定的 ID 号必须与所有协作程序任务中的 ID 号相同。使用此 ID 号，动作才不会在运行时搞混。

`Speed`

数据类型：`speeddata`

适用于运动的速度数据。速度数据规定了工具中心点、外轴和工具方位调整的速率。

`[ \V ]`

Velocity

数据类型：`num`

该参数用于规定指令中TCP的速率，以mm/s计。随后，取代速度数据中指定的相关速率。

`[ \T ]`

Time

数据类型：`num`

该参数用于规定机械臂运动的总时间，以秒计。随后，取代相关的速度数据。

`Tool`

数据类型：`tooldata`

当机器人移动时工具处于使用状态。工具中心点是移向指定目标点的点。

`[ \WObj ]`

Work Object

数据类型：`wobjdata`

同指令中的机械臂位置相关的工件（坐标系）。

此边缘可以缺失，此时位置与世界坐标系关联。另一方面，如果使用了静态 TCP 或协调外部轴，则必须指定此变元才能执行相对此工件的线性移动。

`[ \Corr ]`

Correction

数据类型：`switch`

下一页继续

当存在该参数时，通过指令CorrWrite，将写入修正条目的修正数据添加至路径和目的位置。

[\TLoad]

Total load

数据类型：loaddata

\TLoad主动轴描述了移动中使用的总负载。总负载就是相关的工具负载加上该工具正在处理的有效负载。如果使用了\TLoad自变数，那么就不考虑当前tooldata中的loaddata。

如果\TLoad自变数被设置成load0，那么就不考虑\TLoad自变数，而是以当前tooldata中的loaddata作为代替。

想要使用\TLoad自变数，就必需将系统参数ModalPayLoadMode的数值设置成0。如果将ModalPayLoadMode设置成0，那么就再也无法使用指令GripLoad。

可用服务例程“负载标识”（LoadIdentify）来识别总负载。如果系统参数ModalPayLoadMode被设置成0，且系统正在运行该服务例程，那么操作员便可将相关工具的loaddata复制到一个现有的或新的loaddata永久变量中。

如果使用了关联到系统输入项SimMode（仿真模式）上的一个数字输入信号，那么便可在没有任何有效负载的情况下试运行该程序。如果该数字输入信号被设置成1，那么就不考虑可选自变数\TLoad中的loaddata，而是以当前tooldata中的loaddata作为代替。

注意

处理有效负载的默认功能是使用指令GripLoad，因此系统参数ModalPayLoadMode的默认值为1。

程序执行

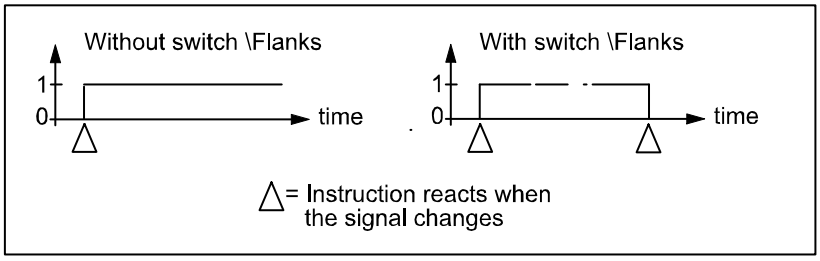
关于圆周运动的信息，请参见指令MoveC。

运动通常以停止点结束，即机械臂在目的点停止。

当使用飞焊搜索，即指定\Sup参数，或未指定任何开关时，机械臂通常将继续移动至编程的目的点。当使用开关\Stop、\PStop或\SStop进行搜索时，机械臂在探测到第一个搜索匹配时停止移动。

SearchC指令返回当数字信号或持续变量值变为所需值时的TCP位置，如下图所示。

本图显示了如何使用侧面触发的信号探测（仅当信号第一次改变时，储存该位置）。



xx0500002237

下一页继续

1.216 SearchC - 使用机械臂沿圆周进行搜索

RobotWare - OS

续前页

更多示例

有关于如何使用指令SearchC的更多例子阐述如下。

例 1

```
SearchC \Sup, di1\Flanks, sp, cirpoint, p10, v100, probe;
```

probe 的TCP沿圆周朝位置p10移动。当信号di1 的值改变为主动或被动时，将位置储存在sp中。如果信号值改变两次，则程序会产生错误。

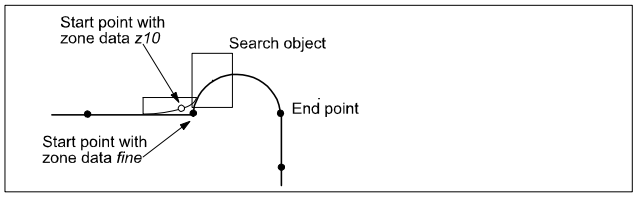
限制

符合指令MoveC的一般限制

必须小心地使用定位指令（领先SearchC）的区域数据。在这种情况下，搜索起点（即当I/O信号准备反应）并非先前定位指令的编程目的点，而是沿实际机械臂路径的一个点。下图阐明了当使用除fine以外的区域数据时，可能出错的实例。

在已经通过圆周点之后，不得重启指令SearchC。否则，机械臂将不会选择编程路径（与编程相比，朝另一方向沿圆周路径定位）。

本图表明如何在工件不正确的一侧进行匹配，因为使用了错误的区域数据。



xx0500002238

下一页继续

警告

同步移动协调时的搜索限制：

- 如果使用关于一个程序任务的 SearchL、SearchC或SearchExtJ以及其他程序任务中的一些其他移动指令，则仅可能使用飞焊搜索以及开关\Sup。此外，仅可能通过TRYNEXT来进行错误恢复。
- 如果使用相关程序任务以及协调同步运动中的指令 SearchL、SearchC 或 SearchExtJ中的一些，并产生相同数字信号输入信号的搜索匹配，则可能使用所有搜索功能。这将同时在所有搜索指令中产生搜索匹配。在所有相关程序任务中，所有错误恢复亦必须相同。

当搜索生效时，通过指令 StorePath，不可能储存当前路径。

TCP速度为20 - 1000 mm/s 0.1 - 0.3 mm的搜索匹配位置的重复精度。

使用50 mm/s的搜索速率时的典型停止距离：

- TCP不在路径上（开关\Stop）时，为1-3 mm
- TCP在路径上（开关\PStop）时，为15-25 mm
- TCP接近路径上（开关\SStop）时，为4-8 mm

在传送带上进行搜索的限制：

- 当搜索匹配时，将使机械臂停止，或如果搜索失败，则使搜索方向与传送带移动方向相同，并在搜索停止以及移动至安全位置之后继续。当搜索失败时，通过错误处理，以移动至安全位置。
- 当在传送带上进行搜索时，关于搜索匹配位置的重复精度将较差，其取决于传送带的速度以及该速度的稳定性。

错误处理

当出现以下情况时，将在搜索期间报错：

- 信号变量是RAPID中声明的变量，且尚未同I/O配置以及指令AliasIO中确定的I/O信号相连，这将产生错误ERR_NO_ALIAS_DEF。
- 未出现信号探测，这将产生错误ERR_WHLSEARCH。
- 出现多个信号探测，仅当使用\Sup参数时，这将产生错误ERR_WHLSEARCH。
- 在搜索过程开始或失去同信号的联系时，信号已经拥有正值。仅当省略\Flanks参数时，这将产生错误ERR_SIGSUPSEARCH。
- 持续变量值在搜索过程开始时为 TRUE - 这仅当\Flanks 变元缺失时会生成错误 ERR_PERSSUPSEARCH。

根据所选的运行模式，错误可以用不同方法来处理。

- **持续向前/指令向前**/ERR_WHLSEARCH：无位置返回，且始终继续朝编程目的点移动。将系统变量ERRNO设置为ERR_WHLSEARCH，且可通过程序的错误处理器来处理错误。
- **连续向前 / 指令向前** / ERR_SIGSUPSEARCH 以及 ERR_PERSSUPSEARCH：不返回位置，移动在搜索路径的开始处尽可能快的停止。根据所使用的变元（信号或持续变量），系统变量 ERRNO 设为 ERR_SIGSUPSEARCH 或 ERR_PERSSUPSEARCH，错误在例行程序的错误处理程序中处理。
- **指令向后**：在向后执行时，指令会进行移动而不进行任何监测。

下一页继续

1.216 SearchC - 使用机械臂沿圆周进行搜索

RobotWare - OS

续前页

语法

```
SearchC
  [ '\' Stop ',']  | [ '\' PStop ',']  | [ '\' SStop ',']  | [ '\'
       Sup ',']
  [ Signal':=' ] < variable (VAR) of signaldi > |
  [ PersBool ':=' ] < persistent (PERS) of bool >
  ['\' Flanks] |
  ['\' PosFlank] |
  ['\' NegFlank] |
  ['\' HighLevel] |
  ['\' LowLevel] ','
  [ SearchPoint':=' ] < var or pers (INOUT) of robtarget > ','
  [ CirPoint':=' ] < expression (IN) of robtarget > ','
  [ ToPoint':=' ] < expression (IN) of robtarget > ','
  [ '\' ID ':=' < expression (IN) of identno >]','
  [ Speed':=' ] < expression (IN) of speeddata >
  [ '\' V ':=' < expression (IN) of num > ]|
  [ '\' T ':=' < expression (IN) of num > ] ','
  [ Tool ':=' ] < persistent (PERS) of tooldata >
  [ '\' WObj':=' < persistent (PERS) of wobjdata > ]
  [ '\' Corr ]
  [ '\' TLoad' :=' < persistent (PERS) of loaddata > ] ';'
```

相关信息

信息，关于	请参阅
线性搜索	第*564*页的*SearchL - 使用机械臂沿直线进行搜索*
写入纠正条目	第*137*页的*CorrWrite - 写入修正发电机*
使机械臂沿圆周移动	第*341*页的*MoveC - 使机械臂沿圆周移动*
圆形运动	技术参考手册 - *RAPID*语言概览
载荷的定义	第*1421*页的*loaddata - 加载数据*
速度的定义	第*1480*页的*speeddata - 速度数据*
工具的定义	第*1502*页的*tooldata - 工具数据*
工件的定义	第*1523*页的*wobjdata - 工件数据*
使用错误处理程序	技术参考手册 - *RAPID*语言概览
一般动作	技术参考手册 - *RAPID*语言概览
关于如何使用TLoad总负载的例子。	第*388*页的*MoveL - 使机械臂沿直线移动*
定义机器人的载荷	第*223*页的*GripLoad - 定义机械臂的有效负载*
LoadIdentify，载荷识别服务例行程序	操作员手册 - 带 *FlexPendant* 的 *IRC5*
用于在仿真模式下无载荷运行机器人的系统输入信号 *SimMode*。 （主题 I/O，类型 System Input，行动值，*SimMode*）	技术参考手册 - 系统参数
激活和停用载荷的系统参数 *ModalPayLoadMode*。 （主题 Controller，类型 System Misc，行动值，*ModalPayLoadMode*）	技术参考手册 - 系统参数

1.217 SearchExtJ - 在没有TCP的情况下，搜索一个或多个机械单元

手册用法

当仅移动线性或旋转外轴时，SearchExtJ（*Search External Joints*）用于搜索外轴位置。在没有TCP的情况下，外轴可属于一个或多个机械单元。

在移动时，机器人会监控一个数字输入信号或持续变量。当信号持续变量的值变为所需值时，机器人立即读取当前位置。

如果出现以下情况，则仅可使用该指令：

- 将实际程序任务定义为运动任务
- 在没有TCP的情况下，本任务控制一个或多个机械单元

使用搜索指令时，重要的是配置I/O系统，以便为设置物理信号留以极短的时间延迟，直至系统获得关于设置的信息（使用I/O单元以及中断控制，而非查询控制）。如何进行上述操作，不同的现场总线可能会采用不同的方法。如果使用DeviceNet，则ABB单元DSQC 651（AD Combi I/O）和DSQC 652（数字I/O）将留以短暂的时间延迟，因为他们正在使用状态变化型连接。如果使用其他现场总线，则确保正确配置网络，从而获得正确的条件。

基本示例

以下实例介绍了指令SearchExtJ：

另请参阅第*560*页的更多示例

例 1

```
SearchExtJ di1, searchp, jpos10, vrot20;
```

机械单元以及旋转轴以速度vrot20，朝位置jpos10移动。当信号di1的数值改变为有效时，将位置储存在searchp中。

例 2

```
SearchExJ \Stop, di2, posx, jpos20, vlin50;
```

机械单元以及线性轴朝位置jpos20移动。当信号di2的值改变为有效时，将位置储存在posx中，并立即停止进行中的移动。

例 3

```
PERS bool mypers:=FALSE;
...
SearchExJ \Stop, di2, posx, jpos20, vlin50;
```

带有线性轴的机械单元朝位置 jpos20 移动，。当持续变量 mypers 的值变为 TRUE 时，位置存储在 posx 而进行中的移动则立即停止。

变元

```
SearchExtJ [\Stop] | [\PStop] | [\SStop] | [\Sup] Signal | PersBool
     [\Flanks] | [\PosFlank] | [\NegFlank] | [\HighLevel] |
     [\LowLevel] SearchJointPos ToJointPos [\ID] [\UseEOffs] Speed
     [\T]
```

[\Stop]

Stiff Stop

数据类型：switch

下一页继续

1.217 SearchExtJ - 在没有TCP的情况下，搜索一个或多个机械单元
RobotWare - OS
续前页

当搜索信号值变为有效或持续变量值变为 TRUE 时，机器人会以硬停止方式尽快停止动作。外部轴会在停止前移动一小段距离，且不会移回搜索位置（即信号改变时的位置）。

[\PStop]

Path Stop

数据类型：switch

当搜索信号值变为有效或持续变量值变为 TRUE 时，机器人会以路径停止方式（程序停止）尽快停止动作。外部轴会在停止前移动相当长的距离，且不会移回搜索位置（即信号改变时的位置）。

[\SStop]

Soft Stop

数据类型：switch

当搜索信号值变为有效或持续变量值变为 TRUE 时，机器人会以快速软停止方式尽快停止动作。外部轴会在停止前只移动一小段距离，且不会移回搜索位置（即信号改变时的位置）。

与SStop. 相比，Stop更快。与PStop.相比，SStop更快。

[\Sup]

Supervision

数据类型：switch

在整个移动过程中，搜索指令对信号激活或持续变量值变化都很敏感（飞速搜索），也就是说，即使在报告了第一次次信号变化或持续变量更改后，也是如此。如果在搜索期间发生了一次以上的匹配，则会生成一个可恢复错误，机器人处于 ToPoint。

如果省略（完全未使用任何开关）参数\Stop、\PStop、\SStop或 \Sup：

- 继续移动（飞焊搜索）至ToJointPos参数（与参数\Sup相同）中指定的位置
- 因一个搜索匹配而报错，但是不会因多个搜索匹配而报错（第一个搜索匹配作为SearchJointPos返回）

Signal

数据类型：signaldi

待监督信号的名称。

PersBool

数据类型：bool

要监控的持续变量。

[\Flanks]

数据类型：switch

信号的正负边缘对搜索匹配有效。如果使用变元 PersBool，则变量的值变化对搜索匹配有效。

对于信号：如果变元 \Flanks 缺失，则仅信号的正边缘对搜索匹配有效，信号监控将在搜索过程开始时激活。这意味着如果信号在搜索过程开始时已经为正值，或与信号的通信中断，则机器人移动将会尽快停止。此时会生成用户可恢复错误 ERR_SIGSUPSEARCH，此错误可由错误处理程序处理。

下一页继续

对于持续变量：如果变元 \Flanks 缺失，则仅当其值变为 TRUE 时属于有效搜索匹配，变量监控将在搜索过程开始时激活。这意味着如果持续变量在搜索过程开始时已经为正值，则机器人移动将会尽快停止。此时会生成用户可恢复错误 ERR_PERSSUPSEARCH，此错误可由错误处理程序处理。

[\PosFlank]

数据类型：switch

信号的正边缘对搜索匹配有效（在使用持续变量时则值变为 TRUE 时有效）。

[\NegFlank]

数据类型：switch

信号的负边缘对搜索匹配有效（在使用持续变量时则值变为 FALSE 时有效）。

[\HighLevel]

数据类型：switch

与不使用 \Flanks 开关时功能相同。

对于信号：信号的正边缘对搜索匹配有效，信号监控将在搜索过程开始时激活。这意味着如果信号在搜索过程开始时已经为正值，或与信号的通信中断，则机器人移动将会尽快停止。此时会生成用户可恢复错误 ERR_SIGSUPSEARCH，此错误可由错误处理程序处理。

对于持续变量：仅当其值变为 TRUE 时属于有效搜索匹配，变量监控将在搜索过程开始时激活。这意味着如果持续变量在搜索过程开始时已经为正值，则机器人移动将会尽快停止。此时会生成用户可恢复错误 ERR_PERSSUPSEARCH，此错误可由错误处理程序处理。

[\LowLevel]

数据类型：switch

对于信号：信号的负边缘对搜索匹配有效，信号监控将在搜索过程开始时激活。这意味着如果信号在搜索过程开始时已经为 0 值，或与信号的通信中断，则机器人移动将会尽快停止。此时会生成用户可恢复错误 ERR_SIGSUPSEARCH，此错误可由错误处理程序处理。

对于持续变量：仅当其值变为 FALSE 时属于有效搜索匹配，变量监控将在搜索过程开始时激活。这意味着如果持续变量在搜索过程开始时已经为 FALSE 值，则机器人移动将会尽快停止。此时会生成用户可恢复错误 ERR_PERSSUPSEARCH，此错误可由错误处理程序处理。

SearchJointPos

数据类型：jointtarget

已触发搜索信号时的外轴位置。该位置将所有有效的ExtOffs纳入考虑。

ToJointPos

数据类型：jointtarget

外轴的目的点。其定义为已命名的位置，或直接储存在指令中（在指令中标记有*）。SearchExtJ始终将停止点作为目的地的区域数据。

[\ID]

Synchronization id

数据类型： identno

下一页继续

1.217 SearchExtJ - 在没有TCP的情况下，搜索一个或多个机械单元
RobotWare - OS
续前页

如果移动是同步或协调同步的，变元 [\ID] 在 MultiMove 系统中是强制的。这个变元在任何其他情况下都不允许使用。指定的 ID 号必须与所有协作程序任务中的 ID 号相同。使用此 ID 号，动作才不会在运行时搞混。

[\UseEOffs]

Use External Offset

数据类型：switch

当使用参数UseEOffs时，针对SearchExtJ指令启用外轴的偏移量，其通过指令EOffsSet来设置。关于外偏移量的更多信息，请参见指令EOffsSet。

Speed

数据类型： speeddata

适用于运动的速度数据。速度数据规定了线性或旋转外轴的速率。

[\T]

Time

数据类型：num

该参数用于规定机械单元运动期间的总时间，以秒计。随后，取代相关的速度数据。

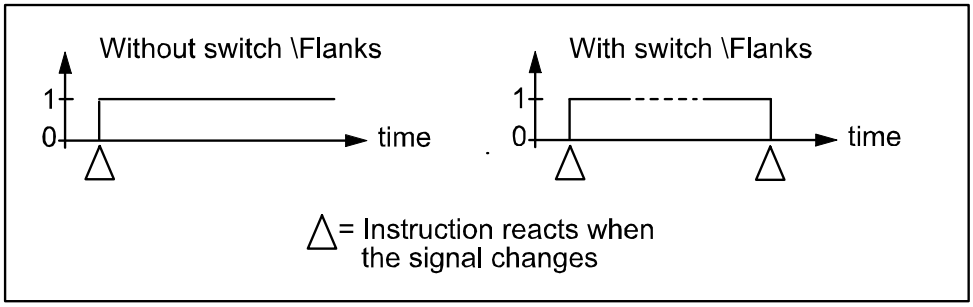
程序执行

关于机械单元在没有TCP的情况下移动的信息，请参见指令MoveExtJ。

移动始终以停止点结束，即外轴在目的点停止。如果使用飞焊搜索，即指定\Sup参数，或未指定开关，则始终继续朝编程目的点移动。如果使用开关\Stop、\PStop或\SStop进行搜索，则在探测到第一个搜索匹配时，移动停止。

SearchExtJ指令返回当数字信号或持续变量值变为所需值时的外部轴位置，如下图所示。

本图显示了如何使用侧面触发的信号探测（仅当信号第一次改变时，储存该位置）。



xx0500002243

更多示例

有关于如何使用指令SearchExtJ的更多例子阐述如下。

例 1

```
SearchExtJ \Sup, di1\Flanks, searchp,jpos10, vrot20;
```

机械单元朝位置jpos10移动。当信号di1 的值改变为主动或被动时，将位置储存在searchp中。如果信号值改变两次，则在完成搜索过程之后，程序将产生错误。

下一页继续

例 2

```
SearchExtJ \Stop, di1, sp, jpos20, vlin50;
MoveExtJ sp, vlin50, fine \Inpos := inpos50;
```

在搜索过程开始时，将对信号dil进行检查，且如果信号已经拥有正值，或失去与信号的通信，则移动停止。否则，机械单元将朝前移动至位置jpos20。当信号di1的值改变为有效时，则将位置储存在sp中。机械单元返回至该点（使用准确定义的停止点）。

错误处理

当出现以下情况时，将在搜索期间报错：

- 信号变量是RAPID中声明的变量，且尚未同I/O配置以及指令AliasIO中确定的I/O信号相连，这将产生错误ERR_NO_ALIASIO_DEF。
- 未出现信号探测，这将产生错误ERR_WHLSEARCH。
- 出现多个信号探测，但是，仅当使用\Sup参数时，这将产生错误ERR_WHLSEARCH。
- 在搜索过程开始或失去同信号的联系时，信号已经拥有正值。但是，仅当省略\Flanks参数时，这将产生错误ERR_SIGSUPSEARCH。
- 持续变量值在搜索过程开始时为 TRUE - 这仅当\Flanks 变元缺失时会生成错误 ERR_PERSSUPSEARCH。

根据所选的运行模式，错误可以用不同方法来处理。

- **持续向前/指令向前**/ERR_WHLSEARCH：无位置返回，且始终继续朝编程目的点移动。将系统变量ERRNO设置为ERR_WHLSEARCH，且可通过程序的错误处理器来处理错误。
- **连续向前** / **指令向前** / ERR_SIGSUPSEARCH 以及 ERR_PERSSUPSEARCH：不返回位置，移动在搜索路径的开始处尽可能快的停止。根据所使用的变元（信号或持续变量），系统变量 ERRNO 设为 ERR_SIGSUPSEARCH 或 ERR_PERSSUPSEARCH，错误在例行程序的错误处理程序中处理。
- **指令向后**：在向后执行时，指令会进行移动而不进行任何监测。

示例

```
VAR num fk;
...
MoveExtJ jpos10, vrot100, fine;
SearchExtJ \Stop, di1, sp, jpos20, vrot5;
...
ERROR
  IF ERRNO=ERR_WHLSEARCH THEN
    StorePath;
    MoveExtJ jpos10, vrot50, fine;
    RestoPath;
    ClearPath;
    StartMove;
    RETRY;
  ELSEIF ERRNO=ERR_SIGSUPSEARCH THEN
    TPWrite "The signal of the SearchExtJ instruction is already
         high!";
```

下一页继续

1.217 SearchExtJ - 在没有TCP的情况下，搜索一个或多个机械单元
RobotWare - OS
续前页

```
    TPReadFK fk,"Try again after manual reset of signal ?","YES",
         stEmpty, stEmpty, stEmpty, "NO";
    IF fk = 1 THEN
      StorePath;
      MoveExtJ jpos10, vrot50, fine;
      RestoPath;
      ClearPath;
      StartMove;
      RETRY;
    ELSE
      Stop;
    ENDIF
  ENDIF
```

如果在搜索过程开始时信号便已生效，或失去同信号的通信，则将启用用户对话框（TPReadFK ...;）。重置信号，按下用户对话框上的YES，机械单元返回jpos10，并再次进行尝试。否则，程序将停止执行。

如果信号在搜索过程开始时为被动，则机械单元将从位置jpos10搜索到位置jpos20。如果未出现信号探测，则机械臂将返回jpos10，并再次进行尝试。

限制

同步移动协调时的搜索限制：

- 如果使用关于一个程序任务的 SearchL、SearchC或SearchExtJ以及另一程序任务中的一些其他移动指令，则仅可能使用飞焊搜索以及开关\Sup。此外，仅可能通过TRYNEXT来进行错误恢复。
- 如果使用相关程序任务以及协调同步运动中的指令SearchL、SearchC 或 SearchExtJ中的一些，并产生相同数字信号输入信号的搜索匹配，则可能使用所有搜索功能。这将同时在所有搜索指令中产生搜索匹配。在所有相关程序任务中，所有错误恢复亦必须相同。
- 当搜索生效时，通过指令StorePath，不可能储存当前路径。

语法

```
SearchExtJ
  [ ’\’ Stop ’,’] | [ ’\’ PStop ’,’] | [ ’\’ SStop ’,’] | [ ’\’
       Sup ’,’]
  [ Signal ’:=’ ] < variable (VAR) of signaldi > |
  [ PersBool ':=' ] < persistent (PERS) of bool >
  [’\’ Flanks] |
  [’\’ PosFlank] |
  [’\’ NegFlank] |
  [’\’ HighLevel] |
  [’\’ LowLevel] ’,’
  [ SearchJointPos’ :=’ ] < var or pers (INOUT) of jointtarget >
       ’,’
  [ ToJointPos’ :=’ ] < expression (IN) of jointtarget >
  [ ’\’ ID ’:=’ < expression (IN) of identno >]’,’
  [ ’\’ UseEOffs’ ,’ ]
  [ Speed ’:=’ ] < expression (IN) of speeddata >
  [ ’\’ T ’:=’ < expression (IN) of num > ] ’;’
```

下一页继续

相关信息

信息，关于	请参阅
在没有TCP的情况下，移动机械单元	第*364*页的*MoveExtJ - 在没有TCP的情况下，移动一个或数个机械单元*
接头位置的定义	第*1418*页的*jointtarget - 接头位置数据*
速度的定义	第*1480*页的*speeddata - 速度数据*
使用错误处理程序	技术参考手册 - *RAPID*语言概览
一般动作	技术参考手册 - *RAPID*语言概览

1.218 SearchL - 使用机械臂沿直线进行搜索

手册用法

当沿直线移动工具中心点时，SearchL（*Search Linear*）用于搜索位置。

在移动时，机器人会监控一个数字输入信号或持续变量。当信号持续变量的值变为所需值时，机器人立即读取当前位置。

当由机械臂固定的工具为用于表面探测的探针时，通常可使用该指令。使用SearchL指令，可获得工件的概略坐标。

本指令仅可用于主任务T_ROB1，或者如果在*MultiMove*系统中，则可用于运动任务中。

使用搜索指令时，重要的是配置I/O系统，以便为设置系统物理信号留以极短的时间，从而获得关于设置的信息（使用I/O单元以及中断控制，而非查询控制）。如何进行上述操作，不同的现场总线可能会采用不同的方法。如果使用DeviceNet，则ABB单元DSQC 651（AD Combi I/O）和DSQC 652（数字I/O）将留以短暂的时间，因为他们正在使用状态变化型连接。如果使用其他现场总线，则确保以正确的方式来配置网络，从而获得正确的条件。

基本示例

以下实例介绍了指令SearchL：

另请参阅第*568*页的更多示例

例 1

```
SearchL di1, sp, p10, v100, probe;
```

以v100的速度，使probe的TCP沿直线朝位置p10 移动。当信号di1的值改变为有效时，将位置储存在sp中。

例 2

```
SearchL \Stop, di2, sp, p10, v100, probe;
```

将probe的TCP沿直线朝位置p10移动。当信号di2的数值改变为有效时，将位置储存在sp中，且机械臂立即停止。

例 3

```
PERS bool mypers:=FALSE;
...
SearchL mypers, sp, p10, v100, probe;
```

probe 的 TCP 朝位置 p10 以直线方式移动，速度为 v100。当持续变量 mypers 变为 TRUE 时，位置存储到 sp。

变元

```
SearchL [\Stop] | [\PStop] | [\SStop] | [\Sup] Signal | PersBool
     [\Flanks] | [\PosFlank] | [\NegFlank] | [\HighLevel] |
     [\LowLevel] SearchPoint ToPoint [\ID] Speed [\V] | [\T] Tool
     [\WObj] [\Corr] [\TLoad]
```

[\Stop]

Stiff Stop

数据类型：switch

下一页继续

当搜索信号值变为有效或持续变量值变为 TRUE 时，机器人会尽快停止动作，不会保持 TCP 在路径上（硬停止）。机器人在停止前会移动一小段距离，且不会移回搜索位置，即信号或持续变量改变时的位置。

警告

如果TCP速度低于100 mm/s，为停止搜索，仅允许硬停止（开关 \Stop）。当以更高的速度硬停止时，一些轴可朝不可预知的方向移动。

`[ \PStop ]`

Path Stop

数据类型：`switch`

当搜索信号值变为有效或持续变量值变为 TRUE 时，机器人会尽快停止动作，同时保持 TCP 在路径上（软停止）。机器人在停止前会移动一小段距离，且不会移回搜索位置，即信号或持续变量改变时的位置。机器人在停止前会移动一小段距离，且不会移回搜索位置，即信号或持续变量改变时的位置。

`[ \SStop ]`

Soft Stop

数据类型：`switch`

当搜索信号值变为有效或持续变量值变为 TRUE 时，机器人会尽快停止移动同时保持 TCP 靠近或在路径上（软停止）。机器人在停止前只会移动一小段距离，且不会移回搜索位置，即信号或持续变量改变时的位置。SStop 比 PStop 速度更快，但是当机器人运行速度大于 100 mm/s 时，会停在前进方向的切线上，这会导致其稍稍滑出路径。

`[ \Sup ]`

Supervision

数据类型：`switch`

在整个移动过程中，搜索指令对信号激活或持续变量值变化都很敏感（飞速搜索），也就是说，即使在报告了第一次次信号变化或持续变量更改后，也是如此。如果在搜索期间发生了一次以上的匹配，则会生成一个可恢复错误，机器人处于 ToPoint。

如果省略参数\Stop、\PStop、\SStop或 \Sup，则（完全未使用任何开关）：

- 继续移动（飞焊搜索）至ToPoint参数（与参数\Sup相同）中指定的位置
- 因无搜索匹配而报错，但是不会因多个搜索匹配而报错（第一个搜索匹配作为SearchPoint返回）

`Signal`

数据类型：`signaldi`

待监督信号的名称。

`PersBool`

数据类型：`bool`

要监控的持续变量。

`[ \Flanks ]`

数据类型：`switch`

下一页继续

信号的正负边缘对搜索匹配有效。如果使用变元 PersBool，则变量的值变化对搜索匹配有效。

对于信号：如果变元 \Flanks 缺失，则仅信号的正边缘对搜索匹配有效，信号监控将在搜索过程开始时激活。这意味着如果信号在搜索过程开始时已经为正值，或与信号的通信中断，则机器人移动将会尽快停止，同时保持TCP在路径上（软停止）。此时会生成用户可恢复错误 ERR_SIGSUPSEARCH，此错误可由错误处理程序处理。

对于持续变量：如果变元 \Flanks 缺失，则仅当其值变为 TRUE 时属于有效搜索匹配，变量监控将在搜索过程开始时激活。这意味着如果持续变量在搜索过程开始时已经为正值，则机器人移动将会尽快停止，同时保持TCP在路径上（软停止）。此时会生成用户可恢复错误 ERR_PERSSUPSEARCH，此错误可由错误处理程序处理。

[\PosFlank]

数据类型：switch

信号的正边缘对搜索匹配有效（在使用持续变量时则值变为 TRUE 时有效）。

[\NegFlank]

数据类型：switch

信号的负边缘对搜索匹配有效（在使用持续变量时则值变为 FALSE 时有效）。

[\HighLevel]

数据类型：switch

与不使用 \Flanks 开关时功能相同。

对于信号：信号的正边缘对搜索匹配有效，信号监控将在搜索过程开始时激活。这意味着如果信号在搜索过程开始时已经为正值，或与信号的通信中断，则机器人移动将会尽快停止，同时保持 TCP 在路径上（软停止）。此时会生成用户可恢复错误 ERR_SIGSUPSEARCH，此错误可由错误处理程序处理。

对于持续变量：仅当其值变为 TRUE 时属于有效搜索匹配，变量监控将在搜索过程开始时激活。这意味着如果持续变量在搜索过程开始时已经为正值，则机器人移动将会尽快停止，同时保持 TCP 在路径上（软停止）。此时会生成用户可恢复错误 ERR_PERSSUPSEARCH，此错误可由错误处理程序处理。

[\LowLevel]

数据类型：switch

对于信号：信号的负边缘对搜索匹配有效，信号监控将在搜索过程开始时激活。这意味着如果信号在搜索过程开始时已经为 0 值，或与信号的通信中断，则机器人移动将会尽快停止，同时保持 TCP 在路径上（软停止）。此时会生成用户可恢复错误 ERR_SIGSUPSEARCH，此错误可由错误处理程序处理。

对于持续变量：仅当其值变为 TRUE 时属于有效搜索匹配，变量监控将在搜索过程开始时激活。这意味着如果持续变量在搜索过程开始时已经为 FALSE 值，则机器人移动将会尽快停止，同时保持 TCP 在路径上（软停止）。此时会生成用户可恢复错误 ERR_PERSSUPSEARCH，此错误可由错误处理程序处理。

SearchPoint

数据类型： robtarget

TCP和外轴在已经触发搜索信号时的位置。在最外面坐标系中指定的位置将指定的工具、工件和有效ProgDisp/ExtOffs坐标系纳入考虑。

下一页继续

ToPoint

数据类型： robtarget

机械臂和外轴的目的点。其定义为已命名的位置，或直接储存在指令中（在指令中标记有*）。SearchL始终将停止点作为目的地的区域数据。

[\ID]

Synchronization id

数据类型： identno

如果移动是同步或协调同步的，变元 [\ID] 在 MultiMove 系统中是强制的。这个变元在任何其他情况下都不允许使用。指定的 ID 号必须与所有协作程序任务中的 ID 号相同。使用此 ID 号，动作才不会在运行时搞混。

Speed

数据类型： speeddata

适用于运动的速度数据。速度数据规定了工具中心点、外轴和工具方位调整的速率。

[\V]

Velocity

数据类型：num

该参数用于规定指令中TCP的速率，以mm/s计。随后，取代速度数据中指定的相关速率。

[\T]

Time

数据类型：num

该参数用于规定机械臂运动的总时间，以秒计。随后，取代相关的速度数据。

Tool

数据类型： tooldata

当机器人移动时工具处于使用状态。工具中心点是移向指定目标点的点。

[\WObj]

Work Object

数据类型： wobjdata

指令中机器人位置关联的工件（坐标系）。

此边缘可以缺失，此时位置与世界坐标系关联。另一方面，如果使用了静态 TCP 或协调外部轴，则必须指定此变元才能执行相对此工件的线性移动。

[\Corr]

Correction

数据类型：switch

如果存在该参数，则将通过指令CorrWrite而写入修正条目的修正数据添加到路径和目的位置。

[\TLoad]

Total load

数据类型：loaddata

下一页继续

续前页

\TLoad主动轴描述了移动中使用的总负载。总负载就是相关的工具负载加上该工具正在处理的有效负载。如果使用了\TLoad自变数，那么就不考虑当前tooldata中的loaddata。

如果\TLoad自变数被设置成load0，那么就不考虑\TLoad自变数，而是以当前tooldata中的loaddata作为代替。

想要使用\TLoad自变数，就必需将系统参数ModalPayLoadMode的数值设置成0。如果将ModalPayLoadMode设置成0，那么就再也无法使用指令GripLoad。

可用服务例程“负载标识”（LoadIdentify）来识别总负载。如果系统参数ModalPayLoadMode被设置成0，且系统正在运行该服务例程，那么操作员便可将相关工具的loaddata复制到一个现有的或新的loaddata永久变量中。

如果使用了关联到系统输入项SimMode（仿真模式）上的一个数字输入信号，那么便可在没有任何有效负载的情况下试运行该程序。如果该数字输入信号被设置成1，那么就不考虑可选自变数\TLoad中的loaddata，而是以当前tooldata中的loaddata作为代替。

注意

处理有效负载的默认功能是使用指令GripLoad，因此系统参数ModalPayLoadMode的默认值为1。

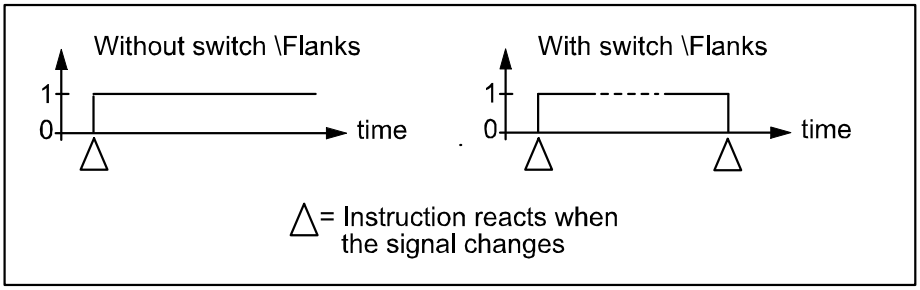
程序执行

有关线性移动的信息请参阅 MoveL 指令。

移动始终以停止点结束，即机械臂在目的点停止。如果使用飞焊搜索，即指定 \Sup 参数，或未指定开关，则机械臂始终继续朝编程目的点移动。如果使用开关 \Stop、\PStop或\SStop进行搜索，则在探测到第一个搜索匹配时，机械臂停止移动。

SearchL 指令储存当数字信号或持续变量值变为所需值时的 TCP 位置，如下图所示。

本图显示了如何使用侧面触发的信号探测（仅当信号第一次改变时，储存该位置）。



xx0500002243

更多示例

有关于如何使用指令SearchL的更多例子阐述如下。

例 1

```
SearchL \Sup, di1 \Flanks, sp, p10, v100, probe;
```

probe的TCP沿直线朝位置p10移动。当信号di1 的值改变为主动或被动时，将位置储存在sp中。如果信号值改变两次，则在完成搜索过程之后，程序会产生错误。

下一页继续

例 2

```
SearchL \Stop, di1, sp, p10, v100, tool1;
MoveL sp, v100, fine \Inpos := inpos50, tool1;
PDispOn *, tool1;
MoveL p100, v100, z10, tool1;
MoveL p110, v100, z10, tool1;
MoveL p120, v100, z10, tool1;
PDispOff;
```

在搜索过程开始时，将对信号di1进行检查，且如果信号已经拥有正值，或失去同信号的通信，则机械臂停止。否则，tool1的TCP沿直线朝位置p10移动。当信号di1的值改变为有效时，将位置储存在sp.中。通过使用准确定义的停止点，机械臂返回该点。通过使用程序位移，机械臂随后移动至搜索位置sp。

例 3

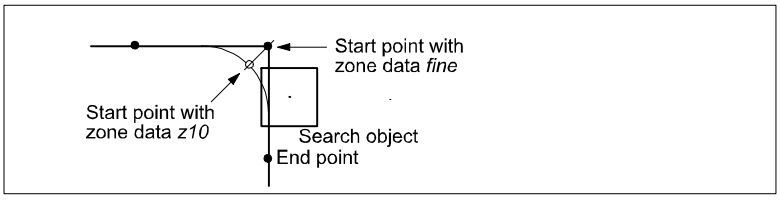
```
PERS bool MyTrigger:=FALSE;
...
SearchL \Stop, MyTrigger, sp, p10, v100, tool1;
MoveL sp, v100, fine \Inpos := inpos50, tool1;
PDispOn *, tool1;
MoveL p100, v100, z10, tool1;
MoveL p110, v100, z10, tool1;
MoveL p120, v100, z10, tool1;
PDispOff;
```

在搜索过程开始时，将会检查持续变量 MyTrigger 的值，如果该变量已经是 TRUE，则机器人停止。否则 tool1 的 TCP 以直线方式朝 p10 位置移动。当持续变量的值 MyTrigger 变为 TRUE，位置存储在 sp. 中。机器人使用精确定义的停止点返回该点。使用程序的偏移量，机器人然后会相对搜索位置 sp 移动。

限制

必须小心地使用定位指令（领先SearchL）的区域数据。在这种情况下，搜索起点（即当I/O信号准备反应）并非先前定位指令的编程目的点，而是沿实际机械臂路径的一个点。下图阐明了当使用除fine以外的区域数据时，可能出错的实例。

下图表明在工件不正确的一侧进行匹配，因为使用了错误的区域数据。

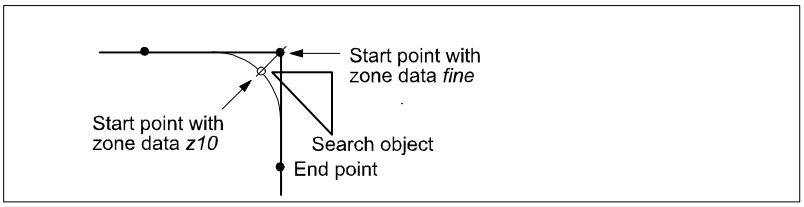


xx0500002244

下一页继续

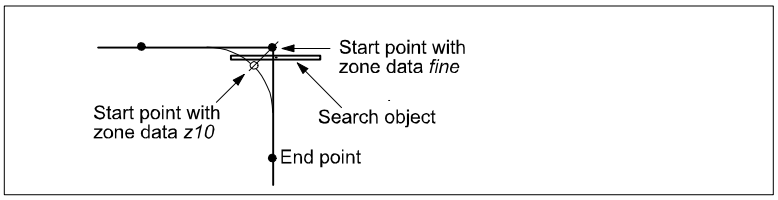
1.218 SearchL - 使用机械臂沿直线进行搜索
RobotWare - OS
续前页

下图表明未探测到匹配，因为使用了错误的区域数据。



xx0500002245

下图表明未探测到匹配，因为使用了错误的区域数据。



xx0500002246

同步移动协调时的搜索限制：

- 如果使用关于一个程序任务的 SearchL、SearchC 或SearchExtJ以及其他程序任务中的一些其他移动指令，则仅可能使用飞焊搜索以及开关\Sup。此外，仅可能通过TRYNEXT来进行错误恢复。
- 如果使用相关程序任务以及协调同步运动中的指令 SearchL、SearchC或SearchExtJ中的一些，并产生相同数字信号输入信号的搜索匹配，则可能使用所有搜索功能。这将同时在所有搜索指令中产生搜索匹配。在所有相关程序任务中，所有错误恢复亦必须相同。

当搜索生效时，通过指令StorePath，不允许储存当前路径。

TCP速度为20 - 1000 mm/s 0.1 - 0.3 mm的搜索匹配位置的重复精度。

使用50 mm/s的搜索速率时的典型停止距离：

- TCP不在路径上（开关\Stop）时，为1-3 mm
- TCP在路径上（开关\PStop）时，为15-25 mm
- TCP接近路径上（开关\SStop）时，为4-8 mm

在传送带上进行搜索的限制：

- 当搜索匹配时，将使机械臂停止，或如果搜索失败，则使搜索方向与传送带移动方向相同，并在搜索停止以及移动至安全位置之后继续。当搜索失败时，通过错误处理，以移动至安全位置。
- 当在传送带上进行搜索时，关于搜索匹配位置的重复精度将较差，其取决于传送带的速度以及该速度的稳定性。

错误处理

当出现以下情况时，将在搜索期间报错：

- 信号变量是RAPID中声明的变量，且尚未同I/O配置以及指令AliasIO中确定的I/O信号相连，这将产生错误ERR_NO_ALIASIO_DEF。
- 未进行检测 - 这将会生成错误 ERR_WHLSEARCH。

下一页继续

- 发生多次检测 – 这将会生成错误 ERR_WHLSEARCH，除非使用了 \Sup 变元。
- 在搜索过程开始或失去同信号的联系时，信号已经拥有正值。仅当省略\Flanks参数时，这将产生错误ERR_SIGSUPSEARCH。
- 持续变量值在搜索过程开始时为 TRUE - 这仅当\Flanks 变元缺失时会生成错误 ERR_PERSSUPSEARCH。

根据所选的运行模式，错误可以用不同方法来处理。

- 持续向前/指令向前/ERR_WHLSEARCH：无位置返回，且始终继续朝编程目的点移动。将系统变量ERRNO设置为ERR_WHLSEARCH，且可通过程序的错误处理器来处理错误。
- 连续向前 / 指令向前 / ERR_SIGSUPSEARCH 以及 ERR_PERSSUPSEARCH：不返回位置，移动在搜索路径的开始处尽可能快的停止。根据所使用的变元（信号或持续变量），系统变量 ERRNO 设为 ERR_SIGSUPSEARCH 或 ERR_PERSSUPSEARCH，错误在例行程序的错误处理程序中处理。
- 指令向后：在向后执行时，指令会进行移动而不进行任何监测。

示例

```
VAR num fk;
...
MoveL p10, v100, fine, tool1;
SearchL \Stop, di1, sp, p20, v100, tool1;
...
ERROR
  IF ERRNO=ERR_WHLSEARCH THEN
    StorePath;
    MoveL p10, v100, fine, tool1;
    RestoPath;
    ClearPath;
    StartMove;
    RETRY;
  ELSEIF ERRNO=ERR_SIGSUPSEARCH THEN
    TPWrite "The signal of the SearchL instruction is already
         high!";
    TPReadFK fk,"Try again after manual reset of signal ?","YES",
         stEmpty, stEmpty, stEmpty, "NO";
    IF fk = 1 THEN
      StorePath;
      MoveL p10, v100, fine, tool1;
      RestoPath;
      ClearPath;
      StartMove;
      RETRY;
    ELSE
      Stop;
    ENDIF
  ENDIF
```

如果在搜索过程开始时信号便已生效，或失去同信号的通信，则将启用用户对话框（TPReadFK ...;）。重置信号，按下用户对话框上的YES，机械臂返回p10，并再次进行尝试。否则，程序将停止执行。

下一页继续

1.218 SearchL - 使用机械臂沿直线进行搜索

RobotWare - OS

续前页

如果信号在搜索过程开始时为被动，则机械臂将从位置p10搜索到位置p20。如果未出现信号探测，则机械臂将返回p10，并再次进行尝试。

语法

```
SearchL
  [ '\' Stop ','] | [ '\' PStop ','] | [ '\' SStop ','] | [ '\'
       Sup ','']
  [ Signal ':=' ] < variable (VAR) of signaldi > |
  [ PersBool ':=' ] < persistent (PERS) of bool >
  ['\' Flanks] |
  ['\' PosFlank] |
  ['\' NegFlank] |
  ['\' HighLevel] |
  ['\' LowLevel] ','
  [ SearchPoint ':=' ] < var or pers (INOUT) of robtarget > ','
  [ ToPoint ':=' ] < expression (IN) of robtarget >
  [ '\' ID ':=' < expression (IN) of identno >] ','
  [ Speed ':=' ] < expression (IN) of speeddata >
  [ '\' V ':=' < expression (IN) of num > ] |
  [ '\' T ':=' < expression (IN) of num > ] ','
  [ Tool ':=' ] < persistent (PERS) of tooldata >
  [ '\' WObj ':=' < persistent (PERS) of wobjdata > ]
  [ '\' Corr ]
  [ '\' TLoad ':=' < persistent (PERS) of loaddata > ] ';'
```

相关信息

信息，关于	请参阅
沿圆周搜索	第*549*页的*SearchC - 使用机械臂沿圆周进行搜索*
写入纠正条目	第*137*页的*CorrWrite - 写入修正发电机*
直线移动机器人	第*388*页的*MoveL - 使机械臂沿直线移动*
线性移动	技术参考手册 - *RAPID*语言概览
载荷的定义	第*1421*页的*loaddata - 加载数据*
速度的定义	第*1480*页的*speeddata - 速度数据*
工具的定义	第*1502*页的*tooldata - 工具数据*
工件的定义	第*1523*页的*wobjdata - 工件数据*
使用错误处理程序	技术参考手册 - *RAPID*语言概览
一般动作	技术参考手册 - *RAPID*语言概览
关于如何使用TLoad总负载的例子。	第*388*页的*MoveL - 使机械臂沿直线移动*
定义机器人的载荷	第*223*页的*GripLoad - 定义机械臂的有效负载*
LoadIdentify，载荷识别服务例行程序	操作员手册 - 带 *FlexPendant* 的 *IRC5*
用于在仿真模式下无载荷运行机器人的系统输入信号 *SimMode*。（主题 I/O，类型 System Input，行动值，*SimMode*）	技术参考手册 - 系统参数

下一页继续

信息，关于	请参阅
激活和停用载荷的系统参数 *ModalPayLoadMode*。 （主题 Controller，类型 System Misc，行动值，*ModalPayLoadMode*）	技术参考手册 - 系统参数

1.219 SenDevice - 与传感器设备相连

手册用法

SenDevice用于连接与串行传感器接口相连的传感器设备。

传感器接口与位于串行通道（使用RTP1传输层协议）上方的传感器进行通信。

此为关于传感器通道配置的实例。

COM_PHY_CHANNEL:

- Name “COM1:”
- Connector “COM1”
- Baudrate 19200

COM_TRP:

- Name “sen1:”
- Type “RTP1”
- PhyChannel “COM1”

基本示例

以下实例介绍了指令SenDevice：

例 1

```
! Define variable numbers
CONST num SensorOn := 6;
CONST num XCoord := 8;
CONST num YCoord := 9;
CONST num ZCoord := 10;
VAR pos SensorPos;
! Connect to the sensor device“ sen1:” (defined in sio.cfg).
SenDevice "sen1:";
! Request start of sensor meassurements
WriteVar "sen1:", SensorOn, 1;
! Read a cartesian position from the sensor.
SensorPos.x := ReadVar "sen1:", XCoord;
SensorPos.y := ReadVar "sen1:", YCoord;
SensorPos.z := ReadVar "sen1:", ZCoord;
! Stop sensor
WriteVar "sen1:", SensorOn, 0;
```

变元

```
SenDevice device
```

device

数据类型：string

在sio.cfg中配置了I/O设备名称，以供传感器使用。

语法

```
ReadBlock
  [ device‘ :=’ ] < expression(IN) of string>’,’
  [ BlockNo’ :=’ ] < expression (IN) of num > ‘,’
```

下一页继续

```
[ FileName' :=' ] < expression (IN) of string > ';'
```

相关信息

信息，关于	请参阅
写入传感器变量	第*927*页的*WriteVar - 写入变量*
读取传感器变量	第*1210*页的*ReadVar - 从设备读取变量*
写入传感器数据块	第*917*页的*WriteBlock - 将数据块写入设备*
传感器通信配置	技术参考手册 - 系统参数

1.220 Set - 设置数字信号输出信号

手册用法

Set用于将数字信号输出信号的值设置为一。

基本示例

以下实例介绍了指令Set：

例 1

```
Set do15;
```

将信号do15设置为1。

例 2

```
Set weldon;
```

将信号weldon设置为1。

变元

```
Set Signal
```

Signal

数据类型：signaldo

有待设置为一的信号的名称。

程序执行

在信号获得其新值之前，存在短暂延迟。如果你想要继续程序执行，直至信号已获得其新值，则可以使用指令SetDO以及可选参数\Sync。

真实值取决于信号的配置。如果在系统参数中反转信号，则该指令将物理通道设置为零。

错误处理

系统会生成下列可恢复错误，并在错误处理器中处理这些错误。系统变量ERRNO将被设置成：

如果信号变量是RAPID中声明的变量，则ERR_NO_ALIASIO_DEF。尚未同I/O配置以及指令AliasIO中确定的I/O信号相连。

如果与I/O单元无接触，则ERR_NORUNUNIT。

如果无法访问I/O信号（仅对ICI现场总线有效），则ERR_SIG_NOT_VALID。

语法

```
Set
  [ Signal ':=' ] < variable (VAR) of signaldo > ';'
```

相关信息

信息，关于	请参阅
将数字信号输出信号设置为零	第*509*页的*Reset - 重置数字信号输出信号*
改变数字信号输出信号值	第*589*页的*SetDO - 改变数字信号输出信号值*
输入/输出指令	技术参考手册 - *RAPID*语言概览

下一页继续

信息，关于	请参阅
输入/输出功能性概述	技术参考手册 - *RAPID*语言概览
I/O配置	技术参考手册 - 系统参数

1.221 SetAllDataVal - 在定义设置下，设置所有数据对象的值

手册用法

SetAllDataVal（*Set All Data Value*）可为了其类型符合指定语法的所有数据对象设置一个新值。

基本示例

以下实例介绍了指令SetAllDataVal：

```
VAR mydata mydata0:=0;
...
SetAllDataVal "mydata"\TypeMod:="mytypes"\Hidden,mydata0;
```

这会将系统中数据类型mydata 的所有数据对象设置为与变量mydata0所有拥有的值相同（参见0的例子）。在模块mytypes中定义了用户定义的数据类型mydata。

变元

```
SetAllDataVal Type [\TypeMod] [\Object] [\Hidden] Value
```

Type

数据类型：string

待设置数据对象的类型名称。

[\TypeMod]

类型、模块

数据类型：string

使用用户定义的数据类型时，用以定义数据类型的模块名称。

[\Object]

数据类型：string

默认行为是设置上述数据类型的所有数据对象，但是该选项使得通过正规表达式来命名一个或多个对象成为可能（亦参见指令SetDataSearch）

[\Hidden]

数据类型：switch

这同时匹配位于由调用链中的一些程序所隐藏的程序中的数据对象（程序数据或参数）。

Value

数据类型：anytype

用以保存待设置新值的变量。数据类型必须与待设置对象的数据类型相同。

程序执行

如果关于Type或TypeMod的规格错误，则指令将失效。

如果相匹配的数据对象为一个数组，则将该数组的所有元素设置为指定值。

如果相匹配的数据对象为只读数据，则数值将不会改变。

如果系统不含任何相匹配的数据对象，则指令将允许接受，并成功地返回。

下一页继续

限制

关于半值数据类型，不可能搜索相关的值数据类型。例如，如果搜索dionum，则不存在关于信号signaldi的搜索匹配，且如果搜索num，则不存在关于信号signalgi或signalai的搜索匹配。

不可能设置在RAPID模式下建立、声明为LOCAL的变量值。

错误处理

可能会产生下列可恢复错误。错误可以由错误处理程序处理。系统变量ERRNO将设置为：

名称	错误原因
ERR_SYMBOL_TYPE	在参数Value中使用的数据对象和变量具有不同的类型。如果使用ALIAS数据类型，则亦将出现此错误，尽管上述类型可能具有相同的基础数据类型。

语法

```
SetAllDataVal
  [ Type ':=' ] < expression (IN) of string >
  ['\'TypeMod' :='<expression (IN) of string>]
  ['\'Object' :='<expression (IN) of string>]
  ['\'Hidden ] ','
  [ Value ':='] <variable (VAR) of anytype>';'
```

相关信息

信息，关于	请参阅
定义在搜索会话中设置的符号	第*582*页的*SetDataSearch - 定义在搜索序列中设置的符号*
获取下一个匹配的符号	第*1095*页的*GetNextSym - 获取下一个匹配的符号*
获取数据对象的值	第*213*页的*GetDataVal - 获得数据对象的值*
设置数据对象的值	第*586*页的*SetDataVal - 设置数据对象的值*
相关的数据类型datapos	第*1382*页的*datapos - 数据类型的封闭块*
Advanced RAPID	应用手册 - 控制器软件*IRC5*

1.222 SetAO - 改变模拟信号输出信号的值

手册用法

SetAO用于改变模拟信号输出信号的值。

基本示例

以下实例介绍了指令SetAO：

另请参阅第*581*页的更多示例

例 1

```
SetAO ao2, 5.5;
```

将信号ao2设置为5.5.

变元

```
SetAO Signal Value
```

Signal

数据类型：signalao

待改变模拟信号输出信号的名称。

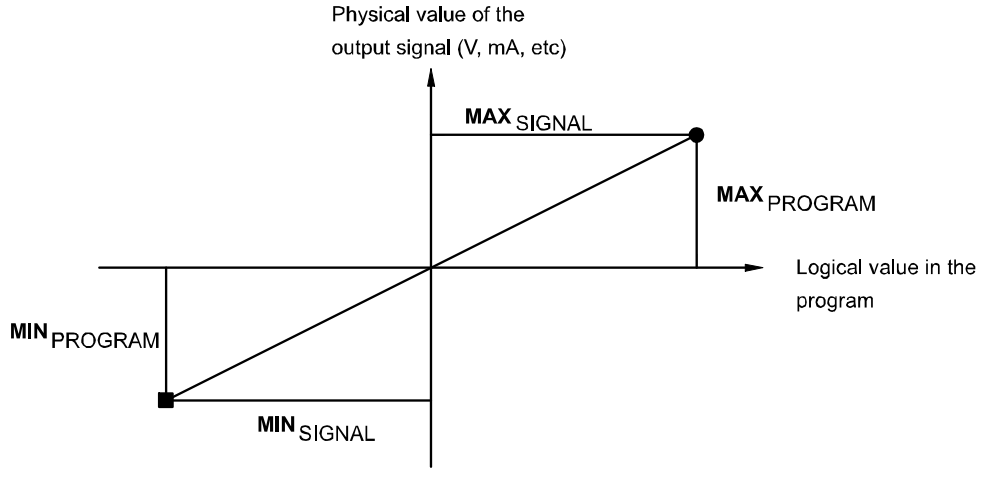
Value

数据类型：num

信号的期望值。

程序执行

在将编程值发送至物理通道之前，对于编程值进行测量（按照系统参数）。下图显示了关于如何测量模拟信号值的图表。



xx0500002408

错误处理

可能会产生下列可恢复错误。错误可以用错误处理器进行处理。将系统变量ERRNO设置为：

ERR_AO_LIM

下一页继续

如果关于指定模拟信号输出信号Signal的编程Value参数超出限值。

如果信号变量是RAPID中声明的变量，则ERR_NO_ALIASIO_DEF。尚未同I/O配置以及指令AliasIO中确定的I/O信号相连。

ERR_NORUNUNIT

如果与I/O单元无接触。

如果无法访问I/O信号（仅对ICI现场总线有效），则ERR_SIG_NOT_VALID。

更多示例

有关于指令SetAO的更多例子阐述如下。

例 1

```
SetAO weldcurr, curr_outp;
```

将信号weldcurr设置为与变量curr_outp当前值相同的值。

语法

```
SetAO
  [ Signal ':=' ] < variable (VAR) of signalao > ','
  [ Value ':=' ] < expression (IN) of num > ';'
```

相关信息

信息，关于	请参阅
输入/输出指令	技术参考手册 - *RAPID*语言概览
输入/输出功能性概述	技术参考手册 - *RAPID*语言概览
I/O配置	技术参考手册 - 系统参数

1.223 SetDataSearch - 定义在搜索序列中设置的符号

手册用法

SetDataSearch与函数GetNextSym一同使用，以从相关系统中检索出数据对象。

基本示例

以下实例介绍了指令SetDataSearch：

例 1

```
VAR datapos block;
VAR string name;
...
SetDataSearch "robtarget"\InTask;
WHILE GetNextSym(name,block \Recursive) DO
...
```

本会话将发现任务中的所有robtarget对象。

变元

```
SetDataSearch Type [\TypeMod] [\Object] [\PersSym]
     [\VarSym][\ConstSym] [\InTask] | [\InMod]
     [\InRout][\GlobalSym] | [\LocalSym]
```

Type

数据类型：string

待检索数据对象的数据类型名称。

[\TypeMod]

Type Module

数据类型：string

使用用户定义的数据类型时，用以定义数据类型的模块名称。

[\Object]

数据类型：string

默认行为是设置上述数据类型的所有数据对象，但是该选项使得通过正规表达式来命名一个或多个数据对象成为可能。

正规表达式是一种强大的机制，可指定用以匹配数据对象名称的语法。字符串由普通字符和元字符组成。元字符是一种特殊的运算符，用以代表字符串中的一个或多个普通字符，旨在拓展研究。可能查看字符串是否匹配指定模式，或在字符串内搜索匹配指定模式的子字符串。

在正规表达式内，所有字母数字字符均自我匹配。也就是说，模式“abc”将仅匹配名为“abc”的数据类型。为匹配所有数据类型，在包含字符序列“abc”的名称中，有必要增加一些元字符。针对于此的正规表达式为“.*abc.*”。

可利用的元字符设置如下所示。

表达式	含义
.	任意单一字符。
[s]	非空集合s中的任意单一字符，其中，s系指字符顺序。可将范围指定为c-c。

下一页继续

表达式	含义
[^s]	不在集合s中的任意单一字符。
r*	关于正规表达式r的零个或多个事件。
r+	关于正规表达式r的一个或多个事件
r?	关于正规表达式r的零个或一个事件。
(r)	正规表达式r。用于将正规表达式分离开来。
r | r'	正规表达式r或r'。
.*	任意字符序列（零、一或多个字符）。

默认行为是接受任意符号，但是，如果指定PersSym、VarSym或ConstSym中之一或多个，则仅接受匹配规格的符号：

[\PersSym]

Persistent Symbols

数据类型：switch

接受永久变量（PERS）符号。

[\VarSym]

Variable Symbols

数据类型：switch

接受变量（VAR）符号。

[\ConstSym]

Constant Symbols

数据类型：switch

接受常量（CONST）符号。

如果未指定标记\InTask或\InMod 之一，则以系统等级开始搜索。系统等级是符号树中所有其他符号定义的根源。在系统等级上，放置所有内置符号，并处理任务等级。在任务等级上，放置所有全局符号，并处理模块等级。

如果在GetNextSym中设置\Recursive标记，则搜索会话将输入系统等级之下的所有加载模块和程序。

[\InTask]

In Task

数据类型：switch

以任务等级开始搜索。在任务等级上，放置所有已加载的全局符号，并处理模块等级。

如果在GetNextSym中设置\Recursive标记，则搜索会话将输入任务等级之下的所有加载模块和程序。

[\InMod]

In Module

数据类型：string

以指定模块等级开始搜索。在模块等级上，放置在指定模块中声明的所有已加载全局和局部符号，并处理程序等级。

下一页继续

1.223 SetDataSearch - 定义在搜索序列中设置的符号

RobotWare - OS

续前页

如果在GetNextSym中设置\Recursive标记，则搜索会话将输入指定模块等级之下的所有加载程序（在指定模块中声明）。

[\InRout]

In Routine

数据类型：string

仅以指定程序等级进行搜索。

必须在参数\InMod.中指定程序的模块名称

默认行为是匹配局部和全局模块符号，但是，如果指定\GlobalSym或\LocalSym之一，则仅接受匹配规格的符号：

[\GlobalSym]

Global Symbols

数据类型：switch

跳过局部模块符号。

[\LocalSym]

Local Symbols

数据类型：switch

跳过全局模块符号。

程序执行

如果关于Type、TypeMod、InMod或InRout之一的规格错误，则指令将失效。

如果系统没有任何匹配的对象，则指令将允许接受，并成功地返回，但是，第一个GetNextSym 将返回FALSE。

限制

无法在符号搜索集合中确定数组数据对象，亦无法在搜索序列中发现数组数据对象。

关于半值数据类型，不可能搜索相关的值数据类型。例如，如果搜索dionum，则不存在关于信号signaldi的搜索匹配，且如果搜索num，则不存在关于信号signalgi或signalai的搜索匹配。

永远不会发现已安装且称之为LOCAL的内置符号，无论参数\GlobalSym、\LocalSym或其他的使用情况如何。

始终会发现已安装且称之为全局或TASK的内置符号，无论参数\GlobalSym、\LocalSym或其他的使用情况如何。

不可能使用SetDataSearch来搜索通过RAPID代码而定义的一些ALIAS数据类型的数据。不存在任何关于预定义ALIAS数据类型的限制。

语法

```
SetDataSearch
  [ Type ':=' ] < expression (IN) of string >
  ['\'TypeMod ':='<expression (IN) of string>]
  ['\'Object ':='<expression (IN) of string>]
  ['\'PersSym ]
  ['\'VarSym ]
  ['\'ConstSym ]
```

下一页继续

```
['\'InTask ]
| ['\'InMod' :='<expression (IN) of string>]
['\'InRout ':='<expression (IN) of string>]
['\'GlobalSym ]
| ['\'LocalSym]' ;'
```

相关信息

信息，关于	请参阅
获取下一个匹配的符号	第*1095*页的*GetNextSym - 获取下一个匹配的符号*
获取数据对象的值	第*213*页的*GetDataVal - 获得数据对象的值*
设置许多数据对象的值	第*578*页的*SetAllDataVal - 在定义设置下，设置所有数据对象的值*
相关的数据类型datapos	第*1382*页的*datapos - 数据类型的封闭块*
Advanced RAPID	应用手册 - 控制器软件*IRC5*

1.224 SetDataVal - 设置数据对象的值

手册用法

SetDataVal（*Set Data Value*）使其有可能设置通过字符串变量而指定的数据对象的值。

基本示例

以下实例介绍了指令SetDataVal：

例 1

```
VAR num value:=3;
...
SetDataVal "reg"+ValToStr(ReadNum(mycom)),value;
```

这将通过从串行通道mycom收到的数字，设置寄存器的值3。

例 2

```
VAR datapos block;
VAR bool truevar:=TRUE;
...
SetDataSearch "bool" \Object:="my.*" \InMod:="mymod"\LocalSym;
WHILE GetNextSym(name,block) DO
  SetDataVal name\Block:=block,truevar;
ENDWHILE
```

该会话将模块mymod中以my开始的所有局部bool设置为TRUE。

例 3

```
VAR string StringArrVar_copy{2};
...
StringArrVar_copy{1} := "test1";
StringArrVar_copy{2} := "test2";
SetDataVal "StringArrVar", StringArrVar_copy;
```

该会话将设置数组StringArrVar ，以包含两个字符串test1 和test2.

变元

```
SetDataVal Object [\Block]|[\TaskRef]|[\TaskName] Value
```

Object

数据类型：string

数据对象的名称。

[\Block]

数据类型：datapos

数据对象的封闭块。仅可通过GetNextSym功能来取得该封闭块。

如果省略该参数，则将设置当前程序执行范围中可见数据对象的值。

[\TaskRef]

Task Reference

数据类型：taskid

下一页继续

用以搜索指定数据对象的程序任务识别号。使用此参数时，可搜索其他任务中的PERS或TASKPERS声明，任何其他声明均将引起错误。

对于系统中的所有程序任务，数据类型taskid的预定义变量将有效。可变识别号将为“任务名”+“Id”，例如，针对T_ROB1任务，可变识别号将为T_ROB1Id。

[\TaskName]

数据类型：string

用以搜索指定数据对象的程序任务名称。使用此参数时，可搜索其他任务中的PERS或TASKPERS声明，任何其他声明均将引起错误。

Value

数据类型：anytype

用以保存待设置新值的变量。数据类型必须与待设置数据对象的数据类型相同。必须从变量中取得设置值，但是，可将其储存在变量或永久数据对象中。

错误处理

如果出现以下情况，则将系统变量ERRNO设置为ERR_SYM_ACCESS：

- 不存在数据对象
- 数据对象为只读数据
- 数据对象是程序数据或程序参数，且不得位于当前有效程序中。
- 在其他任务中搜索其他声明，然后搜索PERS或TASKPERS

当使用参数TaskRef或TaskName时，可搜索其他任务中的PERS或TASKPERS声明，任何其他声明均将引起错误，且系统变量ERRNO得以设置为ERR_SYM_ACCESS。搜索在其他任务中被声明为LOCAL的PERS，亦将引起错误，且系统变量ERRNO得以设置为ERR_SYM_ACCESS。

如果在参数Value中使用的数据对象和变量具有不同维度，则将系统变量ERRNO设置为ERR_INVDIM。

如果在参数Value使用的数据对象和变量为不同的类型，则将系统变量ERRNO设置为ERR_SYMBOL_TYPE。如果使用ALIAS数据类型，则亦将出现该错误，尽管不同类型可能拥有相同的基础数据类型。

可以用程序错误处理器来处理错误。

限制

关于半值数据类型，不可能搜索相关的值数据类型。例如，如果搜索dionum，则将不会获得关于信号signaldi的搜索匹配，且如果搜索num，则将不会获得关于信号signalgi或signalai的搜索匹配。

不可能设置在RAPID模式下建立、声明为LOCAL的变量值。

语法

```
SetDataVal
  [ Object ':=' ] < expression (IN) of string >
  ['\'Block' :='<variable (VAR) of datapos>]
  |[ '\'TaskRef' :=' <variable (VAR) of taskid>]
  |[ '\'TaskName' :=' <expression (IN) of string>] ',']
  [ Value ':=' ] <variable (VAR) of anytype>]';'
```

下一页继续

1.224 SetDataVal - 设置数据对象的值
RobotWare - OS
续前页

相关信息

信息，关于	请参阅
定义在搜索会话中设置的符号	第*582*页的*SetDataSearch - 定义在搜索序列中设置的符号*
获取下一个匹配的符号	第*1095*页的*GetNextSym - 获取下一个匹配的符号*
获取数据对象的值	第*213*页的*GetDataVal - 获得数据对象的值*
设置许多数据对象的值	第*578*页的*SetAllDataVal - 在定义设置下，设置所有数据对象的值*
相关的数据类型`datapos`	第*1382*页的*datapos - 数据类型的封闭块*
Advanced RAPID	应用手册 - 控制器软件*IRC5*

1.225 SetDO - 改变数字信号输出信号值

手册用法

无论是否存在时间延迟或同步，SetDO用于改变数字信号输出信号的值。

基本示例

以下实例介绍了指令SetDO：

例 1

```
SetDO do15, 1;
```

将信号do15设置为1.

例 2

```
SetDO weld, off;
```

将信号weld设置为off.

例 3

```
SetDO \SDelay := 0.2, weld, high;
```

将信号weld设置为high ，且时间延迟为0.2 s。通过下一指令，继续程序执行。

例 4

```
SetDO \Sync ,do1, 0;
```

将信号do1设置为0。程序执行进入等待，直至从物理上将信号设置为指定值。

变元

```
SetDO [ \SDelay ]|[ \Sync ] Signal Value
```

[\SDelay]

Signal Delay

数据类型：num

延迟时间改变（以秒计，最多2000s）。通过下一指令，直接继续程序执行。在给定的时间延迟之后，改变信号，且随后程序执行不受影响。

[\Sync]

Synchronization

数据类型：switch

如果使用该参数，则程序执行将进入等待，直至从物理上将信号设置为指定值。

Signal

数据类型：signaldo

待改变信号的名称。

Value

数据类型：dionum

信号的期望值，0或1。

指定值	将数字信号输出设置为
0	0
除0以外的任意值	1

下一页继续

1.225 SetDO - 改变数字信号输出信号值
RobotWare - OS
续前页

程序执行

真实值取决于信号的配置。如果在系统参数中反转信号，则物理通道的值相反。

如果未使用参数\SDelay或\Sync，则将尽快地设置信号，并将立即执行下一指令，无需等待从物理上设置信号。

限制

如果SetDO以及\SDelay参数后跟随位于相同信号上的新SetDO，无论是否存在\SDelay参数，当在第一个SetDO延迟时间到期之前，如果执行第二个SetDO，则将取消第一个SetDO。

错误处理

系统会生成下列可恢复错误，并在错误处理器中处理这些错误。系统变量ERRNO将被设置成：

如果SDelay参数的值超出最大容许值（2000 s），则ERR_ARGVALERR。

如果信号变量是RAPID中声明的变量，则ERR_NO_ALIASIO_DEF。尚未同I/O配置以及指令AliasIO中确定的I/O信号相连。

如果与I/O单元无接触，则ERR_NORUNUNIT。

如果无法访问I/O信号（仅对ICI现场总线有效），则ERR_SIG_NOT_VALID。

语法

```
SetDO
  [ '\' SDelay ':=' < expression (IN) of num > ',' ]
  |['\'Sync',']
  [ Signal ':=' ] < variable (VAR) of signaldo > ','
  [ Value ':=' ] < expression (IN) of dionum > ';'
```

相关信息

信息，关于	请参阅
输入/输出指令	技术参考手册 - *RAPID*语言概览
输入/输出功能性概述	技术参考手册 - *RAPID*语言概览
I/O配置	技术参考手册 - 系统参数

1.226 SetGO - 改变一组数字信号输出信号的值

手册用法

无论是否存在时间延迟，SetGO用于改变一组数字信号输出信号的值。

基本示例

以下实例介绍了指令SetGO：

例 1

```
SetGO go2, 12;
```

将信号go2设置为12。如果go2包含4个信号，例如，输出6-9，则将输出6和7设置为零，并将输出8和9设置为一。

例 2

```
SetGO \SDelay := 0.4, go2, 10;
```

将信号go2设置为10。如果go2包含4个信号，例如，输出6-9，则将输出6和8设置为零，并将输出7和9设置为一，且延迟0.4 s。通过下一指令，继续程序执行。

例 3

SetGO go32, 4294967295;

将信号go32设置为4294967295。go32包含32个信号，且均设置为一。

变元

```
SetGO [ \SDelay ] Signal Value | Dvalue
```

[\SDelay]

Signal Delay

数据类型：num

延迟时间改变（以秒计，最多2000s）。通过下一指令，直接继续程序执行。在指定的时间延迟之后，改变信号，且随后程序执行不受影响。

如果省略参数，则直接改变信号值。

Signal

数据类型：signalgo

待改变信号组的名称。

Value

数据类型：num

下表显示了信号组的期望值（正整数）。

允许值取决于信号组中的信号数量。num数据类型可保存一组23个信号或更少信号的值。

Dvalue

数据类型：dnum

下表显示了信号组的期望值（正整数）。

下一页继续

1.226 SetGO - 改变一组数字信号输出信号的值
RobotWare - OS
续前页

允许值取决于信号组中的信号数量。dnum数据类型可保存一组32个信号或更少信号的值。

信号数量	允许值	允许D值
1	0-1	0-1
2	0-3	0-3
3	0-7	0-7
4	0-15	0-15
5	0-31	0-31
6	0-63	0-63
7	0-127	0-127
8	0-255	0-255
9	0-511	0-511
10	0-1023	0-1023
11	0-2047	0-2047
12	0-4095	0-4095
13	0-8191	0-8191
14	0-16383	0-16383
15	0-32767	0-32767
16	0-65535	0-65535
17	0-131071	0-131071
18	0-262143	0-262143
19	0-524287	0-524287
20	0-1048575	0-1048575
21	0-2097151	0-2097151
22	0-4194303	0-4194303
23	0-8388607	0-8388607
24	*	0-16777215
25	*	0-33554431
26	*	0-67108863
27	*	0-134217727
28	*	0-268435455
29	*	0-536870911
30	*	0-1073741823
31	*	0-2147483647
32	*	0-4294967295

*) num 类型的 Value 变元只能容纳最大 23 个信号，相对于 dnum 类型的 Dvalue 能容纳的 32 个信号要少多了。

下一页继续

程序执行

将编程值转换为未签名的二进制数字。在信号组上发送二进制数字，从而将信号组中的单个信号设置为0或1。因为在一段较短的时间内，可能未定义信号值的内部延迟。

限制

如果使用参数Value，则可用于信号组的最大信号数量为23，如果使用参数Dvalue，则可用于信号组的最大信号数量为32。该限制对于使用组信号的所有指令和功能有效。

错误处理

系统会生成下列可恢复错误，并在错误处理器中处理这些错误。系统变量ERRNO将被设置成：

如果SDelay参数的值超出最大容许值（2000 s），则ERR_ARGVALERR。

如果关于指定数字组输出信号Signal的已编程的Value或Dvalue参数超出限制，则ERR_GO_LIM。

如果信号变量是RAPID中声明的变量，则ERR_NO_ALIASIO_DEF。尚未同I/O配置以及指令AliasIO中确定的I/O信号相连。

如果与I/O单元无接触，则ERR_NORUNUNIT。

如果无法访问I/O信号（仅对ICI现场总线有效），则ERR_SIG_NOT_VALID。

语法

```
SetGO
  [ '\' SDelay ':=' < expression (IN) of num > ',' ]
  [ Signal ':=' ] < variable (VAR) of signalgo > ','
  [ Value ':=' ] < expression (IN) of num >
  | [ Dvalue' :=' ] < expression (IN) of dnum > ';'
```

相关信息

信息，关于	请参阅
其他输入/输出指令	技术参考手册 - *RAPID*语言概览
输入/输出功能性概述	技术参考手册 - *RAPID*语言概览
I/O（系统参数）的配置	技术参考手册 - 系统参数

1.227 SetSysData - 设置系统数据

手册用法

SetSysData激活指定数据类型的指定系统数据名称。

使用该指令，可能改变实际或相关运动任务中有关机械臂的当前有效工具、工件、有效负载或总负载。

基本示例

以下实例介绍了指令SetSysData：

例 1

```
SetSysData tool5;
```

启用工具tool5。

```
SetSysData tool0 \ObjectName := "tool6";
```

启用工具tool6。

```
SetSysData anytool \ObjectName := "tool2";
```

启用工具tool2。

变元

```
SetSysData SourceObject [\ObjectName]
```

SourceObject

数据类型：anytype

永久变量应当与当前系统数据一样有效。

该参数的数据类型同时规定了有关于实际或相关运动任务中的机械臂且有待启用的系统数据类型。

数据类型	系统数据的类型
tooldata	工具
wobjdata	工件坐标
loaddata	有效负载/总负载

无法使用整个数组或记录分量。

[\ObjectName]

数据类型：string

如果已指定该可选参数，则会规定待启用数据对象的名称（在参数SourceObject中指定的覆盖名称）。始终从参数SourceObject获取待启用数据对象的数据类型。

程序执行

根据参数，设置关于工具、工件、有效负载或总负载的当前有效系统数据对象。

注意，该指令仅启用新的数据对象（或与先前相同），且从未改变任意数据对象的值。

语法

```
SetSysData
[ SourceObject':='] < persistent(PERS) of anytype>
['\'ObjectName':=' < expression (IN) of string> ] ';'
```

下一页继续

相关信息

信息，关于	请参阅
工具的定义	第*1502*页的*tooldata - 工具数据*
工件的定义	第*1523*页的*wobjdata - 工件数据*
有效负载的定义	第*1421*页的*loaddata - 加载数据*
获取系统数据	第*216*页的*GetSysData - 获取系统数据*
关于如何使用TLoad总负载的例子。	第*388*页的*MoveL - 使机械臂沿直线移动*
激活和停用载荷的系统参数*ModalPayLoadMode*。 （主题 Controller，类型 System Misc，行动值，*ModalPayLoadMode*）	技术参考手册 - 系统参数
Advanced RAPID	应用手册 - 控制器软件*IRC5*

1.228 SetupCyclicBool - 设置进行循环求值的逻辑条件

手册用法

将采用SetupCyclicBool来设置进行循环求值并被赋予永久布尔变量的逻辑条件。

基本示例

下列示例说明了指令SetupCyclicBool。

另请参阅第597页的更多示例

例 1

```
PERS bool cyclicflag1;

PROC main()
  SetupCyclicBool cyclicflag1, di1=1 AND do2=1;
  ...
```

设置逻辑条件di1=1 AND do2=1的循环求值，并将结果分配给永久布尔变量cyclicflag1。

变元

```
SetupCyclicBool Flag Cond
```

Flag

数据类型：bool

保存逻辑条件值的永久布尔变量。

变量必须被声明为

Cond

数据类型：bool

应进行循环求值的逻辑表达式。

表达式可包括：

- bool、num和dnum类型的常量或永久变量（bool、num和dnum的别名）。
- 全局数字输入和输出信号
- 运算符：‘NOT’、‘AND’、‘OR’、‘XOR’、‘=’、‘(’、‘)’

程序执行

凭借该指令，可以设置更复杂的条件，取而代之，将采用循环标记来看是否满足条件。

每隔12ms进行逻辑条件循环求值和永久布尔变量赋值。

限制

表达式的求值结果必须为布尔值TRUE或FALSE。表达式的所有部分的求值结果也必须为布尔值TRUE或FALSE。

下一页继续

错误处理

系统会生成下列可恢复错误，并在错误处理器中处理这些错误。系统变量ERRNO将被设置成：

ERR_NO_ALIASIO_DEF	该信号变量是在RAPID中声明的一个变量，与用指令AliasIO在I / O配置中定义的I / O信号无关。
ERR_NORUNUNIT	与I / O单元间没有接触。
ERR_SIG_NOT_VALID	无法访问该I / O信号（仅对ICI现场总线有效）。

更多示例

下文给出了更多指令SetupCyclicBool的示例。

例 1

```
ALIAS bool aliasBool;
PERS bool cyclicflag1;
TASK PERS aliasBool cyclicflag2:=FALSE;
PERS aliasBool flag1:=FALSE;
TASK PERS aliasBool flag2:=FALSE;
CONST num HIGH:=1;
CONST num LOW:=0;

PROC main()
  SetupCyclicBool cyclicflag1, (di1=HIGH AND di2=HIGH AND di3=LOW)
       OR flag1=TRUE;
  SetupCyclicBool cyclicflag2, di4=HIGH AND flag2=TRUE;
  ...
  WaitUntil cyclicflag1=TRUE;
  IF cyclicflag2 = TRUE THEN
    MoveL p1, v1000, z30, tool2;
  ELSE
    MoveL p2, v1000, z30, tool2;
  ENDIF
  ...
```

上述示例设置了2个表达式的循环求值。在设置cyclicflag1前，执行将处于等待状态。cyclicflag2决定了机器人的目标移动位置。

例 2

```
!This condition is wrong:
SetupCyclicBool m1, 5;

!This condition is correct:
SetupCyclicBool m1, myNum = 5;
```

由于值5并非布尔值，因此，第一个条件错误。由于可将比较求值为布尔条件，即，TRUE或FALSE，所以，第二个条件正确。

语法

```
SetupCyclicBool
  [ Flag ':=' ] <persistent (PERS) of bool> ','
  [ Cond ':=' ] <expression (IN) of bool> ';'
```

下一页继续

1.228 SetupCyclicBool - 设置进行循环求值的逻辑条件
续前页

相关信息

信息，关于	请参阅
撤销进行循环求值的逻辑条件	第*502*页的*RemoveCyclicBool* - 撤销进行循环求值的逻辑条件
撤除所有进行循环求值的逻辑条件	第*501*页的*RemoveAllCyclicBool* - 撤除所有进行循环求值的逻辑条件
进行循环求值的逻辑条件，*Cyclic bool*。	应用手册 - 控制器软件*IRC5*

1.229 SetupSuperv - 设置CAP信号监控条件

手册用法

SetupSuperv将用于设置受监控的I/O信号的条件。将在不同列表中采集这些条件：

- PRE
- PRE_START
- END_PRE
- START
- MAIN
- END_MAIN
- START_POST1
- POST1
- END_POST1
- START_POST2
- POST2
- END_POST2

关于监控列表方面的更多信息请参见应用手册 - *Continuous Application Platform*。

作为一个可选参数，可指定输出信号，如果给定条件失效，那么此输出信号将被设到高位。

基本示例

```
PROC main()
  InitSuperv;
  SetupSuperv diWR_EST, ACT, SUPERV_MAIN \ErrIndSig:= do_WR_Sup;
  SetupSuperv diGA_EST, ACT, SUPERV_MAIN;
  CapL p2, v100, cdata1, weavestart, weave, fine, tWeldGun;
ENDPROC
```

SetupSuperv将用于设置信号监控，如果信号*diWR_EST*在SUPERV_MAIN阶段失效，那么，数字输出信号*do_WR_Sup*将被设置到高位。

仅在监控数据变更的情况下才应执行SetupSuperv指令。如果监控数据从未改变，那么将监控数据放入从最开始的位置执行的模块中，会比较妥当。

变元

```
SetupSupervSignal Condition Listtype [\ErrIndSig]
```

Signal

数据类型：signaldi

受监控的数字信号。

Condition

数据类型：num

代表下列可用条件之一的名称：

ACT:	用于状态监控。监控期间的预计信号状态：有源。如果信号变为无源，那么监控将启动。

下一页继续

1.229 SetupSuperv - 设置CAP信号监控条件
Continuous Application Platform (CAP)
续前页

PAS:	用于状态监控。监控期间的预计信号状态：无源。如果信号变为有源，那么监控将启动。
POS_EDGE:	用于握手监控。监控结束时的预计信号状态：有源。如果信号在选定超时范围内未变为有源，那么监控将启动。
NEG_EDGE:	用于握手监控。监控结束时的预计信号状态：无源。如果信号在选定超时范围内未变为无源，那么监控将启动。

Listtype

数据类型：num

代表不同列表（比如，进程中的各阶段）的编号的名称：

- SUPERV_PRE
- SUPERV_PRE_START
- SUPERV_END_PRE
- SUPERV_START
- SUPERV_MAIN
- SUPERV_END_MAIN
- SUPERV_START_POST1
- SUPERV_POST1
- SUPERV_END_POST1
- SUPERV_START_POST2
- SUPERV_POST2
- SUPERV_END_POST2

[\ErrIndSig]

数据类型：signaldo

用于指示在发生失败的情况下哪个条件失效。当失败发生时，此信号的值设为1。这是一项可选参数。

程序执行

给定信号及其条件将加入选定列表中，如果信号失败，那么，CapL/CapC指令将报告在指定阶段期间发生监控错误以及哪个信号失败。

错误

CAP_SPV_LIM

超出最大监控设置数。

CAP_SPV_UNK_LST

监控列表未知。

限制

仅可以监控数字输入信号。

状态监控适用于整个CAP指令序列（参见应用手册 - *Continuous Application Platform* 中的监控和进程阶段章节）。

下一页继续

语法

```
SetupSuperv
  [Signal ':='] < variable (VAR) of signaldi > ','
  [Condition ':='] < variable (IN) of num > ','
  [Listtype ':='] < variable (IN) of num >
  [\ErrIndSig ':=' < variable (VAR) of signaldo >] ';'
```

相关信息

信息，关于	请参阅
Continuous Application Platform	应用手册 - *Continuous Application Platform*
InitSuperv指令	第*258*页的*InitSuperv - 重置CAP*的所有监控
RemoveSuperv指令	第*505*页的*RemoveSuperv - 撤除一个信号的条件*

1.230 SiConnect - 传感器接口连接

手册用法

SiConnect用于同外部设备建立连接。

基本示例

有关于指令SiConnect的基本例子阐述如下。

另请参阅第602页的更多示例

例 1

```
PERS sensor AnyDevice;
...
  SiConnect AnyDevice;
```

同称之为AnyDevice的设备建立连接。

变元

```
SiConnect Sensor [\NoStop]
```

Sensor

数据类型：sensor

关于外部待连接设备的描述符。本参数是一个永久变量，其名称必须与安装文件*Settings.xml*中客户端所指定的名称相同。

[\NoStop]

数据类型：switch

当探测到关于传感器的通信错误时，\NoStop将防止系统停止。如果机械臂移动不取决于传感器时，则可以使用上述指令。使用\NoStop时，系统将继续移动，即使失去与传感器的通信。

如果使用\NoStop，则通过使用IError或IPers，可能在TRAP程序中进行错误处理。

程序执行

加载当前传感器配置，并建立同外部设备的连接。

即使将程序指针设置为主要，传感器仍然保持连接。

更多示例

有关于如何使用指令SiConnect的更多例子阐述如下。

例 1

```
PERS sensor AnyDevice;
PERS robdata DataOut := [[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]];
PERS sensdata DataIn :=
     ["No",[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]];
VAR num SampleRate:=64;
...
! Setup Interface Procedure
PROC RRI_Open()
  SiConnect AnyDevice;
```

下一页继续

```
    ! Send and receive data cyclic with 64 ms rate
    SiGetCyclic AnyDevice, DataIn, SampleRate;
    SiSetCyclic AnyDevice, DataOut, SampleRate;
ENDPROC
```

调用程序RRI_Open时，首先打开同名为AnyDevice的设备的连接。随后，开始以速率SampleRate开始进行循环传输。

例 2

```
PERS sensor AnyDevice;
...
  SiConnect AnyDevice \NoStop;
  ! Send and receive data cyclic with 64 ms rate
  SiGetCyclic AnyDevice, DataIn, SampleRate;
  SiSetCyclic AnyDevice, DataOut, SampleRate;
...
TRAP sensorChange
  IF AnyDevice.state = STATE_ERROR THEN
  ...
  ENDIF
ENDTRAP
```

如果与AnyDevice的连接损坏，则通过可选参数\NoStop，与称之为AnyDevice的设备建立连接，由此防止系统停止。处理TRAP程序中的错误状态。

错误处理

如果将UDP用作通信协议，则不会保证连接成功，因此，在进行连接时，不可能进行错误处理。

如果将TCP用作通信协议，则将系统变量ERRNO设置为ERR_COMM_INIT。如果连接操作失败，则通过错误处理器来处理该错误。

成功连接后，开关\NoStop使其可能处理探测到的通信错误。\NoStop意味着继续移动和执行RAPID，且TRAP程序可用于处理使用IError的特殊错误或使用IPers的特殊状态变更。

注意

IPers和IError并非安全中断，因此，如果在停止后探测到错误，则将不会执行TRAP。处理此问题的一种方法是在重启架中进行SiConnect \NoStop，以确保应用尝试与客户端重新建立连接。

语法

```
SiConnect
  [ Sensor ':=' ] < persistent (PERS) of sensor >
  [ '\' NoStop ] ';'
```

相关信息

信息，关于	请参阅
关闭与外部系统的连接。	第*605*页的*SiClose - 传感器接口关闭*。
关于循环传输的登记数据。	第*610*页的*SiSetCyclic - 传感器接口设置循环*。

下一页继续

1.230 SiConnect - 传感器接口连接
Robot Reference Interface
续前页

信息，关于	请参阅
赞成循环数据传输。	第*606*页的*SiGetCyclic - 传感器接口获得循环*
外部设备的描述符。	第*1470*页的*sensor - 外部设备描述符*。
设备的通信状况。	第*1472*页的*sensorstate - 设备的通信状况*。
Robot Reference Interface	应用手册 - 控制器软件*IRC5*

1.231 SiClose - 传感器接口关闭

手册用法

SiClose关闭同外部设备的现有连接。

基本示例

有关于指令SiClose的基本例子阐述如下。

例 1

```
PERS sensor AnyDevice;
...
  SiClose AnyDevice;
```

关闭与称之为AnyDevice的设备的连接。

变元

```
SiClose Sensor
```

Sensor

数据类型：sensor

关于应当关闭的外部设备的描述符。本参数是一个永久变量，其名称必须与安装文件*Settings.xml*中客户端所指定的名称相同。

程序执行

关闭同外部设备的现有连接。

错误处理

如果将UDP用作通信协议，则不会保证关闭操作成功，因此，不可能进行错误处理。

如果将TCP用作通信协议，则将系统变量ERRNO设置为ERR_COMM_INIT。如果关闭操作失败，则通过错误处理器来处理该错误。

语法

```
SiClose
  [ Sensor ':=' ] < persistent (PERS) of sensor > ';'
```

相关信息

信息，关于	请参阅
建立与外部系统的连接。	第602页的*SiConnect - 传感器接口连接*。
关于循环传输的登记数据。	第610页的*SiSetCyclic - 传感器接口设置循环*。
赞成循环数据传输。	第606页的*SiGetCyclic - 传感器接口获得循环*。
外部设备的描述符。	第1470页的*sensor - 外部设备描述符*。
设备的通信状况。	第1472页的*sensorstate - 设备的通信状况*。
Robot Reference Interface	应用手册 - 控制器软件*IRC5*

1.232 SiGetCyclic - 传感器接口获得循环

手册用法

SiGetCyclic预定关于外部设备循环传输的数据。

基本示例

有关于指令SiGetCyclic的基本例子阐述如下。

另请参阅第*606*页的更多示例

例 1

```
SiConnect AnyDevice;
! Receive data cyclic with 64 ms rate
SiGetCyclic AnyDevice, DataIn, 64;
```

本例子表明如何同外部设备建立连接，并从设备AnyDevice建立循环传输。

变元

```
SiGetCyclic Sensor Data Rate
```

Sensor

数据类型：sensor

关于用以接收循环数据的外部设备的描述符。本参数是一个永久变量，其名称必须与安装文件*Settings.xml*中客户端所指定的名称相同。

Data

数据类型：anytype

包含从参数Sensor中指定客户端所收到数据的永久数据对象的引用。必须将变量定义为文件*Configuration.xml*中的*Readable*。

Rate

数据类型：num

传输率，以毫秒计（仅支持4ms的倍数）。

程序执行

指令SiGetCyclic预定关于外部设备循环传输的数据。

针对SiGetCyclic和SiSetCyclic指令，传输率为0，即停止（未记录/未预定）给定数据或数据集的循环传输。

更多示例

有关于如何使用指令SiGetCyclic的更多例子阐述如下。

例 1

```
PERS sensor AnyDevice;
PERS robdata DataOut := [[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]];
PERS sensdata DataIn :=
     ["No",[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]];
VAR num SampleRate:=64;
...
! Setup Interface Procedure
```

下一页继续

```
PROC RRI_Open()
  SiConnect AnyDevice;
  ! Send and receive data cyclic with 64 ms rate
  SiGetCyclic AnyDevice, DataIn, SampleRate;
  SiSetCyclic AnyDevice, DataOut, SampleRate;
ENDPROC
```

调用程序RRI_Open时，首先打开同名为AnyDevice的设备的连接。随后，开始以速率SampleRate开始进行循环传输。

语法

```
SiGetCyclic
  [ Sensor ':=' ] < persistent (PERS) of sensor > ','
  [ Data ':=' ] < persistent (PERS) of anytype > ','
  [ Rate ':=' ] < expression (IN) of num > ] ';'
```

相关信息

信息，关于	请参阅
建立与外部系统的连接。	第*602*页的*SiConnect - 传感器接口连接*。
关闭与外部系统的连接。	第*605*页的*SiClose - 传感器接口关闭*。
关于循环传输的登记数据。	第*610*页的*SiSetCyclic - 传感器接口设置循环*。
外部设备的描述符。	第*1470*页的*sensor - 外部设备描述符*。
设备的通信状况。	第*1472*页的*sensorstate - 设备的通信状况*。
Robot Reference Interface	应用手册 - 控制器软件*IRC5*

1.233 SingArea - 确定奇点周围的插补

手册用法

SingArea用于定义机械臂如何在奇异点附近移动。

SingArea亦用于定义关于拥有不到六个轴的机械臂的线性和圆周插补，在轴4锁定为0或+-180度的情况下，可编程六轴机械臂运行。

本指令仅可用于主任务T_ROB1，或者如果在*MultiMove*系统中，则可用于运动任务中。

基本示例

以下实例介绍了指令SingArea：

例 1

```
SingArea \Wrist;
```

可略微改变工具方位，以通过奇异点（生产线中的轴4和轴6）。

拥有不到六个轴的机械臂，可能无法达到插补的工具方位。通过使用SingArea \Wrist，机械臂可实现移动，但是工具方位将会略微改变。

例 2

```
SingArea \LockAxis4;
```

通过将轴4锁定在0或+-180度，可编程六轴机械臂运行，从而避免在轴5接近于零时的奇异点问题。

通过将轴4锁定在0或+-180度，可达到编程位置。如果当轴4位于0或+-180度时，未编程位置，则当前将获得不同的工具方位。

如果轴4的起始位置偏离锁定位置2度以上，则第一次移动将表现为通过参数\Wrist来调用SingArea一般。

在所有后续移动中，轴4将保持锁定，直至执行新的SingArea指令。

例 3

```
SingArea \Off;
```

不允许工具方位偏离编程方位。如果通过奇异点，则一个或多个轴可实施彻底的移动，从而导致速率降低。

拥有不到六个轴的机械臂可能无法达到编程的工具方位。因此，机械臂将停止。

变元

```
SingArea [\Wrist]|[\LockAxis4]|[\Off]
```

【 \Wrist 】

数据类型：switch

允许工具方位稍微偏离，以避免腕奇异点。其适用于轴4和轴6平行的情况（轴5为0度）。同时适用于拥有不到六个轴的机械臂的线性和圆周插补，其允许工具方位出现偏离。

【 \LockAxis4 】

数据类型：switch

通过将轴4锁定在0或+-180度，可达到编程位置。如果当轴4位于0或+-180度时，未编程位置，则当前将获得不同的工具方位。

下一页继续

如果轴4的起始位置偏离锁定位置2度以上，则第一次移动将表现为通过参数\Wrist来调用SingArea一般。

【 \Off 】

数据类型：switch

不允许工具方位出现偏离。当未通过奇异点，或不允许方位发生改变时，上述要求适用。

如果未指定任何参数，则将系统设置为\Off。

程序执行

如果已指定参数\Wrist，则对方位进行接头插补，以避免奇异点。在这种情况下，TCP遵循正确的路径，但是工具方位会稍微偏离。当未通过奇异点时，亦将出现上述情况。

如果指定参数\LockAxis4，则将轴4锁定为0或+-180度，以避免奇异点。如果当轴4位于0或+-180度时，未对位置进行编程，则TCP将遵循正确的路径，但是工具方位将会偏离。

指定的插补适用于所有随后的移动，直至执行新的SingArea指令。

仅当执行线性或圆周插补时，移动会受到影响。

针对拥有六个轴的机械臂，默认使用Off参数来自动进行程序执行。拥有不到六个轴的机械臂默认使用Off参数或/Wrist参数。在事件程序SYS_RESET中进行自动设置。

自动设置默认值

- 当使用重启模式重置RAPID时
- 当加载一则新程序或一个新模块时
- 当从起点开始执行程序时
- 当将程序指针移动到main时
- 当将程序指针移动到子程序时
- 移动程序指针造成执行顺序丢失时

语法

```
SingArea
  [ '\' Wrist ] | ['\' LockAxis4 ] | ['\' Off ] ';'
```

相关信息

信息，关于	请参阅
奇异点	技术参考手册 - *RAPID*语言概览
插补	技术参考手册 - *RAPID*语言概览
运动设置数据	第*1430*页的*motsetdata - 运动设置数据*

1.234 SiSetCyclic - 传感器接口设置循环

手册用法

SiSetCyclic登记关于外部设备循环传输的数据。

基本示例

有关于指令SiSetCyclic的基本例子阐述如下。

另请参阅第*610*页的更多示例

例 1

```
PERS sensor AnyDevice;
PERS robdata DataOut := [[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]];
...
  SiConnect AnyDevice;
  SiSetCyclic AnyDevice, DataOut, 40;
```

同称之为AnyDevice的设备建立连接。随后，每隔40 ms，登记关于外部设备AnyDevice循环传输的数据。

变元

```
SiSetCyclic Sensor Data Rate
```

Sensor

数据类型：sensor

关于用以发送数据的外部设备的描述符。

Data

数据类型：anytype

包含发送至参数Sensor所指定客户端数据的任意复杂或获得支持的简单型永久数据对象的引用。必须将变量定义为*Configuration.xml*文件中的*Writable*。

Rate

数据类型：num

传输率，以毫秒计（仅支持4 ms的倍数）。

程序执行

指令SiSetCyclic登记关于外部设备循环传输的数据。

针对SiGetCyclic和SiSetCyclic指令，传输率为0，即停止（未记录/未预定）给定数据或数据集的循环传输。

更多示例

有关于如何使用指令SiSetCyclic的更多例子阐述如下。

例 1

```
PERS sensor AnyDevice;
PERS robdata DataOut := [[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]];
PERS sensdata DataIn :=
     ["No",[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]];
VAR num SampleRate:=64;
...
```

下一页继续

```
! Setup Interface Procedure
PROC RRI_Open()
  SiConnect AnyDevice;
  ! Send and receive data cyclic with 64 ms rate
  SiGetCyclic AnyDevice, DataIn, SampleRate;
  SiSetCyclic AnyDevice, DataOut, SampleRate;
ENDPROC
```

调用程序RRI_Open时，首先打开同名为AnyDevice的设备的连接。随后，开始以速率SampleRate开始进行循环传输。

语法

```
SiSetCyclic
  [ Sensor ':=' ] < persistent (PERS) of sensor > ','
  [ Data ':=' ] < persistent (PERS) of anytype >
  [ Rate ':=' ] < expression (IN) of num > ] ';'
```

相关信息

信息，关于	请参阅
建立与外部系统的连接。	第*602*页的*SiConnect - 传感器接口连接*。
关闭与外部系统的连接。	第*605*页的*SiClose - 传感器接口关闭*。
赞成循环数据传输。	第*606*页的*SiGetCyclic - 传感器接口获得循环*。
外部设备的描述符。	第*1470*页的*sensor - 外部设备描述符*。
设备的通信状况。	第*1472*页的*sensorstate - 设备的通信状况*。
Robot Reference Interface	应用手册 - 控制器软件*IRC5*

1.235 SkipWarn - 跳过最近的警告

手册用法

SkipWarn（*Skip Warning*）用于跳过在采用连续或圆周运行模式执行期间，储存在事件日志中的最新产生的警告消息（未跳过FWD或BWD步骤中的任何警告）。

通过SkipWarn，在没有仅采用警告消息来填充事件日志的情况下，可能在RAPID中反复地进行错误恢复。

基本示例

以下实例介绍了指令SkipWarn：

例 1

```
    %"notexistingproc"%;
    nextinstruction;
ERROR
    IF ERRNO = ERR_REFUNKPRC THEN
        SkipWarn;
        TRYNEXT;
    ENDIF
ENDPROC
```

本程序将执行nextinstruction，且未将警告消息储存在事件日志中。

语法

```
SkipWarn ';'
```

相关信息

信息，关于	请参阅
错误恢复	技术参考手册 - *RAPID*语言概览 技术参考手册 - *RAPID*语言概览
错误编号	第*1395*页的*errnum* - 错误编号

1.236 SocketAccept - 接受输入连接

手册用法

SocketAccept用于接受输入连接请求。SocketAccept仅可用于服务器应用。

基本示例

以下实例介绍了指令SocketAccept：

另请参阅第*614*页的更多示例

例 1

```
VAR socketdev server_socket;
VAR socketdev client_socket;
...
SocketCreate server_socket;
SocketBind server_socket,"192.168.0.1", 1025;
SocketListen server_socket;
SocketAccept server_socket, client_socket;
```

创建服务器套接字，并绑定至地址为192.168.0.1的控制器网络上的端口1025。在执行SocketListen之后，服务器套接字开始监听位于该端口和地址上输入连接。SocketAccept等待所有输入连接，接受连接请求，并返回已建立连接的客户端套接字。

变元

```
SocketAccept Socket ClientSocket [\ClientAddress] [ \Time ]
```

Socket

数据类型：socketdev

正在等待输入连接的服务器套接字。套接字必须已经创建、绑定并准备进行监听。

ClientSocket

数据类型：socketdev

将按照接受的输入连接请求，对返回的新客户端套接字进行更新。

[\ClientAddress]

数据类型：string

将通过已接受输入连接请求的IP地址来进行更新的变量。

[\Time]

数据类型：num

程序执行等待输入连接的最长时间量。如果在任意输入连接之前耗尽时间，则将调用错误处理器，如果存在这样的情况，则采用错误代码ERR_SOCK_TIMEOUT。如果不存在错误处理器，则将停止执行。

如果未使用参数\Time，则等待时间为60秒。为了永久等待，则使用预定义常量WAIT_MAX。

程序执行

服务器套接字将等待所有输入连接请求。当接受输入连接请求时，本指令就绪，且返回的客户端套接字默认连接，并可用于SocketSend和SocketReceive指令。

下一页继续

1.236 SocketAccept - 接受输入连接
Socket Messaging
续前页

更多示例

有关于指令SocketAccept的更多例子阐述如下。

例 1

```
VAR socketdev server_socket;
VAR socketdev client_socket;
VAR string receive_string;
VAR string client_ip;
...
SocketCreate server_socket;
SocketBind server_socket, "192.168.0.1", 1025;
SocketListen server_socket;
WHILE TRUE DO
  SocketAccept server_socket, client_socket
       \ClientAddress:=client_ip;
  SocketReceive client_socket \Str := receive_string;
  SocketSend client_socket \Str := "Hello client with ip-address
       " +client_ip;
  ! Wait for client acknowledge
  ...
  SocketClose client_socket;
ENDWHILE
ERROR
  RETRY;
UNDO
  SocketClose server_socket;
  SocketClose client_socket;
```

创建服务器套接字，并绑定至地址为192.168.0.1的控制器网络上的端口1025。在执行SocketListen后，服务器套接字开始监听位于该端口和地址上的输入连接。SocketAccept将接受来自一些客户端的输入连接，并将客户端地址储存在字符串client_ip中。随后，服务器接收来自客户端的字符串消息，并将消息储存在receive_string。然后，服务器对消息" Hello client with ip-address xxx.xxx.x.x"进行响应，并关闭客户端连接。

此后，服务器准备好同WHILE回路中的相同或其他客户端进行连接。如果PP移动至程序中的main，则关闭所有打开的套接字（始终可以进行SocketClose，即使未创建套接字）。

错误处理

产生了下列可恢复错误，并可以用错误处理器进行处理。将系统变量ERRNO设置为：

ERR_SOCK_CLOSED	关闭套接字（已经关闭或未创建）。使用 SocketCreate来创建新的套接字。
ERR_SOCK_TIMEOUT	不会在超时时间内建立连接。

语法

```
SocketAccept
  [ Socket ':=' ] < variable (VAR) of socketdev > ','
  [ ClientSocket ':=' ] < variable (VAR) of socketdev >
  [ '\' ClientAddress ':=' < variable (VAR) of string> ]
  [ '\' Time ':=' < expression (IN) of num > ] ';'
```

下一页继续

相关信息

信息，关于	请参阅
一般的套接字通信	应用手册 - 控制器软件*IRC5*，套接字消息传送一节
创建新套接字	第*623*页的*SocketCreate* - 创建新套接字
连接远程计算机（仅限客户端）	第*620*页的*SocketConnect* - 连接远程计算机
向远程计算机发送数据	第*635*页的*SocketSend* - 向远程计算机发送数据
接收来自远程计算机的数据	第*627*页的*SocketReceive* - 接收来自远程计算机的数据
关闭套接字	第*618*页的*SocketClose* - 关闭套接字
绑定套接字（仅限服务器）	第*616*页的*SocketBind* - 将套接字与我的*IP*地址和端口绑定
监听连接（仅限服务器）	第*625*页的*SocketListen* - 监听输入连接
获得当前套接字的状态	第*1230*页的*SocketGetStatus* - 获得当前套接字的状态
客户端套接字应用实例	第*635*页的*SocketSend* - 向远程计算机发送数据
服务器套接字应用实例	第*627*页的*SocketReceive* - 接收来自远程计算机的数据

1.237 SocketBind - 将套接字与我的IP地址和端口绑定

手册用法

SocketBind用于将套接字与指定服务器IP地址和端口号绑定。SocketBind仅可用于服务器应用。

基本示例

以下实例介绍了指令SocketBind：

例 1

```
VAR socketdev server_socket;

SocketCreate server_socket;
SocketBind server_socket, "192.168.0.1", 1025;
```

创建服务器套接字，并与地址为192.168.0.1的控制器网络上的端口1025绑定。现在，可在SocketListen指令中使用服务器套接字，以监听位于该端口和地址上的输入连接。

变元

```
SocketBind Socket LocalAddress LocalPort
```

Socket

数据类型：socketdev

有待绑定的服务器套接字。必须创建尚未绑定的套接字。

LocalAddress

数据类型：string

将服务器网络地址与套接字绑定。唯一有效的地址为所有公共WAN地址，或控制器服务端口地址192.168.125.1。

LocalPort

数据类型：num

将服务器端口号与套接字绑定。通常，可自由使用端口1025-4999。

程序执行

将服务器套接字与指定服务器端口和IP地址绑定。

如果已在使用指定端口，则会产生错误。

在启动程序的过程中，使用SocketBind和SocketListen指令，从而使局部地址与套接字相关联，随后，监听指定端口上的输入连接。针对用到的各套接字和端口（TCP/IP），仅建议实施上述操作一次。

如果通过SocketReceiveFrom（UDP/IP）来接收数据，则使用SocketBind指令。

错误处理

系统会生成下列可恢复错误，并在错误处理器中处理这些错误。系统变量ERRNO将被设置成：

ERR_SOCK_CLOSED	关闭套接字（已经关闭或未创建） 将SocketCreate用于创建新的套接字。

下一页继续

ERR_SOCK_ADDR_INUSE	地址和端口已在使用中，且无法再次使用。使用不同的端口号。

语法

```
SocketBind
  [ Socket ':=' ] < variable (VAR) of socketdev > ','
  [ LocalAddress ':=' ] < expression (IN) of string > ','
  [ LocalPort ':=' ] < expression (IN) of num > ';'
```

相关信息

信息，关于	请参阅
一般的套接字通信	应用手册 - 控制器软件*IRC5*
创建新套接字	第*623*页的*SocketCreate* - 创建新套接字
连接远程计算机（仅限客户端）	第*620*页的*SocketConnect* - 连接远程计算机
向远程计算机发送数据	第*635*页的*SocketSend* - 向远程计算机发送数据
接收来自远程计算机的数据	第*627*页的*SocketReceive* - 接收来自远程计算机的数据
关闭套接字	第*618*页的*SocketClose* - 关闭套接字
监听连接（仅限服务器）	第*625*页的*SocketListen* - 监听输入连接
接受连接（仅服务器）	第*613*页的*SocketAccept* - 接受输入连接
获得当前套接字的状态	第*1230*页的*SocketGetStatus* - 获得当前套接字的状态
客户端套接字应用实例	第*635*页的*SocketSend* - 向远程计算机发送数据
服务器套接字应用实例	第*627*页的*SocketReceive* - 接收来自远程计算机的数据
接收来自远程计算机的数据	第*631*页的*SocketReceiveFrom* - 接收来自远程计算机的数据

1.238 SocketClose - 关闭套接字

手册用法

当不再使用套接字连接时，使用SocketClose。

在已经关闭套接字之后，不能将其用于除SocketCreate以外的所有套接字调用。

基本示例

以下实例介绍了指令SocketClose：

例 1

```
SocketClose socket1;
```

关闭套接字，且不能再进行使用。

变元

```
SocketClose Socket
```

Socket

数据类型：socketdev

有待关闭的套接字。

程序执行

将关闭套接字，并将释放其配置资源。

可在任意时间关闭任意套接字。关闭后，不能再使用套接字。在调用SocketCreate之后，可将其再次用于新的连接。

限制

在通过SocketSend发送数据后，随即关闭套接字连接，这会导致失去发送的数据。因为TCP/IP套接字具有在出现通信问题时重新发送数据的内置功能。

为避免有关数据丢失的此类问题，应在SocketClose之前进行以下操作：

- 交换关闭信号，或
- WaitTime 2

通过SocketCreate ... SocketClose，避免快回路，因为在某一特定时间之前，套接字并未真正关闭（TCP/IP功能）。

语法

```
SocketClose
  [ Socket ':=' ] < variable (VAR) of socketdev > ';'
```

相关信息

信息，关于	请参阅
一般的套接字通信	应用手册 - 控制器软件*IRC5*，套接字消息传送一节
创建新套接字	第*623*页的*SocketCreate - 创建新套接字*
连接远程计算机（仅客户端）	第*620*页的*SocketConnect - 连接远程计算机*
向远程计算机发送数据	第*635*页的*SocketSend - 向远程计算机发送数据*

下一页继续

信息，关于	请参阅
接收来自远程计算机的数据	第*627*页的*SocketReceive* - 接收来自远程计算机的数据
绑定套接字（仅限服务器）	第*616*页的*SocketBind* - 将套接字与我的*IP*地址和端口绑定
监听连接（仅限服务器）	第*625*页的*SocketListen* - 监听输入连接
接受连接（仅服务器）	第*613*页的*SocketAccept* - 接受输入连接t
获得当前套接字的状态	第*1230*页的*SocketGetStatus* - 获得当前套接字的状态
客户端套接字应用实例	第*635*页的*SocketSend* - 向远程计算机发送数据
向远程计算机发送数据	第*639*页的*SocketSendTo* - 向远程计算机发送数据
服务器套接字应用实例	第*627*页的*SocketReceive* - 接收来自远程计算机的数据
接收来自远程计算机的数据	第*631*页的*SocketReceiveFrom* - 接收来自远程计算机的数据

1.239 SocketConnect - 连接远程计算机

手册用法

SocketConnect用于将套接字与客户端应用中的远程计算机相连。

基本示例

以下实例介绍了指令SocketConnect：

另请参阅第620页的更多示例

例 1

```
SocketConnect socket1, "192.168.0.1", 1025;
```

尝试与ip地址192.168.0.1和端口1025处的远程计算机相连。

变元

```
SocketConnect Socket Address Port [\Time]
```

Socket

数据类型：socketdev

有待连接的服务器套接字。必须创建尚未连接的套接字。

Address

数据类型：string

远程计算机的地址。必须将远程计算机指定为一个IP地址。不可能使用远程计算机的名称。

Port

数据类型：num

位于远程计算机上的端口。通常，可自由使用端口1025-4999。可能已经采用了低于1025的端口。

[\Time]

数据类型：num

程序执行等待接受或否定连接的最长时间量。如果在满足条件之前耗尽时间，则将调用错误处理器，如果存在这样的情况，则采用错误代码ERR_SOCK_TIMEOUT。如果不存在错误处理器，则将停止执行。

如果未使用参数\Time，则等待时间为60秒。为了永久等待，则使用预定义常量WAIT_MAX。

程序执行

套接字试图连接指定地址和端口上的远程计算机。程序执行将等待，直至连接得以建立、失效或出现超时。

更多示例

有关于指令SocketConnect的更多例子阐述如下。

例 1

```
VAR num retry_no := 0;
VAR socketdev my_socket;
```

下一页继续

```
...
SocketCreate my_socket;
SocketConnect my_socket, "192.168.0.1", 1025;
...
ERROR
  IF ERRNO = ERR_SOCK_TIMEOUT THEN
    IF retry_no < 5 THEN
      WaitTime 1;
      retry_no := retry_no + 1;
      RETRY;
    ELSE
      RAISE;
    ENDIF
  ENDIF
```

创建套接字，并尝试连接远程计算机。如果未在默认超时时间（即60秒）内未建立连接，则错误处理器重新尝试连接。试图重新尝试四次，随后，将错误报告给用户。

错误处理

系统会生成下列可恢复错误，并在错误处理器中处理这些错误。系统变量ERRNO将被设置成：

ERR_SOCK_CLOSED	关闭套接字（已经关闭或未创建）。 将SocketCreate用于创建新的套接字。
ERR_SOCK_TIMEOUT	不会在超时时间内建立连接。

语法

```
SocketConnect
  [ Socket ':=' ] < variable (VAR) of socketdev > ','
  [ Address ':=' ] < expression (IN) of string > ','
  [ Port ':=' ] < expression (IN) of num >
  [ '\' Time ':=' < expression (IN) of num > ] ';'
```

相关信息

信息，关于	参见：
一般的套接字通信	应用手册 - 控制器软件*IRC5*
创建新套接字	第*623*页的*SocketCreate - 创建新套接字*
向远程计算机发送数据	第*635*页的*SocketSend - 向远程计算机发送数据*
接收来自远程计算机的数据	第*627*页的*SocketReceive - 接收来自远程计算机的数据*
绑定套接字（仅限服务器）	第*616*页的*SocketBind - 将套接字与我的IP地址和端口绑定*
监听连接（仅限服务器）	第*625*页的*SocketListen - 监听输入连接*
接受连接（仅服务器）	第*613*页的*SocketAccept - 接受输入连接*
获得当前套接字的状态	第*1230*页的*SocketGetStatus - 获得当前套接字的状态*
客户端套接字应用实例	第*635*页的*SocketSend - 向远程计算机发送数据*

下一页继续

1.239 SocketConnect - 连接远程计算机

Socket Messaging

续前页

信息，关于	参见：
服务器套接字应用实例	第627页的*SocketReceive* - 接收来自远程计算机的数据

1.240 SocketCreate - 创建新套接字

手册用法

SocketCreate用于针对基于通信或非连接通信的连接，创建新的套接字。

带有交付保证的流型协议TCP/IP以及数据电报协议UDP/IP的套接字消息传送均得到支持。可开发服务器和客户端应用。针对数据电报协议UDP/IP，支持采用广播。

基本示例

以下实例介绍了指令SocketCreate：

例 1

```
VAR socketdev socket1;
...
SocketCreate socket1;
```

创建使用流型协议TCP/IP的新套接字设备，并分配到变量socket1。

例 2

```
VAR socketdev udp_sock1;
...
SocketCreate udp_sock1 \UDP;
```

创建使用数据电报协议UDP/IP的新套接字设备，并分配到变量udp_sock1。

变元

```
SocketCreate Socket [\UDP]
```

Socket

数据类型：socketdev

用于储存系统内部套接字数据的变量。

[\UDP]

数据类型：switch

指定套接字应当为数据电报型协议UDP/IP。

程序执行

本指令创建新的套接字设备。

套接字不得业已使用。套接字在SocketCreate和SocketClose之间使用。

限制

可声明任意数量的套接字，但是仅可能同时使用32个套接字。

通过SocketCreate ... SocketClose，避免快回路，因为在某一特定时间之前，套接字并未真正关闭（当使用TCP/IP功能时）。

语法

```
SocketCreate
  [ Socket ':=' ] < variable (VAR) of socketdev >
  [ '\' UDP ] ';'
```

下一页继续

1.240 SocketCreate - 创建新套接字
Socket Messaging
续前页

相关信息

信息，关于	请参阅
一般的套接字通信	应用手册 - 控制器软件*IRC5*，套接字消息传送一节
连接远程计算机（仅限客户端）	第*620*页的*SocketConnect* - 连接远程计算机
向远程计算机发送数据	第*635*页的*SocketSend* - 向远程计算机发送数据
接收来自远程计算机的数据	第*627*页的*SocketReceive* - 接收来自远程计算机的数据
关闭套接字	第*618*页的*SocketClose* - 关闭套接字
绑定套接字（仅限服务器）	第*616*页的*SocketBind* - 将套接字与我的*IP*地址和端口绑定
监听连接（仅限服务器）	第*625*页的*SocketListen* - 监听输入连接
接受连接（仅服务器）	第*613*页的*SocketAccept* - 接受输入连接
获得当前套接字的状态	第*1230*页的*SocketGetStatus* - 获得当前套接字的状态
客户端套接字应用实例	第*635*页的*SocketSend* - 向远程计算机发送数据
向远程计算机发送数据	第*639*页的*SocketSendTo* - 向远程计算机发送数据
服务器套接字应用实例	第*627*页的*SocketReceive* - 接收来自远程计算机的数据
接收来自远程计算机的数据	第*631*页的*SocketReceiveFrom* - 接收来自远程计算机的数据

1.241 SocketListen - 监听输入连接

手册用法

SocketListen用于开始监听输入连接，即开始充当服务器。SocketListen仅可用于服务器应用。

基本示例

以下实例介绍了指令SocketListen：

例 1

```
VAR socketdev server_socket;
VAR socketdev client_socket;
...
SocketCreate server_socket;
SocketBind server_socket, "192.168.0.1", 1025;
SocketListen server_socket;
WHILE listening DO;
  ! Waiting for a connection request
  SocketAccept server_socket, client_socket;
```

创建服务器套接字，并与地址为192.168.0.1的控制器网络上的端口1025绑定。在执行SocketListen后，服务器套接字开始监听位于该端口和地址上的输入连接。

变元

```
SocketListen Socket
```

Socket

数据类型：socketdev

应当开始监听输入连接的服务器套接字。必须已经创建和绑定套接字。

程序执行

服务器套接字开始监听输入连接。当指令就绪时，套接字便准备好接受输入连接。

在启动程序的过程中，使用SocketBind和SocketListen指令，从而使局部地址与套接字相关联，随后，监听指定端口上的输入连接。针对用到的各套接字和端口，仅建议实施上述操作一次。

错误处理

系统会生成下列可恢复错误，并在错误处理器中处理这些错误。系统变量ERRNO将被设置成：

ERR_SOCK_CLOSED	关闭套接字（已经关闭或未创建）。 将SocketCreate用于创建新的套接字。

语法

```
SocketListen
  [ Socket ':=' ] < variable (VAR) of socketdev > ';'
```

下一页继续

1.241 SocketListen - 监听输入连接

Socket Messaging

续前页

相关信息

信息，关于	请参阅
一般的套接字通信	应用手册 - 控制器软件*IRC5*
创建新套接字	第*623*页的*SocketCreate* - 创建新套接字
连接远程计算机（仅限客户端）	第*620*页的*SocketConnect* - 连接远程计算机
向远程计算机发送数据	第*635*页的*SocketSend* - 向远程计算机发送数据
接收来自远程计算机的数据	第*627*页的*SocketReceive* - 接收来自远程计算机的数据
关闭套接字	第*618*页的*SocketClose* - 关闭套接字
绑定套接字（仅限服务器）	第*616*页的*SocketBind* - 将套接字与我的*IP*地址和端口绑定
接受连接（仅服务器）	第*613*页的*SocketAccept* - 接受输入连接
获得当前套接字的状态	第*1230*页的*SocketGetStatus* - 获得当前套接字的状态
客户端套接字应用实例	第*635*页的*SocketSend* - 向远程计算机发送数据
服务器套接字应用实例	第*627*页的*SocketReceive* - 接收来自远程计算机的数据

1.242 SocketReceive - 接收来自远程计算机的数据

手册用法

SocketReceive用于从远程计算机接收数据。SocketReceive可用于客户端和服务器应用。

基本示例

以下实例介绍了指令SocketReceive：

另请参阅第*628*页的更多示例

例 1

```
VAR string str_data;
...
SocketReceive socket1 \Str := str_data;
```

从远程计算机接收数据，并将其储存在字符串变量str_data中。

变元

```
SocketReceive Socket [ \Str ] | [ \RawData ] | [ \Data ]
     [\ReadNoOfBytes] [\NoRecBytes] [\Time]
```

Socket

数据类型：socketdev

在套接字接收数据的客户端应用中，必须已经创建和连接套接字。

在套接字接收数据的服务器应用中，必须已经接受套接字。

[\Str]

数据类型：string

应当储存接收string数据的变量。可处理最多80个字符。

[\RawData]

数据类型：rawbytes

应当储存接收原始数据字节数据的变量。可处理最多1024个rawbytes。

[\Data]

数据类型：array of byte

应当储存接收字节数据的变量。可处理最多1024个byte。

仅可在同一时间使用可选参数\Str、 \RawData和\Data之一。

[\ReadNoOfBytes]

Read number of Bytes

数据类型：num

待读取的字节数。待读取字节的最小值为1，且最大数量为所用数据类型的规模，即如果使用数据类型string的变量，则为80个字节。

如果与始终发送固定字节数的客户端进行通信，则该可选参数可用于指定针对各SocketReceive指令所应读取的相同字节数量。

如果发送方发送原始数据，则接收方需要指定针对已发送的各rawbytes所应接收的4个字节。

下一页继续

1.242 SocketReceive - 接收来自远程计算机的数据

Socket Messaging

续前页

[\NoRecBytes]

Number Received Bytes

数据类型：num

关于储存指定socketdev所需字节数的变量。

通过以下方面，亦可获得相同的结果

- 关于参数\Str中的变量的函数StrLen
- 关于参数\RawData中的变量的函数RawBytesLen

[\Time]

数据类型：num

程序执行等待接收数据的最长时间量。如果在转移数据之前耗尽时间，则将调用错误处理器，如果存在这样的情况，则采用错误代码ERR_SOCK_TIMEOUT。如果不存在错误处理器，则将停止执行。

如果未使用参数\Time，则等待时间为60秒。为了永久等待，则使用预定义常量WAIT_MAX。

程序执行

将等待执行SocketReceive，直至数据在出现超时错误时可用或失效。

通过本指令中使用的数据类型来指定所读取的字节数量。如果使用string数据类型来接收数据，则如果可读取80个字节，则会接收到80个字节。如果使用可选参数ReadNoOfBytes，用户可指定各SocketReceive所应接收的字节数。

通过电缆转移的数据通常为字节，一条消息中最多包含1024个字节。默认不向消息添加任何标题。为实际应用而保留任意标题的使用。

参数	输入数据	电缆数据	输出数据
\Str	1个字符	1个字节（8位）	1个字符
\RawData	1个原始数据字节	1个字节（8位）	1个原始数据字节
\Data	1个字节	1个字节（8位）	1个字节

可能在SocketSend和SocketReceive之间混合使用过的数据类型（string、rawbytes或array of byte）。

更多示例

有关于指令SocketReceive的更多例子阐述如下。

例 1

```
VAR socketdev server_socket;
VAR socketdev client_socket;
VAR string client_ip;

PROC server_messaging()
  VAR string receive_string;
  ...
  ! Create, bind, listen and accept of sockets in error handlers
  SocketReceive client_socket \Str := receive_string;
  SocketSend client_socket \Str := "Hello client with
       ip-address"+client_ip;
```

下一页继续

```
  ! Wait for acknowlegde from client
  ...
  SocketClose server_socket;
  SocketClose client_socket;
ERROR
  IF ERRNO=ERR_SOCK_TIMEOUT THEN
    RETRY;
  ELSEIF ERRNO=SOCK_CLOSED THEN
    server_recover;
    RETRY;
  ELSE
  ! No error recovery handling
  ENDIF
ENDPROC

PROC server_recover()
  SocketClose server_socket;
  SocketClose client_socket;
  SocketCreate server_socket;
  SocketBind server_socket, "192.168.0.1", 1025;
  SocketListen server_socket;
  SocketAccept server_socket,
       client_socket\ClientAddress:=client_ip;
ERROR
  IF ERRNO=ERR_SOCK_TIMEOUT THEN
    RETRY;
  ELSEIF ERRNO=ERR_SOCK_CLOSED THEN
    RETURN;
  ELSE
    ! No error recovery handling
  ENDIF
ENDPROC
```

此为关于创建、绑定、监听和接受错误处理器中套接字的服务器程序实例。采用这种方式，本程序可处理上电失败重启。

在无返回值程序server_recover中，创建服务器套接字，并与地址为192.168.0.1的控制器网络上的端口1025绑定。在执行SocketListen后，服务器套接字开始监听位于该端口和地址上的输入连接。SocketAccept将接受来自一些客户端的输入连接，并将客户端地址储存在字符串client_ip中。

在通信无返回值程序server_messaging中，服务器接收来自客户端的字符串消息，并将消息储存在receive_string中。随后，服务器对消息"Hello client with ip-address xxx.xxx.x.x"做出响应。

错误处理

系统会生成下列可恢复错误，并在错误处理器中处理这些错误。系统变量ERRNO将被设置成：

ERR_SOCK_CLOSED	关闭套接字。连接损坏。
ERR_SOCK_TIMEOUT	在超时时间内未收到任何数据。

下一页继续

1.242 SocketReceive - 接收来自远程计算机的数据

Socket Messaging

续前页

限制

套接字消息传送中不存在内置同步机制，以避免接收到由多种发送消息组成的消息。由程序员来处理与“Ack”消息（客户端或服务器程序中的一系列SocketSend - SocketReceive，且必须在下一系列SocketSend - SocketReceive之前完成）的同步。

在上电失败重启后，关闭所有套接字。可通过错误恢复来处理该问题。参见上文中的例子。

通过SocketCreate ... SocketClose，避免快回路，因为在某一特定时间之前，套接字并未真正关闭（TCP/IP功能）。

可在一次调用中接收到的最大数据尺寸被限制为1024字节。

语法

```
SocketReceive
  [ Socket ':=' ] < variable (VAR) of socketdev >
  [ '\' Str ':=' < variable (VAR) of string > ]
  | [ '\' RawData ':=' < variable (VAR) of rawbytes > ]
  | [ '\' Data ':=' < array {*} (VAR) of byte > ]
  [ '\' ReadNoOfBytes ':=' < expression (IN) of num > ]
  [ '\' NoRecBytes ':=' < variable (VAR) of num > ]
  [ '\' Time ':=' < expression (IN) of num > ] ';'
```

相关信息

信息，关于	请参阅
一般的套接字通信	应用手册 - 控制器软件*IRC5*
创建新套接字	第*623*页的*SocketCreate - 创建新套接字*
连接远程计算机（仅限客户端）	第*620*页的*SocketConnect - 连接远程计算机*
向远程计算机发送数据	第*635*页的*SocketSend - 向远程计算机发送数据*
关闭套接字	第*618*页的*SocketClose - 关闭套接字*
绑定套接字（仅限服务器）	第*616*页的*SocketBind - 将套接字与我的IP地址和端口绑定*
监听连接（仅限服务器）	第*625*页的*SocketListen - 监听输入连接*
接受连接（仅服务器）	第*613*页的*SocketAccept - 接受输入连接*
获得当前套接字的状态	第*1230*页的*SocketGetStatus - 获得当前套接字的状态*
客户端套接字应用实例	第*635*页的*SocketSend - 向远程计算机发送数据*
测试套接字上数据的存在。	第*1232*页的*SocketPeek - 测试套接字上数据的存在*

1.243 SocketReceiveFrom - 接收来自远程计算机的数据

手册用法

SocketReceiveFrom用于从远程计算机接收数据。SocketReceiveFrom可用于客户端和服务器应用。

SocketReceiveFrom用于同数据电报协议UDP/IP的非连接通信。

基本示例

以下实例介绍了指令SocketReceiveFrom：

另请参阅第*628*页的更多示例

例 1

```
VAR string str_data;
VAR string RemoteAddress;
VAR num RemotePort;
...
SocketCreate \UDP;
SocketBind myUDPsock, "192.168.9.100", 4044;
SocketReceiveFrom socket1 \Str := str_data, RemoteAddress,
     RemotePort;
```

接收来自远程计算机的数据，并将其储存在字符串变量str_data中。将远程计算机的地址储存在字符串变量RemoteAddress中，并将端口号储存在数字变量RemotePort中。

变元

```
SocketReceiveFrom Socket [ \Str ] | [ \RawData ] | [ \Data ]
     [\NoRecBytes] RemoteAddress RemotePort [\Time]
```

Socket

数据类型：socketdev

识别绑定套接字的套接字设备。

[\Str]

数据类型：string

应当储存接收string数据的变量。可处理最多80个字符。

[\RawData]

数据类型：rawbytes

应当储存接收原始数据字节数据的变量。可处理最多1024个rawbytes。

[\Data]

数据类型：array of byte

应当储存接收字节数据的变量。可处理最多1024个byte。

仅可在同一时间使用可选参数\Str、 \RawData和\Data之一。

[\NoRecBytes]

Number Received Bytes

数据类型：num

下一页继续

1.243 SocketReceiveFrom - 接收来自远程计算机的数据
Socket Messaging
续前页

关于储存指定socketdev所需字节数的变量。

通过以下方面，亦可获得相同的结果

- 关于参数\Str中的变量的函数StrLen
- 关于参数\RawData中的变量的函数RawBytesLen

RemoteAddress

数据类型：string

包含远程计算机源地址的字符串变量。

RemotePort

数据类型：num

包含远程计算机在发送数据电报包时使用的端口的数字变量。

[\Time]

数据类型：num

程序执行等待接收数据的最长时间量。如果在转移数据之前耗尽时间，则将调用错误处理器，如果存在这样的情况，则采用错误代码ERR_SOCK_TIMEOUT。如果不存在错误处理器，则将停止执行。

如果未使用参数\Time，则等待时间为60秒。为了永久等待，则使用预定义常量WAIT_MAX。

程序执行

SocketReceiveFrom的执行会收到数字电报，并储存源地址和源端口。其将等待，直至在超时错误时数据可用或失效。

通过在指令中使用的数据类型来指定读取的字节数。如果使用string数据类型来接收数据，在可读取80个字节时，便可接收到80个字节。

通过电缆转移的数据通常为字节，一条消息中最多包含1024个字节。默认不向消息添加任何标题。为实际应用而保留任意标题的使用。

参数	输入数据	电缆数据	输出数据
\Str	1个字符	1个字节（8位）	1个字符
\RawData	1个原始数据字节	1个字节（8位）	1个原始数据字节
\Data	1个字节	1个字节（8位）	1个字节

可能在SocketSendTo和SocketReceiveFrom之间混合使用过的数据类型（string、rawbytes或array of byte）。

更多示例

有关于指令SocketReceiveFrom的更多例子阐述如下。

例 1

```
VAR socketdev udp_socket;
VAR string client_ip;
VAR num client_port;

PROC server_messaging()
  VAR string receive_string;
  ...
```

下一页继续

```
  ! Create and bind of sockets in error handlers
  SocketReceiveFrom udp_socket \Str := receive_string, client_ip,
       client_port;
  SocketSendTo udp_socket, client_ip, client_port \Str := "Hello
       client with ip-address"+client_ip;
  ...
  SocketClose udp_socket;
ERROR
  IF ERRNO=ERR_SOCK_TIMEOUT THEN
    RETRY;
  ELSEIF ERRNO=SOCK_CLOSED THEN
    messaging_recover;
    RETRY;
  ELSE
  ! No error recovery handling
  ENDIF
ENDPROC

PROC messaging_recover()
  SocketClose udp_socket;
  SocketCreate udp_socket \UDP;
  SocketBind udp_socket, "192.168.0.1", 1025;
ERROR
  IF ERRNO=ERR_SOCK_CLOSED THEN
    RETURN;
  ELSE
    ! No error recovery handling
  ENDIF
ENDPROC
```

此为关于创建和绑定错误处理器中套接字的服务器程序实例。采用这种方式，本程序可处理上电失败重启。

在通信无返回值程序server_messaging中，服务器接收来自客户端的字符串消息，并将消息储存在receive_string中。随后，服务器对消息"Hello client with ip-address xxx.xxx.x.x"做出响应。

错误处理

系统会生成下列可恢复错误，并在错误处理器中处理这些错误。系统变量ERRNO将被设置成：

ERR_SOCK_CLOSED	关闭套接字。
ERR_SOCK_TIMEOUT	在超时时间内未收到任何数据。

限制

在上电失败重启后，关闭所有套接字。可通过错误恢复来处理该问题。参见上文中的例子。

可在一次调用中接收到的最大数据尺寸被限制为1024字节。

下一页继续

1.243 SocketReceiveFrom - 接收来自远程计算机的数据

Socket Messaging

续前页

语法

```
SocketReceiveFrom
  [ Socket ':=' ] < variable (VAR) of socketdev >
  [ '\' Str ':=' < variable (VAR) of string > ]
  | [ '\' RawData ':=' < variable (VAR) of rawbytes > ]
  | [ '\' Data ':=' < array {*} (VAR) of byte > ]
  [ '\' NoRecBytes ':=' < variable (VAR) of num > ]
  [ RemoteAddress ':=' ] < variable (VAR) of string >
  [ RemotePort ':=' ] < variable (VAR) of num >
  [ '\' Time ':=' < expression (IN) of num > ] ';'
```

相关信息

信息，关于	请参阅
一般的套接字通信	应用手册 - 控制器软件*IRC5*
创建新套接字	第*623*页的*SocketCreate* - 创建新套接字
连接远程计算机（仅限客户端）	第*620*页的*SocketConnect* - 连接远程计算机
向远程计算机发送数据	第*635*页的*SocketSend* - 向远程计算机发送数据
关闭套接字	第*618*页的*SocketClose* - 关闭套接字
绑定套接字（仅限服务器）	第*616*页的*SocketBind* - 将套接字与我的*IP*地址和端口绑定
监听连接（仅限服务器）	第*625*页的*SocketListen* - 监听输入连接
接受连接（仅服务器）	第*613*页的*SocketAccept* - 接受输入连接
获得当前套接字的状态	第*1230*页的*SocketGetStatus* - 获得当前套接字的状态
客户端套接字应用实例	第*635*页的*SocketSend* - 向远程计算机发送数据
向远程计算机发送数据	第*639*页的*SocketSendTo* - 向远程计算机发送数据
测试套接字上数据的存在。	第*1232*页的*SocketPeek* - 测试套接字上数据的存在

1.244 SocketSend - 向远程计算机发送数据

手册用法

SocketSend用于向远程计算机发送数据。SocketSend可用于客户端和服务器应用。

基本示例

以下实例介绍了指令SocketSend：

另请参阅第*636*页的更多示例

例 1

```
SocketSend socket1 \Str := "Hello world";
```

将消息"Hello world"发送给远程计算机。

变元

```
SocketSend Socket [ \Str ] | [ \RawData ] | [ \Data] [ \NoOfBytes
    ]
```

Socket

数据类型：socketdev

在客户端应用中，必须已经创建和连接用于发送的套接字。

在服务器应用中，必须已经接受用于发送的套接字。

[\Str]

数据类型：string

将string发送到远程计算机。

[\RawData]

数据类型：rawbytes

将rawbytes数据发送到远程计算机。

[\Data]

数据类型：array of byte

将byte数组中的数据发送到远程计算机。

仅可在同一时间使用可选参数\Str、\RawData或\Data之一。

[\NoOfBytes]

数据类型：num

如果指定该参数，则仅将该字节数发送到远程计算机。如果\NoOfBytes大于待发送数据结构中的实际字节数，则SocketSend调用将失败。

如果未指定该参数，则将整个数据结构（rawbytes的有效部分）发送到远程计算机。

下一页继续

1.244 SocketSend - 向远程计算机发送数据

Socket Messaging

续前页

程序执行

将指定数据发送到远程计算机。如果连接损坏，则会产生错误。

通过电缆转移的数据通常为字节，一条消息中最多包含1024个字节。默认不向消息添加任何标题。为实际应用而保留任意标题的使用。

参数	输入数据	电缆数据	输出数据
\Str	1个字符	1个字节（8位）	1个字符
\RawData	1个原始数据字节	1个字节（8位）	1个原始数据字节
\Data	1个字节	1个字节（8位）	1个字节

可能在SocketSend和SocketReceive之间混合使用过的数据类型（string、rawbytes或array of byte）。

更多示例

有关于指令SocketSend的更多例子阐述如下。

例 1

```
VAR socketdev client_socket;
VAR string receive_string;

PROC client_messaging()
  ...
  ! Create and connect the socket in error handlers
  SocketSend client_socket \Str := "Hello server";
  SocketReceive client_socket \Str := receive_string;
  ...
  SocketClose client_socket;
ERROR
  IF ERRNO=ERR_SOCK_TIMEOUT THEN
    RETRY;
  ELSEIF ERRNO=ERR_SOCK_CLOSED THEN
    client_recover;
    RETRY;
  ELSE
    ! No error recovery handling
  ENDIF
ENDPROC

PROC client_recover()
  SocketClose client_socket;
  SocketCreate client_socket;
  SocketConnect client_socket, "192.168.0.2", 1025;
ERROR
  IF ERRNO=ERR_SOCK_TIMEOUT THEN
    RETRY;
  ELSEIF ERRNO=ERR_SOCK_CLOSED THEN
    RETURN;
  ELSE
    ! No error recovery handling
  ENDIF
```

下一页继续

```
ENDPROC
```

此为关于创建和连接错误处理器中套接字的客户端程序实例。采用这种方式，本程序可处理上电失败重启。

在无返回值程序client_recover中，创建客户端套接字，并与端口1025上IP地址为192.168.0.2的远程计算机服务器相连。

在通信无返回值程序client_messaging中，客户端将"Hello server"发送到服务器，服务器对客户端的"Hello client"予以响应，并将其储存在变量receive_string中。

例 2

```
VAR socketdev client_socket;
VAR string receive_string;

PROC client_messaging()
  ...
  ! Send cr and lf to the server
  SocketSend client_socket \Str := "\0D\0A";
  ...
ENDPROC
```

这是在字符串中发送不可打印字符（二进制数据）的客户端程序实例。如果同需要此类字符的传感器或其他客户端进行通信，则上述实例有用。

错误处理

系统会生成下列可恢复错误，并在错误处理器中处理这些错误。系统变量ERRNO将被设置成：

ERR_SOCK_CLOSED	The socket is closed. Broken connection.

限制

套接字消息传送中不存在内置同步机制，以避免接收到由多种发送消息组成的消息。由程序员来处理与“Ack”消息（客户端或服务器程序中的一系列SocketSend - SocketReceive，且必须在下一系列SocketSend - SocketReceive之前完成）的同步。

在上电失败重启后，关闭所有套接字。可通过错误恢复来处理该问题。参见上文中的例子。

通过SocketCreate ... SocketClose，避免快回路，因为在某一特定时间之前，套接字并未真正关闭（TCP/IP功能）。

待发送数据的尺寸被限制在1024个字节。

语法

```
SocketSend
  [ Socket ´:=´ ] < variable (VAR) of socketdev >
  [ \Str ´:=´ < expression (IN) of string > ]
  | [ \RawData ´:=´ < variable (VAR) of rawdata > ]
  | [ \Data ´:=´ < array {*} (IN) of byte > ]
  [ ’\’ NoOfBytes ´:=´ < expression (IN) of num > ] ’;’
```

下一页继续

1.244 SocketSend - 向远程计算机发送数据
Socket Messaging
续前页

相关信息

信息，关于	请参阅
一般的套接字通信	应用手册 - 控制器软件*IRC5*
创建新套接字	第*623*页的*SocketCreate* - 创建新套接字
连接远程计算机（仅限客户端）	第*620*页的*SocketConnect* - 连接远程计算机
接收来自远程计算机的数据	第*627*页的*SocketReceive* - 接收来自远程计算机的数据
关闭套接字	第*618*页的*SocketClose* - 关闭套接字
绑定套接字（仅限服务器）	第*616*页的*SocketBind* - 将套接字与我的*IP*地址和端口绑定
监听连接（仅限服务器）	第*625*页的*SocketListen* - 监听输入连接
接受连接（仅服务器）	第*613*页的*SocketAccept* - 接受输入连接
获得当前套接字的状态	第*1230*页的*SocketGetStatus* - 获得当前套接字的状态
服务器套接字应用实例	第*627*页的*SocketReceive* - 接收来自远程计算机的数据
使用字符串文字中的不可打印字符（二进制数据）。	技术参考手册 - *RAPID*语言内核

1.245 SocketSendTo - 向远程计算机发送数据

手册用法

SocketSendTo用于向远程计算机发送数据。SocketSendTo可用于客户端和服务器应用。

SocketSendTo用于同数据电报协议UDP/IP的非连接通信。

基本示例

以下实例介绍了指令SocketSendTo：

另请参阅第*636*页的更多示例

例 1

```
VAR socketdev udp_socket;

SocketCreate udp_socket \UDP;
SocketSendTo udp_socket, Address, Port \Str := "Hello world";
```

将消息"Hello world"发送到IP地址为Address且端口为Port的远程计算机。

变元

```
SocketSendTo Socket RemoteAddress RemotePort [ \Str ] | [ \RawData
     ] | [ \Data] [ \NoOfBytes ]
```

Socket

数据类型：socketdev

必须已经创建了套接字。

RemoteAddress

数据类型：string

远程计算机的地址。必须将远程计算机指定为一个IP地址。不可能使用远程计算机的名称。

RemotePort

数据类型：num

位于远程计算机上的端口。通常，可自由使用端口1025-4999。可能已经采用了低于1025的端口。

[\Str]

数据类型：string

将string发送到远程计算机。

[\RawData]

数据类型：rawbytes

将rawbytes数据发送到远程计算机。

[\Data]

数据类型：array of byte

将byte数组中的数据发送到远程计算机。

仅可在同一时间使用可选参数\Str、\RawData或\Data之一。

下一页继续

1.245 SocketSendTo - 向远程计算机发送数据

Socket Messaging

续前页

```
[ \NoOfBytes ]
```

数据类型：num

如果指定该参数，则仅将该字节数发送到远程计算机。如果\NoOfBytes大于待发送数据结构中的实际字节数，则SocketSendTo调用将失败。

如果未指定该参数，则将整个数据结构（rawbytes的有效部分）发送到远程计算机。

程序执行

将指定数据发送到远程计算机。

通过电缆转移的数据通常为字节，一条消息中最多包含1024个字节。默认不向消息添加任何标题。为实际应用而保留任意标题的使用。

参数	输入数据	电缆数据	输出数据
\Str	1个字符	1个字节（8位）	1个字符
\RawData	1个原始数据字节	1个字节（8位）	1个原始数据字节
\Data	1个字节	1个字节（8位）	1个字节

可能在SocketSendTo和SocketReceiveFrom之间混合使用过的数据类型（string、rawbytes或array of byte）。

更多示例

有关于指令SocketSendTo的更多例子阐述如下。

例 1

```
VAR socketdev client_socket;
VAR string receive_string;
VAR string RemoteAddress;
VAR num RemotePort;

PROC client_messaging()
  ...
  ! Create and bind the socket in error handlers
  SocketSendTo client_socket, "192.168.0.2", 1025 \Str := "Hello
       server";
  SocketReceiveFrom client_socket \Str := receive_string,
       RemoteAddress, RemotePort;
  ...
  SocketClose client_socket;
ERROR
  IF ERRNO=ERR_SOCK_TIMEOUT THEN
    RETRY;
  ELSEIF ERRNO=ERR_SOCK_CLOSED THEN
    client_recover;
    RETRY;
  ELSE
    ! No error recovery handling
  ENDIF
ENDPROC
```

下一页继续

```
PROC client_recover()
  SocketClose client_socket;
  SocketCreate client_socket \UDP;
  SocketBind client_socket, "192.168.0.2", 1025;
ERROR
  IF ERRNO=ERR_SOCK_TIMEOUT THEN
    RETRY;
  ELSEIF ERRNO=ERR_SOCK_CLOSED THEN
    RETURN;
  ELSE
    ! No error recovery handling
  ENDIF
ENDPROC
```

此为关于创建和绑定错误处理器中套接字的客户端程序实例。采用这种方式，本程序可处理上电失败重启。

在无返回值程序client_recover中，创建客户端套接字，并与端口1025上IP地址为192.168.0.2的远程计算机服务器绑定。

在通信无返回值程序client_messaging中，客户端将"Hello server"发送到服务器，服务器对客户端的"Hello client"予以响应，并将其储存在变量receive_string中。

例 2

```
VAR socketdev udp_socket;

PROC message_send()
  ...
  ! Send cr and lf to the server
  SocketSendTo udp_socket, "192.168.0.2", 1025 \Str := "\0D\0A";
  ...
ENDPROC
```

这是在字符串中发送不可打印字符（二进制数据）的程序实例。如果同需要此类字符的传感器或其他客户端进行通信，则上述实例有用。

错误处理

系统会生成下列可恢复错误，并在错误处理器中处理这些错误。系统变量ERRNO将被设置成：

ERR_SOCK_CLOSED	The socket is closed.

限制

在上电失败重启后，关闭所有套接字。可通过错误恢复来处理该问题。参见上文中的例子。

待发送数据的尺寸被限制在1024个字节。

语法

```
SocketSendTo
  [ Socket ':=' ] < variable (VAR) of socketdev >
  [ RemoteAddress ':=' ] < expression (IN) of string >
```

下一页继续

1.245 SocketSendTo - 向远程计算机发送数据

Socket Messaging

续前页

```
[ RemotePort ':=' ] < expression (IN) of num >
[ \Str ':=' < expression (IN) of string > ]
| [ \RawData ':=' < variable (VAR) of rawdata > ]
| [ \Data ':=' < array {*} (IN) of byte > ]
[ '\' NoOfBytes ':=' < expression (IN) of num > ] ';'
```

相关信息

信息，关于	请参阅
一般的套接字通信	应用手册 - 控制器软件*IRC5*
创建新套接字	第*623*页的*SocketCreate* - 创建新套接字
连接远程计算机（仅限客户端）	第*620*页的*SocketConnect* - 连接远程计算机
接收来自远程计算机的数据	第*627*页的*SocketReceive* - 接收来自远程计算机的数据
关闭套接字	第*618*页的*SocketClose* - 关闭套接字
绑定套接字（仅限服务器）	第*616*页的*SocketBind* - 将套接字与我的*IP*地址和端口绑定
监听连接（仅限服务器）	第*625*页的*SocketListen* - 监听输入连接
接受连接（仅服务器）	第*613*页的*SocketAccept* - 接受输入连接
获得当前套接字的状态	第*1230*页的*SocketGetStatus* - 获得当前套接字的状态
服务器套接字应用实例	第*627*页的*SocketReceive* - 接收来自远程计算机的数据
接收来自远程计算机的数据	第*631*页的*SocketReceiveFrom* - 接收来自远程计算机的数据
使用字符串文字中的不可打印字符（二进制数据）。	技术参考手册 - *RAPID*语言内核

1.246 SoftAct - 启用软伺服

手册用法

SoftAct（*Soft Servo Activate*）用于启用机械臂或外部机械单元任意轴上的“软”伺服。

本指令仅可用于主任务T_ROB1，或者如果在*MultiMove*系统中，则可用于任何运动任务中。

基本示例

以下实例介绍了指令SoftAct：

例 1

```
SoftAct 3, 20;
```

采用柔性度值20%，启用机械臂轴3上的软伺服。

例 2

```
SoftAct 1, 90 \Ramp:=150;
```

采用柔性度值90%以及斜面系数150%，启用机械臂轴1上的软伺服。

例 3

```
SoftAct \MechUnit:=orbit1, 1, 40 \Ramp:=120;
```

采用柔性度值40%以及斜面系数120%，启用机械单元orbit1的轴1上的软伺服。

变元

```
SoftAct [\MechUnit] Axis Softness [\Ramp]
```

[\MechUnit]

Mechanical Unit

数据类型：mecunit

机械单元的名称。如果省略该参数，则意味着启用当前程序任务中指定机械臂轴的软伺服。

Axis

数据类型：num

与软伺服共同起作用的机械臂或外轴的数量。

Softness

数据类型：num

柔性度值，以百分比计（0 - 100%）。0%表示最小柔性度（最大刚度），100%表示最大柔性度。

[\Ramp]

数据类型：num

斜面系数，以百分比计（>= 100%）。斜面系数用于控制软伺服的参与度。系数100%表示正常值；值越大，则软伺服参与更为缓慢（更长的斜面）。斜面系数的默认值为100 %。

下一页继续

1.246 SoftAct - 启用软伺服
RobotWare - OS
续前页

程序执行

以针对当前轴而指定的值，启用柔性度。柔性度值适用于所有移动，直至针对当前轴而编程新的柔性度值，或直至通过指令SoftDeact而停用软伺服。

限制

当存在电源故障时，始终停用关于任意机械臂或外轴的软伺服。在电源故障之后重启时，可通过用户程序来处理该限制。

不得启用相同的轴两次，除非在两者之间存在移动指令。因此，应当避免以下程序序列。否则，将在机械臂移动中出现猛拉。

```
SoftAct n , x ;
SoftAct n , y ;
```

（n = 机械臂轴n，x和y为柔性度值）

警告

当启用软伺服时，1类停止的制动距离将更长。

语法

```
SoftAct
  ['\'MechUnit ':=' < variable (VAR) of mecunit>´,´]
  [Axis ':=' ] < expression (IN) of num> ',' 
  [Softness':=' ] < expression (IN) of num> ´,´
  [ '\'Ramp':=' < expression (IN) of num> ]';'
```

相关信息

信息，关于	请参阅
停用软伺服	第645页的*SoftDeact - 停用软伺服*
有关软伺服启用时的行为	技术参考手册 - *RAPID*语言概览
外轴的配置	应用手册 - *Additional axes and stand alone controller*

1.247 SoftDeact - 停用软伺服

手册用法

SoftDeact（*Soft Servo Deactivate*）用于停用所谓的“软”伺服。

基本示例

以下实例介绍了指令SoftDeact：

例 1

```
SoftDeact;
```

停用所有轴上的软伺服。

例 2

```
SoftDeact \Ramp:=150;
```

停用所有轴上的软伺服，且斜面系数为150 %。

变元

```
SoftDeact [\Ramp]
```

[\Ramp]

数据类型：num

斜面系数，以百分比计（>＝100%）。斜面系数用于控制软伺服的停用。系数100%表示正常值。值越大，则软伺服停用更为缓慢（更长的斜面）。斜面系数的默认值为100 %。

程序执行

停用机械单元的软伺服，通过当前程序任务来控制此类机械单元。如果通过非运动任务来实施SoftDeact，则停用由相关运动任务控制的机械单元的软伺服。当处于同步移动模式时，执行SoftDeact，将停用所有同步机械单元的软伺服。

当停用软伺服以及SoftDeact时，机械臂将移动至编程位置，即使机械臂已经在软伺服启用期间，从该位置移动出来。

语法

```
SoftDeact
  [ '\'Ramp ':=' < expression (IN) of num> ]';'
```

相关信息

信息，关于	请参阅
启用软伺服	第643页的*SoftAct - 启用软伺服*

1.248 SpeedLimAxis - 设置轴的速度限制

手册用法

SpeedLimAxis用于设置轴的速度限值。将系统输入信号LimitSpeed设置为1时，完成减速。通过该指令，可能设置随后应当应用的速度限值。

本指令仅可用于主任务T_ROB1，或者如果在*MultiMove*系统中，则可用于任何运动任务中。

基本示例

以下实例介绍了指令SpeedLimAxis：

例 1

```
SpeedLimAxis STN_1, 1, 20;
```

当系统输入LimitSpeed设置为1时，这将会把机械单元STN_1轴1的速度限制在20度/秒。

例 2

```
SpeedLimAxis ROB_1, 1, 10;
SpeedLimAxis ROB_1, 2, 30;
SpeedLimAxis ROB_1, 3, 30;
SpeedLimAxis ROB_1, 4, 30;
SpeedLimAxis ROB_1, 5, 30;
SpeedLimAxis ROB_1, 6, 30;
```

当系统输入LimitSpeed设置为1时，这将会把机械单元ROB_1轴2到轴6的速度限制在30度/秒，并将轴1的速度限制在10度/秒。

变元

```
SpeedLimAxis MechUnit AxisNo AxisSpeed
```

MechUnit

Mechanical Unit

数据类型：mecunit

机械单元名称

AxisNo

数据类型：num

机械单元当前轴的编号。

AxisSpeed

数据类型：num

应当应用的速度。对于旋转轴，速度应当以度/秒计，对于线性轴，速度应当以mm/s计。

程序执行

SpeedLimAxis用于设置指定机械单元的轴的速度限值。并非立即进行减速。将系统输入信号LimitSpeed设置为1时，储存并应用各值。

下一页继续

如果SpeedLimAxis不用于设置轴的限值，则将转而使用手动模式的速度限制。如果特定轴不需要任何限制，则应当输入一个高值。此外，如果指令SpeedLimCheckPoint的使用未设置检查点速度限制，则手动模式的速度限制将用于限制检查点速度。

当系统输入信号LimitSpeed被设置成1时，该速度会减少成相应的已减速度。

当系统输入信号LimitSpeed被设置成0时，该速度会增大成当前移动指令采用的编程速度。

当达到减速时，将系统输出信号LimitSpeed设置为1。当速度开始增加时，将系统输出信号LimitSpeed设置为0。

自动设置速度限制的默认值

- 当使用重启模式重置RAPID时
- 当加载一则新程序或一个新模块时
- 当从起点开始执行程序时
- 当将程序指针移动到main时
- 当将程序指针移动到子程序时
- 移动程序指针造成执行顺序丢失时

更多示例

有关于指令SpeedLimAxis的更多例子阐述如下。

例 1

```
..
VAR intnum sigint1;
VAR intnum sigint2;
..
PROC main()
  ! Setup interrupts reacting on a signal input
  IDelete sigint1;
  CONNECT sigint1 WITH setlimitspeed;
  ISignalDI \SingleSafe, mysensorsignal, 1, sigint1;
  IDelete sigint2;
  CONNECT sigint2 WITH resetlimitspeed;
  ISignalDI \SingleSafe, mysensorsignal, 0, sigint2;
  ..
  MoveL p1, z50, fine, tool2;
  MoveL p2, z50, fine, tool2;
  ..
  MoveL p10, v100, fine, tool2;
  ! Set limitations for checkpoints and axes
  SpeedLimCheckPoint 200;
  SpeedLimAxis ROB_1, 1, 10;
  SpeedLimAxis ROB_1, 2, 10;
  SpeedLimAxis ROB_1, 3, 10;
  SpeedLimAxis ROB_1, 4, 20;
  SpeedLimAxis ROB_1, 5, 20;
  SpeedLimAxis ROB_1, 6, 20;
  WHILE run_loop = TRUE DO
    MoveL p1, vmax, z50, tool2;
    ..
```

下一页继续

```
    MoveL p99, vmax, fine, tool2;
  ENDWHILE
  ! Set the default manual mode max speed
  SpeedLimCheckPoint 0;
  SpeedLimAxis ROB_1, 1, 0;
  SpeedLimAxis ROB_1, 2, 0;
  SpeedLimAxis ROB_1, 3, 0;
  SpeedLimAxis ROB_1, 4, 0;
  SpeedLimAxis ROB_1, 5, 0;
  SpeedLimAxis ROB_1, 6, 0;
  ..
TRAP setlimitspeed
  IDelete sigint1;
  CONNECT sigint1 WITH setlimitspeed;
  ISignalDI \SingleSafe, mysensorsignal, 1, sig1int1;
  ! Set out signal that is cross connected to system input
       LimitSpeed
  SetDO do1LimitSpeed, 1;
ENDTRAP
TRAP resetlimitspeed
  IDelete sigint2;
  CONNECT sigint2 WITH resetlimitspeed;
  ISignalDI \SingleSafe, mysensorsignal, 0, sig1int2;
  ! Reset out signal that is cross connected to system input
       LimitSpeed
  SetDO do1LimitSpeed, 0;
ENDTRAP
```

在机械臂从位置p1移动至p10期间，使用默认速度限制（手动模式速度）。为TCP机械臂和轴的检查点增加新的速度限制。如果信号mysensorsignal的值改变为1时，则TRAP setlimitspeed将应用速度限制。

当信号mysensorsignal的值改变为0时，TRAP resetlimitspeed将移除速度限制。

只要变量run_loop为TRUE，随即将使用速度限制的新设置，并将系统输入信号LimitSpeed设置为1。将run_loop设置为FALSE时，设置默认速度限制（手动模式速度）。

注意

本例子中的TRAP程序仅用于使功能形象化。用于限制速度的信号亦可直接与系统输入信号LimitSpeed相连，或通过安全PLC来相连。

错误处理

系统会生成下列可恢复错误，并在错误处理器中处理这些错误。系统变量ERRNO将被设置成：

ERR_AXIS_PAR	指令中的参数轴错误
ERR_SPEEDLIM_VALUE	参数AxisSpeed中使用的速度过低。

下一页继续

限制

SpeedLimAxis不能用于通电（POWER ON）事件程序。

当降低一个轴或检查点的速度时，其他轴亦将降低至相同的百分比，以便能够沿编程路径运行。沿编程路径的过程速度将会有所不同。

当使用SafeMove连同速度限制时，必须为SafeMove设置裕量，因为SafeMove和运动计算略有不同。

语法

```
SpeedLimAxis
    [ MechUnit':=' ] < variable (VAR) of mecunit> ','
    [ AxisNo':=' ] < expression (IN) of num> ','
    [ AxisSpeed':=' ] < expression (IN) of num> ';'
```

相关信息

信息，关于	请参阅
定位器指令	技术参考手册 - *RAPID*语言概览
设置检查点的速度限制	第*650*页的*SpeedLimCheckPoint* - 设置检查点的速度限制
系统输入输出信号	技术参考手册 - 系统参数

1.249 SpeedLimCheckPoint - 设置检查点的速度限制

手册用法

SpeedLimCheckPoint用于设置TCP机械臂的速度限值。将系统输入信号LimitSpeed设置为1时，完成减速。通过该指令，可能设置随后应当应用的速度限值。

如果任意检查点的运行速度比SpeedLimCheckPoint所设置的限制更快，则进行减速。（欲知关于检查点的更多信息，参见第652页的更多示例。

本指令仅可用于主任务T_ROB1，或者如果在*MultiMove*系统中，则可用于任何运动任务中。

基本示例

以下实例介绍了指令SpeedLimCheckPoint：

例 1

```
VAR num limit_speed:=200;
SpeedLimCheckPoint limit_speed;
```

将系统输入LimitSpeed设置为1时，这将把TCP机械臂的速度限制在200 mm/s。

变元

```
SpeedLimCheckPoint RobSpeed
```

RobSpeed

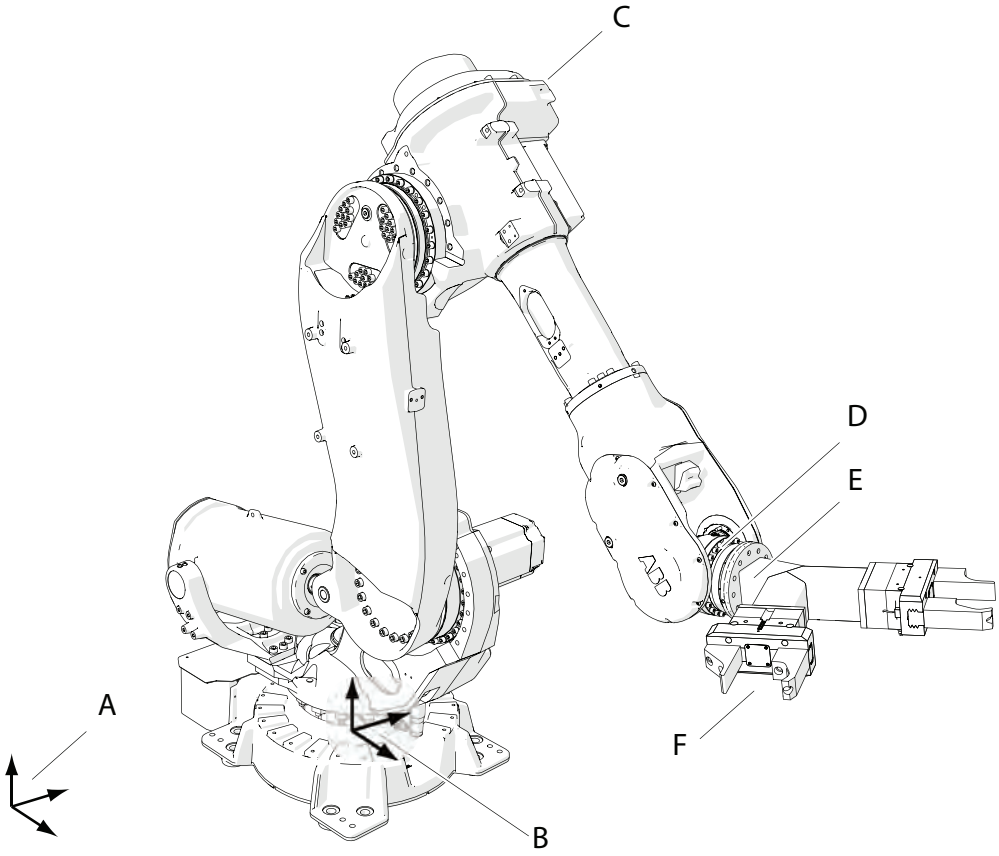
数据类型：num

应当应用的速度限制，以mm/s计。

下一页继续

程序执行

关于检查点的定义，请参见下图。



xx1200000521

A	大地坐标系
B	基坐标系
C	臂检查点
D	腕中心点（WCP）
E	tool0
F	工具中心点

SpeedLimCheckPoint用于设置TCP机械臂4个检查点的速度限值。将被限制的检查点为臂、腕、工具0和有效TCP，如上图所示。未立即完成减速。将系统输入信号LimitSpeed设置为1时，储存并应用各值。相对于基础坐标系，限制检查点的速度。

如果指令SpeedLimCheckPoint不用于设置限制，则将手动模式的速度限制作为限制。如果检查点不需要任何限制，则应当输入一个高值。此外，如果指令SpeedLimAxis的使用未设置轴速度限制，则手动模式的速度限制将用于限制轴速度。

当系统输入信号LimitSpeed被设置成1时，该速度会减少成相应的已减速度。

当系统输入信号LimitSpeed被设置成0时，该速度会增大成当前移动指令采用的编程速度。

下一页继续

1.249 SpeedLimCheckPoint - 设置检查点的速度限制
续前页

当达到减速时，将系统输出信号LimitSpeed设置为1。当速度开始增加时，将系统输出信号LimitSpeed设置为0。

自动设置速度限制的默认值

- 当使用重启模式重置RAPID时
- 当加载一则新程序或一个新模块时
- 当从起点开始执行程序时
- 当将程序指针移动到main时
- 当将程序指针移动到子程序时
- 移动程序指针造成执行顺序丢失时

更多示例

有关于指令SpeedLimCheckPoint的更多例子阐述如下。

例 1

```
..
VAR intnum sigint1;
VAR intnum sigint2;
..
PROC main()
  ! Setup interrupts reacting on a signal input
  IDelete sigint1;
  CONNECT sigint1 WITH setlimitspeed;
  ISignalDI \SingleSafe, mysensorsignal, 1, sigint1;
  IDelete sigint2;
  CONNECT sigint2 WITH resetlimitspeed;
  ISignalDI \SingleSafe, mysensorsignal, 0, sigint2;
  ..
  MoveL p1, z50, fine, tool2;
  MoveL p2, z50, fine, tool2;
  ..
  MoveL p10, v100, fine, tool2;
  ! Set limitations for checkpoints and axes
  SpeedLimCheckPoint 200;
  SpeedLimAxis ROB_1, 1, 10;
  SpeedLimAxis ROB_1, 2, 10;
  SpeedLimAxis ROB_1, 3, 10;
  SpeedLimAxis ROB_1, 4, 20;
  SpeedLimAxis ROB_1, 5, 20;
  SpeedLimAxis ROB_1, 6, 20;
  WHILE run_loop = TRUE DO
    MoveL p1, vmax, z50, tool2;
    ..
    MoveL p99, vmax, fine, tool2;
  ENDWHILE
  ! Set the default manual mode max speed
  SpeedLimCheckPoint 0;
  SpeedLimAxis ROB_1, 1, 0;
  SpeedLimAxis ROB_1, 2, 0;
  SpeedLimAxis ROB_1, 3, 0;
```

下一页继续

```
  SpeedLimAxis ROB_1, 4, 0;
  SpeedLimAxis ROB_1, 5, 0;
  SpeedLimAxis ROB_1, 6, 0;
  ..
TRAP setlimitspeed
  IDelete sigint1;
  CONNECT sigint1 WITH setlimitspeed;
  ISignalDI \SingleSafe, mysensorsignal, 1, sig1int1;
  ! Set out signal that is cross connected to system input
       LimitSpeed
  SetDO do1LimitSpeed, 1;
ENDTRAP
TRAP resetlimitspeed
  IDelete sigint2;
  CONNECT sigint2 WITH resetlimitspeed;
  ISignalDI \SingleSafe, mysensorsignal, 0, sig1int2;
  ! Reset out signal that is cross connected to system input
       LimitSpeed
  SetDO do1LimitSpeed, 0;
ENDTRAP
```

在机械臂从位置p1移动至p10期间，使用默认速度限制（手动模式速度）。为TCP机械臂和轴的检查点增加新的速度限制。如果信号mysensorsignal的值改变为1时，则TRAP setlimitspeed将应用速度限制。

当信号mysensorsignal的值改变为0时，TRAP resetlimitspeed将移除速度限制。

只要变量run_loop为TRUE，随即将使用速度限制的新设置，并将系统输入信号LimitSpeed设置为1。将run_loop设置为FALSE时，设置默认速度限制（手动模式速度）。

注意

本例子中的TRAP程序仅用于使功能形象化。用于限制速度的信号亦可直接与系统输入信号LimitSpeed相连，或通过安全PLC来相连。

错误处理

系统会生成下列可恢复错误，并在错误处理器中处理这些错误。系统变量ERRNO将被设置成：

ERR_SPEEDLIM_VALUE	参数RobSpeed中使用的速度过低。

限制

SpeedLimCheckPoint不能用于通电（POWER ON）事件程序。

如果机械臂位于运动轨迹上，则世界坐标系中的检查点速度会高于基准坐标系中指定检查点的速度限制。世界坐标系中的检查点速度为基准坐标系中的轨迹速度和检查点速度之和。为了同时限制世界坐标系中的检查点速度，确保其和不会超出限制。

当降低一个轴或检查点的速度时，其他轴亦将降低至相同的百分比，以便能够沿编程路径运行。沿编程路径的过程速度将会有所不同。

下一页继续

1.249 SpeedLimCheckPoint - 设置检查点的速度限制

续前页

当使用SafeMove连同速度限制时，必须设置SafeMove的裕量，并进行测试，因为SafeMove和运动计算略有不同。工具TCP的改变不与SafeMove同步。因此，SafeMove中的工具TCP必须比机械臂所用工具更短，或必须为SafeMove最大检查点速度设置额外裕量，并进行测试。

语法

```
SpeedLimCheckPoint
  [ RobSpeed ':=' ] < expression (IN) of num > ';'
```

相关信息

信息，关于	请参阅
定位器指令	技术参考手册 - *RAPID*语言概览
设置轴的速度限制	第*646*页的*SpeedLimAxis* - 设置轴的速度限制
定义手臂负载	技术参考手册 - 系统参数
系统输入输出信号	技术参考手册 - 系统参数

1.250 SpeedRefresh - 更新持续运动速度覆盖

手册用法

SpeedRefresh用于改变当前运动程序任务中进行中的机械臂移动的速度。通过该指令，可能创建关于某些传感器输入的某种粗速度适应。

本指令仅可用于主任务T_ROB1，或者如果在*MultiMove*系统中，则可用于任何运动任务中。

基本示例

以下实例介绍了指令SpeedRefresh：

例 1

```
VAR num change_speed:=70;
SpeedRefresh change_speed;
```

这将把当前速度覆盖改变为70%。

变元

```
SpeedRefresh Override
```

Override

数据类型：num

范围0 ... 100 %内的速度覆盖值。

程序执行

将更新当前运动程序任务中关于机械臂和外部单元进行中的移动的实际速度覆盖值。

当前运动任务中任意机械单元的所有速度数据分量均将受到影响。

> **注意**
>
> 根据SpeedRefresh设置的速度覆盖，不等于根据FlexPendant示教器而设置速度。这是两种不同的值。这两种值的结果以及编程的速度将成为移动中使用的速度。

如果完成PP到Main或已加载新的程序，则将重置通过SpeedRefresh设置的速度，并将应用根据FlexPendant示教器而设置的速度。

更多示例

有关于指令SpeedRefresh的更多例子阐述如下。

例 1

```
VAR intnum time_int;
VAR num override;
...
PROC main()
  CONNECT time_int WITH speed_refresh;
  ITimer 0.1, time_int;
  ISleep time_int;
  ...
  MoveL p1, v100, fine, tool2;
  ! Read current speed override set from FlexPendant
```

下一页继续

1.250 SpeedRefresh - 更新持续运动速度覆盖

RobotWare - OS

续前页

```
  override := CSpeedOverride (\CTask);
  IWatch time_int;
  MoveL p2, v100, fine, tool2;
  IDelete time_int;
  ! Reset to FlexPendant old speed override
  WaitTime 0.5;
  SpeedRefresh override;
  ...
TRAP speed_refresh
  VAR speed_corr;
  ! Analog input signal value from sensor, value 0 ... 10
  speed_corr := (ai_sensor * 10);
  SpeedRefresh speed_corr;
  ERROR
    IF ERRNO = ERR_SPEED_REFRESH_LIM THEN
      IF speed_corr > 100 speed_corr := 100;
      IF speed_corr < 0 speed_corr := 0;
      RETRY;
    ENDIF
ENDTRAP
```

在机械臂从位置p1移动到 p2期间，在TRAP speed_refresh中，每0.1 s更新一次速度覆盖值。模拟信号输入信号ai_sensor用于计算指令SpeedRefresh的Overide值。在机械臂于p1和p2之间移动前后，未执行TRAP。保存根据FlexPendant示教器进行的手动速度覆盖。在此之后，机械臂必须达到p2。

错误处理

如果Override拥有0到100 %范围以外的值，则将ERRNO变量设置为ERR_SPEED_REFRESH_LIM。该错误可恢复，并可通过ERROR处理器进行处理。

限制

注意，通过SpeedRefresh，速度覆盖将不会在短暂时间内完成。下达命令与对物理机械臂产生影响之间将存在0.3 - 0.5秒的滞后。

在SpeedRefresh序列后，用户负责根据RAPID程序，重置速度覆盖值。

如果SpeedRefresh用于START或RESET事件程序，则设置的速度始终为实际的FlexPendant示教器速度覆盖。

语法

```
SpeedRefresh
  [ Override ':=' ] < expression (IN) of num > ';'
```

相关信息

信息，关于	请参阅
定位器指令	技术参考手册 - *RAPID*语言概览
速度的定义	第*1480*页的*speeddata - 速度数据*
读取当前的速度覆盖	第*1038*页的*CSpeedOverride - 读取当前的覆盖速度*

1.251 SpyStart - 开始记录执行时间数据

手册用法

SpyStart用于开始记录执行期间的指令和时间数据。

将执行数据储存在文件中，以供随后分析。

储存的数据旨在排除RAPID程序故障，尤其适用于多任务系统（仅必须在一个程序任务中配备SpyStart - SpyStop）。

基本示例

以下实例介绍了指令SpyStart：

例 1

```
SpyStart "HOME:/spy.log";
```

开始在HOME上的文件spy.log中记录执行时间数据：磁盘。

变元

```
SpyStart File
```

文件

数据类型：string

将包含执行数据的文件的路径和名称。

程序执行

打开指定文件以供写入，并开始在文件中记录执行时间数据。

记录执行时间数据，直至：

- 执行指令SpyStop
- 从起点开始程序执行
- 加载新程序
- 下一次重启

SpyStart将受自动条件重置的影响

限制

避免使用软盘（选项）来进行记录，因为软盘的写入非常耗时。

禁止在生产程序中使用间谍功能，因为该功能会增加循环时间，并消耗在用大容量存储设备的内存。

错误处理

如果无法打开SpyStart指令中的文件，则将系统变量ERRNO设置为ERR_FILEOPEN（参见“数据类型-errnum”）。随后，可通过错误处理器来处理该错误。

文件格式

任务	指令	进	代码	出
MAIN	FOR i FROM 1 TO 3 DO	0	就绪	0
MAIN	mynum:=mynum+i;	1	就绪	1
MAIN	ENDFOR	2	就绪	2

下一页继续

1.251 SpyStart - 开始记录执行时间数据

RobotWare - OS

续前页

任务	指令	进	代码	出
MAIN	mynum:=mynum+i;	2	就绪	2
MAIN	ENDFOR	2	就绪	2
MAIN	mynum:=mynum+i;	2	就绪	2
MAIN	ENDFOR	2	就绪	3
MAIN	SetDo1,1;	3	就绪	3
MAIN	IF di1=0 THEN	3	就绪	4
MAIN	MoveL p1, v1000, fine, tool0;	4	等待	14
MAIN	MoveL p1, v1000, fine, tool0;	111	就绪	111
MAIN	ENDIF	108	就绪	108
MAIN	MoveL p2, v1000, fine, tool0;	111	等待	118
MAIN	MoveL p2, v1000, fine, tool0;	326	就绪	326
MAIN	SpyStop;	326	就绪	

“任务”一栏表明了执行的程序任务。

“指令”一栏表明了在指定程序任务中执行的指令。

“进”一栏表明了输入已执行指令的时间，以ms计。

“代码”一栏表明了在“出”完成时，指令的“就绪”或“等待”状态。

“出”一栏表明了离开已执行指令的时间，以ms计。

所有时间均以ms计（相对值）。

SYSTEM TRAP系指系统正在进行除执行RAPID指令以外的操作。

如果过程调用一些NOSTEPIN语句无返回值程序（模块），则输出清单将仅显示调用无返回值程序的名称。每当执行NOSTEPIN语句程序中的指令时，重复上述操作。

语法

```
SpyStart
  [File':=']<expression (IN) of string>';'
```

相关信息

信息，关于	请参阅
停止记录执行数据	第659页的*SpyStop - 停止记录时间执行数据*
自动条件	技术参考手册 - 系统参数

1.252 SpyStop - 停止记录时间执行数据

手册用法

SpyStop用于在执行期间，停止记录时间数据。

将可用于优化执行周期时间的数据保存在文件中，以供随后分析。

基本示例

以下实例介绍了指令SpyStop ：

另请参阅第*659*页的更多示例

例 1

```
SpyStop;
```

停止记录由先前SpyStart指令所指定文件中的执行时间数据。

程序执行

停止记录执行数据，并关闭由先前SpyStart指令所指定的文件。如果之前未执行任何SpyStart指令，则忽略SpyStop指令。

更多示例

有关于指令SpyStop的更多例子阐述如下。

例 1

```
IF debug = TRUE SpyStart "HOME:/spy.log";
produce_sheets;
IF debug = TRUE SpyStop;
```

如果调试标记真实，则开始记录HOME上文件spy.log中的执行数据：磁盘。执行实际生产；停止记录，并关闭文件spy.log。

限制

避免使用软盘（选项）来进行记录，因为软盘的写入非常耗时。

禁止在生产程序中使用间谍功能，因为该功能会增加循环时间，并消耗在用大容量存储设备的内存。

语法

```
SpyStop’;’
```

相关信息

信息，关于	请参阅
开始记录执行数据	第*657*页的*SpyStart - 开始记录执行时间数据*

1.253 StartLoad - 执行期间，加载普通程序模块

手册用法

StartLoad用于在执行期间，开始将普通程序模块加载到程序内存中。

当正在进行加载时，可同时执行其他指令。在可以使用加载模块的符号/程序之前，必须通过指令WaitLoad，将加载模块与程序任务相连。

将加载普通程序模块添加至程序内存中已经存在的模块中。

可通过静力学（默认）或动力学模式来加载程序或系统程序模块。根据使用的模式，一些操作将卸载模块，或不会对模块造成影响。

静态模式

下表显示了两种不同的操作如何影响静力学加载程序或系统程序模块。

	从TP设置PP到Main。	打开新的RAPID程序
普通程序模块	未受影响	卸载
系统程序模块	未受影响	未受影响

动态模型

下表显示了两种不同的操作如何影响动力学加载程序或系统程序模块。

	从TP设置PP到Main。	打开新的RAPID程序
普通程序模块	卸载	卸载
系统程序模块	卸载	卸载

通过指令UnLoad，可卸载静态和动态加载的模块。

基本示例

以下实例介绍了指令StartLoad：

另请参阅第*661*页的更多示例

例 1

```
VAR loadsession load1;

! Start loading of new program module PART_B containing routine
     routine_b in dynamic mode
StartLoad \Dynamic, diskhome \File:="PART_B.MOD", load1;

! Executing in parallel in old module PART_A containing routine_a
%"routine_a"%;

! Unload of old program module PART_A
UnLoad diskhome \File:="PART_A.MOD";
! Wait until loading and linking of new program module PART_B is
     ready
WaitLoad load1;

! Execution in new program module PART_B
%"routine_b"%;
```

下一页继续

通过指令StartLoad，开始将来自diskhome的程序模块PART_B.MOD加载到程序内存中。在加载的同时，通过模块 PART_A.MOD，程序执行routine_a 。随后，指令WaitLoad进行等待，直至完成加载和链接。以动力学模式加载模块。

变量load1保存由StartLoad更新以及由WaitLoad参考的加载会话的识别号。

为保存链接时间，可通过使用选项参数\UnLoadPath.，将指令UnLoad和WaitLoad结合成指令WaitLoad

变元

```
StartLoad [\Dynamic] FilePath [\File] LoadNo
```

[\Dynamic]

数据类型：switch

开关以动力学模式启用普通程序模块加载。否则，以静力学模式进行加载。

FilePath

数据类型：string

将文件路径和文件名称加载到程序内存中。当使用参数\File 时，应当排除文件名称。

[\File]

数据类型：string

当在参数FilePath中排除文件名称时，则必须通过该参数来进行定义。

LoadNo

数据类型：loadsession

此为应当在指令WaitLoad中使用的加载会话的引用，从而将已加载的普通程序模块与程序任务相连。

程序执行

StartLoad的执行将仅下达加载指令，随后，直接继续下一指令，无需等待完成加载。

指令WaitLoad将首先等待完成加载，随后其将进行链接和初始化。已加载模块的初始化将所有模块等级的变量设置为其初始值。

针对加载操作StartLoad - WaitLoad，将默认接受未解决的参考，但是，在执行未解决的参考时，其将成为运行时错误。

为取得易于理解和维持的良好程序结构，应当从主模块完成普通程序模块的所有加载和卸载，且执行期间，其始终存在于程序内存中。

关于包含主程序 main无返回值程序（以及另一main无返回值程序）的程序的加载，请参见指令Load，第*312*页的*Load - 执行期间，加载普通程序模块*。

更多示例

有关于如何使用指令StartLoad的更多例子阐述如下。

例 1

```
StartLoad \Dynamic, "HOME:/DOORDIR/DOOR1.MOD", load1;
```

将来自路径DOORDIR中的HOME: 的普通程序模块DOOR1.MOD加载到程序内存中。以动态模式来加载普通程序模块。

下一页继续

1.253 StartLoad - 执行期间，加载普通程序模块

RobotWare - OS

续前页

例 2

```
StartLoad \Dynamic, "HOME:" \File:="/DOORDIR/DOOR1.MOD", load1;
```

与例1相同，但是采用另一种语法。

例 3

```
StartLoad "HOME:" \File:="/DOORDIR/DOOR1.MOD", load1;
```

与上文例1和例2相同，但是，以静力学模式来加载模块。

例 4

```
StartLoad \Dynamic, "HOME:" \File:="/DOORDIR/DOOR1.MOD", load1;
WaitLoad load1;
```

与以下相同

```
Load \Dynamic, "HOME:" \File:="/DOORDIR/DOOR1.MOD";
```

错误处理

如果不能找到指令中的指定文件，则将系统变量ERRNO设置为ERR_FILNOTFND。随后，可用错误处理器对该错误进行处理。

如果参数LoadNo 中指定的变量已在使用，则将系统变量ERRNO设置为ERR_LOADNO_INUSE。随后，可以用错误处理器来处理该错误。

语法

```
StartLoad
  [´\´Dynamic ´,´]
  [FilePath’ :=’] <expression (IN) of string>
  [’\’File ’:=’ <expression (IN) of string> ] ’,’
  [LoadNo ’:=’] <variable (VAR) of loadsession>’;’
```

相关信息

信息，关于	请参阅
将加载的模块与任务相连	第*881*页的*WaitLoad - 将加载的模块与任务相连*
加载会话	第*1427*页的*loadsession - 程序加载会话*
加载普通程序模块	第*312*页的*Load - 执行期间，加载普通程序模块*
卸载普通程序模块	第*843*页的*UnLoad - 执行期间，卸载普通程序模块*
取消普通程序模块的加载	第*63*页的*CancelLoad - 取消模块加载*
通过后期绑定进行过程调用	技术参考手册 - *RAPID*语言概览

1.254 StartMove - 重启机械臂移动

手册用法

在停止移动之后，StartMove用于恢复机械臂、外轴移动和随附过程。

- 通过指令StopMove。
- 在执行StorePath ... RestoPath序列之后。
- 在出现以异步方式引起的移动错误之后，例如，在通过ERROR处理器处理之后的ERR_PATH_STOP或特定过程错误。

针对基座系统，可能在以下类别的程序任务中使用该指令：

- 关于重启该任务中的移动的主任务T_ROB1。
- 关于重启主任务中移动的所有其他任务。

针对MultiMove系统，可能在以下类别的程序任务中使用该指令：

- 运动任务，用于重启该任务中的移动。
- 非运动任务，用于重启相关运动任务中的移动。此外，如果在属于协调同步任务组的一个相关运动任务中重启移动，便会在所有合作任务中重启移动。

基本示例

以下实例介绍了指令StartMove：

例 1

```
StopMove;
WaitDI ready_input,1;
StartMove;
```

设置输入ready_input时，机械臂再次开始移动。

例 2

```
...
MoveL p100, v100, z10, tool1;
StorePath;
p:= CRobT(\Tool:=tool1);
! New temporary movement
MoveL p1, v100, fine, tool1;
...
MoveL p, v100, fine, tool1;
RestoPath;
StartMove;
...
```

在返回停止位置p（在本例子中，相当于p100）后，机械臂再次开始以基础路径等级移动。

变元

```
StartMove [\AllMotionTasks]
```

[\AllMotionTasks]

数据类型：switch

重启系统中所有机械单元的移动。仅可在非运动程序任务中使用开关[\AllMotionTasks]。

下一页继续

1.254 StartMove - 重启机械臂移动
续前页

程序执行

同时重启与停止移动相关的所有过程，以便恢复运动。

为通过同步协调模式来重启MultiMove应用，必须在涉及协调的所有运动任务中执行StartMove。

通过开关\AllMotionTasks（仅允许非运动程序任务使用），重启关于系统中所有机械单元的移动。

在没有开关\AllMotionTasks的基座系统中，重启以下机械单元的移动：

- 始终为主任务中的机械单元，不依赖于执行StartMove指令的任务。

在没有开关\AllMotionTasks的MultiMove系统中，重启关于以下机械单元的移动：

- 执行StartMove的运动任务中的机械单元。
- 与执行StartMove的非运动任务相关的运动任务中的机械单元。此外，如果在属于协调同步任务组的一个相关运动任务中重启机械单元，则在所有的合作任务中重启机械单元。

错误处理

如果机械臂过于远离路径（超过10 mm或20度），以致无法重启中断的运动，则将系统变量ERRNO设置为ERR_PATHDIST。

如果在执行StartMove时，机械臂处于停止状态，则将系统变量ERRNO设置为ERR_STARTMOVE。如果定时正确，则紧急停止亦可引起ERR_STARTMOVE。

当恢复StartMove的路径移动时，如果多次停止程序执行，则将系统变量ERRNO设置为ERR_PROGSTOP

如果在执行StartMove时，机械臂正在移动，则将系统变量ERRNO设置为ERR_ALRDY_MOVING。

随后，可用错误处理器来处理此类错误：

- ERR_PATHDIST时，在试图RETRY之前，使机械臂更接近路径。
- ERR_STARTMOVE、ERR_PROGSTOP或ERR_ALRDY_MOVING时，在试图RETRY之前，等待一段时间。

限制

在进行StopMove - 针对某些运动任务的StartMove序列时，仅允许进行若干非运动任务之一。

如果用任意错误处理器来执行StartMove，则不可能进行任何错误恢复。

语法

```
StartMove
  ['\'AllMotionTasks]';'
```

相关信息

信息，关于	请参阅
停止移动	第*690*页的*StopMove - 停止机械臂的移动*
继续移动	第*665*页的*StartMoveRetry - 重启机械臂移动和执行*
更多示例	第*695*页的*StorePath - 发生中断时，存储路径* 第*512*页的*RestoPath - 中断之后，恢复路径*

1.255 StartMoveRetry - 重启机械臂移动和执行

手册用法

StartMoveRetry用于恢复机械臂和外轴移动以及随附过程，同时通过ERROR处理器重新执行。

可在以下类型的程序任务中，通过ERROR处理器来使用该指令：

- 基座系统中的主任务T_ROB1
- *MultiMove*系统中的任意运动任务

基本示例

以下实例介绍了指令StartMoveRetry：

例 1

```
VAR robtarget p_err;
...
MoveL p1\ID:=50, v1000, z30, tool1 \WObj:=stn1;
...
ERROR
  IF ERRNO = ERR_PATH_STOP THEN
    StorePath;
    p_err := CRobT(\Tool:= tool1 \WObj:=wobj0);
    ! Fix the problem
    MoveL p_err, v100, fine, tool1;
    RestoPath;
    StartMoveRetry;
  ENDIF
ENDPROC
```

此为关于MultiMove系统以及协调同步移动（在某种旋转工件上工作的两个机械臂）的实例。

在移动至位置p1期间，其他协作的机械臂出现了一些过程错误，以致协调同步移动停止。随后，该机械臂出现错误ERR_PATH_STOP，并转移到ERROR处理器执行。

在ERROR处理器中，进行以下操作：

- StorePath保存原始路径，转至新的路径等级，并以独立模式设置MultiMove系统。
- 如果机械臂存在问题，则开始在新路径等级上移动。
- 在RestoPath之前，返回错误位置。
- RestoPath返回至原始路径等级，并再次将MultiMove系统设置为同步模式。
- StartMoveRetry重启中断后的移动以及任意过程。同时转移执行，以恢复正常的执行。

程序执行

StartMoveRetry按以下顺序进行：

- 恢复路径
- 重启与停止的移动相关的所有过程
- 重启中断的运动

下一页继续

1.255 StartMoveRetry - 重启机械臂移动和执行
RobotWare - OS
续前页

- 程序执行的RETRY

StartMoveRetry与不可分割操作中的StartMove和RETRY相同。

仅重启执行StartMoveRetry的程序任务中的机械单元。

限制

仅可用于运动任务中的ERROR处理器。

在执行协调同步移动的MultiMove系统中，必须在ERROR处理器中遵循以下编程规则：

- 必须将StartMoveRetry用于所有协作程序任务。
- 如果任意ERROR处理器需要移动，则必须在所有协作程序任务中使用指令StorePath ... RestoPath。
- 如果在StorePath等级上移动机械臂，则在执行RestoPath之前，本程序必须使机械臂返回错误位置。

错误处理

如果机械臂过于远离路径（超过10 mm或20度），以致无法重启中断的运动，则将系统变量ERRNO设置为ERR_PATHDIST。

如果在执行StartMoveRetry时，机械臂处于停止状态，则将系统变量ERRNO设置为ERR_STARTMOVE。

在恢复StartMoveRetry的路径移动期间，如果多次停止程序执行，则将系统变量ERRNO设置为ERR_PROGSTOP.

如果在执行StartMoveRetry时，机械臂正在移动，则将系统变量ERRNO设置为ERR_ALRDY_MOVING。

不可能从此类错误进行任何错误恢复，因为仅可在一些错误处理器中执行StartMoveRetry。

语法

```
StartMoveRetry ';'
```

相关信息

信息，关于	请参阅
停止移动	第*690*页的*StopMove - 停止机械臂的移动*
继续移动	第*663*页的*StartMove - 重启机械臂移动*
在错误后恢复执行	第*514*页的*RETRY - 在错误后恢复执行*
储存/恢复路径	第*695*页的*StorePath - 发生中断时，存储路径* 第*512*页的*RestoPath - 中断之后，恢复路径*

1.256 STCalib - 校准伺服工具

手册用法

STCalib用于校准工具焊嘴之间的距离。在更换焊嘴或工具后，必须进行上述操作，且建议在实施焊枪头塑型后或使用工具一段时间后进行上述操作。

注意！在校准期间，进行两次启闭移动的工具。第一次关闭移动将探测焊嘴接触位置。

基本示例

以下实例介绍了指令STCalib：

例 1

```
VAR num curr_tip_wear;
VAR num retval;
CONST num max_adjustment := 20;

STCalib gun1 \ToolChg;
```

在更换工具后，校准伺服焊枪。在继续下一RAPID指令之前，等待完成焊枪校准。

例 2

```
STCalib gun1 \ToolChg \Conc;
```

在更换工具后，校准伺服焊枪。继续下一RAPID指令，无需等待完成焊枪校准。

例 3

```
STCalib gun1 \TipChg;
```

更换焊嘴后，校准伺服焊枪。

例 4

```
STCalib gun1 \TipWear \RetTipWear := curr_tip_wear;
```

焊枪头磨损后，校准伺服焊枪。在变量curr_tip_wear中保存焊枪头磨损。

例 5

```
STCalib gun1 \TipChg \RetPosAdj:=retval;
IF retval > max_adjustment THEN
TPWrite "The tips are lost!";
...
```

更换焊嘴后，校准伺服焊枪。检查焊嘴是否丢失。

例 6

```
STCalib gun1 \TipChg \PrePos:=10;
```

更换焊嘴后，校准伺服焊枪。快速移动至位置10 mm，随后，开始以较低的速度搜索接触位置。

例 7

无效组合实例：

```
STCalib gun1 \TipWear \RetTipWear := curr_tip_wear \Conc;
```

实施焊枪头磨损校准。无需等待完成焊枪校准，继续进行下一RAPID指令。在这种情况下，参数curr_tip_wear将不会保持任何有效值，因为使用了\Conc开关（在开完成校准过程之前，将开始执行下一RAPID指令）。

下一页继续

1.256 STCalib - 校准伺服工具

Servo Tool Control

续前页

变元

```
STCalib ToolName [\ToolChg] | [\TipChg] | [\TipWear] [\RetTipWear]
    [\RetPosAdj] [\PrePos] [\Conc]
```

`ToolName`

数据类型：`string`

机械单元名称

`[\ToolChg]`

数据类型：`switch`

更换工具之后的校准。

`[\TipChg]`

数据类型：`switch`

更换焊嘴之后的校准。

`[\TipWear]`

数据类型：`switch`

焊枪头磨损之后的校准。

`[\RetTipWear]`

数据类型：`num`

造成的焊枪头磨损[mm]。

`[\RetPosAdj]`

数据类型：`num`

自最后一次校准起的位置调整[mm]。

`[\PrePos]`

数据类型：`num`

在以较慢速度开始搜索接触位置之前，该位置以较高速度移动[mm]。

`[\Conc]`

数据类型：`switch`

当焊枪移动时，执行后续指令。本参数可用于缩短周期时间。当同时控制两个焊枪时，上述操作有用。

程序执行

校准模式

如果存在机械单元，则下达校准伺服工具的命令。根据开关来完成校准，参见下文。如果使用RetTipWear参数，则更新焊枪头磨损。

更换工具之后的校准：

本工具将以较慢的速度关闭，以等待相接触焊嘴快速启闭，以获得较低的压力，并在一个序列中再次打开。焊枪头磨损将保持不变。

更换焊嘴之后的校准：

本工具将以较慢的速度关闭，以等待相接触焊嘴快速启闭，以获得较低的压力，并在一个序列中再次打开。焊枪头磨损将得以重置。

焊枪头磨损之后的校准：

下一页继续

本工具将以较快的速度关闭至接触位置，快速启闭以获得较低的压力，并在一个序列中再次打开。焊枪头磨损将得以更新。

注意！如果使用开关Conc，则在本指令起动时，便认为其就绪，因此，返回值RetTipWear将不可用。在这种情况下，将通过函数STIsOpen来返回RetTipWear。有关更多的细节，请参见RobotWareOS函数 - STIsOpen。

位置调整

例如，如果在更换焊嘴后，焊嘴丢失，则可使用可选参数RetPosAdj来进行探测。本参数将保存自最后一次校准起的位置调整值。该值可为正值或负值。

使用预设位置

为加快校准，可能定义预设位置。开始校准时，焊枪臂将快速运行至预设位置，停止，随后继续缓慢地*)向前，以探测焊嘴接触位置。如果使用预设位置，则小心地进行选择！重要的是并未接触焊嘴，直至达到预设位置之后！否则，校准精度将变差，并可能出现运动监督错误。如果预设位置大于当前焊枪位置，则将忽略预设位置（从而不会放慢校准速度）。

*)如果使用\TipWear选项，则第二次移动亦将较为快速。

错误处理

如果指定伺服工具名称并非配置伺服工具，则将系统变量ERRNO设置为ERR_NO_SGUN。

如果在调用STCalib时未打开焊枪，则将系统变量ERRNO设置为ERR_SGUN_NOTOPEN。

如果未启用伺服工具机械单元，则将系统变量ERRNO设置为ERR_SGUN_NOTACT。使用指令ActUnit，以启用伺服工具。

如果未初始化伺服工具位置，则将系统变量ERRNO设置为ERR_SGUN_NOTINIT。在首次安装焊枪或完成精细校准之后，必须初始化伺服工具位置。使用服务程序ManServiceCalib或实施焊嘴更换校准。将重置焊枪头磨损。

如果伺服工具焊嘴未同步，则将系统变量ERRNO设置为ERR_SGUN_NOTSYNC。如果已丢失和/或更新转数计数器，则必须使伺服工具焊嘴同步。将不会丢失诸如焊枪头磨损等过程数据。

如果从背景任务调用指令，且存在紧急停止，则将完成本指令，并将系统变量ERRNO设置为ERR_SGUN_ESTOP。注意，如果从主任务调用本指令，则将在本指令下停止程序指针，并将从程序重启起点来重启本指令。

如果将参数PrePos指定为小于零的值，则将系统变量ERRNO设置为ERR_SGUN_NEGVAL。

如果从背景任务调用本指令，且本系统处于电机关闭状态，则将系统变量ERRNO设置为ERR_SGUN_MOTOFF。

可用RAPID错误处理器来处理所有上述错误。

语法

```
STCalib
  [ 'ToolName' :=' ] < expression (IN) of string > ',
  [ '\'ToolChg] | ['\'TipChg] | [ '\'TipWear]
  [' \'RetTipWear' :=' < variable or persistent(INOUT) of num >
       ]';'
  [ '\'RetPosAdj' :=' < variable or persistent(INOUT) of num > ]';'
  [ '\'PrePos' :=' < expression (IN) of num > ]'
  [ '\'Conc' ];'
```

下一页继续

1.256 STCalib - 校准伺服工具
Servo Tool Control
续前页

相关信息

信息，关于	请参阅
打开伺服工具	第*688*页的*STOpen - 打开伺服工具*
关闭伺服工具	第*671*页的*STClose - 关闭伺服工具*

1.257 STClose - 关闭伺服工具

手册用法

STClose用于关闭伺服工具。

基本示例

以下实例介绍了指令STClose：

例 1

```
VAR num curr_thickness1;
VAR num curr_thickness2;

STClose gun1, 1000, 5;
```

关闭焊枪头压力为1000 N，且板厚5 mm的伺服焊枪。等待，直至焊枪在继续下一RAPID指令之前关闭。

例 2

```
STClose gun1, 2000, 3\RetThickness:=curr_thickness;
```

关闭焊枪头压力为2000 N，且板厚3 mm的伺服焊枪。获取在变量curr_thickness中测得的厚度。

例 3

并行模式：

```
STClose gun1, 1000, 5 \Conc;
STClose gun2, 2000, 3 \Conc;
```

关闭焊枪头压力为1000 N，且板厚5 mm的伺服gun1。无需等待关闭gun1，继续程序执行，并关闭焊枪头压力为2000N，且板厚3 mm的伺服gun2。继续执行Rapid程序，无需等待gun2关闭。

例 4

```
IF STIsClosed (gun1)\RetThickness:=curr_thickness1 THEN
  IF curr_thickness1 < 0.2 Set weld_start1;
ENDIF
IF STIsClosed (gun2)\RetThickness:=curr_thickness2 THEN
  IF curr_thickness2 < 0.2 Set weld_start2;
ENDIF
```

获取在函数STIsClosed变量curr_thickness1和curr_thickness2中测得的厚度。

例 5

无效组合实例：

```
STClose gun1, 2000, 3\RetThickness:=curr_thickness \Conc;
```

关闭伺服焊枪，并继续RAPID程序执行。在这种情况下，由于使用了\Conc开关，参数curr_thickness将不会保存任何有效值（将在关闭焊枪之前，开始执行下一RAPID指令）。

变元

```
STClose ToolName TipForce Thickness [\RetThickness][\Conc]
```

下一页继续

1.257 STClose - 关闭伺服工具

Servo Tool Control

续前页

`ToolName`

数据类型：`string`

机械单元名称

`TipForce`

数据类型：`num`

所需焊枪头压力[N]。

`Thickness`

数据类型：`num`

关于伺服工具的预计接触位置[mm]。

`[\RetThickness]`

数据类型：`num`

如果未使用`\Conc`开关，则获得的厚度[mm]为唯一值。

`[\Conc]`

数据类型：`switch`

当焊枪移动时，执行后续指令。本参数可用于缩短周期时间。当同时控制两个焊枪时，上述操作有用。

程序执行

如果存在机械单元，则预订伺服工具，以关闭至所需厚度和压力。

关闭过程将开始使工具臂移动至所需接触位置（厚度）。在该位置停止移动，并完成从位置控制模式到力控制模式的转换。

按照针对相关外轴的系统参数规定，以最大速度和加速度移动工具臂。针对其他轴的移动，以手动模式降低速度。

如果指定可选参数`RetThickness`，则当实现所需焊枪头压力时，本指令就绪，并返回所实现的厚度。

注意！如果使用开关`Conc`，则在本指令起动时，便认为其就绪，因此，返回值`RetThickness`将不可用。在这种情况下，将通过函数`STIsClosed`来返回`RetThickness`。有关更多的细节，请参见**RobotWare OS函数 - `STIsClosed`**。

在编程机械臂移动期间，只要机械臂移动不包括工具臂移动，则有可能关闭工具。

欲知更多细节，请参见伺服工具运动控制。

错误处理

如果指定伺服工具名称并非配置伺服工具，则将系统变量`ERRNO`设置为`ERR_NO_SGUN`。

如果在调用`STClose`时未打开焊枪，则将系统变量`ERRNO`设置为`ERR_SGUN_NOTOPEN`。

如果未启用伺服工具机械单元，则将系统变量`ERRNO`设置为`ERR_SGUN_NOTACT`。使用指令`ActUnit`，以启用伺服工具。

如果未初始化伺服工具位置，则将系统变量`ERRNO`设置为`ERR_SGUN_NOTINIT`。在首次安装焊枪或完成精细校准之后，必须初始化伺服工具位置。使用服务程序`ManServiceCalib`或实施焊嘴更换校准。将重置焊枪头磨损。

下一页继续

如果伺服工具焊嘴未同步，则将系统变量ERRNO设置为ERR_SGUN_NOTSYNC。如果已丢失和/或更新转数计数器，则必须使伺服工具焊嘴同步。将不会丢失诸如焊枪头磨损等过程数据。

如果从背景任务调用指令，且如果存在紧急停止，则将完成本指令，并将系统变量ERRNO设置为ERR_SGUN_ESTOP。注意，如果从主任务调用本指令，则将在本指令下停止程序指针，并将从程序重启起点来重启本指令。

如果从背景任务调用本指令，且如果本系统处于电机关闭状态，则将系统变量ERRNO设置为ERR_SGUN_MOTOFF。

可用Rapid错误处理器来处理所有上述错误。

语法

```
STClose
  [ 'ToolName ':=' ] < expression (IN) of string > ‘,’
  [ 'Tipforce' :=' ] < expression (IN) of num > ‘,’
  [ 'Thickness' :='] < expression (IN) of num > ]
  [‘\’ 'RetThickness' :=' < variable or persistent (INOUT) of num
       > ]
  ['\'Conc]
```

相关信息

信息，关于	请参阅
打开伺服工具	第688页的*STOpen - 打开伺服工具*

1.258 StepBwdPath - 在路径上向后移动一步

手册用法

StepBwdPath用于使TCP从一则RESTART事件程序开始沿机械臂路径向后移动。

由用户负责引入重启过程标记，因此，仅在过程重启而非所有程序重启时执行RESTART事件程序中的StepBwdPath。

本指令仅可用于主任务T_ROB1，或者如果在MultiMove系统中，则可用于运动任务中。

基本示例

以下实例介绍了指令StepBwdPath：

例 1

```
StepBwdPath 30, 1;
StepBwdPath 30, 1;
```

第一个指令向后移动30 mm。第二个指令进一步向后移动30 mm。

变元

```
StepBwdPath StepLength StepTime
```

StepLength

数据类型：num

指定在该步骤期间向后移动的距离，以毫米计。该参数必须为正值。

StepTime

数据类型：num

废弃该参数。将其设置为1。

程序执行

机械臂在其路径上向后移动指定的距离。在停止之前，反向路径完全相同。在快速停止或紧急停止的情况下，在完成恢复阶段之后，调用RESTART事件程序，因此，当执行该指令时，机械臂将已经返回其路径。

该移动的实际速度为关于移动命令的编程速度，但是限制为250 mm/s。

在MultiMove系统-同步协调移动中，以下属性有效：

- 所有相关机械单元同时向后移动并进行协调
- 在任意相关程序任务中执行的各StepBwdPath，会导致向后移动一步（无需任何StartMove）。
- 为重启和继续已中断的过程移动，必须在所有相关程序任务中执行指令StartMove

限制

在停止程序后，可能在具有以下限制的路径上向后移动：

- 将StepBwdPath移动限制在最后的精点以及通常为五的移动命令历史长度。

如果试图超越此类限制，则将调用ERRNO设置为ERR_BWDLIMIT的错误处理器。

下一页继续

语法

```
StepBwdPath
  [ StepLength':=' ] < expression (IN) of num >','
  [ StepTime ':=' ] < expression (IN) of num >';'
```

相关信息

信息，关于	请参阅
一般动作	技术参考手册 - *RAPID*语言概览
定位器指令	技术参考手册 - *RAPID*语言概览
Advanced RAPID	应用手册 - 控制器软件*IRC5*

1.259 STIndGun - 以独立模式设置焊枪

手册用法

STIndGun（*Servo Tool independent gun*）用于以独立模式设置焊枪，此后，将焊枪移动至指定独立位置。焊枪将保持独立模式，直至执行指令STIndGunReset。

独立模式期间，将焊枪控制与机械臂分离开来。可关闭、打开、校准焊枪或将其移动至新的独立位置，但是，其将不会跟随经协调的机械臂移动。

如果焊枪实施不依赖于机械臂任务的任务，例如，固定式焊枪的焊枪头塑型，则独立模式适用。

基本示例

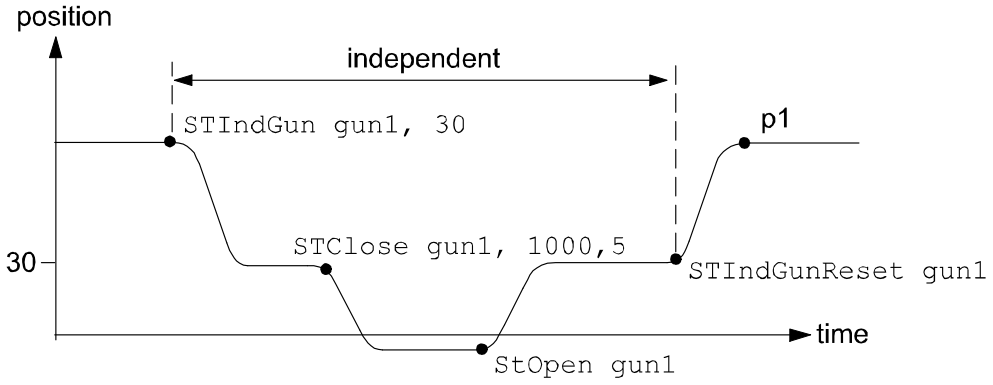
以下实例介绍了指令STIndGun：

例 1

当主任务中的机械臂可继续移动指令时，可在背景任务中运行该无返回值程序。

```
PROC tipdress()

  ! Note that the gun will move to current robtarget position, if
  ! already in independent mode.
  STIndGunReset gun1;
  ...
  STIndGun gun1, 30;
  StClose gun1, 1000, 5;
  WaitTime 10;
  STOpen gun1;
  ...
  STIndGunReset gun1;

ENDPROC
```

启用独立模式，并使焊枪移动至独立位置（30 mm）。在独立模式期间，执行指令StClose、WaitTime和STOpen，其不会妨碍机械臂运动。本指令StIndGunReset将使焊枪退出独立模式，并将焊枪移动至当前的机器人位置。



xx0500002342

位置p1取决于由机械臂刚刚处于的机器人位置中所给定焊枪的位置。

下一页继续

变元

```
STIndGun ToolName GunPos
```

ToolName

数据类型：string

机械单元名称

GunPos

数据类型：num

伺服焊枪的位置（冲程），以mm计。

语法

```
STIndGun
  [ ToolName ':=' ] < expression (IN) of string > ','
  [ GunPos ':=' < expression (IN) of num > ]';'
```

1.260 STIndGunReset - 以独立模式重置焊枪

手册用法

STIndGunReset（伺服工具独立焊枪重置）用于以独立模式重置焊枪，此后，将焊枪移动至当前机器人位置。

基本示例

以下实例介绍了指令STIndGunReset：

```
STIndGunReset gun1;
```

变元

```
STIndGunReset ToolName
```

ToolName

数据类型：string

机械单元名称

程序执行

本指令将以独立模式重置焊枪，并使焊枪移动至当前机器人位置。在该运动期间，焊枪协调速度必须为零，否则，将出现错误。如果机械臂保持不动，或如果当前机械臂移动包括焊枪的“零移动”，则协调速度将为零。

语法

```
STIndGunReset
  [ToolName ´:=´]<expression (IN) of string>´;´
```

1.261 SToolRotCalib - 关于固定工具的TCP和旋转的校准

手册用法

SToolRotCalib *(Stationary Tool Rotation Calibration)* 用于校准固定工具的TCP和旋转。

机械臂及其运动的位置始终与其工具坐标系相关，即TCP和工具方位。为获得最佳精度，重要的是尽可能正确地定义工具坐标系。

通过使用FlexPendant示教器的手动方法，亦可完成校准（在操作手册 - 带*FlexPendant*示教器的*IRC5*，编程与测试一节中有作描述）。

描述

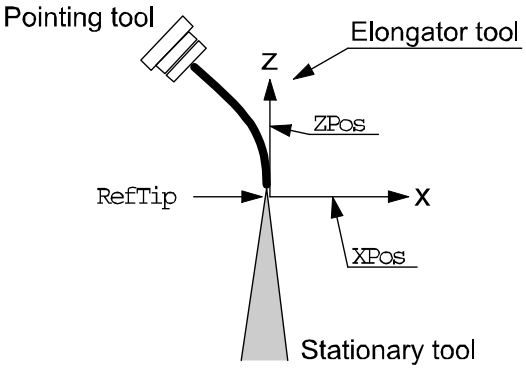
为定义固定工具的TCP和旋转，需要安装在机械臂末端执行器上的可移动指向工具。

在使用指令SToolRotCalib之前，必须满足一些先决条件：

- 必须安装有待校准的固定工具，并通过正确分量robhold（FALSE）来进行定义。
- 必须通过正确的TCP值来定义和校准指向工具（robhold TRUE）。
- 如果使用具有绝对精度的机械臂，则应当确定指向工具的负载和重心。LoadIdentify可用于负载定义。
- 在机械臂点动前，必须启用指向工具、wobj0和PDispOff 。
- 使尽可能靠近固定工具TCP（工具坐标系原点）的指向工具TCP点动，并定义参考点RefTip的robtarget。
- 在不改变工具方位的情况下，使机械臂点动，以便指向工具的TCP正指向工具坐标系正z轴上的某点，并定义关于点ZPos的robtarget。
- 在不改变工具方位的情况下，使机械臂点动，以便指向工具的TCP正指向工具坐标系正x轴上的某点，并定义关于点XPos的robtarget。

可使用某种类型的延伸器工具，来帮助指出正z轴和x轴。

关于机器人位置RefTip、ZPos和XPos的定义，请参见下图。



xx0500002343

注意

不建议修改指令SToolRotCalib中的位置RefTip、ZPos和XPos。

下一页继续

1.261 SToolRotCalib - 关于固定工具的TCP和旋转的校准

RobotWare - OS

续前页

基本示例

以下实例介绍了指令 SToolRotCalib：

例 1

```
! Created with pointing TCP pointing at the stationary tool
! coordinate system
CONST robtarget pos_tip := [...];
CONST robtarget pos_z := [...];
CONST robtarget pos_x := [...];

PERS tooldata tool1:= [ FALSE, [[0, 0, 0], [1, 0, 0 ,0]], [0, [0,
     0, 0], [1, 0, 0, 0], 0, 0, 0]];

!Instructions for creating or ModPos of pos_tip, pos_z and pos_x
MoveJ pos_tip, v10, fine, point_tool;
MoveJ pos_z, v10, fine, point_tool;
MoveJ pos_x, v10, fine, point_tool;

SToolRotCalib pos_tip, pos_z, pos_x, tool1;
```

计算和更新世界坐标系中TCP的位置（tframe.trans）以及tool1的工具方位（tframe.rot）。

变元

```
SToolRotCalib RefTip ZPos XPos Tool
```

RefTip

数据类型：robtarget

指向工具TCP指向待校准固定工具TCP的点。

ZPos

数据类型：robtarget

确定正z方向的延伸器点。

XPos

数据类型： robtarget

确定正x方向的延伸器点。

Tool

数据类型：tooldata

有待校准的工具的永久变量。

程序执行

本系统计算和更新指定 tooldata中的TCP（tframe.trans）和工具方位（tfame.rot）。本计算基于指定3robtarget。未改变 tooldata中的剩余数据。

语法

```
SToolRotCalib
  [ RefTip ':=' ] < expression (IN) of robtarget > ','
  [ ZPos ':=' ] < expression (IN) of robtarget > ','
  [ XPos ':=' ] < expression (IN) of robtarget > ','
```

下一页继续

```
[ Tool ':=' ] < persistent (PERS) of tooldata > ';'
```

相关信息

信息，关于	请参阅
关于移动工具的TCP的校准	第*412*页的*MToolTCPCalib - 关于移动工具的TCP的校准*
关于移动工具的旋转的校准	第*409*页的*MToolRotCalib - 移动工具旋转校准*
关于固定工具的TCP的校准	第*412*页的*MToolTCPCalib - 关于移动工具的TCP的校准*

1.262 SToolTCPCalib - 关于固定工具的TCP的校准

手册用法

`SToolTCPCalib` *(Stationary Tool TCP Calibration)* 用于校准固定工具的工具中心点 - TCP。

机械臂及其运动的位置始终与其工具坐标系相关，即TCP和工具方位。为获得最佳精度，重要的是尽可能正确地定义工具坐标系。

通过使用FlexPendant示教器的手动方法，亦可完成校准（在操作手册 - 带*FlexPendant*示教器的*IRC5*，编程与测试一节中有做描述）。

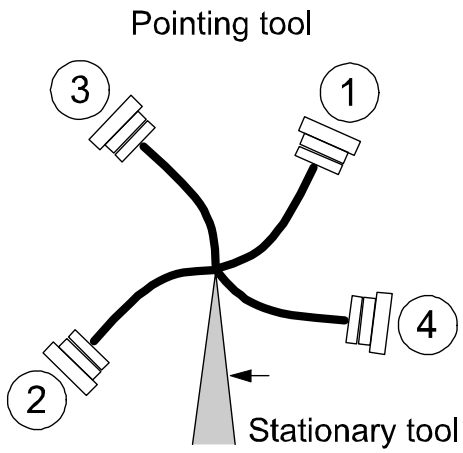
描述

为定义固定工具的TCP，需要安装在机械臂末端执行器上的可移动指向工具。

下列为使用指令`SToolTCPCalib`之前的先决条件：

- 必须安装有待校准的固定工具，并通过正确分量`robhold`（`FALSE`）来进行定义。
- 必须通过正确的TCP值来定义和校准指向工具（`robhold TRUE`）。
- 如果使用具有绝对精度的机械臂，则应当确定指向工具的负载和重心。`LoadIdentify`可用于负载定义。
- 在机械臂点动前，必须启用指向工具、`wobj0`和`PDispOff`。
- 使尽可能靠近固定工具TCP的指向工具TCP点动，并定义第一个点p1的`robtarget`。
- 进一步定义具有不同姿态的三个位置，p2、p3和p4。
- 建议TCP指向不同的方向，以获得可靠的统计结果。

关于4个机器人位置（p1...p4）的定义，请参见下图。



xx0500002344

注意

不建议修改指令`SToolTCPCalib`中的位置`Pos1`至`Pos4`。

4个位置之间的姿态调整应尽可能大，将机械臂置于不同的配置中。在校准后检查TCP的质量亦为一种良好的做法。可通过工具的方位调整来实施上述操作，以检查TCP是否静止不动。

下一页继续

基本示例

以下实例介绍了指令SToolTCPCalib：

例 1

```
! Created with pointing TCP pointing at the stationary TCP
CONST robtarget p1 := [...];
CONST robtarget p2 := [...];
CONST robtarget p3 := [...];
CONST robtarget p4 := [...];

PERS tooldata tool1:= [ FALSE, [[0, 0, 0], [1, 0, 0 ,0]], [0,001,
     [0, 0, 0.001], [1, 0, 0, 0], 0, 0, 0]];
VAR num max_err;
VAR num mean_err;
! Instructions for creating or ModPos of p1 - p4
MoveJ p1, v10, fine, point_tool;
MoveJ p2, v10, fine, point_tool;
MoveJ p3, v10, fine, point_tool;
MoveJ p4, v10, fine, point_tool;

SToolTCPCalib p1, p2, p3, p4, tool1, max_err, mean_err;
```

将校准和更新tool1的TCP值（tframe.trans）。max_err和mean_err将分别维持与计算出的TCP的最大误差以及平均误差，以mm计。

变元

```
SToolTCPCalib Pos1 Pos2 Pos3 Pos4 Tool MaxErr MeanErr
```

Pos1

数据类型：robtarget

第一个进刀点。

Pos2

数据类型：robtarget

第二个进刀点。

Pos3

数据类型：robtarget

第三个进刀点。

Pos4

数据类型：robtarget

第四个进刀点。

Tool

数据类型：tooldata

有待校准的工具的永久变量。

MaxErr

数据类型：num

一个进刀点的最大误差，以mm计。

下一页继续

1.262 SToolTCPCalib - 关于固定工具的TCP的校准

RobotWare - OS

续前页

`MeanErr`

数据类型：`num`

进刀点距离计算出的TCP的平均距离，即如何准确地定位相对于固定TCP的机械臂。

程序执行

本系统计算和更新指定 `tooldata`中世界坐标系（`tfame.trans`）中的TCP值。计算基于指定4`robtarget`。tooldata中的剩余数据，例如，工具方位（`tframe.rot`），不会发生改变。

语法

```
SToolTCPCalib
  [ Pos1 ':=' ] < expression (IN) of robtarget > ','
  [ Pos2 ':=' ] < expression (IN) of robtarget > ','
  [ Pos3 ':=' ] < expression (IN) of robtarget > ','
  [ Pos4 ':=' ] < expression (IN) of robtarget > ','
  [ Tool ':=' ] < persistent (PERS) of tooldata > ','
  [ MaxErr ':=' ] < variable (VAR) of num > ','
  [ MeanErr' :=' ] < variable (VAR) of num > ';'
```

相关信息

信息，关于	请参阅
关于移动工具的TCP的校准	第*682*页的*SToolTCPCalib - 关于固定工具的TCP的校准*
关于移动工具的旋转的校准	第*409*页的*MToolRotCalib - 移动工具旋转校准*
关于固定工具的TCP和旋转的校准	第*679*页的*SToolRotCalib - 关于固定工具的TCP和旋转的校准*

1.263 Stop - 停止程序执行

手册用法

Stop用于停止程序执行。在Stop指令就绪之前，将完成当前执行的所有移动。

基本示例

以下实例介绍了指令Stop：

另请参阅第*686*页的更多示例

例 1

```
TPWrite "The line to the host computer is broken";
Stop;
```

在将消息写入FlexPendant示教器之后，停止程序执行。

变元

```
Stop [ \NoRegain ] | [ \AllMoveTasks ]
```

[\NoRegain]

数据类型：switch

指定下一程序的起点，无论受影响的机械单元是否应当返回停止位置。

如果已设置参数\NoRegain，则机械臂和外轴将不会返回停止位置（如果他们已远离停止位置）。

如果省略该参数，且如果机械臂或外轴已从停止位置慢慢远离，则机械臂会在FlexPendant示教器上显示问题。随后，用户可回答机械臂是否应当返回停止位置。

[\AllMoveTasks]

数据类型：switch

指定所有运行中的普通任务以及实际任务中应当停止的程序。

如果省略本参数，则仅将停止执行本指令的任务中的程序。

程序执行

当在实际运动任务中受影响的机械单元当时正在进行的移动已达到零速度并保持静止时，本指令停止程序执行。随后，可从下一指令重启程序执行。

如果在没有任何开关的情况下使用本指令，则仅有该任务中的程序将受到影响。

如果在一个任务（普通、静力学或半静力学）中使用AllMoveTasks开关，则该任务中的程序以及所有普通任务均将停止。请参见更多关于系统参数文件中的任务声明

由于其仅关注运动路径，因此，仅可能在运动任务中使用NoRegain开关。

如果在事件程序中存在Stop指令，则将停止执行程序，并继续按照第*686*页的*Stop*所示来执行。

如果在MultiMove系统的事件程序中存在Stop\AllMoveTasks指令，则按照第*686*页的*Stop*所述，继续进行包含本指令的任务，并按照第*686*页的*Stop \AllMoveTasks*

下一页继续

1.263 Stop - 停止程序执行
RobotWare - OS
续前页

（与执行事件程序期间的普通程序停止的作用相同）所述，继续执行事件程序的所有其他运动任务。

Stop

事件例程	由Stop指令起效
POWER ON	针对所有任务停止执行。在下一开始命令后的事件程序中，不继续执行。
START	针对所有任务停止执行。在下一开始命令后的事件程序中，继续执行。
RESTART	针对所有任务停止执行。在下一开始命令后的事件程序中，继续执行。
STOP	停止执行。在下一开始命令后的事件程序中，不继续执行。
QSTOP	停止执行。在下一开始命令后的事件程序中，不继续执行。
RESET	停止执行。在下一开始命令后的事件程序中，不继续执行。

Stop \AllMoveTasks

事件例程	由Stop \AllMoveTasks指令起效
POWER ON	针对所有任务停止执行。在下一开始命令后的事件程序中，不继续执行。
START	针对所有任务停止执行。在下一开始命令后的事件程序中，继续执行。
RESTART	针对所有任务停止执行。在下一开始命令后的事件程序中，继续执行。
STOP	针对所有任务停止执行。在下一开始命令后的事件程序中，不继续执行。
QSTOP	针对所有任务停止执行。在下一开始命令后的事件程序中，不继续执行。
RESET	停止执行。在下一开始命令后的事件程序中，不继续执行。

更多示例

有关于如何使用指令Stop的更多例子阐述如下。

例 1

```
MoveL p1, v500, fine, tool1;
TPWrite "Jog the robot to the position for pallet corner 1";
Stop \NoRegain;
p1_read := CRobT(\Tool:=tool1 \WObj:=wobj0);
MoveL p2, v500, z50, tool1;
```

通过位于p1的机械臂，停止程序执行。运算符点动，机械臂移动至p1_read。关于下一次程序起动，机械臂并未恢复至p1，因此，可将位置p1_read储存在程序中。

语法

```
Stop
  [ '\' NoRegain ] '|'
  [ '\' AllMoveTasks ]';'
```

下一页继续

相关信息

信息，关于	请参阅
终止程序执行	第*202*页的*EXIT - 终止程序执行*
仅停止机械臂的移动	第*690*页的*StopMove - 停止机械臂的移动*
停止关于调制的程序	第*42*页的*Break - 中断程序执行*

1.264 STOpen - 打开伺服工具

手册用法

STOpen用于打开伺服工具。

基本示例

以下实例介绍了指令STOpen：

例 1

```
STOpen gun1;
```

打开伺服工具gun1。等待，直至在继续下一RAPID指令之前打开焊枪。

例 2

```
STOpen gun1 \Conc;
```

打开伺服工具gun1。继续下一RAPID指令，无需等待焊枪打开。

例 3

```
STOpen "SERVOGUN"\WaitZeroSpeed;
```

停止伺服工具SERVOGUN，等待，直至已完成所有协调移动，随后，打开伺服工具SERVOGUN。

变元

```
STOpen ToolName [\WaitZeroSpeed] [\Conc]
```

ToolName

数据类型：string

机械单元名称

[\WaitZeroSpeed]

数据类型：switch

停止伺服工具，等待，直至已完成所有协调移动，随后，打开伺服工具。

[\Conc]

数据类型：switch

当焊枪移动时，执行后续指令。本参数可用于缩短周期时间。当同时控制两个焊枪时，上述操作有用。

程序执行

如果存在机械单元，则下达打开伺服工具的命令。将焊枪头压力降低为零，并使工具臂返回预关闭位置。

按照针对相关外轴的系统参数规定，以最大速度和加速度移动工具臂。针对其他轴的移动，以手动模式降低速度。

在编程机械臂移动期间，只要机械臂移动不包括工具臂的移动，则有可能打开工具。如果在此类移动期间打开工具，则将显示错误50251 Tool opening failed。开关WaitZeroSpeed可用于降低出现该错误的风险。

如果使用开关Conc，则认为本指令将在打开伺服工具之前就绪。建议在STOpen之后使用函数STIsOpen，以避免以并发模式出现的任何问题。

下一页继续

欲知更多细节，请参见伺服工具运动控制。

错误处理

如果指定伺服工具名称并非配置伺服工具，则将系统变量ERRNO设置为ERR_NO_SGUN。

如果未启用伺服工具机械单元，则将系统变量ERRNO设置为ERR_SGUN_NOTACT。使用指令ActUnit，以启用伺服工具。

如果未初始化伺服工具位置，则将系统变量ERRNO设置为ERR_SGUN_NOTINIT。在首次安装焊枪或完成精细校准之后，必须初始化伺服工具位置。使用服务程序ManServiceCalib或实施焊嘴更换校准。将重置焊枪头磨损。

如果伺服工具焊嘴未同步，则将系统变量ERRNO设置为ERR_SGUN_NOTSYNC。如果已丢失和/或更新转数计数器，则必须使伺服工具焊嘴同步。将不会丢失诸如焊枪头磨损等过程数据。

可用RAPID错误处理器来处理所有上述错误。

注意

如果从背景任务调用本指令，且存在紧急停止，则本指令将在不产生任何错误的情况下完成。

语法

```
STOpen
[ 'ToolName ':=' ] < expression (IN) of string > ',’
[ '\'WaitZeroSpeed]' ,'
['\'Conc]'
```

相关信息

信息，关于	请参阅
关闭伺服工具	第671页的*STClose - 关闭伺服工具*

1.265 StopMove - 停止机械臂的移动

手册用法

StopMove用于停止机械臂和外轴的移动以及暂时随附的过程。如果给定指令StartMove，则移动和过程会恢复。

例如，当出现中断时，该指令可用于软中断程序，以暂时停止机械臂。

针对基座系统，可能在以下类别的程序任务中使用该指令：

- 关于停止该任务中的移动的主任务T_ROB1。
- 关于停止主任务中移动的所有其他任务。

针对MultiMove系统，可能在以下类别的程序任务中使用该指令：

- 关于停止该任务中的移动的运动任务。
- 非运动任务，用于停止相关运动任务中的移动。此外，如果在属于协调同步任务组的一个运动任务中停止移动，便会在所有合作任务中停止移动。

基本示例

以下实例介绍了指令StopMove：

另请参阅第*691*页的更多示例

例 1

```
StopMove;
WaitDI ready_input, 1;
StartMove;
```

停止机械臂移动，直至输入ready_input得以设置。

变元

```
StopMove [\Quick] [\AllMotionTasks]
```

[\Quick]

数据类型：switch

尽快停止本路径上的机械臂。

在没有可选参数\Quick的情况下，机械臂在路径上停止，但是制动距离更长（与普通程序停止相同）。

[\AllMotionTasks]

数据类型：switch

停止系统中所有机械单元的移动。仅可在非运动程序任务中使用开关[\AllMotionTasks]。

程序执行

在没有进行制动的情况下，机械臂和外轴的移动停止。当停止移动时，同时停止与过程中的移动相关的所有过程。

在等待机械臂和外轴停止（保持静止）后，继续程序执行。

通过开关\AllMotionTasks（仅允许非运动程序任务使用），停止关于系统中所有机械单元的移动。

下一页继续

在没有开关\AllMotionTasks的基座系统中，停止以下机械单元的移动：

- 始终为主任务中的机械单元，不依赖于执行StopMove指令的任务。

在没有开关\AllMotionTasks的MultiMove系统中，停止关于以下机械单元的移动：

- 执行StopMove的运动任务中的机械单元。
- 与执行StopMove的非运动任务相关的运动任务中的机械单元。此外，如果在属于协调同步任务组的一个相关运动任务中停止机械单元，则在所有的合作任务中停止机械单元。

当从开始起动运动任务时，从运动任务自身产生的运动任务中的StopMove状态将自动重置。

从一些非运动任务产生的相关运动任务中的StopMove状态将自动重置：

- 如果为普通非运动任务，位于该任务的开始位置。
- 如果为半静力学非运动任务，当本任务从起点开始时，位于上电失败重启处。
- 如果为静力学非运动任务，当本任务从起点开始时，位于installation 起点处。

更多示例

有关于指令StopMove的更多例子阐述如下。

例 1

```
VAR intnum intno1;
...
PROC main()
  ...
  CONNECT intno1 WITH go_to_home_pos;
  ISignalDI di1,1,intno1;
  ...

TRAP go_to_home_pos
  VAR robtarget p10;
  StopMove;
  StorePath;
  p10:=CRobT(\Tool:=tool1 \WObj:=wobj0);
  MoveL home,v500,fine,tool1;
  WaitDI di1,0;
  Move L p10,v500,fine,tool1;
  RestoPath;
  StartMove;
ENDTRAP
```

将输入di1设置为1时，启用中断，进而启用中断程序go_to_home_pos。停止当前移动，机械臂转而移动至home位置。将di1设置为0时，机械臂返回至出现中断的位置，并继续沿编程路径移动。

例 2

```
VAR intnum intno1;
...
PROC main()
  ...
  CONNECT intno1 WITH go_to_home_pos;
  ISignalDI di1,1,intno1;
```

下一页继续

1.265 StopMove - 停止机械臂的移动
RobotWare - OS
续前页

```
  ...

TRAP go_to_home_pos ()
  VAR robtarget p10;
  StorePath;
  p10:=CRobT(\Tool:=tool1 \WObj:=wobj0);
  MoveL home,v500,fine,tool1;
  WaitDI di1,0;
  MoveL p10,v500,fine,tool1;
  RestoPath;
  StartMove;
ENDTRAP
```

与先前的例子类似，但是，机械臂并未移动至起始位置，直至完成当前移动指令。

限制

在进行StopMove - 针对某些运动任务的StartMove序列时，仅允许进行若干非运动任务之一。

语法

```
StopMove
  ['\'Quick]
  ['\'AllMotionTasks]';'
```

相关信息

信息，关于	请参阅
继续移动	第*663*页的*StartMove* - 重启机械臂移动 第*665*页的*StartMoveRetry* - 重启机械臂移动和执行
储存-恢复路径	第*695*页的*StorePath* - 发生中断时，存储路径 第*512*页的*RestoPath* - 中断之后，恢复路径

1.266 StopMoveReset - 重置系统停止移动状态

手册用法

StopMoveReset用于在不开始任何移动的情况下，重置系统停止移动状态。

以异步方式引起的移动错误，例如，移动期间的ERR_PATH_STOP或特定过程错误，可用ERROR处理器进行处理。当出现此类错误时，立即停止移动，并针对实际程序任务设置系统停止移动标记。这意味着，如果在程序指针位于ERROR处理器内部时启动任何程序，则不会重启移动。

在以下行动之一后，且出现此类移动错误后，将重启移动：

- 执行StartMove或StartMoveRetry。
- 执行StopMoveReset，将在下一程序起点重启移动。

基本示例

以下实例介绍了指令StopMoveReset：

例 1

```
...
ArcL p101, v100, seam1, weld1, weave1, z10, gun1;
...
ERROR
  IF ERRNO=AW_WELD_ERR OR ERRNO=ERR_PATH_STOP THEN
    ! Execute something but without any restart of the movement
    ! ProgStop - ProgStart must be allowed
    ...
    ! No idea to try to recover from this error, so let the error

    ...
    ! Reset the move stop flag, so it’s possible to manual restart

    ! stopped
    StopMoveReset;
  ENDIF
ENDPROC
```

在上述情况后，ERROR处理器已经执行了ENDPROC，程序执行停止，指针位于ArcL指令开始位置。下一程序开始从出现原始移动错误的位置，重启程序和移动。

变元

```
StopMoveReset [\AllMotionTasks]
```

[\AllMotionTasks]

数据类型：switch

针对系统中的所有机械单元，重置系统停止移动状态。仅可在非运动程序任务中使用开关[\AllMotionTasks]。

程序执行

为通过同步协调模式来重置MultiMove应用，必须在涉及协调的所有运动任务中执行StopMoveReset。

下一页继续

1.266 StopMoveReset - 重置系统停止移动状态

RobotWare - OS

续前页

通过开关\AllMotionTasks（仅允许非运动程序任务使用），完成系统中所有机械单元的重置工作。

在没有开关\AllMotionTasks的基座系统中，始终完成主任务的重置工作，且不依赖于执行StopMoveReset指令的任务。

针对基座系统，可能在以下类别的程序任务中使用StopMoveReset：

- 主任务T_ROB1，用于重置在该任务中的停止移动状态。
- 用于重置主任务中停止移动状态的所有其他任务。

针对MultiMove系统，可能在以下类别的程序任务中使用该指令：

- 运动任务，用于重置在该任务中的停止移动状态。
- 非运动任务，用于重置相关运动任务中的停止移动状态。此外，如果在属于协调同步任务组的一个相关运动任务中重置停止移动状态，便会在所有合作任务中重置停止移动状态。

语法

```
StopMoveReset
  ['\'AllMotionTasks]';'
```

相关信息

信息，关于	请参阅
停止移动	第*690*页的*StopMove - 停止机械臂的移动*
继续移动	第*663*页的*StartMove - 重启机械臂移动* 第*665*页的*StartMoveRetry - 重启机械臂移动和执行*
储存-恢复路径	第*695*页的*StorePath - 发生中断时，存储路径* 第*512*页的*RestoPath - 中断之后，恢复路径*

1.267 StorePath - 发生中断时，存储路径

手册用法

StorePath用于储存执行中的移动路径，例如，当出现错误或中断时。随后，错误处理器或软中断程序可开始新的临时移动，最后，重启先前保存的原始移动。

例如，当出现错误时，该指令可用于转到服务位置，或清洁焊枪。

本指令仅可用于主任务T_ROB1，或者如果在MultiMove系统中，则可用于运动任务中。

基本示例

以下实例介绍了指令StorePath：

另请参阅第*695*页的更多示例

例 1

```
StorePath;
```

保存当前移动路径，以供随后使用。将系统设置为独立移动模式。

例 2

```
StorePath \KeepSync;
```

保存当前移动路径，以供随后使用。保持同步移动模式。

变元

```
StorePath [\KeepSync]
```

[\KeepSync]

Keep Synchronization

数据类型：switch

在StorePath \KeepSync之后，保持同步移动模式。仅当系统在 StorePath \KeepSync调用之前处于同步移动模式，方可使用KeepSync开关。

在MultiMove协调同步系统中没有可选参数\KeepSync的情况下，将本系统设置为独立的半协调移动模式。在所有相关任务中执行StorePath后，如果进一步使用协调工件，则本系统处于半协调模式。否则，本系统处于独立模式。如果处于半协调模式，则建议在所有相关任务中的WaitSyncTask之前，始终在控制用户坐标系的机械单元中开始移动。

程序执行

保存机械臂和外轴的当前移动路径。此后，可通过软中断程序或错误处理器，开始另一个移动。当已调整关于错误或中断的原因时，可重启已保存的移动路径。

更多示例

有关于如何使用指令StorePath的更多例子阐述如下。

例 1

```
TRAP machine_ready
  VAR robtarget p1;
  StorePath;
  p1 := CRobT();
  MoveL p100, v100, fine, tool1;
```

下一页继续

1.267 StorePath - 发生中断时，存储路径
RobotWare - OS
续前页

```
    ...
    MoveL p1, v100, fine, tool1;
    RestoPath;
    StartMove;
  ENDTRAP
```

当出现启用软中断程序machine_ready的中断时，机械臂当时正在执行的移动路径得以在本指令（ToPoint）结束时停止并保存。此后，机械臂通过更换机器中的零件，对中断进行补救。随后，重启普通移动。

限制

通过指令StorePath，仅保存了移动路径数据。

如果用户想要下达在新路径等级上移动的命令，随后，在StorePath移动至保存于路径上的停止位置之后，以及RestoPath移动至保存于路径上的停止位置之前，必须直接储存实际停止位置。

一次只能储存一个移动路径。

语法

```
StorePath
  ['\'KeepSync]';'
```

相关信息

信息，关于	请参阅
恢复路径	第*512*页的*RestoPath* - 中断之后，恢复路径
更多示例	第*512*页的*RestoPath* - 中断之后，恢复路径 第*440*页的*PathRecStart* - 起动路径记录器 第*715*页的*SyncMoveResume* - 设置同步协调移动 第*717*页的*SyncMoveSuspend* - 设置独立-半协调移动

1.268 STTune - 调节伺服工具

手册用法

STTune用于调节/改变伺服工具参数。临时根据原始值来改变本参数，其在系统参数中建立。执行本指令后，新调整值将立即生效。

STTune适用于调节无返回值程序。调节无返回值程序通常用于寻找参数的最佳值。当使用不同的参数调节值时，重复试验（即包含伺服工具移动的程序执行）。

校准或工具关闭期间，不得使用STTune。

基本示例

以下实例介绍了指令STTune：

例 1

```
STTune SEOLO_RG, 0.050, CloseTimeAdjust;
```

暂时将伺服工具参数CloseTimeAdjust设置为0.050秒。

变元

```
STTune MecUnit TuneValue Type
```

`MecUnit`

数据类型：mecunit

机械单元名称

`TuneValue`

数据类型：num

新调节值。

`Type`

数据类型：tunegtype

参数类型。可用于调节的伺服工具参数包括RampTorqRefOpen、RampTorqRefClose、KV、SpeedLimit、CollAlarmTorq、CollContactPos、CollisionSpeed、CloseTimeAdjust、ForceReadyDelayT、PostSyncTime、CalibTime、CalibForceLow和CalibForceHigh。在系统参数中，预定义上述类型，并定义原始值。

描述

`RampTorqRefOpen`

调节系统参数Ramp when decrease force，其决定在打开工具时，压力释放的快慢程度。单位为Nm/s，且典型值为200。

相关的系统参数：主题*Motion*、类型*Force master*、参数ramp_torque_ref_opening。

`RampTorqRefClose`

调节系统参数Ramp when increase force，其决定在打开工具时，压力累积的快慢程度。单位为Nm/s，且典型值为80。

相关的系统参数：主题*Motion*、类型*Force master*、参数ramp_torque_ref_closing。

下一页继续

1.268 STTune - 调节伺服工具
Servo Tool Control
续前页

KV

调节系统参数KV，其用于速度限制。单位为Nms/rad，且典型值为1。欲知更多细节，请参见外轴文件。

相关的系统参数：主题*Motion*、类型 *Force master*、参数Kv。

SpeedLimit

调节系统参数Speed limit，其用于速度限制。单位为rad/s（电动机转速），且典型值为60。欲知更多细节，请参见外轴文件。

相关的系统参数：主题*Motion*、类型*Force master*、参数speed_limit。

CollAlarmTorq

调节系统参数Collision alarm torque，其用于新焊嘴的自动校准。单位为Nm（电动机扭矩），且典型值为1。欲知更多细节，请参见外轴文件。

相关的系统参数：主题*Motion*、类型*Force master*、参数alarm_torque。

CollContactPos

调节系统参数Collision delta pos，其用于新焊嘴的自动校准。单位为m，且典型值为0,002。欲知更多细节，请参见外轴文件。

相关的系统参数：主题*Motion*、类型*Force master*、参数distance_to_contact_position。

CollisionSpeed

调节系统参数Collision speed，其用于新焊嘴的自动校准。单位为m/s，且典型值为0,02。欲知更多细节，请参见外轴文件。

相关的系统参数：主题*Motion*、类型*Force master*、参数col_speed。

CloseTimeAdjust

当工具焊嘴在工具关闭期间达到接触点时的常量时间调整（正或负）。当同步的预关闭用于焊接时，可用于稍微延迟关闭。

相关的系统参数：主题*Motion*、类型*SG process*、参数min_close_time_adjust。

ForceReadyDelayT

在发送焊接就绪信号之前且实现编程压力之后的常量时间延迟（s）。

相关的系统参数：主题*Motion*、类型*SG process*、参数pre_sync_delay_time。

PostSyncTime

在焊接后，对下一个机械臂移动的释放时间预测（s）。可调节该调节类型，以使焊枪的打开与下一个机械臂移动同步。如果参数设置过高，则同步可能失败。

相关的系统参数：主题*Motion*、类型*SG process*、参数post_sync_time。

CalibTime

在完成定位工具焊嘴修正之前的校准期间的等待时间。为获得最佳结果，不得使用过低的值，例如，0.5 s。

相关的系统参数：主题*Motion*、类型*SG process*、参数calib_time。

CalibForceLow

焊枪头磨损校准期间使用的最小焊枪头压力（N）。为获得关于厚度探测的最佳结果，建议使用最小的编程焊接压力。

下一页继续

相关的系统参数：主题*Motion*、类型*SG process*、参数`calib_force_low`。

`CalibForceHigh`

焊枪头磨损校准期间使用的最大焊枪头压力（N）。为获得关于厚度探测的最佳结果，建议使用最大的编程焊接压力。

相关的系统参数：主题*Motion*、类型*SG process*、参数`calib_force_high`。

程序执行

针对指定的机械单元，启用指定的调节类型和调节值。该值适用于所有移动，直至针对当前机械单元而编制新值，或通过使用指令`STTuneReset`来重置调节类型和值。

可永久性地改变系统参数中的原始调节值。

自动设置默认伺服工具调节值

- 通过执行指令`STTuneReset`。
- 重启时。

错误处理

如果指定伺服工具名称并非配置伺服工具，则将系统变量`ERRNO`设置为`ERR_NO_SGUN`。

可以用Rapid错误处理器来处理错误。

语法

```
STTune
  [ MecUnit ':=' ] < variable (VAR) of mecunit > ','
  [ TuneValue' :=' ] < expression (IN) of num > ','
  [ 'Type ':='] < expression (IN) of tunegtype > ]';'
```

相关信息

信息，关于	请参阅
伺服工具参数的恢复	第*825*页的*TuneReset* - 重置伺服调节
伺服工具的调节	应用手册 - *Additional axes and stand alone controller*

1.269 STTuneReset - 重置伺服工具调节

手册用法

如果伺服工具参数已经被STTune指令所改变，则STTuneReset用于恢复伺服工具参数的原始值。

基本示例

以下实例介绍了指令STTuneReset：

例 1

```
STTuneReset SEOLO_RG;
```

针对机械单元SEOLO_RG，恢复伺服工具参数的原始值。

变元

```
STTuneReset MecUnit
```

MecUnit

数据类型：mecunit

机械单元名称

程序执行

恢复原始伺服工具参数。

同时在重启时实现。

错误处理

如果指定伺服工具名称并非配置伺服工具，则将系统变量ERRNO设置为ERR_NO_SGUN。

可以用Rapid错误处理器来处理错误。

语法

```
STTuneReset
  [ MecUnit ':=' ] < variable (VAR) of mecunit > ','
```

相关信息

信息，关于	请参阅
伺服工具参数的调节	第697页的*STTune - 调节伺服工具*
伺服工具参数的调节	应用手册 - *Additional axes and stand alone controller*

1.270 SupSyncSensorOff - 停止同步传感器监督

手册用法

SupSyncSensorOff用于停止对机械臂移动和同步传感器移动的监督。

基本示例

有关于指令SupSyncSensorOff的基本例子阐述如下。

示例

```
SupSyncSensorOff SSYNC1;
```

不再监督传感器。

变元

```
SupSyncSensorOff MechUnit
```

MechUnit

Mechanical unit

数据类型：mecunit

机械单元名称

语法

```
SupSyncSensorOff
  [ MechUnit ':=' ] < variable (VAR) of mecunit> ';'
```

相关信息

信息，关于	请参阅
启动对同步化传感器的监控	第*702*页的*SupSyncSensorOn - 起动同步传感器监督*
与传感器同步	第*721*页的*SyncToSensor - 同步至传感器*
Machine Synchronization	应用手册 - 控制器软件*IRC5*

1.271 SupSyncSensorOn - 起动同步传感器监督

手册用法

SupSyncSensorOn用于起动机械臂移动和同步传感器移动之间的监督。

基本示例

有关于指令SupSyncSensorOn的基本例子阐述如下。

示例

```
SupSyncSensorOn Ssync1, 150, 100, 50
```

当传感器位于50和150之间时，监督机械单元Ssync1。如果机械臂与传感器之间的距离小于100，则终止监督。

变元

```
SupSyncSensorOn MechUnit MaxSyncSup SafetyDist MinSyncSup
    [\SafetyDelay]
```

MechUnit

Mechanical unit

数据类型：mecunit

机械单元名称

MaxSyncSup

Maximal Synchronized supervised position

数据类型：num

机械臂将监督传感器，直至传感器通过最大同步位置。当通过该点时，停止监督。单位为mm。

SafetyDist

Safety distance

数据类型：num

Safetydist是预期机器位置与实际机器位置之间的差异的限制。其必须为负值，即模型应当始终在实际机器之前移动。在减少机器位置的情况下，相对于最大反向位置差异（以及最小提前距离），限值必须为负。在增加机器位置的情况下，相对于最小正向位置差异（以及最小提前距离），限值必须为正。

如果机械臂与传感器之间的距离小于安全距离，则机械臂将触发警报。触发警报时，将停止监督。

单位为mm。

MinSyncSup

Minimal synchronized supervised position

数据类型：num

当传感器位于从MinSyncSup位置到MaxSyncSup位置定义的窗口中时，机械臂将开始监督。单位为mm。

[\SafetyDelay]

Safety delay

下一页继续

数据类型：num

SafetyDelay用于调整机械臂编程位置与传感器监督位置之间的延迟。单位为秒。

限制

如果在完成指令WaitSensor之前使用SupSynSensorOn，则机械臂将停止。

语法

```
SupSyncSensorOn
  [ MechUnit ':=' ] <variable (VAR) of mecunit> ','
  [ MaxSyncSup ':=' ] < expression (IN) of num > ','
  [ SafetyDist ':=' ] < expression (IN) of num > ','
  [ MinSyncSup ':=' ] < expression (IN) of num >
  [ \SafetyDelay ':=' ] < expression (IN) of num > ';'
```

相关信息

信息，关于	请参阅
停止对同步化传感器的监控	第*701*页的*SupSyncSensorOff - 停止同步传感器监督*
与传感器同步	第*721*页的*SyncToSensor - 同步至传感器*
Machine Synchronization	应用手册 - 控制器软件*IRC5*

1.272 SyncMoveOff - 结束协调同步移动

手册用法

SyncMoveOff用于结束一系列的同步移动以及绝大多数情况下的协调移动。首先，所有相关的程序任务将等待在停止点同步，随后，将相关程序任务的运动规划器设置为独立模式。

本指令SyncMoveOff仅可用于含选项*Coordinated Robots*的*MultiMove*系统以及定义为Motion Task的程序任务中。

警告

为实现安全同步功能，各交会点（参数SyncID）必须拥有唯一的名称。交会点名称必须同时与应当满足的所有程序任务相同。

基本示例

以下实例介绍了指令SyncMoveOff：

另请参阅第*705*页的更多示例

例 1

```
!Program example in task T_ROB1

PERS tasks task_list{2} := [ ["T_ROB1"], ["T_ROB2"] ];
VAR syncident sync1;
VAR syncident sync2;

...
SyncMoveOn sync1, task_list;
...
SyncMoveOff sync2;
...

!Program example in task T_ROB2

PERS tasks task_list{2} := [ ["T_ROB1"], ["T_ROB2"] ];
VAR syncident sync1;
VAR syncident sync2;

...
SyncMoveOn sync1, task_list;
...
SyncMoveOff sync2;
...
```

首先实现SyncMoveOff以及识别号sync2的程序任务，等待其他任务实现SyncMoveOff以及相同的识别号sync2。在该同步点sync2，将用于相关程序任务的运动规划器设置为独立模式。此后，任务T_ROB1和T_ROB2均继续执行。

变元

```
SyncMoveOff SyncID [\TimeOut]
```

下一页继续

SyncID

Synchronization Identity

数据类型： syncident

指定非同步（交会）点名称的变量。数据类型 syncident为非值类型。其仅作为用于命名非同步点的标识符。

必须定义变量，使其在所有协作程序任务中拥有相同的名称。建议始终定义各任务中的变量全局（VAR syncident ...）。

[\TimeOut]

数据类型：num

等待其他程序任务达到非同步点的最长时间。以秒来定义时限（分辨率为0,001s）。

如果在所有程序任务已达到非同步点之前耗尽时间，则将调用错误处理器。如果存在这样的情况，则其错误代码为ERR_SYNCMOVEOFF。如果不存在错误处理器，则将停止执行。

如果省略该参数，则程序任务将永久等待。

程序执行

首次实现SyncMoveOff的程序任务等待所有其他指定任务实现具有相同SyncID识别号的SyncMoveOff。在该SyncID非同步点，将相关程序任务的运动规划器设置为独立模式。此后，相关的程序任务继续执行。

将相关程序任务的运动规划器设置为非同步模式。这意味着：

- 所有RAPID程序任务以及此类任务的所有移动均可相互独立地进行。
- 所有移动指令均不得标记上任何ID编号。参见指令MoveL。

可能因为测试FlexPendant示教器 - 任务选择面板而排除程序任务。指令SyncMoveOn和SyncMoveOff将仍然对程序任务数量的减少起作用，即便仅针对一个程序任务。

更多示例

有关于如何使用指令SyncMoveOff的更多例子阐述如下。

简单同步移动实例

```
!Program example in task T_ROB1
PERS tasks task_list{2} := [ ["T_ROB1"], ["T_ROB2"] ];
VAR syncident sync1;
VAR syncident sync2;
VAR syncident sync3;

PROC main()
  ...
  MoveL p_zone, vmax, z50, tcp1;
  WaitSyncTask sync1, task_list;
  MoveL p_fine, v1000, fine, tcp1;
  syncmove;
  ...
ENDPROC

PROC syncmove()
  SyncMoveOn sync2, task_list;
```

下一页继续

1.272 SyncMoveOff - 结束协调同步移动

RW-MRS Synchronized

续前页

```
  MoveL * \ID:=10, v100, z10, tcp1 \WObj:= rob2_obj;
  MoveL * \ID:=20, v100, fine, tcp1 \WObj:= rob2_obj;
  SyncMoveOff sync3;
  UNDO
    SyncMoveUndo;
ENDPROC

!Program example in task T_ROB2
PERS tasks task_list{2} := [ ["T_ROB1"], ["T_ROB2"] ];
VAR syncident sync1;
VAR syncident sync2;
VAR syncident sync3;

PROC main()
  ...
  MoveL p_zone, vmax, z50, obj2;
  WaitSyncTask sync1, task_list;
  MoveL p_fine, v1000, fine, obj2;
  syncmove;
  ...
ENDPROC

PROC syncmove()
  SyncMoveOn sync2, task_list;
  MoveL * \ID:=10, v100, z10, obj2;
  MoveL * \ID:=20, v100, fine, obj2 ;
  SyncMoveOff sync3;
  UNDO
    SyncMoveUndo;
ENDPROC
```

首先，程序任务T_ROB1和T_ROB2正在识别号为sync1的WaitSyncTask处相互等待，并通过关于向前移动的角路径来编程，以节省周期时间。

随后，程序任务正在识别号为sync2的SyncMoveOn处相互等待，并通过关于先前移动的必要停止点来编程。此后，将相关程序任务的运动规划器设置为同步模式。

此后，当T_ROB1正在将tcp1移动至运动对象obj2上的ID点10和20时，T_ROB2正在将obj2移动至世界坐标系中的ID点10和20。

随后，程序任务正在识别号为sync3的SyncMoveOff处相互等待，并通过关于先前移动的必要停止点来编程。此后，将相关程序任务的运动规划器设置为独立模式。

包含错误恢复的实例

```
!Program example with use of time-out function
VAR syncident sync3;

...
SyncMoveOff sync3 \TimeOut := 60;
...
ERROR
  IF ERRNO = ERR_SYNCMOVEOFF THEN
    RETRY;
  ENDIF
```

下一页继续

本程序任务等待指令SyncMoveOff以及一些其他程序任务达到相同的同步点sync3。在等待60秒后，通过与ERR_SYNCMOVEOFF相同的ERRNO，调用错误处理器。随后，再次调用本指令SyncMoveOff ，以额外等待60秒。

半联动移动和联动移动示例

```
!Example with semicoordinated and synchronized movement
!Program example in task T_ROB1
PERS tasks task_list{2} := [ ["T_ROB1"], ["T_ROB2"] ];
PERS wobjdata rob2_obj:= [FALSE,FALSE,"ROB_2",
     [[0,0,0],[1,0,0,0]],[[155.241,-51.5938,57.6297],
     [0.493981,0.506191,-0.501597,0.49815]]];
VAR syncident sync0;
VAR syncident sync1;
VAR syncident sync2;
VAR syncident sync3;
VAR syncident sync4;

PROC main()
  ...
  WaitSyncTask sync0, task_list;
  MoveL p1_90, v100, fine, tcp1 \WObj:= rob2_obj;
  WaitSyncTask sync1, task_list;
  SyncMoveOn sync2, task_list;
  MoveL p1_100 \ID:=10, v100, fine, tcp1 \WObj:= rob2_obj;
  SyncMoveOff sync3;
  !Wait until the movement has been finished in T_ROB2
  WaitSyncTask sync3, task_list;
  !Now a semicoordinated movement can be performed
  MoveL p1_120, v100, z10, tcp1 \WObj:= rob2_obj;
  MoveL p1_130, v100, fine, tcp1 \WObj:= rob2_obj;
  WaitSyncTask sync4, task_list;
  ...
ENDPROC

!Program example in task T_ROB2
PERS tasks task_list{2} := [ ["T_ROB1"], ["T_ROB2"] ];
VAR syncident sync0;
VAR syncident sync1;
VAR syncident sync2;
VAR syncident sync3;
VAR syncident sync4;

PROC main()
  ...
  MoveL p_fine, v1000, fine, tcp2;
  WaitSyncTask sync0, task_list;
  !Wait until the movement in T_ROB1 task is finished
  WaitSyncTask sync1, task_list;
  SyncMoveOn sync2, task_list;
  MoveL p2_100 \ID:=10, v100, fine, tcp2;
  SyncMoveOff sync3;
```

下一页继续

1.272 SyncMoveOff - 结束协调同步移动

RW-MRS Synchronized

续前页

```
    !The path has been removed at SyncMoveOff
    !Perform a movement to wanted position for the object to make
         the position available for other tasks
    MoveL p2_100, v100, fine, tcp2;
    WaitSyncTask sync3, task_list;
    WaitSyncTask sync4, task_list;
    MoveL p2_110, v100, z10, tcp2;
    ...
ENDPROC
```

从半联动移动变为同步移动时， 需要采用 WaitSyncTask （当采用标识符 sync1 时）。

从同步移动变为半联动移动时，移动对象的任务（rob2_obj）需要移动到预想的位置。此后，在能开展半联动移动前 ， 需要采用 WaitSyncTask（标识符sync3）。

错误处理

如果因为SyncMoveOff未及时就绪而超时，则将系统变量ERRNO设置为ERR_SYNCMOVEOFF。

可用ERROR处理器来处理该错误。

限制

如果所有相关机械臂均在停止点保持静止，则仅可执行SyncMoveOff指令。

如果该指令位于移动指令之后，则必须使用停止点（zonedata fine）而非飞越点来编程移动指令，否则，将无法在电源故障后重启。

无法在与任意下列特殊系统事件关联的RAPID程序中执行SyncMoveOff：PowerOn、Stop、QStop、Restart、Reset或者Step。

语法

```
SyncMoveOff
  [ SyncID ':=' ] < variable (VAR) of syncident>
  [ '\'TimeOut' :=' < expression (IN) of num> ] ';'
```

相关信息

信息，关于	请参阅
指定协作的程序任务	第*1500*页的*tasks - RAPID*程序任务
同步点的识别号	第*1496*页的*syncident* - 同步点的识别号
起动协调同步移动	第*709*页的*SyncMoveOn* - 起动协调同步移动
设置独立移动	第*719*页的*SyncMoveUndo* - 设置独立移动
测试是否处于同步模式	第*1136*页的*IsSyncMoveOn* - 测试是否处于同步移动模式
含选项Coordinated robots的MultiMove系统	应用手册 - *MultiMove*

1.273 SyncMoveOn - 起动协调同步移动

手册用法

SyncMoveOn用于起动一系列的同步移动以及绝大多数情况下的协调移动。首先，所有相关的程序任务将等待在停止点同步，随后，将相关程序任务的运动规划器设置为同步模式。

本指令SyncMoveOn仅可用于含选项*Coordinated Robots*的*MultiMove*系统以及定义为Motion Task的程序任务中。

警告

为实现安全同步功能，各交会点（参数SyncID）必须拥有唯一的名称。交会点名称必须同时与应当在交会点中满足的所有程序任务相同。

基本示例

以下实例介绍了指令SyncMoveOn：

另请参阅第*710*页的更多示例

例 1

```
!Program example in task T_ROB1

PERS tasks task_list{2} := [ ["T_ROB1"], ["T_ROB2"] ];
VAR syncident sync1;
VAR syncident sync2;

...
SyncMoveOn sync1, task_list;
...
SyncMoveOff sync2;
...

!Program example in task T_ROB2

PERS tasks task_list{2} := [ ["T_ROB1"], ["T_ROB2"] ];
VAR syncident sync1;
VAR syncident sync2;

...
SyncMoveOn sync1, task_list;
...
SyncMoveOff sync2;
...
```

首次达到含识别号sync1的SyncMoveOn的程序任务进行等待，直至其他任务达到含相同识别号sync1的SyncMoveOn。在该同步点sync1，将相关程序任务的运动规划器设置为同步模式。此后，任务T_ROB1和T_ROB2均继续执行，同步直至其达到含相同识别号sync2的SyncMoveOff。

下一页继续

1.273 SyncMoveOn - 起动协调同步移动

RW-MRS Independent

续前页

变元

```
SyncMoveOn SyncID TaskList [\TimeOut]
```

SyncID

Synchronization Identity

数据类型：syncident

指定同步（交会）点名称的变量。数据类型syncident为非值类型，其仅作为用于命名同步点的标识符。

必须定义变量，使其在所有协作程序任务中拥有相同的名称。建议始终定义各任务中的变量全局（VAR syncident ...）。

TaskList

数据类型：tasks

位于任务列表（数组）中的永久变量，指定应当在同步点中满足的程序任务的名称（string），其名称符合参数SyncID。

必须定义永久变量，使其在所有协作程序任务中拥有相同的名称和内容。建议始终在系统中定义变量全局（PERS tasks ...）。

[\TimeOut]

数据类型：num

等待其他程序任务达到同步点的最长时间。以秒来定义时限（分辨率为0.001s）。

如果在所有程序任务已达到同步点之前耗尽时间，则将调用错误处理器。如果存在这样的情况，则其错误代码为ERR_SYNCMOVEON。如果不存在错误处理器，则将停止执行。

如果省略该参数，则程序任务将永久等待。

程序执行

首次实现SyncMoveOn的程序任务等待所有其他指定任务实现具有相同SyncID识别号的SyncMoveOn。在该SyncID同步点，将相关程序任务的运动规划器设置为同步模式。此后，相关的程序任务继续执行。

将相关程序任务的运动规划器设置为同步模式。这意味着：

- TaskList中所有程序任务中的各移动指令与TaskList中其他任务中的移动指令同时起效。
- 规划所有协作移动指令，并插补到相同的运动规划器中。
- 所有移动同时开始和结束。花费最长时间的移动将作为速度控制，其减速与其他移动的工件相关。
- 所有协作移动指令必须标记上相同的ID编号。参见指令MoveL。

可能因为测试FlexPendant示教器 - 任务选择面板而排除程序任务。指令SyncMoveOn将仍然对程序任务数量的减少起作用，即便仅针对一个程序任务。

更多示例

有关于如何使用指令SyncMoveOn的更多例子阐述如下。

例 1

```
!Program example in task T_ROB1
PERS tasks task_list{2} := [["T_ROB1"], ["T_ROB2"]];
```

下一页继续

```
VAR syncident sync1;
VAR syncident sync2;
VAR syncident sync3;

PROC main()
  ...
  MoveL p_zone, vmax, z50, tcp1;
  WaitSyncTask sync1, task_list;
  MoveL p_fine, v1000, fine, tcp1;
  syncmove;
  ...
ENDPROC

PROC syncmove()
  SyncMoveOn sync2, task_list;
  MoveL * \ID:=10, v100, z10, tcp1 \WOBJ:= rob2_obj;
  MoveL * \ID:=20, v100, fine, tcp1 \WOBJ:= rob2_obj;
  SyncMoveOff sync3;
  UNDO
    SyncMoveUndo;
ENDPROC
!Program example in task T_ROB2
PERS tasks task_list{2} := [["T_ROB1"], ["T_ROB2"]];
VAR syncident sync1;
VAR syncident sync2;
VAR syncident sync3;

PROC main()
  ...
  MoveL p_zone, vmax, z50, obj2;
  WaitSyncTask sync1, task_list;
  MoveL p_fine, v1000, fine, obj2;
  syncmove;
  ...
ENDPROC

PROC syncmove()
  SyncMoveOn sync2, task_list;
  MoveL * \ID:=10, v100, z10, obj2;
  MoveL * \ID:=20, v100, fine, obj2;
  SyncMoveOff sync3;
  UNDO
    SyncMoveUndo;
ENDPROC
```

首先，程序任务T_ROB1和T_ROB2正在识别号为sync1的WaitSyncTask处相互等待。通过关于向前移动的角路径来编程，以节省周期时间。

随后，程序任务正在识别号为sync2的SyncMoveOn处相互等待。通过关于先前移动的必要停止点来编程。此后，将相关程序任务的运动规划器设置为同步模式。

下一页继续

1.273 SyncMoveOn - 起动协调同步移动
RW-MRS Independent
续前页

此后，当T_ROB1正在将tcp1移动至运动对象obj2上的ID 点10和20时，T_ROB2正在将obj2移动至世界坐标系中的ID点10和20。

例 2

```
!Program example with use of time-out function
VAR syncident sync3;

...
SyncMoveOn sync3, task_list \TimeOut :=60;
...
ERROR
  IF ERRNO = ERR_SYNCMOVEON THEN
    RETRY;
  ENDIF
```

本程序任务等待程序任务T_ROB2的指令SyncMoveOn，以达到相同的同步点sync3。在等待60秒后，通过与ERR_SYNCMOVEON相同的ERRNO，调用错误处理器。随后，再次调用本指令SyncMoveOn，以额外等待60秒。

例3-包含三个任务的程序实例

```
!Program example in task T_ROB1
PERS tasks task_list1 {2} :=[["T_ROB1"], ["T_ROB2"]];
PERS tasks task_list2 {3} :=[["T_ROB1"], ["T_ROB2"], ["T_ROB3"]];
VAR syncident sync1;
...
VAR syncident sync5;

...
  SyncMoveOn sync1, task_list1;
  ...
  SyncMoveOff sync2;
  WaitSyncTask sync3, task_list2;
  SyncMoveOn sync4, task_list2;
  ...
  SyncMoveOff sync5;
...

!Program example in task T_ROB2

PERS tasks task_list1 {2} := [["T_ROB1"], ["T_ROB2"]];
PERS tasks task_list2 {3} := [["T_ROB1"], ["T_ROB2"], ["T_ROB3"]];
VAR syncident sync1;
...
VAR syncident sync5;

...
  SyncMoveOn sync1, task_list1;
  ...
  SyncMoveOff sync2;
  WaitSyncTask sync3, task_list2;
  SyncMoveOn sync4, task_list2;
  ...
```

下一页继续

```
  SyncMoveOff sync5;
...

!Program example in task T_ROB3

PERS tasks task_list2 {3} := [["T_ROB1"], ["T_ROB2"], ["T_ROB3"]];
VAR syncident sync3;
VAR syncident sync4;
VAR syncident sync5;

...
  WaitSyncTask sync3, task_list2;
  SyncMoveOn sync4, task_list2;
  ...
  SyncMoveOff sync5;
...
```

在本例子中，首先，程序任务T_ROB1和T_ROB2同步移动，且T_ROB3独立移动。在更进一步的程序中，所有三个任务均同步移动。为防止在T_ROB1和T_ROB2的首次同步结束之前，以T_ROB3来执行SyncMoveOn指令，应使用指令WaitSyncTask。

错误处理

如果因为SyncMoveOn未及时就绪而超时，则将系统变量ERRNO设置为ERR_SYNCMOVEON。

可用ERROR处理器来处理该错误。

限制

如果所有相关机械臂均在停止点保持静止，则仅可执行SyncMoveOn指令。

同时仅可启用一个协调同步移动组。

如果该指令位于移动指令之后，则必须使用停止点（zonedata fine）而非飞越点来编程移动指令，否则，将无法在电源故障后重启。

无法在与任意下列特殊系统事件关联的RAPID程序中执行SyncMoveOn：PowerOn、Stop、QStop、Restart、Reset或者Step。

语法

```
SyncMoveOn
  [ SyncID ':=' ] < variable (VAR) of syncident> ','
  [ TaskList ':=' ] < persistent array {*} (PERS) of tasks> ','
  [ '\' TimeOut ':=' < expression (IN) of num > ] ';'
```

相关信息

信息，关于	请参阅
指定协作的程序任务	第*1500*页的*tasks - RAPID*程序任务
同步点的识别号	第*1496*页的*syncident* - 同步点的识别号
结束协调同步移动	第*704*页的*SyncMoveOff* - 结束协调同步移动
设置独立移动	第*719*页的*SyncMoveUndo* - 设置独立移动

下一页继续

1.273 SyncMoveOn - 起动协调同步移动

RW-MRS Independent

续前页

信息，关于	请参阅
测试是否处于同步模式	第*1136*页的*IsSyncMoveOn* - 测试是否处于同步移动模式
含选项Coordinated Robots的MultiMove系统	应用手册 - *MultiMove*
等待同步任务	第*890*页的*WaitSyncTask* - 在同步点等待其他程序任务

1.274 SyncMoveResume - 设置同步协调移动

手册用法

SyncMoveResume 用于从独立移动模式返回同步移动。本指令仅可用于StorePath等级，例如，在已经执行StorePath \KeepSync后，且在已经执行SyncMoveSuspend后，本系统处于独立运动模式。为了能够使用本指令，在执行StorePath和SyncMoveSuspend指令之前，本系统必须已经处于同步运动模式。

本指令SyncMoveResume仅可用于含选项*Coordinated Robots*和*Path Recovery*的*MultiMove*系统以及定义为运动任务的程序任务中。

基本示例

以下实例介绍了指令SyncMoveResume：

例 1

```
ERROR
  StorePath \KeepSync;
  ! Save position
  p11 := CRobT(\Tool:=tool2);
  ! Move in synchronized motion mode
  MoveL p12\ID:=111, v50, fine, tool2;
  SyncMoveSuspend;
  ! Move in independent mode somewhere, e.g. to a cleaning station
  p13 := CRobT();
  MoveL p14, v100, fine, tool2;
  ! Do something at cleaning station
  MoveL p13, v100, fine, tool2;
  SyncMoveResume;
  ! Move in synchronized motion mode back to start position p11
  MoveL p11\ID:=111, fine, z20, tool2;
  RestoPath;
  StartMove;
  RETRY;
```

出现某种可恢复错误。以同步模式来维持本系统，并同步移动至某点，例如，沿路径向后移动。此后，独立移动至清理站。随后，机械臂返回出现错误的点，且程序从因错误而中断的位置继续。

程序执行

当系统采用StorePath等级上的独立移动模式时，SyncMoveResume压力恢复同步模式。

进入独立模式之前，所有正在以同步移动而执行中的任务均需要SyncMoveResume。如果一个运动任务执行SyncMoveResume，则该任务将等待，直至先前采用同步移动模式的所有任务均执行SyncMoveResume指令。此后，相关程序任务继续执行。

限制

SyncMoveResume仅可用于返回同步移动模式，且仅可用于StorePath等级。

如果该指令位于移动指令之后，则必须使用停止点（zonedata fine）而非飞越点来编程移动指令，否则，将无法在电源故障后重启。

下一页继续

1.274 SyncMoveResume - 设置同步协调移动

Path Recovery

续前页

无法在与任意下列特殊系统事件关联的RAPID程序中执行SyncMoveResume：PowerOn、Stop、QStop、Restart,、Reset或者Step。

语法

```
SyncMoveResume ';'
```

相关信息

信息，关于	请参阅
指定协作的程序任务	第*1500*页的*tasks - RAPID*程序任务
起动协调同步移动	第*709*页的*SyncMoveOn* - 起动协调同步移动
结束协调同步移动	第*704*页的*SyncMoveOff* - 结束协调同步移动
测试是否处于同步模式	第*709*页的*SyncMoveOn* - 起动协调同步移动
储存路径	第*695*页的*StorePath* - 发生中断时，存储路径
恢复路径	第*512*页的*RestoPath* - 中断之后，恢复路径
暂停同步移动	第*717*页的*SyncMoveSuspend* - 设置独立-半协调移动

1.275 SyncMoveSuspend - 设置独立-半协调移动

手册用法

SyncMoveSuspend 用于暂停同步移动模式，并将系统设置为独立-半协调移动模式。本指令仅可用于StorePath等级，例如，在已经执行StorePath或StorePath \KeepSync之后，且系统采用同步移动模式。

本指令SyncMoveSuspend仅可用于含选项*Coordinated Robots*和*Path Recovery*的*MultiMove System*以及定义为运动任务的程序任务中。

基本示例

以下实例介绍了指令SyncMoveSuspend：

例 1

```
ERROR
  StorePath \KeepSync;
  ! Save position
  p11 := CRobT(\Tool:=tool2);
  ! Move in synchronized motion mode
  MoveL p12\ID:=111, v50, fine, tool2;
  SyncMoveSuspend;
  ! Move in independent mode somewhere, e.g. to a cleaning station
  p13 := CRobT();
  MoveL p14, v100, fine, tool2;
  ! Do something at cleaning station
  MoveL p13, v100, fine, tool2;
  SyncMoveResume;
  ! Move in synchronized motion mode back to start position p11
  MoveL p11\ID:=111, fine, z20, tool2;
  RestoPath;
  StartMove;
  RETRY;
```

出现某种可恢复错误。以同步模式来维持本系统，并同步移动至某点，例如，沿路径向后移动。此后，独立移动至清理站。随后，机械臂返回出现错误的点，且程序从因错误而中断的位置继续。

程序执行

SyncMoveSuspend 强制重置同步移动，并将系统设置为独立-半协调移动模式。

所有同步运动任务均需要SyncMoveSuspend，以采用独立-半协调移动模式来设置系统。如果一个运动任务执行SyncMoveSuspend，则该任务将等待，直至其他任务已经执行了SyncMoveSuspend指令。

在所有相关任务中执行SyncMoveSuspend后，如果进一步使用协调工件，则本系统会采用半协调模式。否则，其会采用独立模式。如果采用半协调模式，建议始终从控制相关任务中WaitSyncTask之前的用户坐标的机械单元中的移动开始。

限制

SyncMoveSuspend指令仅在StorePath等级上，暂停同步模式。在从StorePath等级返回后，将系统设置为其在StorePath之前采用的模式。

下一页继续

1.275 SyncMoveSuspend - 设置独立-半协调移动

Path Recovery

续前页

如果该指令位于移动指令之后，则必须使用停止点（zonedata `fine`）而非飞越点来编程移动指令，否则，将无法在电源故障后重启。

无法在与任意下列特殊系统事件关联的RAPID程序中执行`SyncMoveSuspend`：PowerOn、Stop、QStop、Restart,、Reset或者Step。

语法

```
SyncMoveSuspend ';'
```

相关信息

信息，关于	请参阅
指定协作的程序任务	第*1500*页的*tasks - RAPID*程序任务
起动协调同步移动	第*709*页的*SyncMoveOn* - 起动协调同步移动
结束协调同步移动	第*704*页的*SyncMoveOff* - 结束协调同步移动
测试是否处于同步模式	第*1136*页的*IsSyncMoveOn* - 测试是否处于同步移动模式
储存路径	第*695*页的*StorePath* - 发生中断时，存储路径
恢复路径	第*512*页的*RestoPath* - 中断之后，恢复路径
恢复同步移动	第*715*页的*SyncMoveResume* - 设置同步协调移动

1.276 SyncMoveUndo - 设置独立移动

手册用法

SyncMoveUndo用于强制重置同步协调移动，并将系统设置为独立移动模式。

本指令SyncMoveUndo仅可用于含选项*Coordinated Robots*的*MultiMove*系统以及定义为Motion Task的程序任务中。

基本示例

以下实例介绍了指令SyncMoveUndo：

例 1

位于任务T_ROB1中的程序实例

```
PERS tasks task_list{2} := [ ["T_ROB1"], ["T_ROB2"] ];
VAR syncident sync1;
VAR syncident sync2;
VAR syncident sync3;
PROC main()

...
  MoveL p_zone, vmax, z50, tcp1;
  WaitSyncTask sync1, task_list;
  MoveL p_fine, v1000, fine, tcp1;
  syncmove;
  ...
ENDPROC

PROC syncmove()
  SyncMoveOn sync2, task_list;
  MoveL * \ID:=10, v100, z10, tcp1 \WOBJ:= rob2_obj;
  MoveL * \ID:=20, v100, fine, tcp1 \WOBJ:= rob2_obj;
  SyncMoveOff sync3;
UNDO
  SyncMoveUndo;
ENDPROC
```

如果在无返回值程序syncmove内执行本程序，且程序指针移出无返回值程序syncmove时，停止本程序，则执行UNDO处理器内的所有指令。在该例子中，执行指令SyncMoveUndo，并将系统设置为独立移动模式。

程序执行

强制重置同步协调移动，并将系统设置为独立移动模式。

能够在一个程序任务中执行SyncMoveUndo，从而将整个系统设置为独立移动模式。如果本系统已经采用独立移动模式，则可在不产生任何错误的情况下执行本指令。

亦将本系统设置为默认独立移动模式

- 当使用重启模式重置**RAPID**时
- 当加载一则新程序或一个新模块时
- 当从起点开始执行程序时
- 当将程序指针移动到main时

下一页继续

1.276 SyncMoveUndo - 设置独立移动

RobotWare - OS

续前页

- 当将程序指针移动到子程序时
- 移动程序指针造成执行顺序丢失时

语法

```
SyncMoveUndo ';'
```

相关信息

信息，关于	请参阅
指定协作的程序任务	第*1500*页的*tasks - RAPID*程序任务
同步点的识别号	第*1496*页的*syncident* - 同步点的识别号
起动协调同步移动	第*709*页的*SyncMoveOn* - 起动协调同步移动
结束协调同步移动	第*704*页的*SyncMoveOff* - 结束协调同步移动
测试是否处于同步模式	第*1136*页的*IsSyncMoveOn* - 测试是否处于同步移动模式

1.277 SyncToSensor - 同步至传感器

手册用法

SyncToSensor用于起动或停止机械臂与传感器的同步移动。

基本示例

有关于指令SyncToSensor的基本例子阐述如下。

例 1

```
WaitSensor Ssync1;
MoveL *, v1000, z10, tool, \WObj:=wobj0;
SyncToSensor Ssync1\On;
MoveL *, v1000, z20, tool, \WObj:=wobj0;
MoveL *, v1000, z20, tool, \WObj:=wobj0;
SyncToSensor Ssync1\Off;
```

变元

```
SyncToSensor MechUnit [\MaxSync] [\On] | [\Off]
```

MechUnit

Mechanical Unit

数据类型：mecunit

运动中的机械单元与指令中的机械臂位置相关。

[\MaxSync]

数据类型：num

机械臂将与传感器同步移动，直至传感器通过MaxSync位置。此后，机械臂将以编程速度非同步移动。如果未定义可选参数MaxSync，则机械臂将同步移动，直至执行指令SyncToSensor Ssync1\Off。

[\On]

数据类型：switch

在使用参数\On的指令后，机械臂将与传感器同步移动。

[\Off]

数据类型：switch

在使用参数\Off的指令后，机械臂将与传感器非同步移动。

程序执行

SyncToSensor Ssync1 \On意味着机械臂开始与传感器Ssync1同步移动。因此，在传感器通过储存于robtarget中的外部位置时，机械臂同时通过robtarget。

SyncToSensor Ssync1 \Off意味着机械臂停止与传感器同步移动。

限制

当传感器尚未通过WaitSensor连接时，如果发出指令SyncToSensor Ssync1 \On，则机械臂将停止。

下一页继续

1.277 SyncToSensor - 同步至传感器

续前页

语法

```
SyncToSensor
  [ MechUnit ':=' ] < variable (VAR) of mecunit >
  [ \MaxSync] [ '\' On] | [ '\' Off] ';'
```

相关信息

信息，关于	请参阅
启动对同步化传感器的监控	第*702*页的*SupSyncSensorOn - 起动同步传感器监督*
与传感器同步	第*721*页的*SyncToSensor - 同步至传感器*
等候传感器的连接	第*887*页的*WaitSensor - 等待传感器连接*
将对象丢到传感器上	第*147*页的*DropSensor - 使物体落于传感器上*
Machine Synchronization	应用手册 - 控制器软件*IRC5*

1.278 SystemStopAction - 停止机器人系统

手册用法

根据错误或问题的严重程度，SystemStopAction可用于以不同的方式来停止机器人系统。

基本示例

以下实例介绍了指令SystemStopAction：

例 1

```
SystemStopAction \Stop;
```

这将停止所有运动任务中的程序执行和机械臂移动。重启程序执行前，无需采取特定行动。

例 2

```
SystemStopAction \StopBlock;
```

这将停止所有运动任务中的程序执行和机械臂移动。在可以重启程序执行前，必须移动所有程序指针。

例 3

```
SystemStopAction \Halt;
```

这将导致电机关闭、程序执行停止以及所有运动任务中的机械臂移动。必须在可重启程序执行前，完成电机开启。

变元

```
SystemStopAction [\Stop] [\StopBlock] [\Halt]
```

[\Stop]

数据类型：switch

\Stop用于停止所有运动任务中的程序执行和机械臂移动。重启程序执行前，无需采取特定行动。

[\StopBlock]

数据类型：switch

\StopBlock用于停止所有运动任务中的程序执行和机械臂移动。在可以重启程序执行前，必须移动所有程序指针。

[\Halt]

数据类型：switch

\Halt将导致电机关闭、程序执行停止以及所有运动任务中的机械臂移动。必须在可重启程序执行前，完成电机开启。

限制

如果在SystemStopAction \StopBlock期间，机械臂正在进行圆周移动，则程序指针和机械臂必须移动至重启程序执行前的圆周移动的起点。

程序执行

根据错误或问题的严重程度，SystemStopAction用于以不同的方式来停止机器人系统。如果该任务为普通任务，则在执行任务的过程中停止程序执行。

下一页继续

1.278 SystemStopAction - 停止机器人系统

RobotWare - OS

续前页

如果在静力学或半静力学任务中执行SystemStopAction，则所有普通任务将停止程序执行，但是，该任务将继续。参见系统参数文件中关于任务声明的更多内容。

语法

```
SystemStopAction
  [ '\'Stop ]
  | [ '\'StopBlock ]
  | [ '\'Halt ]';'
```

相关信息

信息，关于	请参阅
停止程序执行	第*685*页的*Stop - 停止程序执行*
终止程序执行	第*202*页的*EXIT - 终止程序执行*
仅停止机械臂的移动	第*690*页的*StopMove - 停止机械臂的移动*
写入某些错误消息	第*193*页的*ErrLog - 写入错误消息*

1.279 TEST - 根据表达式的值...

手册用法

根据表达式或数据的值，当有待执行不同的指令时，使用TEST。

如果并没有太多的替代选择，则亦可使用IF..ELSE指令。

基本示例

以下实例介绍了指令TEST：

例 1

```
TEST reg1
CASE 1,2,3 :
  routine1;
CASE 4 :
  routine2;
DEFAULT :
  TPWrite "Illegal choice";
  Stop;
ENDTEST
```

根据reg1的值，执行不同的指令。如果该值为1、2或3时，则执行routine1。如果该值为4，则执行routine2。否则，打印出错误消息，并停止执行。

变元

```
TEST Test data {CASE Test value {, Test value} : ...} [ DEFAULT:
     ...] ENDTEST
```

Test data

数据类型：所有

用于比较测试值的数据或表达式。

Test value

数据类型：与test data相同

测试数据必须拥有的值，以供执行相关的指令。

程序执行

将测试数据与第一个CASE条件中的测试值进行比较。如果对比真实，则执行相关指令。此后，通过ENDTEST后的指令，继续程序执行。

如果未满足第一个CASE条件，则对其他CASE条件进行测试等。如果未满足任何条件，则执行与DEFAULT相关的指令（如果存在）。

语法

```
TEST <expression>
{ CASE <test value> { ',' <test value> } ':'
  <statement list> }
[ DEFAULT ':'
  <statement list> ]
ENDTEST
```

下一页继续

1.279 TEST - 根据表达式的值...
RobotWare - OS
续前页

相关信息

信息，关于	请参阅
表达式	技术参考手册 - *RAPID*指令、函数和数据类型

1.280 TestSignDefine - 定义测试信号

手册用法

TestSignDefine用于定义有关机械臂运动系统的一个测试信号。

测试信号不断地反映一些指定运动数据流。例如，适用于一些指定轴的扭矩参考量。通过函数TestSignRead，可读取RAPID中某一特定时间的实际值。

仅可获得外轴的测试信号。根据对于机械臂轴以及外轴非预定测试信号的需求，测试信号亦有效。

基本示例

以下实例介绍了指令TestSignDefine：

例 1

```
TestSignDefine 1, resolver_angle, Orbit, 2, 0.1;
```

与通道1相连的测试信号resolver_angle，将给出有关orbit机械臂，上外轴2的旋转变压器角度，以100 ms的速率进行取样。

变元

```
TestSignDefine Channel SignalId MechUnit Axis SampleTime
```

Channel

数据类型：num

将用于测试信号的通道编号1-12。必须在函数TestSignRead中使用相同的编号，以便读取测试信号的实际值。

SignalId

数据类型：testsignal

测试信号的名称或编号。参见数据类型 testsignal中描述的预定义常量。

MechUnit

Mechanical Unit

数据类型：mecunit

机械单元名称

Axis

数据类型：num

机械单元内的轴编号。

SampleTime

数据类型：num

样本时间，以秒计。

如下表所示，针对样本时间<0.004 s，函数TestSignRead会返回最新可用内部样本的平均值。

样本时间，以秒计	来自TestSignRead的结果
0	每0.5 ms产生的最新8个样本的平均值
0.001	每1 ms产生的最新4个样本的平均值

下一页继续

1.280 TestSignDefine - 定义测试信号
RobotWare - OS
续前页

样本时间，以秒计	来自TestSignRead的结果
0.002	每2 ms产生的最新2个样本的平均值
大于或等于0.004	在指定样本时间产生的瞬时值
0.1	在指定样本时间100 ms产生的瞬时值

程序执行

启用测试信号的定义，且机器人系统开始对测试信号进行取样。
对测试信号的取样始终有效，直至：

- 执行关于实际通道的新TestSignDefine指令。
- 通过执行指令TestSignReset，停用所有测试信号。
- 在系统重启时，停用所有测试信号。

错误处理

如果参数MechUnit中存在错误，则将变量ERRNO设置为ERR_UNIT_PAR。如果参数Axis中存在错误，则将ERRNO设置为ERR_AXIS_PAR。

语法

```
TestSignDefine
  [ Channel ':=' ] < expression (IN) of num>' ,'
  [ SignalId' :=' ] < expression (IN) of testsignal> ','
  [ MechUnit' :=' ] < variable (VAR) of mecunit> ','
  [ Axis ':=' ] < expression (IN) of num> ','
  [ SampleTime' :=' ] < expression (IN) of num > ';'
```

相关信息

信息，关于	请参阅
测试信号	第*1501*页的*testsignal - 测试信号*
读取测试信号	第*1276*页的*TestSignRead - 读取测试信号值*
重置测试信号	第*729*页的*TestSignReset - 重置所有测试信号定义*

1.281 TestSignReset - 重置所有测试信号定义

手册用法

TestSignReset用于停用所有先前定义的测试信号。

基本示例

以下实例介绍了指令TestSignReset：

例 1

```
TestSignReset;
```

停用所有先前定义的测试信号。

程序执行

停用测试信号的定义，且机器人系统停止对测试信号进行取样。

对定义测试信号的取样始终有效，直至：

- 系统重启
- 该指令TestSignReset的执行

语法

```
TestSignReset';'
```

相关信息

信息，关于	请参阅
测试信号	第*1501*页的*testsignal - 测试信号*
定义测试信号	第*727*页的*TestSignDefine - 定义测试信号*
读取测试信号	第*1276*页的*TestSignRead - 读取测试信号值*

1.282 TextTabInstall - 安装文本表格

手册用法

TextTabInstall的作用是在系统中安装一份文本表格。

基本示例

以下实例介绍了指令TextTabInstall：

例 1

```
! System Module with Event Routine to be executed at event
! POWER ON, RESET or START

PROC install_text()
  IF TextTabFreeToUse("text_table_name") THEN
    TextTabInstall "HOME:/text_file.eng";
  ENDIF
ENDPROC
```

首次执行事件程序install_text时，函数TextTabFreeToUse返回TRUE，并在系统中安装文本文件text_file.eng。此后，可通过函数TextTabGet和TextGet，从RAPID系统获取已安装的文本字符串。

下一次执行事件程序install_text时，函数TextTabFreeToUse 返回FALSE，且不重复安装。

变元

```
TextTabInstall File
```

File

数据类型：string

包含将于系统中安装的文本字符串文件的路径和名称。

限制

关于在系统中安装文本表格（文本资源）的限制：

- 不可能在系统中多次安装同一文本表格。
- 不可能从系统中卸载（解除）单一文本表格。从系统卸载文本表格的唯一方式是通过使用重启模式重置系统，以此来重启控制器。随后，将卸载所有文本表格（包括系统和用户定义的文本表格）。

错误处理

如果不能打开TextTabInstall指令中的文件，则将系统变量ERRNO设置为ERR_FILEOPEN。随后，可用错误处理器对该错误进行处理。

语法

```
TextTabInstall
  [ File ':=' ] < expression (IN) of string >';'
```

下一页继续

相关信息

信息，关于	请参阅
测试文本表格是否已解除	第*1280*页的*TextTabFreeToUse* - 测试文本表格是否已解除
文本文件的格式	技术参考手册 - *RAPID*语言内核
获取文本表格编号	第*1282*页的*TextTabGet* - 获取文本表格编号
从系统文本表格中获取文本	第*1278*页的*TextGet* - 从系统文本表格中获取文本
字符串功能	技术参考手册 - *RAPID*语言概览
字符串的定义	第*1489*页的*string* - 字符串
Advanced RAPID	应用手册 - 控制器软件*IRC5*

1.283 TPErase - 擦除在FlexPendant示教器上印刷的文本

手册用法

TPErase（*FlexPendant Erase*）为，用于清除FlexPendant示教器的显示内容。

基本示例

以下实例介绍了指令TPErase：

例 1

```
TPErase;
TPWrite "Execution started";
```

写入Execution started前，清除FlexPendant示教器显示。

程序执行

彻底清除FlexPendant示教器显示器中的所有文本。下一次写入文本时，其将进入显示器的最高线。

语法

```
TPErase;
```

相关信息

信息，关于	请参阅
在FlexPendant示教器上写入	技术参考手册 - *RAPID*语言概览

1.284 TPReadDnum - 从FlexPendant示教器读取编号

手册用法

TPReadDnum（*FlexPendant Read Numerical*）用于从FlexPendant示教器读取编号

基本示例

以下实例介绍了指令 TPReadDnum：

例 1

```
VAR dnum value;

TPReadDnum value, "How many units should be produced?";
```

将文本How many units should be produced?写入FlexPendant示教器显示器。程序执行进入等待，直至已经从FlexPendant示教器上的数字键盘输入编号。将该编号储存在value中。

变元

```
TPReadDnum TPAnswer TPText [\MaxTime][\DIBreak] [\DIPassive]
     [\DOBreak] [\DOPassive] [\BreakFlag]
```

TPAnswer

数据类型：dnum

返回用于通过FlexPendant示教器输入编号的变量。

TPText

数据类型：string

有待写入FlexPendant示教器的信息文本（每行40个字符，最多80个字符）。

[\MaxTime]

数据类型：num

程序执行等待的最长时间。如果在该时间内未输入任何编号，则用错误处理器继续执行程序，除非使用BreakFlag（参见下文）。常量ERR_TP_MAXTIME可用于测试是否已经过最长时间。

[\DIBreak]

Digital Input Break

数据类型：signaldi

可能中断运算符对话框的数字信号。如果将信号设置为1（或已经为1）时未输入任何数字，则用错误处理器继续执行程序，除非使用BreakFlag（参见下文）。可使用常量ERR_TP_DIBREAK来测试是否已出现该情况。

[\DIPassive]

Digital Input Passive

数据类型：switch

当使用DIBreak可选参数时，该开关会覆盖默认行为。将信号设置为1（或已经为1）时，取代反应；将信号DIBreak设置为0（或已经为0）时，应当用错误处理器继续本指令（如果未使用BreakFlag）。常量ERR_TP_DIBREAK可用于测试是否已出现上述情况。

下一页继续

1.284 TPReadDnum - 从FlexPendant示教器读取编号
RobotWare - OS
续前页

[\DOBreak]

Digital Output Break

数据类型：signaldo

支持其他任务终止请求的数字信号。如果将信号设置为1（或已经为1）时未选择任何按钮，则用错误处理器继续执行程序，除非使用BreakFlag（参见下文）。可使用常量ERR_TP_DOBREAK来测试是否已出现该情况。

[\DOPassive]

Digital Output Passive

数据类型：switch

当使用DOBreak可选参数时，该开关会覆盖默认行为。将信号设置为1（或已经为1）时，取代反应；将信号 DOBreak设置为0（或已经为0）时，应当用错误处理器继续本指令（如果未使用BreakFlag）。常量ERR_TP_DOBREAK可用于测试是否已出现上述情况。

[\BreakFlag]

数据类型：errnum

使用MaxTime、DIBreak 或DOBreak时将保存错误代码的变量。如果省略该可选变量，则将执行错误处理器。常量ERR_TP_MAXTIME、ERR_TP_DIBREAK和ERR_TP_DOBREAK可用于选择原因。

程序执行

始终将信息文本写入新行。如果显示器充满文本，则首先将该文本正文向上移动一行。在下一写入的文本上最多可存在7行。

程序执行进入等待，直至通过数字键盘输入数字（随后按下Enter或OK），或由超时或信号行动来中断指令。

关于对相同或其他程序任务中FlexPendant示教器上同时出现的TPReadFK或TPReadDnum请求进行描述的TPReadFK的引用。

错误处理

信号变量是在RAPID中声明的变量。其尚未与通过指令AliasIO而在I/O配置中确定的I/O信号相连。如果使用该信号，则将系统变量ERRNO设置为ERR_NO_ALIASIO_DEF，并用错误处理器继续执行。

如果在运算符输入之前超时（参数\MaxTime），则将系统变量ERRNO设置为ERR_TP_MAXTIME，并用错误处理器继续执行。

如果在运算符输入之前设置数字信号输入（参数\DIBreak），则将系统变量ERRNO设置为ERR_TP_DIBREAK，并用错误处理器继续执行。

如果在运算符输入之前出现数字信号输出（参数\DOBreak），则将系统变量ERRNO设置为ERR_TP_DOBREAK，并用错误处理器继续执行。

如果没有客户端（例如，FlexPendant示教器）来关注指令，则将系统变量ERRNO设置为ERR_TP_NO_CLIENT，并用错误处理器继续执行。

随后，可通过错误处理器来处理此类情形。

下一页继续

语法

```
TPReadDnum
  [TPAnswer':='] <var or pers (INOUT) of dnum>','
  [TPText':='] <expression (IN) of string>
  ['\'MaxTime':=' <expression (IN) of num>]
  ['\'DIBreak':=' <variable (VAR) of signaldi>]
  ['\'DIPassive]
  ['\'DOBreak':=' <variable (VAR) of signaldo>]
  ['\'DOPassive]
  ['\'BreakFlag':=' <var or pers (INOUT) of errnum>] ';'
```

相关信息

信息，关于	请参阅
FlexPendant示教器的写入和读取	技术参考手册 - *RAPID*语言概览
在FlexPendant示教器上输入一个数字	操作员手册 - 带 *FlexPendant* 的 *IRC5*
关于如何使用参数MaxTime、DIBreak和BreakFlag的实例	第*736*页的*TPReadFK - 读取功能键*
清除运算符窗口	第*732*页的*TPErase - 擦除在FlexPendant示教器上印刷的文本*

1.285 TPReadFK - 读取功能键

手册用法

TPReadFK（*FlexPendant Read Function Key*）用于对功能键编写文本以及查找按下的是哪个键。

基本示例

下列示例说明了指令TPReadFK：

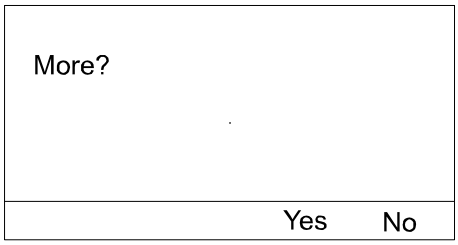
另请参阅第*738*页的更多示例

例 1

```
TPReadFK reg1, "More?", stEmpty, stEmpty, stEmpty, "Yes", "No";
```

在FlexPendant示教器显示器上写入文本More?，并分别通过文本字符串Yes和No（参见下图），启用功能键4和5。程序执行进入等待，直至按下功能键4或5之一。也就是说，根据按下的键，向reg1 分配4或5。

本图表明，运算符可通过功能键来输入信息。



xx0500002345

变元

```
TPReadFK TPAnswer TPText TPFK1 TPFK2 TPFK3 TPFK4 TPFK5 [\MaxTime]
     [\DIBreak] [\DIPassive] [\DOBreak] [\DOPassive] [\BreakFlag]
```

TPAnswer

数据类型：num

根据按下的键，返回数值1..5的变量。如果按下功能键1，则返回1，以此类推。

TPText

数据类型：string

有待写入显示器的信息文本（每行40个字符，最多80个字符）。

TPFKx

Function key text

数据类型：string

有待写入适当功能键的文本（最多45个字符）。TPFK1为最左边的键。

通过预定义字符串常量stEmpty以及空字符串值（“”），指定不含文本的功能键。

[\MaxTime]

数据类型：num

下一页继续

程序执行等待的最长时间，以秒计。如果在该时间内未按下任何功能键，则用错误处理器继续执行程序，除非使用BreakFlag（参见下文）。常量ERR_TP_MAXTIME可用于测试是否已经过最长时间。

[\DIBreak]

Digital Input Break

数据类型：signaldi

可能中断运算符对话框的数字信号。如果将信号设置为1（或已经为1）时未按下任何功能键，则用错误处理器继续执行程序，除非使用BreakFlag（参见下文）。可使用常量ERR_TP_DIBREAK来测试是否已出现该情况。

[\DIPassive]

Digital Input Passive

数据类型：switch

当使用DIBreak可选参数时，该开关会覆盖默认行为。将信号设置为1（或已经为1）时，取代反应；将信号DIBreak设置为0（或已经为0）时，应当用错误处理器继续本指令（如果未使用BreakFlag）。常量ERR_TP_DIBREAK可用于测试是否已出现上述情况。

[\DOBreak]

Digital Output Break

数据类型：signaldo

支持其他任务终止请求的数字信号。如果将信号设置为1（或已经为1）时未选择任何按钮，则用错误处理器继续执行程序，除非使用BreakFlag（参见下文）。可使用常量ERR_TP_DOBREAK来测试是否已出现该情况。

[\DOPassive]

Digital Output Passive

数据类型：switch

当使用DOBreak可选参数时，该开关会覆盖默认行为。将信号设置为1（或已经为1）时，取代反应；将信号 DOBreak设置为0（或已经为0）时，应当用错误处理器继续本指令（如果未使用BreakFlag）。常量ERR_TP_DOBREAK可用于测试是否已出现上述情况。

[\BreakFlag]

数据类型：errnum

使用MaxTime、DIBreak或DOBreak时将保存错误代码的变量。如果省略该可选变量，则将执行错误处理器。常量ERR_TP_MAXTIME, 、ERR_TP_DIBREAK和ERR_TP_DOBREAK可用于选择原因。

程序执行

始终将信息文本写入新行。如果显示器充满文本，则首先将该文本正文向上移动一行。在新写入的文本上最多可存在7行。

在适当的功能键上写入文本。

程序执行进入等待，直至按下已启用的功能键之一。

下一页继续

1.285 TPReadFK - 读取功能键
RobotWare - OS
续前页

相同或其他程序任务的FlexPendant示教器上并行TPReadFK或TPReadNum请求（TP请求）的描述：

- 将不会关注（新纳入队列）其他程序任务的新TP请求
- 将关注（以往纳入队列）相同程序任务中TRAP的新TP请求
- 关注（以往纳入队列）程序停止
- 关注（以往纳入队列）程序停止状态中的新TP请求

更多示例

有关于如何使用指令TPReadFK的更多例子阐述如下。

例 1

```
VAR errnum errvar;
...
TPReadFK reg1, "Go to service position?", stEmpty, stEmpty, stEmpty,
     "Yes","No"
\MaxTime:= 600
  \DIBreak:= di5\BreakFlag:= errvar;
IF reg1 = 4 OR errvar = ERR_TP_DIBREAK THEN
  MoveL service, v500, fine, tool1;
  Stop;
ENDIF
IF errvar = ERR_TP_MAXTIME EXIT;
```

如果按下向前功能键（"Yes"），或如果启用输入5，则将机械臂移动至服务位置。如果在10分钟内未给出任何回答，则终止执行。

错误处理

信号变量是在RAPID中声明的变量。其尚未与通过指令AliasIO而在I/O配置中确定的I/O信号相连。如果使用该信号，则将系统变量ERRNO设置为ERR_NO_ALIASIO_DEF，并用错误处理器继续执行。

如果在运算符输入之前出现超时（参数\MaxTime），则将系统变量ERRNO设置为ERR_TP_MAXTIME，并用错误处理器继续执行。

如果在运算符输入之前设置数字信号输入（参数\DIBreak），则将系统变量ERRNO设置为ERR_TP_DIBREAK，并用错误处理器继续执行。

如果在运算符输入之前出现数字信号输出（参数\DOBreak），则将系统变量ERRNO设置为ERR_TP_DOBREAK，并用错误处理器继续执行。

如果没有客户端（例如，FlexPendant示教器）来关注指令，则将系统变量ERRNO设置为ERR_TP_NO_CLIENT，并用错误处理器继续执行。

随后，可通过错误处理器来处理此类情形。

限制

比如在回路中频繁执行TPReadFK时，避免为超时参数\MaxTime采用两个值。这会造成系统性能出现无法预测的行为，比如，FlexPendant示教器响应减缓。

预定义数据

```
CONST string stEmpty := "";
```

预定义常量stEmpty可用于不含文本的功能键。

下一页继续

语法

```
TPReadFK
  [TPAnswer ':='] <var or pers (INOUT) of num>','
  [TPText ':='] <expression (IN) of string>','
  [TPFK1 ':='] <expression (IN) of string>','
  [TPFK2 ':='] <expression (IN) of string>','
  [TPFK3 ':='] <expression (IN) of string>','
  [TPFK4 ':='] <expression (IN) of string>','
  [TPFK5 ':='] <expression (IN) of string>
  ['\' MaxTime ':=' <expression (IN) of num>]
  ['\' DIBreak ':=' <variable (VAR) of signaldi>]
  ['\' DIPassive]
  ['\' DOBreak ':=' <variable (VAR) of signaldo>]
  ['\' DOPassive]
  ['\' BreakFlag ':=' <var or pers (INOUT) of errnum>]';'
```

相关信息

信息，关于	请参阅
FlexPendant示教器的写入和读取	技术参考手册 - *RAPID*语言概览
通过FlexPendant示教器回复	操作员手册 - 带 *FlexPendant* 的 *IRC5*
清除运算符窗口	第*732*页的*TPErase* - 擦除在*FlexPendant*示教器上印刷的文本

1.286 TPReadNum - 从FlexPendant示教器读取编号

手册用法

TPReadNum（*FlexPendant Read Numerical*）为，用于从FlexPendant示教器读取编号

基本示例

以下实例介绍了指令 TPReadNum：

另请参阅第*741*页的更多示例

例 1

```
TPReadNum reg1, "How many units should be produced?";
```

将文本How many units should be produced?写入FlexPendant示教器显示器。程序执行进入等待，直至已经从FlexPendant示教器上的数字键盘输入编号。将该编号储存在reg1中。

变元

```
TPReadNum TPAnswer TPText [\MaxTime][\DIBreak] [\DIPassive]
    [\DOBreak] [\DOPassive] [\BreakFlag]
```

TPAnswer

数据类型：num

返回用于通过FlexPendant示教器输入编号的变量。

TPText

数据类型：string

有待写入FlexPendant示教器的信息文本（每行40个字符，最多80个字符）。

[\MaxTime]

数据类型：num

程序执行等待的最长时间。如果在该时间内未输入任何编号，则用错误处理器继续执行程序，除非使用BreakFlag（参见下文）。常量ERR_TP_MAXTIME可用于测试是否已经过最长时间。

[\DIBreak]

Digital Input Break

数据类型：signaldi

可能中断运算符对话框的数字信号。如果将信号设置为1（或已经为1）时未输入任何数字，则用错误处理器继续执行程序，除非使用BreakFlag（参见下文）。可使用常量ERR_TP_DIBREAK来测试是否已出现该情况。

[\DIPassive]

Digital Input Passive

数据类型：switch

当使用DIBreak可选参数时，该开关会覆盖默认行为。将信号设置为1（或已经为1）时，取代反应；将信号DIBreak设置为0（或已经为0）时，应当用错误处理器继续本指令（如果未使用BreakFlag）。常量ERR_TP_DIBREAK可用于测试是否已出现上述情况。

下一页继续

[\DOBreak]

Digital Output Break

数据类型：signaldo

支持其他任务终止请求的数字信号。如果将信号设置为1（或已经为1）时未选择任何按钮，则用错误处理器继续执行程序，除非使用BreakFlag（参见下文）。可使用常量ERR_TP_DOBREAK来测试是否已出现该情况。

[\DOPassive]

Digital Output Passive

数据类型：switch

当使用DOBreak可选参数时，该开关会覆盖默认行为。将信号设置为1（或已经为1）时，取代反应；将信号 DOBreak设置为0（或已经为0）时，应当用错误处理器继续本指令（如果未使用BreakFlag）。常量ERR_TP_DOBREAK可用于测试是否已出现上述情况。

[\BreakFlag]

数据类型：errnum

使用MaxTime、DIBreak或DOBreak时将保存错误代码的变量。如果省略该可选变量，则将执行错误处理器。常量ERR_TP_MAXTIME、ERR_TP_DIBREAK和ERR_TP_DOBREAK可用于选择原因。

程序执行

始终将信息文本写入新行。如果显示器充满文本，则首先将该文本正文向上移动一行。在下一写入的文本上最多可存在7行。

程序执行进入等待，直至通过数字键盘输入数字（随后按下Enter或OK），或由超时或信号行动来中断指令。

关于对相同或其他程序任务中FlexPendant示教器上同时出现的TPReadFK或TPReadNum请求进行描述的TPReadFK的引用。

更多示例

有关于如何使用指令TPReadNum的更多例子阐述如下。

例 1

```
TPReadNum reg1, "How many units should be produced?";
FOR i FROM 1 TO reg1 DO
  produce_part;
ENDFOR
```

在FlexPendant示教器显示器上写入的文本How many units should be produced?。随后，重复程序produce_part，通过FlexPendant示教器输入的次数。

错误处理

信号变量是在RAPID中声明的变量。其尚未与通过指令AliasIO而在I/O配置中确定的I/O信号相连。如果使用该信号，则将系统变量ERRNO设置为ERR_NO_ALIASIO_DEF，并用错误处理器继续执行。

下一页继续

1.286 TPReadNum - 从FlexPendant示教器读取编号

RobotWare - OS

续前页

如果在运算符输入之前出现超时（参数\MaxTime），则将系统变量ERRNO设置为ERR_TP_MAXTIME，并用错误处理器继续执行。

如果在运算符输入之前设置数字信号输入（参数\DIBreak），则将系统变量ERRNO设置为ERR_TP_DIBREAK，并用错误处理器继续执行。

如果在运算符输入之前设置数字信号输出（参数\DOBreak），则将系统变量ERRNO设置为ERR_TP_DOBREAK，并用错误处理器继续执行。

如果没有客户端（例如，FlexPendant示教器）来关注指令，则将系统变量ERRNO设置为ERR_TP_NO_CLIENT，并用错误处理器继续执行。

随后，可通过错误处理器来处理此类情形。

语法

```
TPReadNum
  [TPAnswer':='] <var or pers (INOUT) of num>','
  [TPText':='] <expression (IN) of string>
  ['\'MaxTime':=' <expression (IN) of num>]
  ['\'DIBreak':=' <variable (VAR) of signaldi>]
  ['\'DIPassive]
  ['\'DOBreak':=' <variable (VAR) of signaldo>]
  ['\'DOPassive]
  ['\'BreakFlag':=' <var or pers (INOUT) of errnum>] ';'
```

相关信息

信息，关于	请参阅
FlexPendant示教器的写入和读取	技术参考手册 - *RAPID*语言概览
在FlexPendant示教器上输入一个数字	操作员手册 - 带 *FlexPendant* 的 *IRC5*
关于如何使用参数MaxTime、DIBreak和BreakFlag的实例	第*736*页的*TPReadFK* - 读取功能键
清除运算符窗口	第*732*页的*TPErase* - 擦除在*FlexPendant*示教器上印刷的文本

1.287 TPShow - 位于FlexPendant示教器上的开关窗口

手册用法

TPShow（*FlexPendant Show*）用于从RAPID选择FlexPendant示教器窗口。

基本示例

以下实例介绍了指令TPShow：

例 1

```
TPShow TP_LATEST;
```

执行本指令后，将启用于当前FlexPendant示教器窗口前最新使用的FlexPendant示教器窗口。

变元

```
TPShow Window
```

Window

数据类型：tpnum

窗口TP_LATEST将显示在当前FlexPendant示教器窗口前最新使用的FlexPendant示教器窗口。

预定义数据

```
CONST tpnum TP_LATEST := 2;
```

程序执行

将启用选定的FlexPendant示教器窗口。

语法

```
TPShow
  [Window':='] <expression (IN) of tpnum> ';'
```

相关信息

信息，关于	请参阅
使用FlexPendant示教器进行通信	技术参考手册 - *RAPID*语言概览
FlexPendant示教器窗口编号	第*1508*页的*tpnum - FlexPendant*示教器窗口编号
清除运算符窗口	第*732*页的*TPErase* - 擦除在*FlexPendant*示教器上印刷的文本

1.288 TPWrite - 写入FlexPendant示教器

手册用法

TPWrite（*FlexPendant Write*）用于在FlexPendant示教器上写入文本。可将特定数据的值同文本一样写入。

基本示例

以下实例介绍了指令TPWrite：

例 1

```
TPWrite "Execution started";
```

在FlexPendant示教器上写入文本Execution started。

例 2

```
TPWrite "No of produced parts="\Num:=reg1;
```

如果reg1保存值5，则在FlexPendant示教器上写入文本No of produced parts=5。

例 3

```
VAR string my_robot;
...
my_robot := RobName();
IF my_robot="" THEN
  TPWrite "This task does not control any TCP robot";
ELSE
  TPWrite "This task controls TCP robot with name "+ my_robot;
ENDIF
```

将本程序任务控制的TCP机械臂的名称写入FlexPendant示教器。如果未控制任何TCP机械臂，则写入：本任务未控制机械臂。

变元

```
TPWrite String [\Num] | [\Bool] | [\Pos] | [\Orient] | [\Dnum]
```

String

数据类型：string

有待写入的文本字符串（每行40个字符，最多80个字符）。

[\Num]

Numeric

数据类型：num

将在文本字符串后写入其数值的数据。

[\Bool]

Boolean

数据类型：bool

将在文本字符串后写入其逻辑值的数据。

[\Pos]

Position

数据类型：pos

下一页继续

将在文本字符串后写入其位置的数据。

[\Orient]

Orientation

数据类型：orient

将在文本字符串后写入其方位的数据。

[\Dnum]

Numeric

数据类型：dnum

将在文本字符串后写入其数值的数据。

程序执行

在FlexPendant示教器上写入的文本始终以新的一行开始。当显示器充满文本时（11行），则该文本首先上升一行。

如果使用参数\Num、\Dnum，\Bool、 \Pos或 \Orient 之一，则在将其添加至第一个字符串之前，首先将其值转换为文本字符串。从值到文本字符串的转换如下所示：

变元	值	文本串
\Num	23	"23"
\Num	1.141367	"1.14137"
\Bool	TRUE	"TRUE"
\Pos	[1817.3,905.17,879.11]	"[1817.3,905.17,879.11]"
\Orient	[0.96593,0,0.25882,0]	"[0.96593,0,0.25882,0]"
\Dnum	4294967295	"4294967295"

通过标准RAPID格式，将值转换为字符串。这意味着，原则上为6个有效位。如果小数部分小于0.000005或大于0.999995，则将该数字四舍五入为整数。

限制

参数\Num、\Dnum，\Bool、 \Pos和 \Orient互相排斥，因此，不可同时用于同一指令。

语法

```
TPWrite
  [TPText':='] <expression (IN) of string>
  ['\'Num':=' <expression (IN) of num> ]
  | ['\'Bool':=' <expression (IN) of bool> ]
  | ['\'Pos':=' <expression (IN) of pos> ]
  | ['\'Orient':=' <expression (IN) of orient> ]
  | ['\'Dnum':=' <expression (IN) of dnum> ]';'
```

相关信息

信息，关于	请参阅
清除和读取FlexPendant示教器	技术参考手册 - *RAPID*语言概览

下一页继续

1.288 TPWrite - 写入FlexPendant示教器
RobotWare - OS
续前页

信息，关于	请参阅
清除运算符窗口	第*732*页的*TPErase - 擦除在FlexPendant示教器上印刷的文本*

1.289 TriggC - 关于事件的机械臂圆周移动

手册用法

当机械臂正在圆周路径上移动时，TriggC（*Trigg Circular*）用于设置输出信号和/或在固定位置运行中断程序。

使用指令TriggIO、TriggEquip、TriggInt、TriggCheckIO、TriggSpeed或TriggRampAO，可定义一个或多个（最多8个）事件，然后，在指令TriggC中参考此类定义。

本指令仅可用于主任务T_ROB1，或者如果在*MultiMove*系统中，则可用于运动任务中。

基本示例

以下实例介绍了指令TriggC：

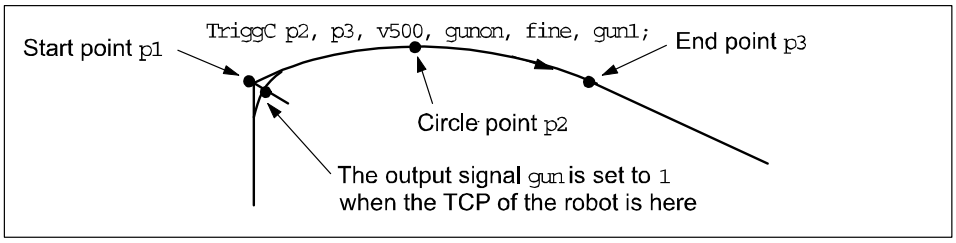
另请参阅第*751*页的更多示例

例 1

```
VAR triggdata gunon;

TriggIO gunon, 0 \Start \DOp:=gun, 1;
MoveL p1, v500, z50, gun1;
TriggC p2, p3, v500, gunon, fine, gun1;
```

当机械臂的TCP通过点p1角路径中点时，设置数字信号输出信号gun。

本图显示了固定位置I/O事件的实例。



xx0500002267

变元

```
TriggC [\Conc] CirPoint ToPoint [\ID] Speed [\T] Trigg_1 |
     TriggArray{*} [\T2] [\T3] [\T4] [\T5] [\T6] [\T7] [\T8] Zone
     [\Inpos] Tool [\WObj] [ \Corr ] [\TLoad]
```

`[ \Conc ]`

Concurrent

数据类型：switch

当机械臂正在运动时，执行后续指令。通常不使用参数，但是，当使用飞越点时，可使用参数以避免由过载CPU所引起的多余停止。当编程点在高速度下极为接近时，该参数适用。当不需要同外设备与机械臂运动之间的外设备和同步进行通信时，参数亦适用。其亦可用于调节机械臂路径的执行，以避免警告50024-角路径故障或错误50082-减速限制。

下一页继续

1.289 TriggC - 关于事件的机械臂圆周移动
RobotWare - OS
续前页

运用参数\Conc，将连续运动指令的数量限制为5。在包含StorePath-RestoPath运动指令以及参数的程序段中，不允许\Conc。

如果省略该参数，且ToPoint并非停止点，则在机械臂达到编程区之前，执行后续指令一段时间。

不能将该参数用于MultiMove系统中的协调同步移动。

CirPoint

数据类型：robtarget

机械臂的圆周点。有关对圆周运动的更为详细的描述，请参见指令MoveC。将圆周点定义为已命名的位置，或直接储存在指令中（在指令中标记有*）。

ToPoint

数据类型：robtarget

机器人和外部轴的目标点。定义为已命名的位置或直接存储在指令中（在指令中加 * 标记）。

[\ID]

Synchronization id

数据类型： identno

如果移动是同步或协调同步的，变元 [\ID] 在 MultiMove 系统中是强制的。这个变元在任何其他情况下都不允许使用。指定的 ID 号必须与所有协作程序任务中的 ID 号相同。使用此 ID 号，动作才不会在运行时搞混。

Speed

数据类型： speeddata

适用于运动的速度数据。速度数据规定了关于工具中心点、工具方位调整和外轴的速率。

[\T]

Time

数据类型：num

该参数用于规定机械臂运动的总时间，以秒计。随后，取代相关的速度数据。

Trigg_1

数据类型：triggdata

利用指令TriggIO、TriggEquip、TriggInt、TriggCheckIO、TriggSpeed或TriggRampAO较早期地在程序中定义涉及触发条件和触发活动的变量。

TriggArray

Trigg Data Array Parameter

数据类型：triggdata

指代先前在程序中通过使用指令TriggIO、TriggEquip、TriggInt、TriggSpeed、TriggCheckIO或TriggRampAO而定义的触发条件和触发活动的数组变量。

数组中的限制为25个元素，且必须定义1到25个已确定的触发条件。

当使用TriggArray参数时，不可能同时使用可选参数T2、T3、T4、T5、T6、T7或T8。

下一页继续

[\T2]

Trigg 2

数据类型：triggdata

利用指令TriggIO、TriggEquip、TriggInt、TriggCheckIO、TriggSpeed或TriggRampAO较早期地在程序中定义涉及触发条件和触发活动的变量。

[\T3]

Trigg 3

数据类型：triggdata

利用指令TriggIO、TriggEquip、TriggInt、TriggCheckIO、TriggSpeed或TriggRampAO较早期地在程序中定义涉及触发条件和触发活动的变量。

[\T4]

Trigg 4

数据类型：triggdata

利用指令TriggIO、TriggEquip、TriggInt、TriggCheckIO、TriggSpeed或TriggRampAO较早期地在程序中定义涉及触发条件和触发活动的变量。

[\T5]

Trigg 5

数据类型：triggdata

利用指令TriggIO、TriggEquip、TriggInt、TriggCheck、TriggSpeed或TriggRampAO较早期地在程序中定义涉及触发条件和触发活动的变量。

[\T6]

Trigg 6

数据类型：triggdata

利用指令TriggIO、TriggEquip、TriggInt、TriggCheckIO、TriggSpeed或TriggRampAO较早期地在程序中定义涉及触发条件和触发活动的变量。

[\T7]

Trigg 7

数据类型：triggdata

利用指令TriggIO、TriggEquip、TriggInt、TriggCheckIO、TriggSpeed或TriggRampAO较早期地在程序中定义涉及触发条件和触发活动的变量。

[\T8]

Trigg 8

数据类型：triggdata

利用指令TriggIO、TriggEquip、TriggInt、TriggCheckIO、TriggSpeed或TriggRampAO较早期地在程序中定义涉及触发条件和触发活动的变量。

Zone

数据类型：zonedata

相关移动的区域数据。区域数据描述了所生成拐角路径的大小。

下一页继续

1.289 TriggC - 关于事件的机械臂圆周移动
RobotWare - OS
续前页

[\Inpos]

In position

数据类型： stoppointdata

该参数用于规定停止点中机械臂TCP位置的收敛准则。停止点数据取代Zone参数中的指定区域。

Tool

数据类型： tooldata

当机器人移动时工具处于使用状态。工具中心点是移向指定目标点的点。

[\WObj]

Work Object

数据类型： wobjdata

指令中机器人位置关联的工件（坐标系）。

此边缘可以缺失，此时位置与世界坐标系关联。另一方面，如果使用了静态 TCP 或协调外部轴，则必须指定此变元才能执行相对此工件的线性移动。

[\Corr]

Correction

数据类型：switch

如果存在该参数，则将通过指令CorrWrite而写入修正条目的修正数据添加到路径和目的位置。

[\TLoad]

Total load

数据类型：loaddata

\TLoad主动轴描述了移动中使用的总负载。总负载就是相关的工具负载加上该工具正在处理的有效负载。如果使用了\TLoad自变数，那么就不考虑当前tooldata中的loaddata。

如果\TLoad自变数被设置成load0，那么就不考虑\TLoad自变数，而是以当前tooldata中的loaddata作为代替。

想要使用\TLoad自变数，就必需将系统参数ModalPayLoadMode的数值设置成0。如果将ModalPayLoadMode设置成0，那么就再也无法使用指令GripLoad。

可用服务例程“负载标识”（LoadIdentify）来识别总负载。如果系统参数ModalPayLoadMode被设置成0，且系统正在运行该服务例程，那么操作员便可将相关工具的loaddata复制到一个现有的或新的loaddata永久变量中。

如果使用了关联到系统输入项SimMode（仿真模式）上的一个数字输入信号，那么便可在没有任何有效负载的情况下试运行该程序。如果该数字输入信号被设置成1，那么就不考虑可选自变数\TLoad中的loaddata，而是以当前tooldata中的loaddata作为代替。

注意

处理有效负载的默认功能是使用指令GripLoad，因此系统参数ModalPayLoadMode的默认值为1。

下一页继续

程序执行

关于圆周运动的信息，请参见指令MoveC。

因为当机械臂定位越来越接近终点时，触发条件得以满足，因此，实施指定触发活动。在指令终点前的一定距离处、或在指令起点后一定距离处，或在指令终点前的特定时间点（限于短时间），触发条件得以满足。

逐步向前执行期间，实施I/O活动，但是并未运行中断程序。逐步向后执行期间，并未实施任何触发活动。

更多示例

有关于如何使用指令TriggC的更多例子阐述如下。

例 1

```
VAR intnum intno1;
VAR triggdata trigg1;
...
PROC main()
  ...
  CONNECT intno1 WITH trap1;
  TriggInt trigg1, 0.1 \Time, intno1;
  ...
  TriggC p1, p2, v500, trigg1, fine, gun1;
  TriggC p3, p4, v500, trigg1, fine, gun1;
  ...
  IDelete intno1;
```

当工作点分别在点p2或p4前的位置0.1 s处，运行中断程序trap1。

例 2

```
VAR num Distance:=0;
VAR triggdata trigg_array{25};
VAR signaldo myaliassignaldo;
VAR string signalname;
...
PROC main()
  ...
  FOR i FROM 1 TO 25 DO
    signalname:="do";
    signalname:=signalname+ValToStr(i);
    AliasIO signalname, myaliassignaldo;
    TriggEquip trigg_array{i}, Distance \Start, 0
         \DOp:=myaliassignaldo, SetValue:=1;
    Distance:=Distance+10;
  ENDFOR
  TriggC p1, p2, v500, trigg_array, z30, tool2;
  MoveC p3, p4, v500, z30, tool2;
  ...
```

在移动至p2期间，设置数字信号输出信号do1到do25。信号设置之间的距离为10 mm。

下一页继续

1.289 TriggC - 关于事件的机械臂圆周移动
RobotWare - OS
续前页

错误处理

如果在某些相关TriggSpeed指令中的指定模拟信号输出信号AOp的编程ScaleValue参数连同该指令中的编程Speed，导致模拟信号超出限制，则将系统变量ERRNO设置为ERR_AO_LIM。

如果与系统参数中使用的事件预设时间相比，某些相关TriggSpeed指令中的编程DipLag参数过大，则将系统变量ERRNO设置为ERR_DIPLAG_LIM。

如果在输入指令时，没有与I/O单元接触，则可将系统变量ERRNO设置为ERR_NORUNUNIT，使用的触发数据取决于运行中的I/O单元，即在触发数据中使用的信号。

如果已经超出了使用参数\Conc的连续运动指令数，则将系统变量ERRNO设置为ERR_CONC_MAX。

可用错误处理器来处理此类错误。

限制

符合指令MoveC的一般限制

如果当前起动点偏离正常点，以致指令TriggC的总定位长度短于正常水平，则若干或全部触发条件可能立即得以满足，并位于相同位置。在此类情况下，实施触发活动的顺序将变得不明确。用户程序中的程序逻辑可能不基于有关“不完整移动”触发活动的正常顺序。

警告

当机械臂位于圆点后的位置中时，不得从起点开始指令TriggC。否则，机械臂将不会选择编程路径（与编程相比，朝另一方向沿圆周路径定位）。

语法

```
TriggC
  [ ’\’ Conc ’,’]
  [ CirPoint’ :=’ ] < expression (IN) of robtarget > ’,’
  [ ToPoint’ :=’ ] < expression (IN) of robtarget > ’,’
  [ ’\’ ID ’:=’ < expression (IN) of identno >] ’,’
  [ Speed ’:=’ ] < expression (IN) of speeddata >
  [ ’\’ T ’:=’ < expression (IN) of num > ] ’,’
  [Trigg_1 ’:=’ ] < variable (VAR) of triggdata > |
  [TriggArray ’:=’ ] < array variable {*} (VAR) of triggdata >
  [ ’\’ T2 ’:=’ < variable (VAR) of triggdata > ]
  [ ’\’ T3 ’:=’ < variable (VAR) of triggdata > ]
  [ ’\’ T4 ’:=’ < variable (VAR) of triggdata > ]
  [ ’\’ T5 ’:=’ < variable (VAR) of triggdata > ]
  [ ’\’ T6 ’:=’ < variable (VAR) of triggdata > ]
  [ ’\’ T7 ’:=’ < variable (VAR) of triggdata > ]
  [ ’\’ T8 ’:=’ < variable (VAR) of triggdata > ] ´,’
  [Zone ’:=’ ] < expression (IN) of zonedata >
  [ ’\’ Inpos’ :=’ < expression (IN) of stoppointdata > ]´,’
  [ Tool ’:=’ ] < persistent (PERS) of tooldata >
  [ ’\’ WObj’ :=’ < persistent (PERS) of wobjdata > ]
  [ ’\’ Corr ]
```

下一页继续

```
[ '\' TLoad' :=' < persistent (PERS) of loaddata > ] ';'
```

相关信息

信息，关于	请参阅
触发时的线性移动	第*785*页的*TriggL - 关于事件的机械臂线性运动*
触发时的接头移动	第*778*页的*TriggJ - 关于事件的轴式机械臂运动*
使机械臂沿圆周移动	第*341*页的*MoveC - 使机械臂沿圆周移动*
触发的定义	第*773*页的*TriggIO - 定义停止点附近的固定位置或时间I/O事件* 第*763*页的*TriggEquip - 定义路径上的固定位置和时间I/O事件* 第*769*页的*TriggInt - 定义与位置相关的中断* 第*754*页的*TriggCheckIO - 定义位于固定位置的IO检查* 第*804*页的*TriggRampAO - 定义路径上的固定位置斜坡AO事件* 第*810*页的*TriggSpeed - 定义与固定位置-时间尺度事件成比例的TCP速度模拟信号输出*
处理`triggdata`	第*1510*页的触发数据 - 定位事件，触发 第*761*页的*TriggDataReset - 重置触发数据变量中的内容* 第*759*页的*TriggDataCopy - 复制触发数据变量中的内容* 第*1284*页的*TriggDataValid - 检查触发数据变量中的内容是否有效*
写入纠正条目	第*137*页的*CorrWrite - 写入修正发电机*
圆形运动	技术参考手册 - *RAPID*语言概览
载荷的定义	第*1421*页的*loaddata - 加载数据*
速度的定义	第*1480*页的*speeddata - 速度数据*
停止点数据的定义	第*1483*页的*stoppointdata - 停止点数据*
工具的定义	第*1502*页的*tooldata - 工具数据*
工件的定义	第*1523*页的*wobjdata - 工件数据*
区域数据的定义	第*1531*页的*zonedata - 区域数据*
一般动作	技术参考手册 - *RAPID*语言概览
关于如何使用`TLoad`总负载的例子。	第*388*页的*MoveL - 使机械臂沿直线移动*
定义机器人的载荷	第*223*页的*GripLoad - 定义机械臂的有效负载*
LoadIdentify，载荷识别服务例行程序	操作员手册 - 带 *FlexPendant* 的 *IRC5*
用于在仿真模式下无载荷运行机器人的系统输入信号 *SimMode*。 （主题 I/O，类型 System Input，行动值，*SimMode*）	技术参考手册 - 系统参数
激活和停用载荷的系统参数 *ModalPayLoadMode*。 （主题 Controller，类型 System Misc，行动值，*ModalPayLoadMode*）	技术参考手册 - 系统参数

1.290 TriggCheckIO - 定义位于固定位置的IO检查

手册用法

TriggCheckIO用于定义有关测试机械臂移动路径沿线固定位置处的数字、数字组值或模拟信号输入或输出信号的条件。如果本条件得以满足，则将不会存在特定行动。否则，在机械臂已尽快在路径上随意停止后，将运行中断程序。

为获得固定位置I/O检查，TriggCheckIO对控制系统中的滞后（伺服与机械臂之间的滞后）进行补偿。

使用确定的数据，以供在后续TriggL、TriggC或TriggJ 指令之一或多个中实施。

本指令仅可用于主任务T_ROB1，或者如果在*MultiMove*系统中，则可用于运动任务中。

基本示例

以下实例介绍了指令TriggCheckIO：

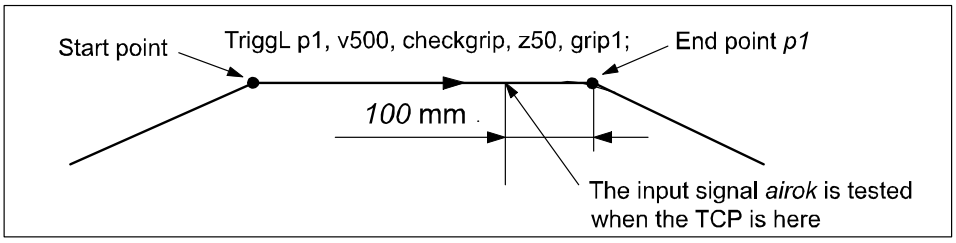
另请参阅第*757*页的更多示例

例 1

```
VAR triggdata checkgrip;
VAR intnum intno1;

PROC main()
  CONNECT intno1 WITH trap1;
  TriggCheckIO checkgrip, 100, airok, EQ, 1, intno1;

  TriggL p1, v500, checkgrip, z50, grip1;
```

当TCP位于点p1之前100 mm时，检查数字信号输入信号airok ，以获得值1。如果该信号得以设置，则继续正常执行程序。若未设置，则运行中断程序trap1。

本图显示了固定位置I/O检查的实例。



xx0500002254

变元

```
TriggCheckIO TriggData Distance [\Start] | [\Time] Signal Relation
     CheckValue | CheckDvalue [\StopMove] Interrupt [\Inhib]
     [\Mode]
```

TriggData

数据类型：triggdata

从该指令返回用于储存triggdata的变量。随后，将此类triggdata 用于TriggL、TriggC或TriggJ指令。

下一页继续

Distance

数据类型：num

定义应当在路径上进行I/O检查的位置。

定义为距移动路径终点的距离，以mm计（正值）（未设置参数\Start或\Time时适用）。

有关更多细节，请参见第*756*页的程序执行。

[\Start]

数据类型：switch

当关于参数Distance的距离始于移动起点而非终点时使用。

[\Time]

数据类型：switch

当关于参数Distance的指定值实际为时间（以秒计，且未正值）而非距离时使用。

在机械臂达到本指令终点前，以时间表示的固定位置I/O仅可用于短时间（<0.5 s）。欲知更多细节，参见限制一节。

Signal

数据类型：signalxx

将测试的信号的名称。可能为任意类型的IO信号。

Relation

数据类型：opnum

规定如何将信号的实际值与参数CheckValue定义的值进行比较。关于待使用的预定义常量列表，参见opnum数据类型。

CheckValue

数据类型：num

与输入或输出信号的实际值进行比较的值（在当前信号的容许范围内）。如果信号为数字信号，则其必须为整数值。

如果信号为数字组信号，则容许值取决于该组中信号的数量。可在CheckValue参数中使用的最大值为8388608，且该值为23位数字组信号可拥有的最大值（参见num的范围）。

CheckDvalue

数据类型：dnum

与输入或输出信号的实际值进行比较的值（在当前信号的容许范围内）。如果信号为数字信号，则其必须为整数值。

如果信号为数字组信号，则容许值取决于该组中信号的数量。一个数字组信号可拥有的最大信号位为32。通过dnum变量，可能覆盖值范围0-4294967295，其为32位数字信号可拥有的值范围。

[\StopMove]

数据类型：switch

规定如果未满足条件，则在运行中断程序之前，机械臂将在路径上尽快停止。

下一页继续

1.290 TriggCheckIO - 定义位于固定位置的IO检查

RobotWare - OS

续前页

`Interrupt`

数据类型：`intnum`

用于确定运行中断程序的变量。

`[ \Inhib ]`

Inhibit

数据类型：`bool`

抑制中断子程序执行的永久变量标记的名称。

如果采用此项可选参数并且在到达I/O检查位置-时间，指定标记的实际值为真，那么不会开展检查。

`[ \Mode ]`

数据类型：`triggmode`

用于在定义触发器时指定不同的行动模式。

程序执行

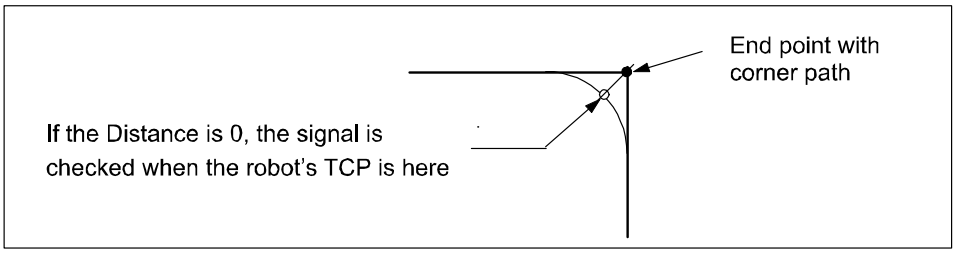
当运行指令`TriggCheckIO`时，将触发条件储存在参数`TriggData`的指定变量中。

此后，当执行指令`TriggL`、`TriggC`或`TriggJ`之一时，以下方面适用于`TriggCheckIO`中的定义：

本表描述了参数`Distance`中指定的距离：

线性移动	直线距离
圆形运动	圆弧长度
非线性移动	沿路径的适当弧长（以获取适当的准确度，距离不得超过弧长的一半）。

本图显示了位于角路径上的固定位置I/O检查。



xx0500002256

如果距终点（起点）的指定距离不在当前指令（`TriggL...`）的移动长度内，则当通过起点（终点）时，将完成固定位置I/O检查。

当机械臂的TCP位于路径上的指定位置时，将通过系统进行以下I/O检查：

- 读取I/O信号的值。
- 根据指定的`Relation`，将读取值与`CheckValue`进行比较。
- 如果比较结果为TRUE，则未进行更多的工作。
- 如果比较结果为FALSE，则进行以下工作：
- 如果存在可选参数`\StopMove`，则机械臂在路径上尽快停止。
- 生成和执行指定的软中断程序。

下一页继续

更多示例

有关于如何使用指令TriggCheckIO的更多例子阐述如下。

例 1

```
VAR triggdata checkgate;
VAR intnum gateclosed;

PROC main()
  CONNECT gateclosed WITH waitgate;
  TriggCheckIO checkgate,150, gatedi, EQ, 1 \StopMove, gateclosed;
  TriggL p1, v600, checkgate, z50, grip1;
  ...
TRAP waitgate
  ! Block movement
  StopMove;
  ! log some information
  ...
  ! Wait until signal is set
  WaitDI gatedi,1;
  ! Unlock block, and resume movement
  StartMove;
ENDTRAP
```

当TCP位于点p1前150 mm时，检查下一工件操作的大门是否打开（检查数字信号输入信号gatedi 是否拥有值1）。如果其打开，则机械臂将会移动至p1并继续。如果未打开，则机械臂在路径上停止，并运行中断程序waitgate。该中断阻碍进一步的移动，记录一些信息，且通常等待条件变好，以执行StartMove指令，从而重启已中断的路径。

错误处理

系统会生成下列可恢复错误，并在错误处理器中处理这些错误。系统变量ERRNO将被设置成：

ERR_AO_LIM

如果关于指定模拟信号输出信号Signal的编程CheckValue或CheckDvalue参数超出限制。

ERR_GO_LIM

如果关于指定数字组输出信号Signal的编程CheckValue或CheckDvalue参数超出限制。

ERR_NO_ALIASIO_DEF

如果信号变量是RAPID中声明的变量，且尚未同I/O配置以及指令AliasIO中确定的I/O信号相连。

限制

I/O检查以及距离（无参数\Time）旨在供飞焊点（角路径）使用。使用停止点的I/O检查以及距离，导致精度低于规定水平。

I/O检查以及时间（含参数\Time）旨在供飞焊点使用。使用飞焊点的I/O检查以及时间，导致精度低于规定水平。

下一页继续

1.290 TriggCheckIO - 定义位于固定位置的IO检查

RobotWare - OS

续前页

仅可根据移动终点，指定I/O检查以及时间。该时间不能超出机械臂当前的制动时间，其最大约为0.5 s（速度为500 mm/s时，IRB2400的典型值为150 ms，且IRB6400的典型值为250 ms）。如果指定时间大于当前的制动时间，则直至开始制动（迟于指定值），方才产生I/O检查。在小而快速的移动期间，可利用当前移动的整个移动时间。

用于测试数字信号输入+/- 5 ms的典型绝对精度值。用于测试数字信号输入+/- 2 ms的典型重复精度值。

语法

```
TriggCheckIO
  [ TriggData ':=' ] < variable (VAR) of triggdata> ','
  [ Distance' :=' ] < expression (IN) of num>
  [ '\' Start ] | [ '\' Time ] ','
  [ Signal ':=' ] < variable (VAR) of anytype> ','
  [ Relation' :=' ] < expression (IN) of opnum> ','
  [ CheckValue' :=' ] < expression (IN) of num>
  | [ CheckDvalue' :=' ] < expression (IN) of dnum>
  [ '\' StopMove] ','
  [ Interrupt' :=' ] < variable(VAR) of intnum>
  [ '\' Inhib' :=' < persistent (PERS) of bool> ]
  [ Mode' :=' ] < expression (IN) of triggmode> ';'
```

相关信息

信息，关于	请参阅
触发的使用	第*785*页的*TriggL* - 关于事件的机械臂线性运动 第*747*页的*TriggC* - 关于事件的机械臂圆周移动 第*778*页的*TriggJ* - 关于事件的轴式机械臂运动
位置-时间I/O事件的定义	第*773*页的*TriggIO* - 定义停止点附近的固定位置或时间I/O事件 第*763*页的*TriggEquip* - 定义路径上的固定位置和时间I/O事件
位置相关中断的定义	第*769*页的*TriggInt* - 定义与位置相关的中断
触发数据的储存	第*1510*页的触发数据 - 定位事件，触发 第*1515*页的*triggmode* - 触发行动模式
比较运算符的定义	第*1438*页的*opnum* - 比较运算符

1.291 TriggDataCopy - 复制触发数据变量中的内容

手册用法

TriggDataCopy用于复制triggdata变量中的内容。

基本示例

以下实例介绍了指令TriggDataCopy。

例 1

```
VAR triggdata trigg_array{25};
...
PROC MyTriggProcL(robtarget myrobt, \VAR triggdata T1 \VAR triggdata
     T2 \VAR triggdata T3)
  VAR num triggcnt:=2;
  ! Reset entire trigg_array array before using it
  FOR i FROM 1 TO 25 DO
    TriggDataReset trigg_array{i};
  ENDFOR
  TriggEquip trigg_array{1}, 10 \Start, 0 \DOp:=do1, SetValue:=1;
  TriggEquip trigg_array{2}, 40 \Start, 0 \DOp:=do2, SetValue:=1;
  ! Check if optional argument is present,
  ! and if any trigger condition has been setup in T1
  IF Present(T1) AND TriggDataValid(T1) THEN
    ! Copy actual trigger condition to trigg_array
    TriggDataCopy T1, trigg_array{triggcnt};
    Incr triggcnt;
  ENDIF
  IF Present(T2) AND TriggDataValid(T2) THEN
    Incr triggcnt;
    TriggDataCopy T2, trigg_array{triggcnt};
  ENDIF
  IF Present(T3) AND TriggDataValid(T3) THEN
    Incr triggcnt;
    TriggDataCopy T3, trigg_array{triggcnt};
  ENDIF
  TriggL p1, v500, trigg_array, z30, tool2;
  ...
```

上述无返回值程序MyTriggProcL使用TriggDataCopy指令，以将triggdata可选参数复制到TriggL指令所用triggdata数组中的正确位置。

变元

```
TriggDataCopy Source Destination
```

Source

数据类型：triggdata

从triggdata变量中复制。

Destination

数据类型：triggdata

下一页继续

1.291 TriggDataCopy - 复制触发数据变量中的内容

续前页

复制到triggdata变量。

程序执行

TriggDataCopy指令用于将数据从一个triggdata变量复制到另一个triggdata变量。当与triggdata数组变量共同起作用时，该指令可能有用。

语法

```
TriggDataCopy
  [Source ':=' ] < variable (VAR) of triggdata > ','
  [Destination ':=' ] < variable (VAR) of triggdata > ';'
```

相关信息

信息，关于	请参阅
触发时的线性移动	第*785*页的*TriggL* - 关于事件的机械臂线性运动
触发时的接头移动	第*778*页的*TriggJ* - 关于事件的轴式机械臂运动
触发时的圆周移动	第*747*页的*TriggC* - 关于事件的机械臂圆周移动
触发的定义	第*773*页的*TriggIO* - 定义停止点附近的固定位置或时间*I/O*事件 第*763*页的*TriggEquip* - 定义路径上的固定位置和时间*I/O*事件 第*769*页的*TriggInt* - 定义与位置相关的中断 第*754*页的*TriggCheckIO* - 定义位于固定位置的*IO*检查 第*804*页的*TriggRampAO* - 定义路径上的固定位置斜坡*AO*事件 第*810*页的*TriggSpeed* - 定义与固定位置-时间尺度事件成比例的*TCP*速度模拟信号输出
处理triggdata	第*1510*页的触发数据 - 定位事件，触发 第*761*页的*TriggDataReset* - 重置触发数据变量中的内容 第*1284*页的*TriggDataValid* - 检查触发数据变量中的内容是否有效

1.292 TriggDataReset - 重置触发数据变量中的内容

手册用法

TriggDataReset用于重置triggdata变量中的内容。

基本示例

以下实例介绍了指令TriggDataReset。

例 1

```
VAR triggdata trigg_array{25};
...
PROC MyTriggProcL(robtarget myrobt, \VAR triggdata T1 \VAR triggdata
     T2 \VAR triggdata T3)
  VAR num triggcnt:=2;
  ! Reset entire trigg_array array before using it
  FOR i FROM 1 TO 25 DO
    TriggDataReset trigg_array{i};
  ENDFOR
  TriggEquip trigg_array{1}, 10 \Start, 0 \DOp:=do1, SetValue:=1;
  TriggEquip trigg_array{2}, 40 \Start, 0 \DOp:=do2, SetValue:=1;
  ! Check if optional argument is present,
  ! and if any trigger condition has been setup in T1
  IF Present(T1) AND TriggDataValid(T1) THEN
    ! Copy actual trigger condition to trigg_array
    TriggDataCopy trigg_array{triggcnt}, T1;
    Incr triggcnt;
  ENDIF
  IF Present(T2) AND TriggDataValid(T2) THEN
    Incr triggcnt;
    TriggDataCopy trigg_array{triggcnt}, T2;
  ENDIF
  IF Present(T3) AND TriggDataValid(T3) THEN
    Incr triggcnt;
    TriggDataCopy trigg_array{triggcnt}, T3;
  ENDIF
  TriggL p1, v500, trigg_array, z30, tool2;
  ...
```

在使用triggdata数组前，上述无返回值程序MyTriggProcL使用TriggDataReset指令来重置该数组。

变元

```
TriggDataReset TriggData
```

TriggData

数据类型：triggdata

用于重置的triggdata变量。

下一页继续

1.292 TriggDataReset - 重置触发数据变量中的内容

续前页

程序执行

TriggDataReset指令用于清除先前在triggdata变量中使用的所有触发条件。当与triggdata数组变量共同起作用时，该指令有用。

语法

```
TriggDataReset
  [TriggData ':=' ] < variable (VAR) of triggdata > ';'
```

相关信息

信息，关于	请参阅
触发时的线性移动	第*785*页的*TriggL* - 关于事件的机械臂线性运动
触发时的接头移动	第*778*页的*TriggJ* - 关于事件的轴式机械臂运动
触发时的圆周移动	第*747*页的*TriggC* - 关于事件的机械臂圆周移动
触发的定义	第*773*页的*TriggIO* - 定义停止点附近的固定位置或时间*I/O*事件 第*763*页的*TriggEquip* - 定义路径上的固定位置和时间*I/O*事件 第*769*页的*TriggInt* - 定义与位置相关的中断 第*754*页的*TriggCheckIO* - 定义位于固定位置的*IO*检查 第*804*页的*TriggRampAO* - 定义路径上的固定位置斜坡*AO*事件 第*810*页的*TriggSpeed* - 定义与固定位置-时间尺度事件成比例的*TCP*速度模拟信号输出
处理triggdata	第*1510*页的触发数据 - 定位事件，触发 第*759*页的*TriggDataCopy* - 复制触发数据变量中的内容 第*1284*页的*TriggDataValid* - 检查触发数据变量中的内容是否有效

1.293 TriggEquip - 定义路径上的固定位置和时间I/O事件

手册用法

TriggEquip（*Trigg Equipment*）用于定义有关设置机械臂移动路径沿线固定位置处的数字、数字组或模拟信号输出信号的条件和行动以及对外部设备中的滞后进行时间补偿的可能性。

如果停止点附近的I/O设置需要较高的精度，则应当始终使用TriggIO（而非TriggEquip）。

使用确定的数据，以供在后续TriggL、TriggC或TriggJ 指令之一或多个中实施。

本指令仅可用于主任务T_ROB1，或者如果在*MultiMove*系统中，则可用于运动任务中。

基本示例

以下实例介绍了指令TriggEquip：

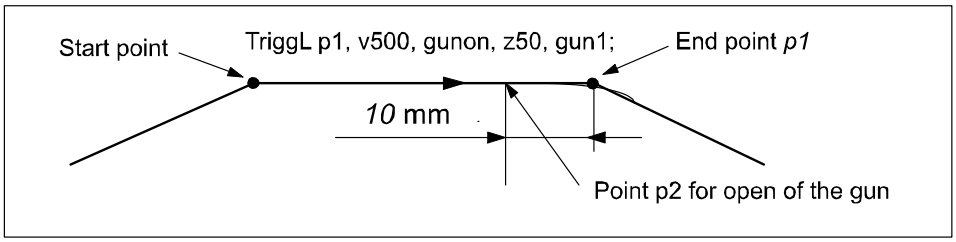
另请参阅第*766*页的更多示例

例 1

```
VAR triggdata gunon;
...
TriggEquip gunon, 10, 0.1 \DOp:=gun, 1;
TriggL p1, v500, gunon, z50, gun1;
```

当其TCP位于虚构点p2前0,1 s时（位于点p1前10 mm），工具gun1开始打开。当TCP达到点p2时，焊枪完全打开。

本图显示了固定位置时间I/O事件的实例。



xx0500002260

变元

```
TriggEquip TriggData Distance [\Start] | [\Next] EquipLag [\DOp]
     | [\GOp]| [\AOp] | [\ProcID] SetValue | SetDvalue [\Inhib]
     [\InhibSetValue] [\Mode]
```

TriggData

数据类型：triggdata

从该指令返回用于储存triggdata的变量。随后，将此类triggdata 用于TriggL、TriggC或TriggJ指令。

Distance

数据类型：num

定义在路径上应出现I/O设备事件的位置。

下一页继续

1.293 TriggEquip - 定义路径上的固定位置和时间I/O事件
RobotWare - OS
续前页

指定为运动路径终点到起点之间的距离（正值，单位：毫米）（在未设置参数\Start和\Next的情况下适用）。

有关更多细节，请参见第765页的程序执行。

[\Start]

数据类型：switch

当关于参数Distance的距离始于移动起点而非终点时使用。

[\Next]

数据类型：switch

当参数Distance的距离朝前指向下一编程点时使用。如果Distance长于到下一精确点的距离，那么将在精确点处执行此事件。

EquipLag

Equipment Lag

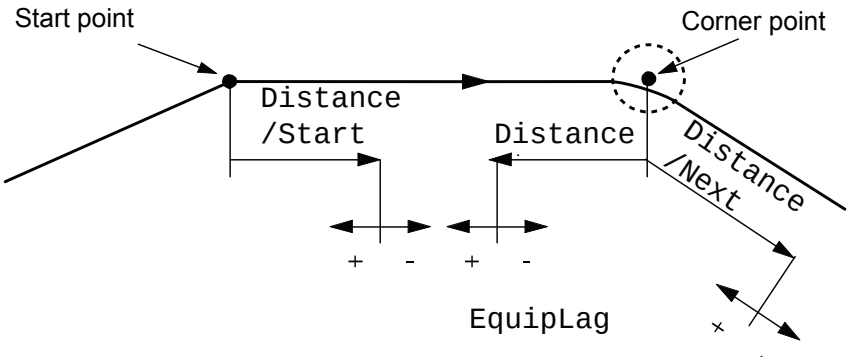
数据类型：num

指定外部设备的滞后，以s计。

关于外部设备滞后补偿，使用正参数值。正参数值意味着I/O信号由TCP达到相对于移动起点或终点的指定距离前一指定时间的机器人系统来设置。

负参数值意味着I/O信号由TCP已通过相对于移动起点或终点的指定距离后一指定时间的机器人系统来设置。

本图显示了参数EquipLag的使用。



xx0500002262

[\DOp]

Digital Output

数据类型：signaldo

应当改变数字信号输出信号时的信号名称。

[\GOp]

Group Output

数据类型：signalgo

应当改变一组数字信号输出信号时的信号名称。

[\AOp]

Analog Output

下一页继续

数据类型：signalao

应当改变模拟信号输出信号时的信号名称。

[\ProcID]

Process Identity

数据类型：num

未针对客户使用而实施。

（用于接收事件的IPM工艺的识别号。通过参数SetValue，指定选择器。）

SetValue

数据类型：num

信号的期望值（处于当前信号的容许范围内）。如果信号为数字信号，则其必须为整数值。如果信号为数字组信号，则容许值取决于该组中信号的数量。可在SetValue参数中使用的最大值为8388608，且该值为23位数字组信号可拥有的最大值（参见num的范围）。

SetDvalue

数据类型：dnum

信号的期望值（处于当前信号的容许范围内）。如果信号为数字信号，则其必须为整数值。如果信号为数字组信号，则容许值取决于该组中信号的数量。一个数字组信号可拥有的最大信号位为32。通过dnum变量，可能覆盖值范围0-4294967295，其为32位数字信号可拥有的值范围。

[\Inhib]

Inhibit

数据类型：bool

用于约束运行时信号设置的永久变量标志的名称。

如果在用于设置信号的位置-时间下，使用该可选参数，且指定标志的实际值为TRUE，则将指定信号（DOp、GOp或AOp）设置为0而非指定值。

[\InhibSetValue]

InhibitSetValue

数据类型：bool, num or dnum

数据类型bool、num或dnum的永久变量的名称，或这三个基础数据类型的别名。

此可选参数只可结合可选参数Inhib一起使用。

如果采用此可选参数，并且在到达设置信号的位置-时间，可选参数Inhib采用的永久变量标记的值为真，那么将读取可选参数InhibSetValue采用的永久变量值，将采用该值来设置DOp、GOp或AOp信号。

如果采用布尔永久变量，那么值“真”将被转换为值“1”，“假”将被转换为值“0”。

[\Mode]

数据类型：triggmode

用于在定义触发器时指定不同的行动模式。

程序执行

当运行指令TriggEquip时，将触发条件储存在参数TriggData的指定变量中。

下一页继续

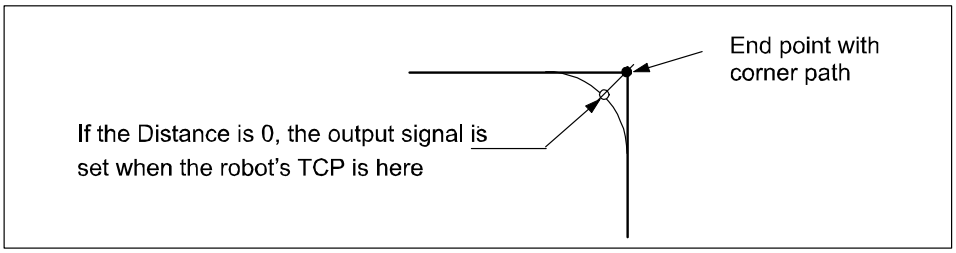
1.293 TriggEquip - 定义路径上的固定位置和时间I/O事件
RobotWare - OS
续前页

此后，当执行指令TriggL、TriggC或TriggJ之一时，则以下方面适用于TriggEquip中的定义：

本表描述了参数Distance中指定的距离：

线性移动	直线距离
圆形运动	圆弧长度
非线性移动	沿路径的适当弧长（以获取适当的准确度，距离不得超过弧长的一半）。

本图显示了位于角路径上的固定位置时间I/O。



xx0500002263

如果距终点（起点）的指定距离不在当前指令（TriggL...）的移动长度内，则当通过起点（终点）时，将产生与位置-时间相关的事件。通过使用参数EquipLag以及负时间（延迟），可在终点后设置I/O信号。

更多示例

有关于如何使用指令TriggEquip的更多例子阐述如下。

例 1

```
VAR triggdata glueflow;
...
TriggEquip glueflow, 1 \Start, 0.05 \AOp:=glue, 5.3;
MoveJ p1, v1000, z50, tool1;
TriggL p2, v500, glueflow, z50, tool1;
```

当TCP通过位于起点p1 后1 mm处的点时，将模拟信号输出信号glue 设置为值5.3，且设备滞后补偿为0.05 s。

例 2

```
...
TriggL p3, v500, glueflow, z50, tool1;
```

当TCP通过位于起点p2后1 mm处的点时，再次将模拟信号输出信号glue 设置为值5.3 。

错误处理

如果关于指定模拟信号输出信号AOp的编程SetValue参数超出限制，则将系统变量ERRNO设置为ERR_AO_LIM。可通过错误处理器来处理该错误。

如果关于指定数字组输出信号GOp的编程SetValue或SetDvalue参数超出限制，则将系统变量ERRNO设置为ERR_GO_LIM。可通过错误处理器来处理该错误。

下一页继续

如果信号变量是在RAPID中声明的变量，且其尚未与通过指令AliasIO而在I/O配置中确定的I/O信号相连，随后，将系统变量ERRNO设置为ERR_NO_ALIASIO_DEF。可通过错误处理器来处理该错误。

限制

I/O事件以及距离旨在供飞焊点（角路径）使用。使用停止点的I/O事件以及距离，导致精度低于规定水平。

关于I/O事件以及距离的精度以及飞焊点的使用，当通过指令TriggL或TriggC，在距起点或终点指定距离处设置数字信号输出时，以下方面适用：

- 以下指定的精度适用于正EquipLag 参数 < 40 ms，相当于机械臂伺服中的滞后（没有改变系统参数Event Preset Time）。不同的机械臂类型，滞后可能有所不同。例如，IRB140的滞后较低。
- 以下指定的精度适用于正EquipLag 参数 < 配置的 Event Preset Time（系统参数）。
- 以下指定的精度不适用于正EquipLag 参数 > 配置的Event Preset Time（系统参数）。在这种情况下，使用适当的方法，该方法不考虑机械臂的动力学限制。必须使用SingArea \Wrist，以获得可接受的精度。
- 以下指定的精度适用于负EquipLag。

用于设置数字信号输出+/- 5 ms的典型绝对精度值。

用于设置数字信号输出+/- 2 ms的典型重复精度值。

语法

```
TriggEquip
  [ TriggData ':=' ] < variable (VAR) of triggdata> ','
  [ Distance' :=' ] < expression (IN) of num>
  [ '\' Start ]
  | [ '\' Next ] ','
  [ EquipLag' :=' ] < expression (IN) of num>
  [ '\' DOp' :=' < variable (VAR) of signaldo> ]
  | [ '\' GOp' :=' < variable (VAR) of signalgo> ]
  | [ '\' AOp' :=' < variable (VAR) of signalao> ]
  | [ '\' ProcID' :=' < expression (IN) of num> ] ','
  [ SetValue' :=' ] < expression (IN) of num>
  | [ SetDvalue' :=' ] < expression (IN) of dnum> ','
  [ '\' Inhib' :=' < persistent (PERS) of bool> ]
  [ '\' InhibSetValue' :=' < persistent (PERS) of anytype> ]
  [ Mode' :=' ] < expression (IN) of triggmode> ';'
```

相关信息

信息，关于	请参阅
触发的使用	第*785*页的*TriggL - 关于事件的机械臂线性运动* 第*747*页的*TriggC - 关于事件的机械臂圆周移动* 第*778*页的*TriggJ - 关于事件的轴式机械臂运动*
其他触发的定义	第*773*页的*TriggIO - 定义停止点附近的固定位置或时间I/O事件* 第*769*页的*TriggInt - 定义与位置相关的中断*

下一页继续

1.293 TriggEquip - 定义路径上的固定位置和时间I/O事件
RobotWare - OS
续前页

信息，关于	请参阅
定义固定位置的I/O检查	第*754*页的*TriggCheckIO* - 定义位于固定位置的*IO*检查
触发数据的储存	第*1510*页的触发数据 - 定位事件，触发 第*1515*页的*triggmode* - 触发行动模式
I/O的设置	第*589*页的*SetDO* - 改变数字信号输出信号值 第*591*页的*SetGO* - 改变一组数字信号输出信号的值 第*580*页的*SetAO* - 改变模拟信号输出信号的值
Event preset time的配置	技术参考手册 - 系统参数

1.294 TriggInt - 定义与位置相关的中断

手册用法

TriggInt用于定义条件和行动，从而在机械臂移动路径上的指定位置运行中断程序。使用确定的数据，以供在后续TriggL、TriggC或TriggJ 指令之一或多个中实施。本指令仅可用于主任务T_ROB1，或者如果在MultiMove系统中，则可用于运动任务中。

基本示例

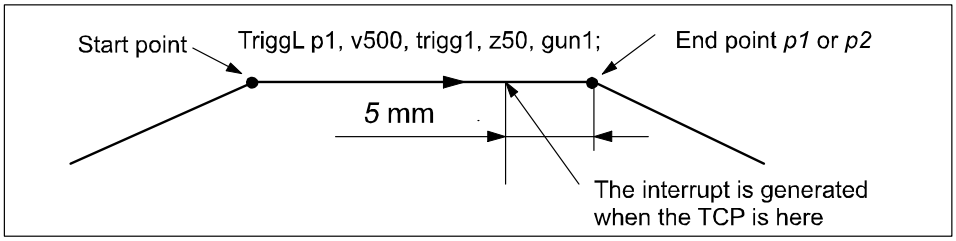
以下实例介绍了指令TriggInt：

例 1

```
VAR intnum intno1;
VAR triggdata trigg1;
...
PROC main()
  CONNECT intno1 WITH trap1;
  TriggInt trigg1, 5, intno1;
  ...
  TriggL p1, v500, trigg1, z50, gun1;
  TriggL p2, v500, trigg1, z50, gun1;
  ...
  IDelete intno1;
```

当TCP分别位于点p1 或p2前5mm处的位置时，运行中断程序trap1。

本图显示了位置相关中断的实例。



xx0500002251

变元

```
TriggInt TriggData Distance [\Start] | [\Time] Interrupt [\Inhib]
     [\Mode]
```

TriggData

数据类型：triggdata

从该指令返回用于储存triggdata的变量。随后，将此类triggdata 用于TriggL、TriggC或TriggJ指令。

Distance

数据类型：num

定义应当在路径上产生中断的位置。

下一页继续

1.294 TriggInt - 定义与位置相关的中断
RobotWare - OS
续前页

定义为距移动路径终点的距离，以mm计（正值）（未设置参数\Start或\Time时适用）。

有关更多细节，请参见第770页的程序执行。

[\Start]

数据类型：switch

当关于参数Distance的距离始于移动起点而非终点时使用。

[\Time]

数据类型：switch

当关于参数Distance的指定值实际为时间（以秒计，且未正值）而非距离时使用。

在机械臂达到本指令终点前，以时间表示的位置相关中断仅可用于短时间（<0.5 s）。欲知更多细节，参见限制一节。

Interrupt

数据类型：intnum

用于确定中断的变量。

[\Inhib]

Inhibit

数据类型：bool

抑制中断子程序执行的永久变量标记的名称。

如果采用此项可选参数并且在到达中断执行位置-时间，指定标记的实际值为真，那么不会执行此中断。

[\Mode]

数据类型：triggmode

用于在定义触发器时指定不同的行动模式。

程序执行

当运行指令TriggInt时，将数据储存在有关参数TriggData的指定变量中，并启用有关参数Interrupt的变量中的指定中断。

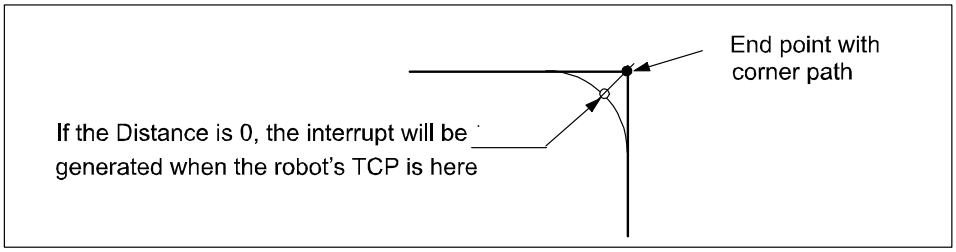
此后，当执行指令TriggL、TriggC或TriggJ之一时，以下方面适用于TriggInt中的定义：

本表描述了参数Distance中指定的距离：

线性移动	直线距离
圆形运动	圆弧长度
非线性移动	沿路径的适当弧长（以获取适当的准确度，距离不得超过弧长的一半）。

下一页继续

本图显示了位于角路径上的位置相关中断。



xx0500002253

如果距终点（起点）的指定距离不在当前指令（TriggL...）的移动长度内，则当通过起点（终点）时，将产生位置相关中断。

本中断被认作是安全中断。安全中断无法同指令ISleep一同处于休眠模式。程序停止时，安全中断事件将列入队列，并逐步执行，且当再次以连续模式启动时，将执行中断。弃用安全中断的唯一时刻便是中断队列已满。随后，将报告错误。本中断将无法经历程序重置，例如，PP到Main。

更多示例

有关于如何使用指令TriggInt的更多例子阐述如下。

例 1

本实例描述了各指令的编程，此类指令相互作用，以产生同位置相关的中断：

```
VAR intnum intno2;
VAR triggdata trigg2;
```

- 不得开始变量intno2和trigg2（的声明）。

```
CONNECT intno2 WITH trap2;
```

- 储存在变量intno2中的中断编号的配置。
- 同中断程序trap2耦合的中断编号。

```
TriggInt trigg2, 0, intno2;
```

- 将变量intno2中的中断编号标记为已使用。
- 启用中断。
- 将确定的触发条件和中断编号储存在变量trigg2中

```
TriggL p1, v500, trigg2, z50, gun1;
```

- 使机械臂移动至点p1。
- 当TCP达到点p1时，产生中断，并运行中断程序trap2。

```
TriggL p2, v500, trigg2, z50, gun1;
```

- 使机械臂移动至点p2。
- 当TCP达到点p2时，产生中断，并再次运行中断程序trap2。

```
IDelete intno2;
```

- 解除分配变量intno2中的中断编号。

限制

中断事件以及距离（无参数\Time）旨在供飞焊点（角路径）使用。使用停止点的中断事件以及距离，导致精度低于规定水平。

下一页继续

1.294 TriggInt - 定义与位置相关的中断

RobotWare - OS

续前页

中断事件以及时间（含参数\Time）旨在供停止点使用。使用飞焊点的中断事件以及时间，导致精度低于规定水平。仅可根据移动终点，指定I/O事件以及时间。该时间不能超出机械臂当前的制动时间，其最大约为0.5 s（速度为500 mm/s时，IRB2400的典型值为150 ms，且IRB6400的典型值为250 ms）。如果指定时间大于当前的制动时间，则直至开始制动（迟于指定值），方才产生本事件。在小而快速的移动期间，可利用当前移动的整个移动时间。

关于产生中断+/- 5 ms的典型绝对精度值。关于产生中断+/- 2 ms的典型重复精度值。根据在中断时实施的移动的类型，在中断产生与响应之间通常存在2到30 ms的延迟（参考*RAPID*参考手册 - *RAPID*概述，基本特征 - 中断一节）。

为获得最佳精度，当设置沿机械臂路径固定位置处的输出时，使用指令TriggIO或TriggEquip，优先考虑中断程序中的指令TriggInt以及SetDO/SetGO/SetAO 。

语法

```
TriggInt
  [ TriggData ':=' ] < variable (VAR) of triggdata> ','
  [ Distance' :=' ] < expression (IN) of num>
  [ '\' Start ] | [ '\' Time ] ','
  [ Interrupt' :=' ] < variable (VAR) of intnum>
  [ '\' Inhib' :=' < persistent (PERS) of bool> ]
  [ Mode' :=' ] < expression (IN) of triggmode> ';'
```

相关信息

信息，关于	请参阅
触发的使用	第*785*页的*TriggL* - 关于事件的机械臂线性运动 第*747*页的*TriggC* - 关于事件的机械臂圆周移动 第*778*页的*TriggJ* - 关于事件的轴式机械臂运动
定位I/O的定义	第*773*页的*TriggIO* - 定义停止点附近的固定位置或时间*I/O*事件 第*763*页的*TriggEquip* - 定义路径上的固定位置和时间*I/O*事件
定义固定位置的I/O检查	第*754*页的*TriggCheckIO* - 定义位于固定位置的*IO*检查
触发数据的储存	第*1510*页的触发数据 - 定位事件，触发 第*1515*页的*triggmode* - 触发行动模式
中断	技术参考手册 - *RAPID*语言概览

1.295 TriggIO - 定义停止点附近的固定位置或时间I/O事件

手册用法

TriggIO用于定义有关设置机械臂移动路径沿线固定位置处的数字、数字组或模拟信号输出信号的条件和行动。

如果停止点附近的I/O设置需要较高精度，则应始终使用TriggIO（而非TriggEquip）。

为获得固定位置I/O事件，TriggIO补偿控制系统中的滞后（机械臂与伺服之间的滞后），而非外部设备中的滞后。为补偿上述两种滞后，应使用TriggEquip。

使用确定的数据，以供在后续TriggL、TriggC或TriggJ指令之一或多个中实施。

本指令仅可用于主T_ROB1 任务，或者如果在*MultiMove*系统中，则可用于运动任务中。

基本示例

以下实例介绍了指令TriggIO：

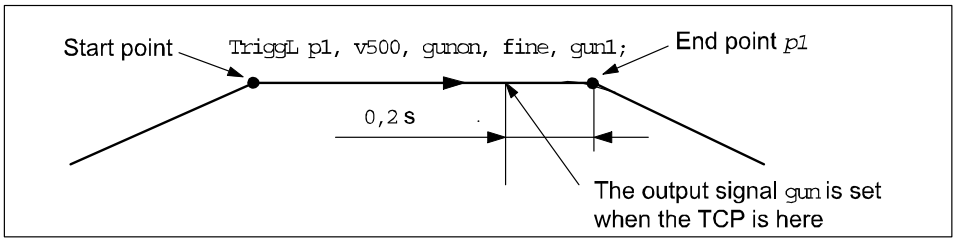
另请参阅第*776*页的更多示例

例 1

```
VAR triggdata gunon;
...
TriggIO gunon, 0.2\Time\DOp:=gun, 1;
TriggL p1, v500, gunon, fine, gun1;
```

当TCP位于点p1前0,2秒时，将数字信号输出信号gun 设置为值1。

本图显示了固定位置I/O事件的实例。



xx0500002247

变元

```
TriggIO TriggData Distance [\Start] | [\Time] [\DOp] | [\GOp]|
     [\AOp] | [\ProcID] SetValue | SetDvalue [\DODelay] [\Inhib]
     [\InhibSetValue] [\Mode]
```

TriggData

数据类型：triggdata

从该指令返回用于储存triggdata的变量。随后，将此类triggdata用于TriggL、TriggC或TriggJ指令。

Distance

数据类型：num

定义在路径上应出现I/O事件的位置。

下一页继续

1.295 TriggIO - 定义停止点附近的固定位置或时间I/O事件
RobotWare - OS
续前页

定义为距移动路径终点的距离，以mm计（正值）（未设置参数\Start或\Time时适用）。

有关更多的细节，参见第775页的程序执行以及第776页的限制一节。

[\Start]

数据类型：switch

当关于参数Distance的距离始于移动起点而非终点时使用。

[\Time]

数据类型：switch

当关于参数Distance的指定值实际为时间（以秒计，且未正值）而非距离时使用。

在机械臂达到本指令终点前，以时间表示的固定位置I/O仅可用于短时间（<0.5 s）。欲知更多细节，参见限制一节。

[\DOp]

Digital Output

数据类型：signaldo

应当改变数字信号输出信号时的信号名称。

[\GOp]

Group Output

数据类型：signalgo

应当改变一组数字信号输出信号时的信号名称。

[\AOp]

Analog Output

数据类型：signalao

应当改变模拟信号输出信号时的信号名称。

[\ProcID]

Process Identity

数据类型：num

未针对客户使用而实施。

（用于接收事件的IPM工艺的识别号。通过参数SetValue，指定选择器。）

SetValue

数据类型：num

信号的期望值（处于当前信号的容许范围内）。如果信号为数字信号，则其必须为整数值。如果信号为数字组信号，则容许值取决于该组中信号的数量。可在SetValue参数中使用的最大值为8388608，且该值为23位数字组信号可拥有的最大值（参见num的范围）。

SetDvalue

数据类型：dnum

信号的期望值（处于当前信号的容许范围内）。如果信号为数字信号，则其必须为整数值。如果信号为数字组信号，则容许值取决于该组中信号的数量。一个数字组信号

下一页继续

可拥有的最大信号位为32。通过dnum变量，可能覆盖值范围0-4294967295，其为32位数字信号可拥有的值范围。

[\DODelay]

Digital Output Delay

数据类型：num

有关数字、组或模拟信号输出信号的时间延迟，以秒计（正值）。

仅用于在机械臂已达到指定位置后，延迟输出信号的设置。如果省略本参数，则将不存在延迟。

延迟不与移动同步。

[\Inhib]

Inhibit

数据类型：bool

用于约束运行时信号设置的永久变量标志的名称。

如果在用于设置信号的位置-时间下，使用该可选参数，且指定标志的实际值为TRUE，则将指定信号（DOp、GOp或AOp）设置为0而非指定值。

[\InhibSetValue]

InhibitSetValue

数据类型：bool, num or dnum

数据类型bool、num或dnum的永久变量的名称，或这三个基础数据类型的别名。

此可选参数只可结合可选参数Inhib一起使用。

如果采用此可选参数，并且在到达设置信号的位置-时间，可选参数Inhib采用的永久变量标记的值为真，那么将读取可选参数InhibSetValue采用的永久变量值，将采用该值来设置DOp、GOp或AOp信号。

如果采用布尔永久变量，那么值“真”将被转换为值“1”，“假”将被转换为值“0”。

[\Mode]

数据类型：triggmode

用于在定义触发器时指定不同的行动模式。

程序执行

当运行指令TriggIO时，将触发条件储存在参数TriggData的指定变量中。

此后，当执行指令TriggL、TriggC或TriggJ之一时，以下方面适用于TriggIO中的定义：

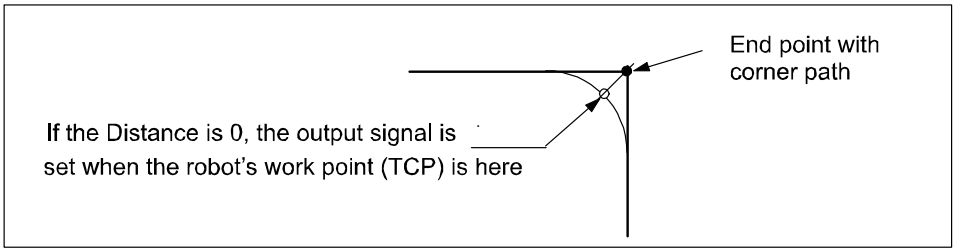
下表描述了参数Distance中指定的距离：

线性移动	直线距离
圆形运动	圆弧长度
非线性移动	沿路径的适当弧长（以获取适当的准确度，距离不得超过弧长的一半）。

下一页继续

1.295 TriggIO - 定义停止点附近的固定位置或时间I/O事件
RobotWare - OS
续前页

本图显示了位于角路径上的固定位置I/O。



xx0500002248

如果距终点（起点）的指定距离不在当前指令（Trigg...）的移动长度内，则当通过起点（终点）时，将产生固定位置I/O。

更多示例

有关于如何使用指令TriggIO的更多例子阐述如下。

例 1

```
VAR triggdata glueflow;

TriggIO glueflow, 1 \Start \AOp:=glue, 5.3;

MoveJ p1, v1000, z50, tool1;
TriggL p2, v500, glueflow, z50, tool1;
```

当工作点（TCP）通过位于起点p1后1 mm处的点时，将模拟信号输出信号glue 设置为值5.3 。

例 2

```
...
TriggL p3, v500, glueflow, z50, tool1;
```

当工作点（TCP）通过位于起点p2后1 mm处的点时，再次将模拟信号输出信号glue设置为值5.3 。

错误处理

如果关于指定模拟信号输出信号AOp的编程SetValue参数超出限制，则将系统变量ERRNO设置为ERR_AO_LIM。可通过错误处理器来处理该错误。

如果关于指定数字组输出信号GOp的编程SetValue或SetDvalue参数超出限制，则将系统变量ERRNO设置为ERR_GO_LIM。可通过错误处理器来处理该错误。

如果信号变量是在RAPID中声明的变量，且其尚未与通过指令AliasIO而在I/O配置中确定的I/O信号相连，随后，将系统变量ERRNO设置为ERR_NO_ALIASIO_DEF。可通过错误处理器来处理该错误。

限制

I/O事件以及距离（无参数\Time）旨在供飞焊点（角路径）使用。使用停止点，距离等于0的I/O事件将延迟触发，直至机械臂已达到精度为+/-24 ms的点。

I/O事件以及时间（含参数\Time）旨在供停止点使用。使用飞焊点的I/O事件以及时间，导致精度低于规定水平。仅可根据移动终点，指定I/O事件以及时间。该时间不能超出机械臂当前的制动时间，其最大约为0.5 s（速度为500 mm/s时，IRB2400的典型

下一页继续

值为150 ms，且IRB6400的典型值为250 ms）。如果指定时间大于当前的制动时间，则直至开始制动（迟于指定值），方才产生本事件。在小而快速的移动期间，可利用当前移动的整个移动时间。

用于设置数字信号输出+/- 5 ms的典型绝对精度值。用于设置数字信号输出+/- 2 ms的典型重复精度值。

语法

```
TriggIO
  [ TriggData ':=' ] < variable (VAR) of triggdata> ','
  [ Distance' :=' ] < expression (IN) of num>
  [ '\' Start ] | [ '\' Time ]
  [ '\' DOp' :=' < variable (VAR) of signaldo> ]
  | [ '\' GOp' :=' < variable (VAR) of signalgo> ]
  | [ '\' AOp' :=' < variable (VAR) of signalao> ]
  | [ '\' ProcID' :=' < expression (IN) of num> ] ','
  [ SetValue' :=' ] < expression (IN) of num>
  | [ SetDvalue' :=' ] < expression (IN) of dnum>
  [ '\' DODelay' :=' < expression (IN) of num> ]
  [ '\' Inhib' :=' < persistent (PERS) of bool> ]
  [ '\' InhibSetValue' :=' < persistent (PERS) of anytype> ]
  [ Mode' :=' ] < expression (IN) of triggmode> ';'
```

相关信息

信息，关于	请参阅
触发的使用	第*785*页的*TriggL* - 关于事件的机械臂线性运动 第*747*页的*TriggC* - 关于事件的机械臂圆周移动 第*778*页的*TriggJ* - 关于事件的轴式机械臂运动
位置-时间I/O事件的定义	第*763*页的*TriggEquip* - 定义路径上的固定位置和时间*I/O*事件
位置相关中断的定义	第*769*页的*TriggInt* - 定义与位置相关的中断
触发数据的储存	第*1510*页的触发数据 - 定位事件，触发 第*1515*页的*triggmode* - 触发行动模式
定义固定位置的I/O检查	第*754*页的*TriggCheckIO* - 定义位于固定位置的*IO*检查
I/O的设置	第*589*页的*SetDO* - 改变数字信号输出信号值 第*591*页的*SetGO* - 改变一组数字信号输出信号的值 第*580*页的*SetAO* - 改变模拟信号输出信号的值

1.296 TriggJ - 关于事件的轴式机械臂运动

手册用法

当无须以直线移动时，在机械臂迅速从一点移动至另一点的同时，TriggJ（*TriggJoint*）用于在大致固定位置设置输出信号和/或运行中断程序。

使用指令TriggIO、TriggEquip、TriggInt、TriggCheckIO、TriggSpeed或TriggRampAO，可定义一个或多个（最多8个）事件，然后，在指令TriggJ中参考此类定义。

本指令仅可用于主任务T_ROB1，或者如果在*MultiMove*系统中，则可用于运动任务中。

基本示例

以下实例介绍了指令TriggJ：

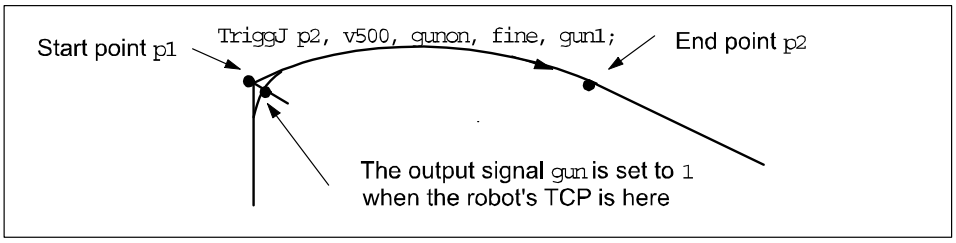
另请参阅第*782*页的更多示例

例 1

```
VAR triggdata gunon;
...
TriggIO gunon, 0 \Start \DOp:=gun, 1;
MoveL p1, v500, z50, gun1;
TriggJ p2, v500, gunon, fine, gun1;
```

当机械臂的TCP通过点p1角路径中点时，设置数字信号输出信号gun。

本图显示了固定位置I/O事件的实例。



xx0500002272

变元

```
TriggJ [\Conc] ToPoint [\ID] Speed [\T] Trigg_1 | TriggArray{*} [
     \T2 ] [ \T3 ] [\T4] [\T5] [\T6] [\T7] [\T8] Zone [\Inpos]
     Tool [\WObj] [\TLoad]
```

[\Conc]

Concurrent

数据类型：switch

当机械臂正在移动时，执行后续指令。当使用飞越点时，本参数可用于避免由过载CPU引起的多余停止。当编程点以高速度靠拢时，该参数有用。

例如，当与外部设备通信，且无需外部设备与机械臂移动之间的同步时，本参数亦有用。其亦可用于调节机械臂路径的执行，以避免警告50024-角路径故障或错误50082-减速限制。

下一页继续

运用参数\Conc，将连续运动指令的数量限制为5。在包含StorePath-RestoPath运动指令以及参数的程序段中，不允许\Conc。

如果省略该参数，则在机械臂已达到指定停止点或指定区域前100 ms之后，执行随后的指令。

不能将该参数用于MultiMove系统中的协调同步移动。

ToPoint

数据类型：robtarget

机器人和外部轴的目标点。定义为已命名的位置或直接存储在指令中（在指令中加 * 标记）。

[\ID]

Synchronization id

数据类型： identno

如果移动是同步或协调同步的，变元 [\ID] 在 MultiMove 系统中是强制的。这个变元在任何其他情况下都不允许使用。指定的 ID 号必须与所有协作程序任务中的 ID 号相同。使用此 ID 号，动作才不会在运行时搞混。

Speed

数据类型： speeddata

适用于运动的速度数据。速度数据规定了关于工具中心点、工具方位调整和外轴的速率。

[\T]

Time

数据类型：num

该参数用于规定机械臂运动的总时间，以秒计。随后，取代相关的速度数据。

Trigg_1

数据类型：triggdata

利用指令TriggIO、TriggEquip、TriggInt、TriggCheckIO、TriggSpeed或TriggRampAO较早期地在程序中定义涉及触发条件和触发活动的变量。

TriggArray

Trigg Data Array Parameter

数据类型：triggdata

指代先前在程序中通过使用指令TriggIO、TriggEquip、TriggInt、TriggSpeed、TriggCheckIO或TriggRampAO而定义的触发条件和触发活动的数组变量。

数组中的限制为25个元素，且必须定义1到25个已确定的触发条件。

当使用TriggArray参数时，不可能同时使用可选参数T2、T3、T4、T5、T6、T7或T8。

[\T2]

触发*2*

数据类型：triggdata

下一页继续

1.296 TriggJ - 关于事件的轴式机械臂运动
RobotWare - OS
续前页

利用指令TriggIO、TriggEquip、TriggInt、TriggCheckIO、TriggSpeed或TriggRampAO较早期地在程序中定义涉及触发条件和触发活动的变量。

[\T3]

Trigg 3

数据类型：triggdata

利用指令TriggIO、TriggEquip、TriggInt、TriggCheckIO、TriggSpeed或TriggRampAO较早期地在程序中定义涉及触发条件和触发活动的变量。

[\T4]

Trigg 4

数据类型：triggdata

利用指令TriggIO、TriggEquip、TriggInt、TriggCheckIO、TriggSpeed或TriggRampAO较早期地在程序中定义涉及触发条件和触发活动的变量。

[\T5]

触发5

数据类型：triggdata

利用指令TriggIO、TriggEquip、TriggInt、TriggCheckIO、TriggSpeed或TriggRampAO较早期地在程序中定义涉及触发条件和触发活动的变量。

[\T6]

Trigg 6

数据类型：triggdata

利用指令TriggIO、TriggEquip、TriggInt、TriggCheckIO、TriggSpeed或TriggRampAO较早期地在程序中定义涉及触发条件和触发活动的变量。

[\T7]

Trigg 7

数据类型：triggdata

利用指令TriggIO、TriggEquip、TriggInt、TriggCheckIO、TriggSpeed或TriggRampAO较早期地在程序中定义涉及触发条件和触发活动的变量。

[\T8]

Trigg 8

数据类型：triggdata

利用指令TriggIO、TriggEquip、TriggInt、TriggCheckIO、TriggSpeed或TriggRampAO较早期地在程序中定义涉及触发条件和触发活动的变量。

Zone

数据类型：zonedata

相关移动的区域数据。区域数据描述了所生成拐角路径的大小。

[\Inpos]

In position

数据类型： stoppointdata

下一页继续

该参数用于规定停止点中机械臂TCP位置的收敛准则。停止点数据取代Zone参数中的指定区域。

Tool

数据类型：tooldata

当机器人移动时工具处于使用状态。工具中心点是移向指定目标点的点。

[\WObj]

Work Object

数据类型：wobjdata

指令中机器人位置关联的工件（坐标系）。

可省略该参数，随后，位置与世界坐标系相关。另一方面，如果使用固定式TCP或协调的外轴，则必须指定该参数，从而执行与工件相关的接头移动。

[\TLoad]

Total load

数据类型：loaddata

\TLoad主动轴描述了移动中使用的总负载。总负载就是相关的工具负载加上该工具正在处理的有效负载。如果使用了\TLoad自变数，那么就不考虑当前tooldata中的loaddata。

如果\TLoad自变数被设置成load0，那么就不考虑\TLoad自变数，而是以当前tooldata中的loaddata作为代替。

想要使用\TLoad自变数，就必需将系统参数ModalPayLoadMode的数值设置成0。如果将ModalPayLoadMode设置成0，那么就再也无法使用指令GripLoad。

可用服务例程“负载标识”（LoadIdentify）来识别总负载。如果系统参数ModalPayLoadMode被设置成0，且系统正在运行该服务例程，那么操作员便可将相关工具的loaddata复制到一个现有的或新的loaddata永久变量中。

如果使用了关联到系统输入项SimMode（仿真模式）上的一个数字输入信号，那么便可在没有任何有效负载的情况下试运行该程序。如果该数字输入信号被设置成1，那么就不考虑可选自变数\TLoad中的loaddata，而是以当前tooldata中的loaddata作为代替。

注意

处理有效负载的默认功能是使用指令GripLoad，因此系统参数ModalPayLoadMode的默认值为1。

程序执行

关于接头移动的信息，请参见指令MoveJ。

因为当机械臂定位越来越接近终点时，触发条件得以满足，因此，实施指定触发活动。在指令终点前的一定距离处、或在指令起点后一定距离处，或在指令终点前的特定时间点（限于短时间），触发条件得以满足。

逐步向前执行期间，实施I/O活动，但是并未运行中断程序。逐步向后执行期间，并未实施任何触发活动。

下一页继续

1.296 TriggJ - 关于事件的轴式机械臂运动
RobotWare - OS
续前页

更多示例

有关于如何使用指令TriggJ的更多例子阐述如下。

例 1

```
VAR intnum intno1;
VAR triggdata trigg1;
...
PROC main()
  CONNECT intno1 WITH trap1;
  TriggInt trigg1, 0.1 \Time, intno1;
  ...
  TriggJ p1, v500, trigg1, fine, gun1;
  TriggJ p2, v500, trigg1, fine, gun1;
  ...
  IDelete intno1;
```

当工作点分别在停止点p1或p2前的位置0.1 s处，运行中断程序trap1。

例 2

```
VAR num Distance:=0;
VAR triggdata trigg_array{25};
VAR signaldo myaliassignaldo;
VAR string signalname;
...
PROC main()
  ...
  FOR i FROM 1 TO 25 DO
    signalname:="do";
    signalname:=signalname+ValToStr(i);
    AliasIO signalname, myaliassignaldo;
    TriggEquip trigg_array{i}, Distance \Start, 0
         \DOp:=myaliassignaldo, SetValue:=1;
    Distance:=Distance+10;
  ENDFOR
  TriggJ p1, v500, trigg_array, z30, tool2;
  MoveJ p2, v500, z30, tool2;
  ...
```

在移动至p1期间，设置数字信号输出信号do1到do25。信号设置之间的距离为10 mm。

错误处理

如果在某些相关TriggSpeed指令中的指定模拟信号输出信号AOp的编程ScaleValue参数连同该指令中的编程Speed，导致模拟信号超出限制，则将系统变量ERRNO设置为ERR_AO_LIM。

如果与系统参数中使用的事件预设时间相比，某些相关TriggSpeed指令中的编程DipLag参数过大，则将系统变量ERRNO设置为ERR_DIPLAG_LIM。

如果在输入指令时，没有与I/O单元接触，则可将系统变量ERRNO设置为ERR_NORUNUNIT，使用的触发数据取决于运行中的I/O单元，即在触发数据中使用的信号。

下一页继续

如果已经超出了使用参数\Conc的连续运动指令数，则将系统变量ERRNO设置为ERR_CONC_MAX。

可用错误处理器来处理此类错误。

限制

如果当前起动点偏离正常点，以致指令TriggJ的总定位长度短于正常水平（例如，在机械臂位于终点时TriggJ开始的时候），则若干或全部触发条件可能立即得以满足，并位于相同位置。在此类情况下，实施触发活动的顺序将变得不明确。用户程序中的程序逻辑可能不基于有关“不完整移动”触发活动的正常顺序。

语法

```
TriggJ
  [ '\' Conc ',']
  [ ToPoint' :=' ] < expression (IN) of robtarget >
  [ '\' ID ':=' < expression (IN) of identno >]','
  [ Speed ':=' ] < expression (IN) of speeddata >
  [ '\' T ':=' < expression (IN) of num > ]','
  [Trigg_1 ':=' ] < variable (VAR) of triggdata > |
  [TriggArray ':=' ] < array variable {*} (VAR) of triggdata >
  [ '\' T2 ':=' < variable (VAR) of triggdata > ]
  [ '\' T3 ':=' < variable (VAR) of triggdata > ]
  [ '\' T4 ':=' < variable (VAR) of triggdata > ]
  [ '\' T5 ':=' < variable (VAR) of triggdata > ]
  [ '\' T6 ':=' < variable (VAR) of triggdata > ]
  [ '\' T7 ':=' < variable (VAR) of triggdata > ]
  [ '\' T8 ':=' < variable (VAR) of triggdata > ] ´,'
  [Zone ':=' ] < expression (IN) of zonedata >
  [ '\' Inpos ':=' < expression (IN) of stoppointdata > ]´,'
  [ Tool ':=' ] < persistent (PERS) of tooldata >
  [ '\' WObj' :='< persistent (PERS) of wobjdata > ]
  [ '\' TLoad' :=' < persistent (PERS) of loaddata > ] ';'
```

相关信息

信息，关于	请参阅
触发时的线性移动	第*785*页的*TriggL - 关于事件的机械臂线性运动*
触发时的圆周移动	第*747*页的*TriggC - 关于事件的机械臂圆周移动*
触发的定义	第*773*页的*TriggIO - 定义停止点附近的固定位置或时间I/O事件* 第*763*页的*TriggEquip - 定义路径上的固定位置和时间I/O事件* 第*804*页的*TriggRampAO - 定义路径上的固定位置斜坡AO事件* 第*769*页的*TriggInt - 定义与位置相关的中断* 第*754*页的*TriggCheckIO - 定义位于固定位置的IO检查*
处理triggdata	第*1510*页的触发数据 - 定位事件，触发 第*761*页的*TriggDataReset - 重置触发数据变量中的内容* 第*759*页的*TriggDataCopy - 复制触发数据变量中的内容* 第*1284*页的*TriggDataValid - 检查触发数据变量中的内容是否有效*

下一页继续

1.296 TriggJ - 关于事件的轴式机械臂运动

RobotWare - OS

续前页

信息，关于	请参阅
通过接头移动，移动机械臂	第*367*页的*MoveJ - 通过接头移动，移动机械臂*
接头移动	技术参考手册 - *RAPID*语言概览
载荷的定义	第*1421*页的*loaddata - 加载数据*
速度的定义	第*1480*页的*speeddata - 速度数据*
停止点数据的定义	第*1483*页的*stoppointdata - 停止点数据*
工具的定义	第*1502*页的*tooldata - 工具数据*
工件的定义	第*1523*页的*wobjdata - 工件数据*
区域数据的定义	第*1531*页的*zonedata - 区域数据*
一般动作	技术参考手册 - *RAPID*语言概览
关于如何使用TLoad总负载的例子。	第*388*页的*MoveL - 使机械臂沿直线移动*
定义机器人的载荷	第*223*页的*GripLoad - 定义机械臂的有效负载*
LoadIdentify，载荷识别服务例行程序	操作员手册 - 带 *FlexPendant* 的 *IRC5*
用于在仿真模式下无载荷运行机器人的系统输入信号 *SimMode*。 （主题 I/O，类型 System Input，行动值，*SimMode*）	技术参考手册 - 系统参数
激活和停用载荷的系统参数 *ModalPayLoadMode*。 （主题 Controller，类型 System Misc，行动值，*ModalPayLoadMode*）	技术参考手册 - 系统参数

1.297 TriggL - 关于事件的机械臂线性运动

手册用法

当机械臂正在进行线性移动时，TriggL（*Trigg Linear*）用于设置输出信号和/或在固定位置运行中断程序。

使用指令TriggIO、TriggEquip、TriggInt、TriggSpeed、TriggCheckIO或TriggRampAO，可定义一个或多个（最多8个）事件。此后，指令TriggL参考此类定义。

本指令仅可用于主任务T_ROB1，或者如果在*MultiMove*系统中，则可用于运动任务中。

基本示例

以下实例介绍了指令TriggL：

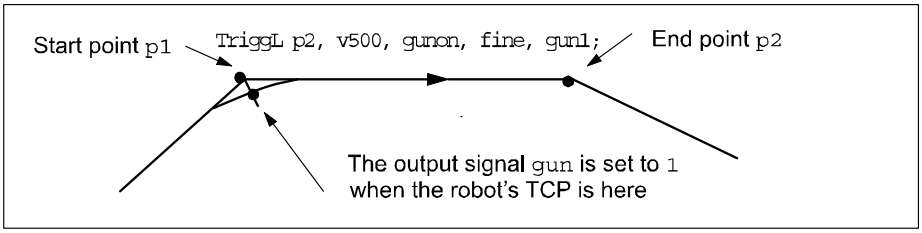
另请参阅第*789*页的更多示例

例 1

```
VAR triggdata gunon;

TriggIO gunon, 0 \Start \DOp:=gun, 1;
MoveJ p1, v500, z50, gun1;
TriggL p2, v500, gunon, fine, gun1;
```

当机械臂的TCP通过点p1角路径中点时，设置数字信号输出信号gun。

本图显示了固定位置I/O事件的实例。



xx0500002291

变元

```
TriggL [\Conc] ToPoint [\ID] Speed [\T] Trigg_1 | TriggArray{*}
     [\T2] [\T3] [\T4] [\T5] [\T6] [\T7] [\T8] Zone [\Inpos] Tool
     [\WObj] [\Corr] [\TLoad]
```

[\Conc]

Concurrent

数据类型：switch

当机械臂正在移动时，执行后续指令。当使用飞越点时，本参数可用于避免由过载CPU引起的多余停止。当编程点以高速度靠拢时，该参数有用。

例如，当与外部设备通信，且无需外部设备与机械臂移动之间的同步时，本参数亦有用。其亦可用于调节机械臂路径的执行，以避免警告50024-角路径故障或错误50082-减速限制。

下一页继续

1.297 TriggL - 关于事件的机械臂线性运动
RobotWare - OS
续前页

运用参数\Conc，将连续运动指令的数量限制为5。在包含StorePath-RestoPath运动指令以及参数的程序段中，不允许\Conc。

如果省略该参数，且ToPoint并非停止点，则在机械臂达到编程区之前，执行后续指令一段时间。

不能将该参数用于MultiMove系统中的协调同步移动。

ToPoint

数据类型：robtarget

机器人和外部轴的目标点。定义为已命名的位置或直接存储在指令中（在指令中加 * 标记）。

[\ID]

Synchronization id

数据类型： identno

如果移动是同步或协调同步的，变元 [\ID] 在 MultiMove 系统中是强制的。这个变元在任何其他情况下都不允许使用。指定的 ID 号必须与所有协作程序任务中的 ID 号相同。使用此 ID 号，动作才不会在运行时搞混。

Speed

数据类型： speeddata

适用于运动的速度数据。速度数据规定了工具中心点、外轴和工具方位调整的速率。

[\T]

Time

数据类型：num

该参数用于规定机械臂运动的总时间，以秒计。随后，取代相关的速度数据。

Trigg_1

数据类型：triggdata

指代先前在程序中通过使用指令TriggIO、TriggEquip、TriggInt、TriggSpeed、TriggCheckIO或TriggRampAO而定义的触发条件和触发活动的变量。

TriggArray

Trigg Data Array Parameter

数据类型：triggdata

指代先前在程序中通过使用指令TriggIO、TriggEquip、TriggInt、TriggSpeed、TriggCheckIO或TriggRampAO而定义的触发条件和触发活动的数组变量。

数组中的限制为25个元素，且必须定义1到25个已确定的触发条件。

当使用TriggArray参数时，不可能同时使用可选参数T2、T3、T4、T5、T6、T7或T8。

[\T2]

Trigg 2

数据类型：triggdata

下一页继续

指代先前在程序中通过使用指令TriggIO、TriggEquip、TriggInt、TriggSpeed、TriggCheckIO或TriggRampAO而定义的触发条件和触发活动的变量。

[\T3]

Trigg 3

数据类型：triggdata

指代先前在程序中通过使用指令TriggIO、TriggEquip、TriggInt、TriggSpeed、TriggCheckIO或TriggRampAO而定义的触发条件和触发活动的变量。

[\T4]

Trigg 4

数据类型：triggdata

指代先前在程序中通过使用指令TriggIO、TriggEquip、TriggInt、TriggSpeed、TriggCheckIO或TriggRampAO而定义的触发条件和触发活动的变量。

[\T5]

Trigg 5

数据类型：triggdata

指代先前在程序中通过使用指令TriggIO、TriggEquip、TriggInt、TriggSpeed、TriggCheckIO或TriggRampAO而定义的触发条件和触发活动的变量。

[\T6]

Trigg 6

数据类型：triggdata

指代先前在程序中通过使用指令TriggIO、TriggEquip、TriggInt、TriggSpeed、TriggCheckIO或TriggRampAO而定义的触发条件和触发活动的变量。

[\T7]

Trigg 7

数据类型：triggdata

指代先前在程序中通过使用指令TriggIO、TriggEquip、TriggInt、TriggSpeed、TriggCheckIO或TriggRampAO而定义的触发条件和触发活动的变量。

[\T8]

Trigg 8

数据类型：triggdata

指代先前在程序中通过使用指令TriggIO、TriggEquip、TriggInt、TriggSpeed、TriggCheckIO或TriggRampAO而定义的触发条件和触发活动的变量。

下一页继续

1.297 TriggL - 关于事件的机械臂线性运动
RobotWare - OS
续前页

Zone

数据类型：zonedata

相关移动的区域数据。区域数据描述了所生成拐角路径的大小。

[\Inpos]

In position

数据类型： stoppointdata

该参数用于规定停止点中机械臂TCP位置的收敛准则。停止点数据取代Zone参数中的指定区域。

Tool

数据类型： tooldata

当机器人移动时工具处于使用状态。工具中心点是移向指定目标点的点。

[\WObj]

Work Object

数据类型： wobjdata

指令中机器人位置关联的工件（坐标系）。

此边缘可以缺失，此时位置与世界坐标系关联。另一方面，如果使用了静态 TCP 或协调外部轴，则必须指定此变元才能执行相对此工件的线性移动。

[\Corr]

Correction

数据类型：switch

如果存在该参数，则将通过指令CorrWrite而写入修正条目的修正数据添加到路径和目的位置。

[\TLoad]

Total load

数据类型：loaddata

\TLoad主动轴描述了移动中使用的总负载。总负载就是相关的工具负载加上该工具正在处理的有效负载。如果使用了\TLoad自变数，那么就不考虑当前tooldata中的loaddata。

如果\TLoad自变数被设置成load0，那么就不考虑\TLoad自变数，而是以当前tooldata中的loaddata作为代替。

想要使用\TLoad自变数，就必需将系统参数ModalPayLoadMode的数值设置成0。如果将ModalPayLoadMode设置成0，那么就再也无法使用指令GripLoad。

可用服务例程“负载标识”（LoadIdentify）来识别总负载。如果系统参数ModalPayLoadMode被设置成0，且系统正在运行该服务例程，那么操作员便可将相关工具的loaddata复制到一个现有的或新的loaddata永久变量中。

如果使用了关联到系统输入项SimMode（仿真模式）上的一个数字输入信号，那么便可在没有任何有效负载的情况下试运行该程序。如果该数字输入信号被设置成1，那么

下一页继续

就不考虑可选自变数\TLoad中的loaddata，而是以当前tooldata中的loaddata作为代替。

> **注意**
>
> 处理有效负载的默认功能是使用指令GripLoad，因此系统参数ModalPayLoadMode的默认值为1。

程序执行

有关线性移动的信息请参阅 MoveL 指令。

因为当机械臂定位越来越接近终点时，触发条件得以满足，因此，实施指定触发活动。在指令终点前的一定距离处、或在指令起点后一定距离处，或在指令终点前的特定时间点（限于短时间），触发条件得以满足。

逐步向前执行期间，实施I/O活动，但是并未运行中断程序。逐步向后执行期间，并未实施任何触发活动。

更多示例

有关于如何使用指令TriggL的更多例子阐述如下。

例 1

```
VAR intnum intno1;
VAR triggdata trigg1;
...
PROC main()
  CONNECT intno1 WITH trap1;
  TriggInt trigg1, 0.1 \Time, intno1;
  ...
  TriggL p1, v500, trigg1, fine, gun1;
  TriggL p2, v500, trigg1, fine, gun1;
  ...
  IDelete intno1;
```

当工作点分别在点p1或p2前的位置0.1 s处，运行中断程序trap1。

例 2

```
VAR num Distance:=0;
VAR triggdata trigg_array{25};
VAR signaldo myaliassignaldo;
VAR string signalname;
...
PROC main()
  ...
  FOR i FROM 1 TO 25 DO
    signalname:="do";
    signalname:=signalname+ValToStr(i);
    AliasIO signalname, myaliassignaldo;
    TriggEquip trigg_array{i}, Distance \Start, 0
         \DOp:=myaliassignaldo, SetValue:=1;
    Distance:=Distance+10;
  ENDFOR
  TriggL p1, v500, trigg_array, z30, tool2;
```

下一页继续

1.297 TriggL - 关于事件的机械臂线性运动
RobotWare - OS
续前页

```
MoveL p2, v500, z30, tool2;
...
```

在移动至p1期间，设置数字信号输出信号do1到do25。信号设置之间的距离为10 mm。

错误处理

如果在某些相关TriggSpeed指令中的指定模拟信号输出信号AOp的编程ScaleValue参数连同该指令中的编程Speed，导致模拟信号超出限制，则将系统变量ERRNO设置为ERR_AO_LIM。

如果与系统参数中使用的事件预设时间相比，某些相关TriggSpeed指令中的编程DipLag参数过大，则将系统变量ERRNO设置为ERR_DIPLAG_LIM。

如果在输入指令时，没有与I/O单元接触，则可将系统变量ERRNO设置为ERR_NORUNUNIT，使用的触发数据取决于运行中的I/O单元，即在触发数据中使用的信号。

如果已经超出了使用参数\Conc的连续运动指令数，则将系统变量ERRNO设置为ERR_CONC_MAX。

可用错误处理器来处理此类错误。

限制

如果当前起动点偏离正常点，以致指令TriggL的总定位长度短于正常水平（例如，在机械臂位于终点时TriggL开始的时候），则若干或全部触发条件可能立即得以满足，并位于相同位置。在此类情况下，实施触发活动的顺序将变得不明确。用户程序中的程序逻辑可能不基于有关“不完整移动”触发活动的正常顺序。

语法

```
TriggL
  ['\' Conc ',']
  [ ToPoint' :=' ] < expression (IN) of robtarget >
  [ '\' ID ':=' < expression (IN) of identno >] ','
  [ Speed ':=' ] < expression (IN) of speeddata >
  [ '\' T ':=' < expression (IN) of num > ]',' 
  [Trigg_1 ':=' ] < variable (VAR) of triggdata > |
  [TriggArray ':=' ] < array variable {*} (VAR) of triggdata >
  [ '\' T2 ':=' < variable (VAR) of triggdata > ]
  [ '\' T3 ':=' < variable (VAR) of triggdata > ]
  [ '\' T4 ':=' < variable (VAR) of triggdata > ]
  [ '\' T5 ':=' < variable (VAR) of triggdata > ]
  [ '\' T6 ':=' < variable (VAR) of triggdata > ]
  [ '\' T7 ':=' < variable (VAR) of triggdata > ]
  [ '\' T8 ':=' < variable (VAR) of triggdata > ] ´,'
  [Zone ':=' ] < expression (IN) of zonedata >
  [ '\' Inpos' :=' < expression (IN) of stoppointdata > ] ´,'
  [ Tool ':=' ] < persistent (PERS) of tooldata >
  [ '\' WObj' :=' < persistent (PERS) of wobjdata > ]
  [ '\' Corr ]
  [ '\' TLoad' :=' < persistent (PERS) of loaddata > ] ';'
```

下一页继续

相关信息

信息，关于	请参阅
触发时的圆周移动	第*747*页的*TriggC - 关于事件的机械臂圆周移动*
触发时的接头移动	第*778*页的*TriggJ - 关于事件的轴式机械臂运动*
触发的定义	第*773*页的*TriggIO - 定义停止点附近的固定位置或时间I/O事件* 第*763*页的*TriggEquip - 定义路径上的固定位置和时间I/O事件* 第*769*页的*TriggInt - 定义与位置相关的中断* 第*754*页的*TriggCheckIO - 定义位于固定位置的IO检查* 第*804*页的*TriggRampAO - 定义路径上的固定位置斜坡AO事件* 第*810*页的*TriggSpeed - 定义与固定位置-时间尺度事件成比例的TCP速度模拟信号输出*
处理`triggdata`	第*1510*页的触发数据 - 定位事件，触发 第*761*页的*TriggDataReset - 重置触发数据变量中的内容* 第*759*页的*TriggDataCopy - 复制触发数据变量中的内容* 第*1284*页的*TriggDataValid - 检查触发数据变量中的内容是否有效*
写入纠正条目	第*137*页的*CorrWrite - 写入修正发电机*
线性移动	技术参考手册 - *RAPID*语言概览
载荷的定义	第*1421*页的*loaddata - 加载数据*
速度的定义	第*1480*页的*speeddata - 速度数据*
停止点数据的定义	第*1483*页的*stoppointdata - 停止点数据*
工具的定义	第*1502*页的*tooldata - 工具数据*
工件的定义	第*1523*页的*wobjdata - 工件数据*
区域数据的定义	第*1531*页的*zonedata - 区域数据*
一般动作	技术参考手册 - *RAPID*语言概览
关于如何使用`TLoad`总负载的例子。	第*388*页的*MoveL - 使机械臂沿直线移动*
定义机器人的载荷	第*223*页的*GripLoad - 定义机械臂的有效负载*
LoadIdentify，载荷识别服务例行程序	操作员手册 - 带 *FlexPendant* 的 *IRC5*
用于在仿真模式下无载荷运行机器人的系统输入信号 *SimMode*。 （主题 I/O，类型 System Input，行动值，*SimMode*）	技术参考手册 - 系统参数
激活和停用载荷的系统参数 *ModalPayLoadMode*。 （主题 Controller，类型 System Misc，行动值，*ModalPayLoadMode*）	技术参考手册 - 系统参数

1.298 TriggJIOs - 接头机械臂移动以及I/O事件

手册用法

当机械臂正在进行接头移动时，TriggJIOs（*Trigg Joint I/O*）用于设置固定位置处的输出信号。

当使用移动以及区域时，TriggJIOs指令得以优化，以获得较好的精度（同TriggEquip/TriggL相比）。

本指令仅可用于主任务T_ROB1，或者如果在*MultiMove*系统中，则可用于运动任务中。

基本示例

以下实例介绍了指令TriggJIOs：

另请参阅第*801*页的更多示例

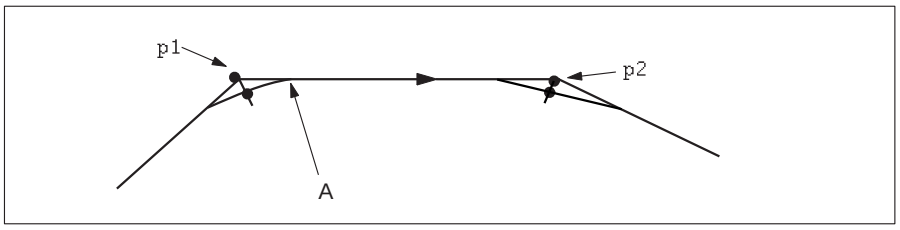
例 1

```
VAR triggios gunon{1};

gunon{1}.used:=TRUE;
gunon{1}.distance:=3;
gunon{1}.start:=TRUE;
gunon{1}.signalname:="gun";
gunon{1}.equiplag:=0;
gunon{1}.setvalue:=1;

MoveJ p1, v500, z50, gun1;
TriggJIOs p2, v500, \TriggData1:=gunon, z50, gun1;
MoveL p3, v500, z50, gun1;
```

当TCP位于点p1后3 mm时，设置信号gun。

RAPID代码和本图显示了固定位置I/O事件的实例。

```
TriggJIOs p2, v500, \TriggData1:=gunon, z50, gun1;
```



xx1500000304

A 当机械臂的TCP位于此处时，将输出信号gun设置为1。

变元

```
TriggJIOs ToPoint [\ID] Speed [\T] [\TriggData1] [\TriggData2]
     [\TriggData3] Zone [\Inpos] Tool [\WObj] [\Corr] [\TLoad]
```

ToPoint

数据类型：robtarget

机器人和外部轴的目标点。定义为已命名的位置或直接存储在指令中（在指令中加 * 标记）。

下一页继续

`[ \ID ]`

Synchronization id

数据类型： `identno`

如果移动是同步或协调同步的，变元 `[ \ID ]` 在 MultiMove 系统中是强制的。这个变元在任何其他情况下都不允许使用。指定的 ID 号必须与所有协作程序任务中的 ID 号相同。使用此 ID 号，动作才不会在运行时搞混。

`Speed`

数据类型： `speeddata`

适用于运动的速度数据。速度数据规定了工具中心点、外轴和工具方位调整的速率。

`[ \T ]`

Time

数据类型：`num`

该参数用于规定机械臂运动的总时间，以秒计。随后，取代相关的速度数据。

`[\TriggData1]`

数据类型：`array of triggios`

指代触发条件和触发活动的变量（数组）。当使用此参数时，设置模拟信号输出信号、数字信号输出信号和数字组输出信号是可能的。如果使用数字组输出信号，则存在有关该组中23个信号的限制。

`[\TriggData2]`

数据类型：`array of triggstrgo`

指代触发条件和触发活动的变量（数组）。当使用该参数时，设置由该组中32个信号组成且最大设置值为4294967295的数字组输出信号是可能的。仅可使用数字组输出信号。

`[\TriggData3]`

数据类型：`array of triggiosdnum`

指代触发条件和触发活动的变量（数组）。当使用此参数时，设置由该组中32个信号组成且最大设置值为4294967295的模拟信号输出信号、数字信号输出信号和数字组输出信号是可能的。

`Zone`

数据类型：`zonedata`

相关移动的区域数据。区域数据描述了所生成拐角路径的大小。

`[ \Inpos ]`

In position

数据类型： `stoppointdata`

该参数用于规定停止点中机械臂TCP位置的收敛准则。停止点数据取代Zone参数中的指定区域。

`Tool`

数据类型： `tooldata`

当机器人移动时工具处于使用状态。工具中心点是移向指定目标点的点。

下一页继续

1.298 TriggJIOs - 接头机械臂移动以及I/O事件
RobotWare - OS
续前页

[\WObj]

Work Object

数据类型： wobjdata

指令中机器人位置关联的工件（坐标系）。

此边缘可以缺失，此时位置与世界坐标系关联。另一方面，如果使用了静态TCP或协调外部轴，则必须指定此变元才能执行相对此工件的线性移动。

[\Corr]

Correction

数据类型：switch

如果存在该参数，则将通过指令CorrWrite而写入修正条目的修正数据添加到路径和目的位置。

[\TLoad]

Total load

数据类型：loaddata

\TLoad主动轴描述了移动中使用的总负载。总负载就是相关的工具负载加上该工具正在处理的有效负载。如果使用了\TLoad自变数，那么就不考虑当前tooldata中的loaddata。

如果\TLoad自变数被设置成load0，那么就不考虑\TLoad自变数，而是以当前tooldata中的loaddata作为代替。

想要使用\TLoad自变数，就必需将系统参数ModalPayLoadMode的数值设置成0。如果将ModalPayLoadMode设置成0，那么就再也无法使用指令GripLoad。

可用服务例程“负载标识”（LoadIdentify）来识别总负载。如果系统参数ModalPayLoadMode被设置成0，且系统正在运行该服务例程，那么操作员便可将相关工具的loaddata复制到一个现有的或新的loaddata永久变量中。

如果使用了关联到系统输入项SimMode（仿真模式）上的一个数字输入信号，那么便可在没有任何有效负载的情况下试运行该程序。如果该数字输入信号被设置成1，那么就不考虑可选自变数\TLoad中的loaddata，而是以当前tooldata中的loaddata作为代替。

注意

处理有效负载的默认功能是使用指令GripLoad，因此系统参数ModalPayLoadMode的默认值为1。

程序执行

关于接头移动的信息，请参见指令MoveJ，*第367页的MoveJ - 通过接头移动，移动机械臂*。

通过指令TriggJIOs，设置关于沿A到B路径的I/O信号的1-50个不同的触发活动是可能的。可使用的信号包括数字信号输出信号、数字组输出信号和模拟信号输出信号。在本指令终点前一特定距离处，或在本指令起点后一特定距离处，触发条件均得以满足。

本指令需要使用TriggData1、TriggData2或TriggData3参数之一或全部。但是，任意触发的使用是可选择的。为抑制触发的使用，可通过数据类型

下一页继续

triggios/triggstrgo/triggiosdnum的数组元素，将分量used设置为FALSE。如果未使用数组元素，则TriggJIOs指令将起MoveJ的作用，且将不会实施任何I/O活动。

如果本程序逐步向前，则实施I/O活动。逐步向后执行期间，将不会实施任何I/O活动。

如果将TriggData1、TriggData2或TriggData3参数中的分量EquipLag设置为负时间（延迟），则可将I/O信号设置在目的点后（ToPoint）。

如果使用参数TriggData2或TriggData3，则使用高达4294967295的值是可能的，该值是一组数字信号所能拥有的最大值（对于本系统而言，一组信号最多包含32个信号）。

更多示例

有关于如何使用指令TriggJIOs的更多例子阐述如下。

例 1

```
VAR triggios mytriggios{3}:= [[TRUE, 3, TRUE, 0, "go1", 55, 0],
     [TRUE, 15, TRUE, 0, "ao1", 10, 0], [TRUE, 3, FALSE, 0, "do1",
     1, 0]];
...
MoveL p1, v500, z50, gun1;
TriggJIOs p2, v500, \TriggData1:=mytriggios, z50, gun1;
MoveL p3, v500, z50, gun1;
```

将数字组输出信号go1设置为距p155 3 mm。将模拟信号输出信号设置为距p110 15 mm。将数字信号输出信号do1设置为距ToPointp23 mm。

例 2

```
VAR triggios mytriggios{3}:= [[TRUE, 3, TRUE, 0, "go1", 55, 0],
     [TRUE, 15, TRUE, 0, "ao1", 10, 0], [TRUE, 3, FALSE, 0, "do1",
     1, 0]];
VAR triggstrgo mytriggstrgo{3}:= [[TRUE, 3, TRUE, 0, "go2", "1",
     0], [TRUE, 15, TRUE, 0, "go2", "800000", 0], [TRUE, 4, FALSE,
     0, "go2", "4294967295", 0]];
VAR triggiosdnum mytriggiosdnum{3}:= [[TRUE, 10, TRUE, 0, "go3",
     4294967295, 0], [TRUE, 10, TRUE, 0, "ao2", 5, 0], [TRUE, 10,
     TRUE, 0, "do2", 1, 0]];
...
MoveL p1, v500, z50, gun1;
TriggJIOs p2, v500, \TriggData1:=mytriggios \TriggData2:=
     mytriggstrgo \TriggData3:=mytriggiosdnum, z50, gun1;
MoveL p3, v500, z50, gun1;
```

将数字组输出信号go1设置为距p155 3 mm。将模拟信号输出信号ao1设置为距p110 15 mm。将数字信号输出信号do1设置为距ToPointp23 mm。通过变量mytriggios，设置此类位置事件。变量mytriggstrgo进行设置，以使位置事件距离p13和15 mm。首先，将信号go2设置为1，随后，将其设置为800000。将本信号设置为距ToPoint p24294967295 4 mm。此为32位数字信号输出信号的最大值。变量mytriggiosdnum设置三个位置，甚至距p110 mm。首先，将信号go3设置为4294967295，随后，将ao2设置为5，最后，将do2设置为1。

下一页继续

1.298 TriggJIOs - 接头机械臂移动以及I/O事件

RobotWare - OS

续前页

错误处理

系统会生成下列可恢复错误，并在错误处理器中处理这些错误。系统变量ERRNO将被设置成：

名称	错误原因
ERR_NORUNUNIT	如果与I/O单元无接触。
ERR_GO_LIM	如果关于指定数字组输出信号signalname的编程setvalue参数超出限制。（用TriggData1、TriggData2或TriggData3进行声明）
ERR_AO_LIM	如果关于指定模拟信号输出信号signalname的编程setvalue参数超出限制。（用TriggData1或TriggData3进行声明）

限制

如果当前起动点偏离正常点，以致指令TriggJIOs的总定位长度短于正常水平（例如，在机械臂位于终点时TriggJIOs开始的时候），则若干或全部触发条件可能立即得以满足，并位于相同位置。在此类情况下，实施触发活动的顺序将变得不明确。用户程序中的程序逻辑可能不基于有关“不完整移动”触发活动的正常顺序。

针对各编程指令，指令TriggJIOs中触发的数量限制为50。如果在更近的距离处出现触发，则本系统可能无法对其进行处理。其取决于移动方式、所用TCP速度以及触发编程的紧密程度。存在此类限制，但是当出现此类问题时，难以对其进行预测。

语法

```
TriggJIOs
  [ ToPoint ':=' ] < expression (IN) of robtarget >
  [ '\' ID ':=' < expression (IN) of identno >] ','
  [ Speed ':=' ] < expression (IN) of speeddata >
  [ '\' T ':=' < expression (IN) of num > ] ','
  [ '\' TriggData1 ':=' ] < array {*} (VAR) of triggios >
  [ '\' TriggData2 ':=' ] < array {*} (VAR) of triggstrgo >
  [ '\' TriggData3 ':=' ] < array {*} (VAR) of triggiosdnum >
  [ Zone ':=' ] < expression (IN) of zonedata >
  [ '\' Inpos ':=' < expression (IN) of stoppointdata > ] ','
  [ Tool ':=' ] < persistent (PERS) of tooldata >
  [ '\' WObj ':=' < persistent (PERS) of wobjdata > ]
  [ '\' Corr ]
  [ '\' TLoad ':=' < persistent (PERS) of loaddata > ] ';'
```

相关信息

信息，关于	请参阅
机械臂线性移动以及I/O事件	第*798*页的*TriggLIOs - 机械臂线性移动以及I/O事件*
触发条件和触发活动的储存	第*1511*页的*triggios - Positioning events, trigg*
由32个信号组成的数字信号组的触发条件和触发活动的储存	第*1517*页的*triggstrgo - Positioning events, trigg*
触发条件和触发活动的储存	第*1513*页的*triggiosdnum - Positioning events, trigg*
事件对象的分配	技术参考手册 - 系统参数
线性移动	技术参考手册 - *RAPID*语言概览

下一页继续

续前页

信息，关于	请参阅
一般动作	技术参考手册 - *RAPID*语言概览
载荷的定义	第*1421*页的*loaddata - 加载数据*
关于如何使用TLoad总负载的例子。	第*388*页的*MoveL - 使机械臂沿直线移动*
定义机器人的载荷	第*223*页的*GripLoad - 定义机械臂的有效负载*
LoadIdentify，载荷识别服务例行程序	操作员手册 - 带 *FlexPendant* 的 *IRC5*
用于在仿真模式下无载荷运行机器人的系统输入信号 *SimMode*。 （主题 I/O，类型 System Input，行动值，*SimMode*）	技术参考手册 - 系统参数
激活和停用载荷的系统参数 *ModalPayLoadMode*。 （主题 Controller，类型 System Misc，行动值，*ModalPayLoadMode*）	技术参考手册 - 系统参数

1.299 TriggLIOs - 机械臂线性移动以及I/O事件

手册用法

当机械臂正在进行线性移动时，TriggLIOs （触发线性I/O）用于设置固定位置处的输出信号。

当使用移动以及区域时，TriggLIOs指令得以优化，以获得较好的精度（同TriggEquip/TriggL相比）。

本指令仅可用于主任务T_ROB1，或者如果在*MultiMove*系统中，则可用于运动任务中。

基本示例

以下实例介绍了指令TriggLIOs：

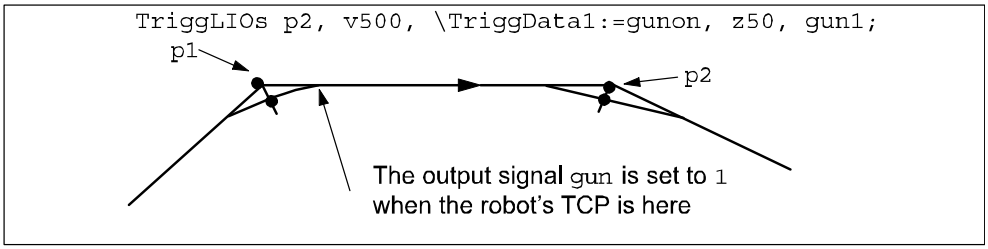
另请参阅第*801*页的更多示例

例 1

```
VAR triggios gunon{1};

gunon{1}.used:=TRUE;
gunon{1}.distance:=3;
gunon{1}.start:=TRUE;
gunon{1}.signalname:="gun";
gunon{1}.equiplag:=0;
gunon{1}.setvalue:=1;

MoveJ p1, v500, z50, gun1;
TriggLIOs p2, v500, \TriggData1:=gunon, z50, gun1;
MoveL p3, v500, z50, gun1;
```

当TCP位于点p1后3 mm时，设置信号gun。

本图显示了固定位置I/O事件的实例。



en0800000157

变元

```
TriggLIOs [\Conc] ToPoint [\ID] Speed [\T] [\TriggData1]
     [\TriggData2] [\TriggData3] Zone [\Inpos] Tool [\WObj] [\Corr]
     [\TLoad]
```

[\Conc]

Concurrent

数据类型：switch

下一页继续

当机械臂正在移动时，执行后续指令。当使用飞越点时，本参数可用于避免由过载CPU引起的多余停止。当编程点以高速度靠拢时，该参数有用。

例如，当与外部设备通信，且无需外部设备与机械臂移动之间的同步时，本参数亦有用。其亦可用于调节机械臂路径的执行，以避免警告50024-角路径故障或错误50082-减速限制。

运用参数\Conc，将连续运动指令的数量限制为5。在包含StorePath-RestoPath运动指令以及参数的程序段中，不允许\Conc。

如果省略该参数，且ToPoint并非停止点，则在机械臂达到编程区之前，执行后续指令一段时间。

不能将该参数用于MultiMove系统中的协调同步移动。

ToPoint

数据类型：robtarget

机器人和外部轴的目标点。定义为已命名的位置或直接存储在指令中（在指令中加 * 标记）。

[\ID]

Synchronization id

数据类型： identno

如果移动是同步或协调同步的，变元 [\ID] 在 MultiMove 系统中是强制的。这个变元在任何其他情况下都不允许使用。指定的 ID 号必须与所有协作程序任务中的 ID 号相同。使用此 ID 号，动作才不会在运行时搞混。

Speed

数据类型： speeddata

适用于运动的速度数据。速度数据规定了工具中心点、外轴和工具方位调整的速率。

[\T]

Time

数据类型：num

该参数用于规定机械臂运动的总时间，以秒计。随后，取代相关的速度数据。

[\TriggData1]

数据类型：array of triggios

指代触发条件和触发活动的变量（数组）。当使用此参数时，设置模拟信号输出信号、数字信号输出信号和数字组输出信号是可能的。如果使用数字组输出信号，则存在有关该组中23个信号的限制。

[\TriggData2]

数据类型：array of triggstrgo

指代触发条件和触发活动的变量（数组）。当使用该参数时，设置由该组中32个信号组成且最大设置值为4294967295的数字组输出信号是可能的。仅可使用数字组输出信号。

[\TriggData3]

数据类型：array of triggiosdnum

下一页继续

1.299 TriggLIOs - 机械臂线性移动以及I/O事件
RobotWare - OS
续前页

指代触发条件和触发活动的变量（数组）。当使用此参数时，设置由该组中32个信号组成且最大设置值为4294967295的模拟信号输出信号、数字信号输出信号和数字组输出信号是可能的。

Zone

数据类型：zonedata

相关移动的区域数据。区域数据描述了所生成拐角路径的大小。

[\Inpos]

In position

数据类型： stoppointdata

该参数用于规定停止点中机械臂TCP位置的收敛准则。停止点数据取代Zone参数中的指定区域。

Tool

数据类型： tooldata

当机器人移动时工具处于使用状态。工具中心点是移向指定目标点的点。

[\WObj]

Work Object

数据类型： wobjdata

指令中机器人位置关联的工件（坐标系）。

此边缘可以缺失，此时位置与世界坐标系关联。另一方面，如果使用了静态TCP或协调外部轴，则必须指定此变元才能执行相对此工件的线性移动。

[\Corr]

Correction

数据类型：switch

如果存在该参数，则将通过指令CorrWrite而写入修正条目的修正数据添加到路径和目的位置。

[\TLoad]

Total load

数据类型：loaddata

\TLoad主动轴描述了移动中使用的总负载。总负载就是相关的工具负载加上该工具正在处理的有效负载。如果使用了\TLoad自变数，那么就不考虑当前tooldata中的loaddata。

如果\TLoad自变数被设置成load0，那么就不考虑\TLoad自变数，而是以当前tooldata中的loaddata作为代替。

想要使用\TLoad自变数，就必需将系统参数ModalPayLoadMode的数值设置成0。如果将ModalPayLoadMode设置成0，那么就再也无法使用指令GripLoad。

可用服务例程“负载标识”（LoadIdentify）来识别总负载。如果系统参数ModalPayLoadMode被设置成0，且系统正在运行该服务例程，那么操作员便可将相关工具的loaddata复制到一个现有的或新的loaddata永久变量中。

如果使用了关联到系统输入项SimMode（仿真模式）上的一个数字输入信号，那么便可在没有任何有效负载的情况下试运行该程序。如果该数字输入信号被设置成1，那么

下一页继续

就不考虑可选自变数\TLoad中的loaddata，而是以当前tooldata中的loaddata作为代替。

> **注意**
>
> 处理有效负载的默认功能是使用指令GripLoad，因此系统参数ModalPayLoadMode的默认值为1。

程序执行

有关线性移动的信息请参阅 MoveL 指令。

通过指令TriggLIOs，设置关于沿A到B路径的I/O信号的1-50个不同的触发活动是可能的。可使用的信号包括数字信号输出信号、数字组输出信号和模拟信号输出信号。在本指令终点前一特定距离处，或在本指令起点后一特定距离处，触发条件均得以满足。

本指令需要使用TriggData1、TriggData2或TriggData3参数之一或全部。但是，任意触发的使用是可选择的。为抑制触发的使用，可通过数据类型triggios/triggstrgo/triggiosdnum的数组元素，将分量used设置为FALSE。如果未使用数组元素，则TriggLIOs指令将起MoveL的作用，且将不会实施任何I/O活动。

如果本程序逐步向前，则实施I/O活动。逐步向后执行期间，将不会实施任何I/O活动。

如果将TriggData1、TriggData2或TriggData3参数中的分量EquipLag设置为负时间（延迟），则可将I/O信号设置在目的点后（ToPoint）。

如果使用参数TriggData2或TriggData3，则使用高达4294967295的值是可能的，该值是一组数字信号所能拥有的最大值（对于本系统而言，一组信号最多包含32个信号）。

更多示例

有关于如何使用指令TriggLIOs的更多例子阐述如下。

例 1

```
VAR triggios mytriggios{3}:= [[TRUE, 3, TRUE, 0, "go1", 55, 0],
     [TRUE, 15, TRUE, 0, "ao1", 10, 0], [TRUE, 3, FALSE, 0, "do1",
     1, 0]];
...
MoveL p1, v500, z50, gun1;
TriggLIOs p2, v500, \TriggData1:=mytriggios, z50, gun1;
MoveL p3, v500, z50, gun1;
```

将数字组输出信号go1设置为距p155 3 mm。将模拟信号输出信号设置为距p110 15 mm。将数字信号输出信号do1设置为距ToPointp23 mm。

例 2

```
VAR triggios mytriggios{3}:= [[TRUE, 3, TRUE, 0, "go1", 55, 0],
     [TRUE, 15, TRUE, 0, "ao1", 10, 0], [TRUE, 3, FALSE, 0, "do1",
     1, 0]];
VAR triggstrgo mytriggstrgo{3}:= [[TRUE, 3, TRUE, 0, "go2", "1",
     0], [TRUE, 15, TRUE, 0, "go2", "800000", 0], [TRUE, 4, FALSE,
     0, "go2", "4294967295", 0]];
```

下一页继续

1.299 TriggLIOs - 机械臂线性移动以及I/O事件

RobotWare - OS

续前页

```
VAR triggiosdnum mytriggiosdnum{3}:= [[TRUE, 10, TRUE, 0, "go3",
     4294967295, 0], [TRUE, 10, TRUE, 0, "ao2", 5, 0], [TRUE, 10,
     TRUE, 0, "do2", 1, 0]];
...
MoveL p1, v500, z50, gun1;
TriggLIOs p2, v500, \TriggData1:=mytriggios \TriggData2:=
     mytriggstrgo \TriggData3:=mytriggiosdnum, z50, gun1;
MoveL p3, v500, z50, gun1;
```

将数字组输出信号go1设置为距p155 3 mm。将模拟信号输出信号ao1设置为距p110 15 mm。将数字信号输出信号do1设置为距ToPointp23 mm。通过变量mytriggios，设置此类位置事件。变量mytriggstrgo进行设置，以使位置事件距离p13和15 mm。首先，将信号go2设置为1，随后，将其设置为800000。将本信号设置为距ToPoint p24294967295 4 mm。此为32位数字信号输出信号的最大值。变量mytriggiosdnum设置三个位置，甚至距p110 mm。首先，将信号go3设置为4294967295，随后，将ao2设置为5，最后，将do2设置为1。

错误处理

系统会生成下列可恢复错误，并在错误处理器中处理这些错误。系统变量ERRNO将被设置成：

名称	错误原因
ERR_NORUNUNIT	如果与I/O单元无接触。
ERR_GO_LIM	如果关于指定数字组输出信号signalname的编程setvalue参数超出限制。（用TriggData1、TriggData2或TriggData3进行声明）
ERR_AO_LIM	如果关于指定模拟信号输出信号signalname的编程setvalue参数超出限制。（用TriggData1或TriggData3进行声明）
ERR_CONC_MAX	如果已经超过运用参数\Conc的连续运动指令的数量。

限制

如果当前起动点偏离正常点，以致指令TriggLIOs的总定位长度短于正常水平（例如，在机械臂位于终点时TriggLIOs开始的时候），则若干或全部触发条件可能立即得以满足，并位于相同位置。在此类情况下，实施触发活动的顺序将变得不明确。用户程序中的程序逻辑可能不基于有关“不完整移动”触发活动的正常顺序。

针对各编程指令，指令TriggLIOs中触发的数量限制为50。如果在更近的距离处出现触发，则本系统可能无法对其进行处理。其取决于移动方式、所用TCP速度以及触发编程的紧密程度。存在此类限制，但是当出现此类问题时，难以对其进行预测。

语法

```
TriggLIOs
  ['\' Conc ',']
  [ ToPoint' :=' ] < expression (IN) of robtarget >
  [ '\' ID ':=' < expression (IN) of identno >] ','
  [ Speed ':=' ] < expression (IN) of speeddata >
  [ '\' T ':=' < expression (IN) of num > ]',
  [ '\' TriggData1' :=' ] < array {*} (VAR) of triggios >
  [ '\' TriggData2' :=' ] < array {*} (VAR) of triggstrgo >
  [ '\' TriggData3' :=' ] < array {*} (VAR) of triggiosdnum >
```

下一页继续

```
[Zone ’:=’ ] < expression (IN) of zonedata >
[ ’\’ Inpos’ :=’ < expression (IN) of stoppointdata > ] ´,’
[ Tool ’:=’ ] < persistent (PERS) of tooldata >
[ ’\’ WObj’ :=’ < persistent (PERS) of wobjdata > ]
[ ’\’ Corr ]
[ ’\’ TLoad’ :=’ < persistent (PERS) of loaddata > ] ’;’
```

相关信息

信息，关于	请参阅
接头机械臂移动以及I/O事件	第*792*页的*TriggJIOs - 接头机械臂移动以及I/O事件*
触发条件和触发活动的储存	第*1511*页的*triggios - Positioning events, trigg*
由32个信号组成的数字信号组的触发条件和触发活动的储存	第*1517*页的*triggstrgo - Positioning events, trigg*
触发条件和触发活动的储存	第*1513*页的*triggiosdnum - Positioning events, trigg*
事件对象的分配	技术参考手册 - 系统参数，运动主题 - 运动规划器 - 内部事件对象的数量一节
线性移动	技术参考手册 - RAPID概述
一般动作	技术参考手册 - RAPID概述
载荷的定义	第*1421*页的*loaddata - 加载数据*
关于如何使用TLoad总负载的例子。	第*388*页的*MoveL - 使机械臂沿直线移动*
定义机器人的载荷	第*223*页的*GripLoad - 定义机械臂的有效负载*
LoadIdentify，载荷识别服务例行程序	操作员手册 - 带 *FlexPendant* 的 *IRC5*
用于在仿真模式下无载荷运行机器人的系统输入信号 *SimMode*。 （主题 I/O，类型 System Input，行动值，*SimMode*）	技术参考手册 - 系统参数
激活和停用载荷的系统参数 *ModalPayLoadMode*。 （主题 Controller，类型 System Misc，行动值，*ModalPayLoadMode*）	技术参考手册 - 系统参数

1.300 TriggRampAO - 定义路径上的固定位置斜坡AO事件

手册用法

TriggRampAO（*Trigg Ramp Analog Output*）用于定义有关增加或减少机械臂移动路径沿线固定位置处模拟信号输出信号值的条件和行动以及对外部设备中的滞后进行时间补偿的可能性。

定义的数据用于在一个或多个后续TriggL、TriggC或TriggJ指令中执行。除此类指令外，TriggRampAO亦可用于CapL或CapC指令。

与同一TriggL/C/J指令相关的触发行动类型可以为TriggRampAO或任意TriggIO、TriggEquip、TriggSpeed、TriggInt或TriggCheckIO指令。允许任意类型的组合，除了仅允许有关相同TriggL/C/J指令中相同信号的一个TriggSpeed行动。

本指令仅可用于主任务T_ROB1，或者如果在*MultiMove*系统中，则可用于运动任务中。

基本示例

以下实例介绍了指令TriggRampAO：

另请参阅第*808*页的更多示例

例 1

```
VAR triggdata ramp_up;
...
TriggRampAO ramp_up, 0 \Start, 0.1, aolaser1, 8, 15;
MoveL p1, v200, z10, gun1;
TriggL p2, v200, ramp_up, z10, gun1;
```

当工具gun1的TCP在位于p1角路径中心之前0.1 s时，模拟信号aolaser1将开始从其当前逻辑值增加至新值8。当机械臂移动15 mm时，将完成整个增加过程。

例 2

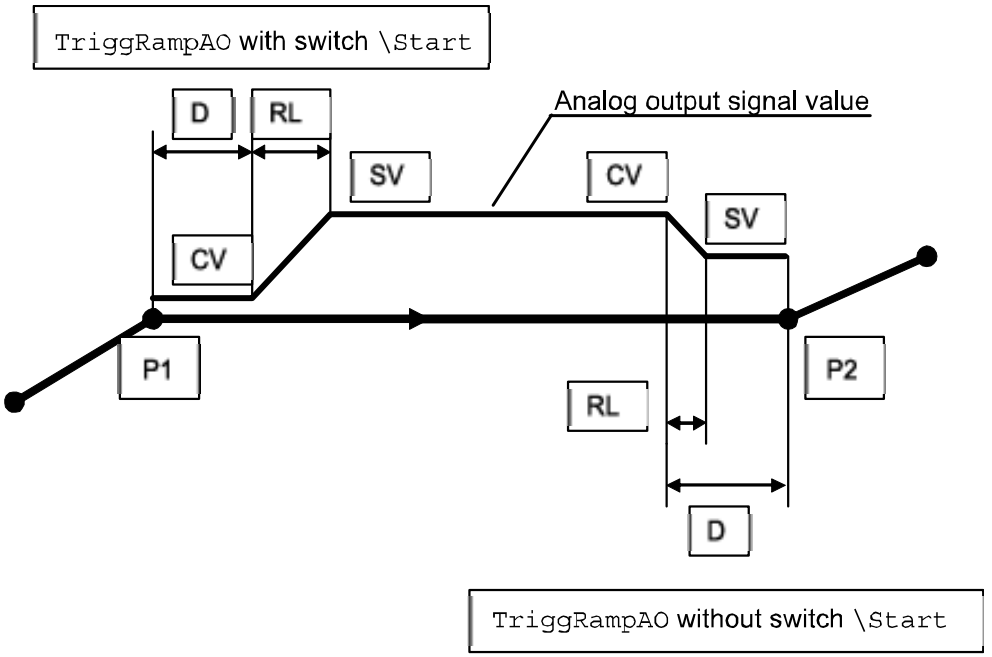
```
VAR triggdata ramp_down;
...
TriggRampAO ramp_down, 15, 0.1, aolaser1, 2, 10;
MoveL p3, v200, z10, gun1;
TriggL p4, v200, ramp_down, z10, gun1;
```

当工具gun1的TCP在位于p4角路径中心之前15 mm加0.1 s时，模拟信号aolaser1将开始从其当前逻辑值减小至新值2。当机械臂移动10时，将完成整个减小过程。

变元

```
TriggRampAO TriggData Distance [\Start] EquipLag AOutput SetValue
     RampLength [\Time] [\Inhib] [\InhibSetValue] [\Mode]
```

下一页继续



xx0600003433

D	参数Distance
RL	参数RampLength
CV	当前模拟信号值
SV	关于模拟新信号值的参数SetValue
P1	关于先前于移动指令的ToPoint
P2	关于实际TrigL/C/J 指令的ToPoint

TriggData

数据类型：triggdata

用于储存从本指令返回的triggdata的变量。随后，此类triggdata可用于后续TriggL、TriggC、TriggJ、CapL或CapC指令中。

Distance

数据类型：num

定义距角路径中心（模拟信号输出的斜坡应由此开始）的距离。

定义为距移动路径终点（ToPoint）的距离，以mm计（正值）（未设置参数\Start时适用）。

有关更多细节，请参见第807页的程序执行。

[\Start]

数据类型：switch

当关于参数Distance的距离与移动起点（先前的ToPoint）而非终点相关时使用。

EquipLag

Equipment Lag

数据类型：num

下一页继续

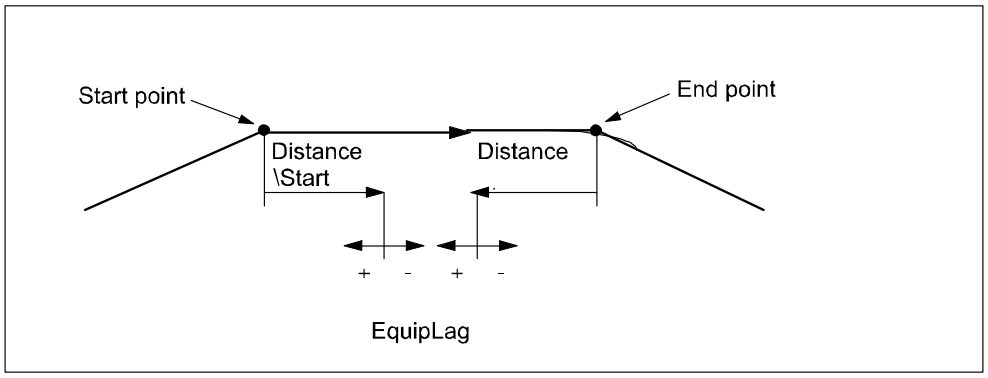
1.300 TriggRampAO - 定义路径上的固定位置斜坡AO事件
RobotWare - OS
续前页

指定外部设备的滞后，以s计。

关于外部设备滞后补偿，使用正参数值。正参数值意味着由TCP达到同移动起点或终点相关的指定距离点之前指定时间的机器人系统来开始使AO信号倾斜。

负参数值意味着由指定时间的机器人系统开始使AO信号倾斜。此后，TCP已通过同移动起点或终点相关的指定距离点。

本图显示了参数EquipLag的使用。



xx0500002262

AOutput

Analog Output

数据类型：signalao

模拟信号输出信号的名称。

SetValue

数据类型：num

模拟信号输出信号应增加或减少的值（必须处于该信号的容许逻辑范围值内）。模拟信号输出信号的当前值开始倾斜。

RampLength

数据类型：num

沿TCP移动路径的倾斜长度，以mm计。

[\Time]

数据类型：switch

如使用，则RampLength规定斜坡时间（以s计）而非斜坡长度。

如果后续TriggL，TriggC或TriggJ规定，应按照时间（参数\T）而非速度来完成整体移动，则必须予以使用。

[\Inhib]

Inhibit

数据类型：bool

用于约束运行时信号设置的永久变量标志的名称。

如果采用此项可选参数并且在到达I/O信号开始渐变的位置-时间，指定标记的实际值为真，那么指定信号（AOutput）将被设为0。

下一页继续

[\InhibSetValue]

InhibitSetValue

数据类型：bool, num or dnum

数据类型bool、num或dnum的永久变量的名称，或这三个基础数据类型的别名。

此可选参数只可结合可选参数Inhib一起使用。

如果采用此可选参数，并且在到达设置信号的位置-时间，可选参数Inhib采用的永久变量标记的值为真，那么将读取可选参数InhibSetValue采用的永久变量值，将采用该值来设置AOutput信号。

如果采用布尔永久变量，那么值“真”将被转换为值“1”，“假”将被转换为值“0”。

[\Mode]

数据类型：triggmode

用于在定义触发器时指定不同的行动模式。

程序执行

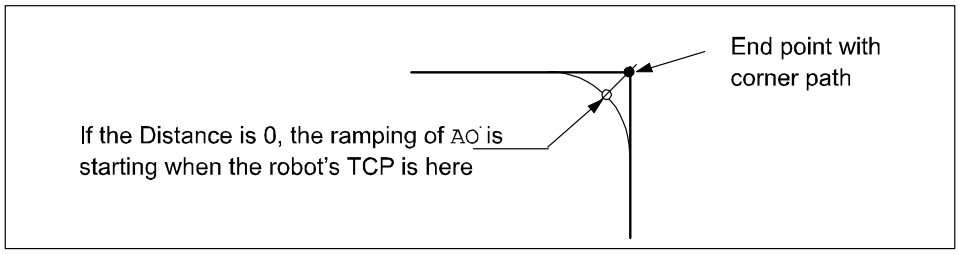
当运行指令TriggRampAO时，将触发条件储存在参数TriggData的指定变量中。

此后，当执行指令TriggL、TriggC或TriggJ之一时，以下方面适用于TriggRampAO中的定义：

本表描述了参数Distance中指定的距离：

线性移动	直线距离
圆形运动	圆弧长度
非线性移动	沿路径的适当弧长（以获取适当的准确度，距离不得超过弧长的一半）。

本图显示了位于角路径中的AO的斜坡。



xx0600003439

同任意TriggL/C/J相关的TriggRampAO的程序执行特征：

- 当机械臂达到机械臂路径上的指定Distance点（并补偿指定EquipLag）时，AO开始倾斜
- 在根据指定RampLength和编程TCP速度而计算出的时期期间，将执行倾斜函数。该计算考虑VelSet、手动速度覆盖，且在MAN模式下最大为250 mm/s，此外无任何其他速度限制。
- 每10 ms，将AO信号值从起始（当前读取）值更新为指定SetValue，从而产生楼梯式的形式。如果计算出的斜坡时间或指定斜坡时间大于0.5 s，则倾斜频率将降低：
  - < = 0.5 s，得出每10 ms最多50步

下一页继续

- < = 1 s，得出每20 ms最多50步
- < = 1.5 s，得出每30 ms等最多50步

同时通过FWD步进而非BWD步进模式，完成TriggRampAO行动。

在任意类型停止（ProgStop、紧急停止等等）下，如果偶然启用倾斜函数：

- 如果增加，则即刻将AO设置为旧值。
- 如果减少，则即刻将AO设置为新SetValue。

更多示例

有关于如何使用指令TriggRampAO的更多例子阐述如下。

例 1

```
VAR triggdata ramp_up;
VAR triggdata ramp_down;
...
TriggRampAO ramp_up, 0 \Start, 0.1, aolaser1, 8, 15;
TriggRampAO ramp_down, 15, 0.1, aolaser1, 2, 10;
MoveL p1, v200, z10, gun1;
TriggL p2, v200, ramp_up, \T2:=ramp_down, z10, gun1;
```

在该例子中，AO的增加和减少均通过相同移动路径上的相同TriggL指令完成。如果移动路径足够长，其在没有任何AO设置干扰的情况下起作用。

当工具gun1的TCP在位于p1处角路径中心之前0.1 s时，模拟信号aolaser1将开始使其逻辑值从当前值增加至新值8。当机械臂移动15 mm时，将完成整个增加过程。

当工具gun1的TCP在位于p2处角路径中心之前15 mm加0.1 s时，模拟信号aolaser1将开始使其逻辑值从当前值8减少至新值2。当机械臂移动10 mm时，将完成整个增加过程。

错误处理

如果关于指定模拟信号输出信号AOutput的编程SetValue参数超出限制，则将系统变量ERRNO设置为ERR_AO_LIM。可通过错误处理器来处理该错误。

如果信号变量是在RAPID中声明的变量，且其尚未与通过指令AliasIO而在I/O配置中确定的I/O信号相连，随后，将系统变量ERRNO设置为ERR_NO_ALIASIO_DEF。可通过错误处理器来处理该错误。

限制

在角路径中或其他加速或减速阶段期间，针对更低的TCP速度，将不会补偿模拟信号输出信号值（AO不与TCP速度成比例）。

将在路径上的指定位置处，仅完成AO倾斜的起点。将通过高精度的“停滞计算”，完成增加或减少：

- 在恒速下，AO倾斜终点的偏离将比规定值低。
- 加速或减速阶段期间，例如在停止点附近，偏离将较高。
- 建议：在增加之前以及减少之后，使用角路径。

如果在相同模拟信号输出信号上使用两个或多个TriggRampAO，且与相同的TriggL/C/J指令相关，且两个或多个RampLength均位于机械臂路径的相同部分，则AO设置将相互作用。

下一页继续

如果距实际ToPoint的指定 Distance不在当前TriggL/C/J 指令的移动长度内，则当通过先前的ToPoint时，将开始位置（+/-时间）相关斜坡AO事件。如果距先前ToPoint的指定Distance不在当前TriggL/C/J指令的移动长度内，则当通过实际的ToPoint时，将开始位置（+/-时间）相关斜坡AO事件（使用参数 \Start）。

在任意类型的停止（ProgStop、紧急停止等等）后，不支持重启倾斜AO函数。

在上电失败重启时，从当前上电失败位置开始起动TriggL/C/J指令。

语法

```
TriggRampAO
  [ TriggData ':=' ] < variable (VAR) of triggdata > ','
  [ Distance ':=' ] < expression (IN) of num >
  [ '\' Start ] ','
  [ EquipLag ':=' ] < expression (IN) of num > ','
  [ AOutput ':=' ] < variable (VAR) of signalao> ','
  [ SetValue ':=' ] < expression (IN) of num> ','
  [ RampLength ':=' ] < expression (IN) of num> ','
  [ '\' Time ]
  [ '\' Inhib' :=' < persistent (PERS) of bool> ]
  [ '\' InhibSetValue' :=' < persistent (PERS) of anytype> ]
  [ Mode' :=' ] < expression (IN) of triggmode> ';'
```

相关信息

信息，关于	请参阅
触发的使用	第*785*页的*TriggL* - 关于事件的机械臂线性运动 第*747*页的*TriggC* - 关于事件的机械臂圆周移动 第*778*页的*TriggJ* - 关于事件的轴式机械臂运动
其他触发的定义	第*763*页的*TriggEquip* - 定义路径上的固定位置和时间*I/O*事件
触发数据的储存	第*1510*页的触发数据 - 定位事件，触发 第*1515*页的*triggmode* - 触发行动模式
模拟信号输出信号的设置	第*580*页的*SetAO* - 改变模拟信号输出信号的值 第*1476*页的*signalxx* - 数字信号和模拟信号
事件预置时间的配置	技术参考手册 - 系统参数

1.301 TriggSpeed - 定义与固定位置-时间尺度事件成比例的TCP速度模拟信号输出

手册用法

TriggSpeed用于定义关于模拟信号输出信号控制的条件和行动，其输出值与实际TCP速度成比例。可在沿机械臂移动路径的固定位置-时间处，指定模拟信号输出的开始、缩放和结束。针对外部设备中的滞后、模拟信号输出的开始、缩放和结束以及针对机械臂的速度下降，使用时间补偿是可能的。

在后续TriggL、TriggC或TriggJ 指令之一或多个中使用已确定的数据。

本指令仅可用于主任务T_ROB1，或者如果在MultiMove系统中，则可用于运动任务中。

基本示例

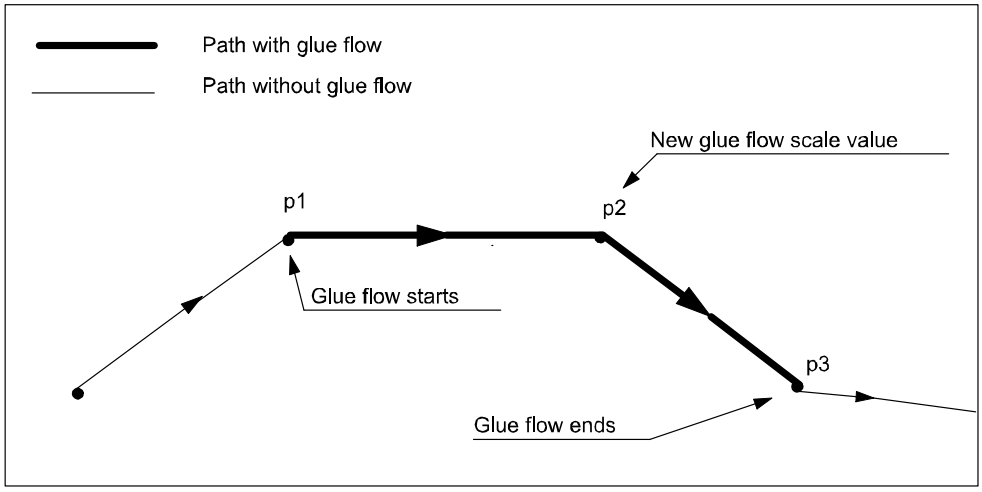
以下实例介绍了指令TriggSpeed：

另请参阅第*814*页的更多示例

例 1

```
VAR triggdata glueflow;
TriggSpeed glueflow, 0, 0.05, glue_ao, 0.8\DipLag:=0.04
     \ErrDO:=glue_err;
TriggL p1, v500, glueflow, z50, gun1;
TriggSpeed glueflow, 10, 0.05, glue_ao, 1;
TriggL p2, v500, glueflow, z10, gun1;
TriggSpeed glueflow, 0, 0.05, glue_ao, 0;
TriggL p3, v500, glueflow, z50, gun1;
```

下图阐明了关于TriggSpeed序列的例子。



xx0500002329

当TCP位于点p1前0.05 s时，启用胶流（模拟输出glue_ao）以及刻度值0.8；当TCP位于点p2前10 mm加0.05 s时，启用新胶流刻度值1；当TCP位于点p3前0.05 s时，启用胶流终点（刻度值0）。

对机械臂的速度下降进行时间补偿，以致模拟信号输出信号glue_ao在TCP速度出现下降前0.04 s受到影响。

下一页继续

如果在glue_ao中计算出的逻辑模拟信号输出值溢出，则设置数字信号输出信号glue_err。如果不再存在溢出，则重置glue_err。

变元

```
TriggSpeed TriggData Distance [\Start] ScaleLag AOp ScaleValue
    [\DipLag] [\ErrDO] [\Inhib] [\InhibSetValue] [\Mode]
```

TriggData

数据类型：triggdata

从该指令返回用于储存triggdata的变量。随后，将此类triggdata 用于TriggL、TriggC或TriggJ指令。

Distance

数据类型：num

定义路径上的位置，以改变模拟信号输出值。

定义为距移动路径终点的距离，以mm计（正值）（未设置参数 Start时适用）。

有关更多细节，请参见第*813*页的程序执行。

[\Start]

数据类型：switch

当关于参数Distance的距离始于移动起点而非终点时使用。

ScaleLag

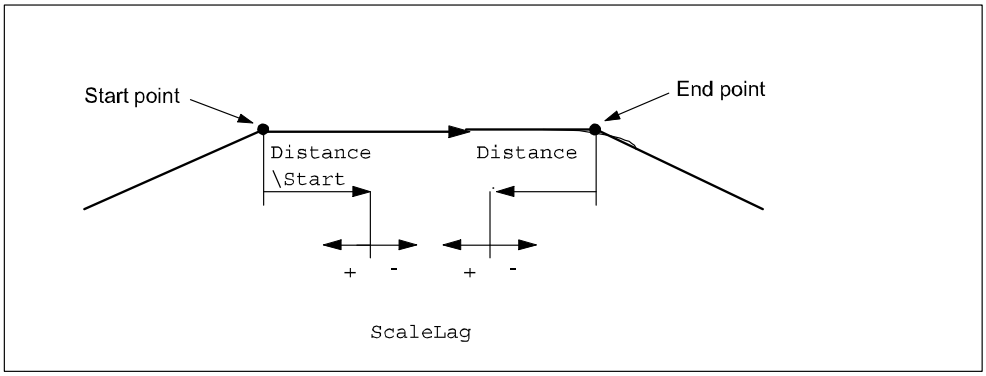
数据类型：num

指定滞后作为外部设备中的时间，以s计（正值），以改变模拟信号输出值（开始、缩放和结束）。

关于外部设备滞后补偿，该参数值意味着模拟信号输出信号由TCP达到相对于移动起点或终点的指定距离前一指定时间的机械臂来设置。

本参数亦可用于延长模拟信号输出，使其超过端点。设置机械臂应当维持模拟信号输出的时间，以秒计。设置时间为负。限制为-0.10秒。

下图阐明了参数ScaleLag的使用。



xx0500002330

AOp

Analog Output

数据类型：signalao

下一页继续

1.301 TriggSpeed - 定义与固定位置-时间尺度事件成比例的TCP速度模拟信号输出
RobotWare - OS
续前页

模拟信号输出信号的名称。

ScaleValue

数据类型：num

模拟信号输出信号的刻度值。

由机械臂计算关于模拟信号的实际输出值：

- 逻辑输出值 = 刻度值*实际TCP速度，以mm/s计
- 实际输出值 = 关于实际模拟信号输出信号以及作为输入的上述逻辑输出值的配置中的相关定义。

[\DipLag]

数据类型：num

由于机械臂速度下降，因此，当改变模拟信号输出值时，指定滞后作为关于外部设备的时间，以s计（正值）。

关于外部设备滞后补偿，该参数值意味着模拟信号输出信号由TCP速度出现下降前一指定时间的机械臂来设置。

该参数仅可由机械臂使用，以满足在一系列TriggSpeed指令中的首个TriggSpeed（结合TriggL、TriggC或TriggJ之一）。第一个指定的参数值适用于序列中的所有以下TriggSpeed。

[\ErrDO]

Error Digital Output

数据类型：signaldo

用于报告模拟值溢出的数字信号输出信号的名称。

如果在移动期间，关于参数AOp中信号的逻辑模拟输出值的计算因超速而产生溢出，则设置该信号，并将实际模拟信号输出值降低至最大值。如果不再存在溢出，则重置信号。

该参数仅可由机械臂使用，以满足在一系列TriggSpeed指令中的首个TriggSpeed（结合TriggL、TriggC或TriggJ之一）。第一个给定的参数值适用于序列中的所有以下TriggSpeed。

[\Inhib]

Inhibit

数据类型：bool

用于约束运行时模拟信号设置的永久变量标志的名称。

当设置模拟信号时，如果使用该可选参数，且指定标记的实际值为TRUE，则将指定信号AOp设置为0而非计算出的值。

该参数仅可由机械臂使用，以满足在一系列TriggSpeed指令中的首个TriggSpeed（结合TriggL、TriggC或TriggJ之一）。第一个给定的参数值适用于序列中的所有以下TriggSpeed。

[\InhibSetValue]

InhibitSetValue

数据类型：bool, num or dnum

数据类型bool、num或dnum的永久变量的名称，或这三个基础数据类型的别名。

下一页继续

此可选参数只可结合可选参数Inhib一起使用。

如果采用此可选参数，并且在到达设置信号的位置-时间，可选参数Inhib采用的永久变量标记的值为真，那么将读取可选参数InhibSetValue采用的永久变量值，将采用该值来设置AOp信号。

如果采用布尔永久变量，那么值“真”将被转换为值“1”，“假”将被转换为值“0”。

[\Mode]

数据类型：triggmode

用于在定义触发器时指定不同的行动模式。

程序执行

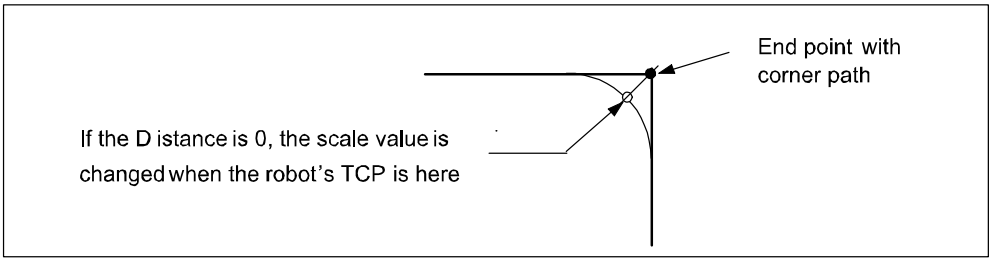
当运行指令TriggSpeed时，将触发条件储存在参数TriggData的指定变量中。

此后，当执行指令TriggL、TriggC或TriggJ之一时，则以下方面适用于TriggSpeed中的定义：

关于用参数Distance指定的距离，请参见下表：

线性移动	直线距离
圆形运动	圆弧长度
非线性移动	沿路径的适当弧长（以获取适当的准确度，距离不得超过弧长的一半）。

下图阐明了角路径上的固定位置-时间刻度值事件。



xx0500002331

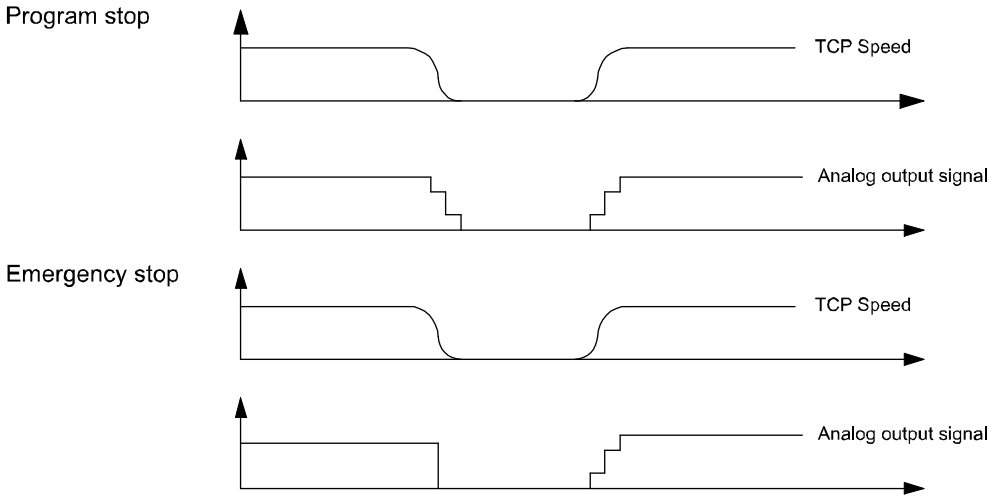
如果距终点（起点）的指定距离不在当前指令（TriggL、TriggC或TriggJ）的移动长度内，则当通过起点（终点）时，将产生位置-时间相关刻度值事件。

由TriggL、TriggC或TriggJ指令之一使用的第一个TriggSpeed，将在系统内部创建一个与模拟信号输出信号名称相同的过程。相同的过程将由所有后续TriggL、TriggC或TriggJ使用，其指代相同的信号名称，并通过TriggSpeed 指令来设置。

下一页继续

1.301 TriggSpeed - 定义与固定位置-时间尺度事件成比例的TCP速度模拟信号输出
RobotWare - OS
续前页

如果程序紧急停止，则本过程将立即把模拟信号输出设置为0。如果程序停止，则模拟信号输出信号将始终与TCP速度成比例，直至机械臂保持静止。使本过程始终“启用”，并准备重启。当机械臂重启时，信号从一开始便直接与TCP速度成比例。



xx0500002332

如果在当前，后续TriggL、TriggC或TriggJ均未在队列中，则在通过值0来处理尺度事件后，本过程将“停用”。

更多示例

有关于指令TriggSpeed的更多例子阐述如下。

例 1

```
VAR triggdata flow;
TriggSpeed flow, 10 \Start, 0.05, flowsignal, 0.5 \DipLag:=0.03;
MoveJ p1, v1000, z50, tool1;
TriggL p2, v500, flow, z50, tool1;
```

在TCP通过位于起点p．后10 mm处的点之前，将模拟信号输出信号flowsignal 设置为逻辑值 = （0.5*实际TCP速度，以mm/s计）0.05 s。在移动至p2期间，将输出值调整为与实际TCP速度成比例。

```
...
TriggL p3, v500, flow, z10, tool1;
```

机械臂从p2移动至p3，且模拟输出值与实际TCP速度成比例。通过角路径z10期间，在机械臂降低TCP速度前0.03 s时，将减少模拟信号输出值。

限制

有关于指令TriggSpeed的限制阐述如下。

位置-时间相关刻度值事件的精度

关于刻度值事件±5 ms的典型绝度精度值。

关于刻度值事件±2 ms的典型重复精度值。

TCP速度下降适应的精度（减速-加速阶段）

关于TCP速度下降适应±5 ms的典型绝对精度值。

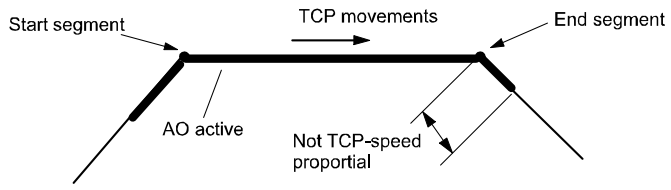
关于TCP速度下降适应±2ms的典型重复精度值（该值取决于已配置的*Path resolution*）。

下一页继续

负ScaleLag

如果使用关于参数ScaleLag的负值来移动下一移动指令的零刻度，则在出现系统停止时，模拟信号输出信号将不会重置。紧急停止将始终重置模拟信号。

在移动指令终点后，模拟信号不再与TCP速度成比例。

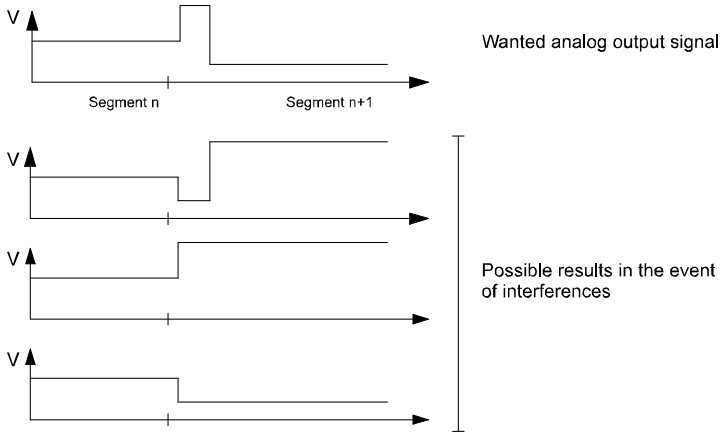


xx0500002333

错误处理

如果信号变量是在RAPID中声明的变量，且其尚未与通过指令AliasIO而在I/O配置中确定的I/O信号相连，随后，将系统变量ERRNO设置为ERR_NO_ALIASIO_DEF。可通过错误处理器来处理该错误。

给定两个连续的移动命令以及TriggL/TriggSpeed指令。参数ScaleLag中的负值使其可能将尺度事件从第一个移动命令移动至第二个移动命令开始。如果第二个移动命令位于刻度起点，则在两个刻度相互干扰时不存在控制。



xx0500002334

相关的系统参数

系统参数*Event Preset Time*用于延迟机械臂，使其在机械臂穿过该位置前，可能启用/控制外部设备。

下表阐明了关于系统参数*Event Preset Time*设置的建议，其中，典型伺服滞后为0.040 s。

ScaleLag	DipLag	用以避免运行时执行错误所需要的*Event Preset Time*	用以获得最佳精度而推荐的*Event Preset Time*
ScaleLag > DipLag	总是	如果DipLag > ServoLag，则DipLag	ScaleLag（以s计）加上0.090 s

下一页继续

1.301 TriggSpeed - 定义与固定位置-时间尺度事件成比例的TCP速度模拟信号输出
RobotWare - OS
续前页

ScaleLag	DipLag	用以避免运行时执行错误所需要的*Event Preset Time*	用以获得最佳精度而推荐的*Event Preset Time*
ScaleLag < DipLag	DipLag < Servo Lag	- " -	0.090 s
- " -	DipLag > Servo Lag	- " -	DipLag（以s计）加上 0.030 s

语法

```
TriggSpeed
  [ TriggData ':=' ] < variable (VAR) of triggdata>','
  [ Distance' :=' ] < expression (IN) of num>
  [ '\' Start ] ','
  [ ScaleLag':=' ] < expression (IN) of num> ','
  [ AOp ':='] < variable (VAR) of signalao> ','
  [ ScaleValue' :=' ] < expression (IN) of num>
  [ '\' DipLag' :=' < expression (IN) of num> ]
  [ '\' ErrDO' :=' < variable (VAR ) of signaldo> ]
  [ '\' Inhib' :=' < persistent (PERS ) of bool >]
  [ '\' InhibSetValue' :=' < persistent (PERS) of anytype> ]
  [ Mode' :=' ] < expression (IN) of triggmode> ';'
```

相关信息

信息，关于	请参阅
触发的使用	第*785*页的*TriggL - 关于事件的机械臂线性运动* 第*747*页的*TriggC - 关于事件的机械臂圆周移动* 第*778*页的*TriggJ - 关于事件的轴式机械臂运动*
其他触发的定义	第*773*页的*TriggIO - 定义停止点附近的固定位置或时间I/O事件* 第*769*页的*TriggInt - 定义与位置相关的中断* 第*763*页的*TriggEquip - 定义路径上的固定位置和时间I/O事件*
触发数据的储存	第*1510*页的触发数据 - 定位事件，触发 第*1515*页的*triggmode - 触发行动模式*
Event preset time的配置	技术参考手册 - 系统参数
Advanced RAPID	应用手册 - 控制器软件*IRC5*

1.302 TriggStopProc - 产生关于停止时触发信号的重启数据

手册用法

本指令TriggStopProc在系统中建立了用于指定过程信号零点设置的内部监督过程，并在系统中生成了各程序停止（STOP）或紧急停止（QSTOP）时指定永久变量中的重启数据。

TriggStopProc和数据类型restartdata旨在用于RAPID（NOSTEPIN程序）中定义的自身过程指令的程序停止(STOP)或紧急停止(QSTOP)之后的重启。

可能通过用户定义的RESTART事件程序，分析当前的重置数据，通过指令StepBwdPath沿路径后退，并在重启移动之前，启用合适的过程信号。

本指令仅可用于主任务T_ROB1，或者如果在*MultiMove*系统中，则可用于任何运动任务中。

关于MultiMove系统的注意事项：在系统中，同一时间仅可启用一个与指定阴影信号名称（参数 ShadowDO）相同的TriggStopProc支持过程。这意味着TriggStopProc对最后执行的程序任务中的程序停止或紧急停止进行监督。

变元

```
TriggStopProc RestartRef [\DO] [\GO1] [\GO2] [\GO3] [\GO4] ShadowDO
```

RestartRef

Restart Reference

数据类型：restartdata

每次停止程序执行后，其重启数据可用的永久变量。

[\DO1]

Digital Output 1

数据类型：signaldo

当停止程序执行时，有待设置为零并通过重启数据进行监督的关于数字过程信号的信号变量。

[\GO1]

Group Output 1

数据类型：signalgo

当停止程序执行时，有待设置为零并通过重启数据进行监督的关于数字组过程信号的信号变量。

[\GO2]

Group Output 2

数据类型：signalgo

当停止程序执行时，有待设置为零并通过重启数据进行监督的关于数字组过程信号的信号变量。

[\GO3]

Group Output 3

数据类型：signalgo

下一页继续

1.302 TriggStopProc - 产生关于停止时触发信号的重启数据
RobotWare - OS
续前页

当停止程序执行时，有待设置为零并通过重启数据进行监督的关于数字组过程信号的信号变量。

[\GO4]

Group Output 4

数据类型：signalgo

当停止程序执行时，有待设置为零并通过重启数据进行监督的关于数字组过程信号的信号变量。

必须使用至少一个可选参数D01, GO1 ... GO4。

ShadowDO

Shadow Digital Output

数据类型：signaldo

关于数字信号的信号变量，其必须反映是否沿机械臂路径启用本过程。

通过STOP或QSTOP时的过程TriggStopProc ，将不会把该信号设置为零，但是，将通过restartdata来监测其值。

程序执行

TriggStopProc的设置和执行

必须从以下两种情况调用TriggStopProc：

- 通过START事件程序或本程序的单元部分（设置PP到main，消除TriggStopProc的内部过程）
- POWERON事件程序（电源关闭，清除关于TriggStopProc的内部过程）

关于TriggStopProc 的过程的内部名称与参数ShadowDO中的信号名称相同。如果TriggStopProc与参数ShadowDO中的信号名称相同，并在相同或另一程序任务中执行两次，则仅最后执行的TriggStopProc有效。

TriggStopProc的执行仅开始对STOP和QSTOP的I/O信号进行监督。

程序停止STOP

本过程TriggStopProc包含以下步骤：

1 等待，直至机械臂在路径上保持静止。
2 储存所有使用的过程信号的当前值（符合restartdata的预置值）。设置所有使用的信号为零，ShadowDO.除外。
3 如果一些过程信号在该时间内改变其值，则在下一时隙（约500 ms）期间，采取以下措施：
   - 再次储存当前值（符合restatdata)的后置值
   - 将该信号设置为零，ShadowDO除外
   - 统计转换信号ShadowDO的值（两侧）的次数
4 用重置数据更新指定的永久变量。

紧急停止（QSTOP）

本过程TriggStopProc包含以下步骤：

1 尽快进行下一步骤。

下一页继续

2 储存所有使用的过程信号的当前值（符合restartdata的预置值）。设置所有使用的信号为零，ShadowDO.除外。
3 如果一些过程信号在该时间内改变其值，则在下一时隙（约500 ms）期间，采取以下措施：
   - 再次储存其当前值（符合restatdata的后置值
   - 将该信号设置为零，ShadowDO除外
   - 统计转换信号ShadowDO的值（两侧）的次数
4 用重置数据更新指定的永久变量。

关于过程重启的临界区域

机械臂伺服和外部设备均具有某种滞后。设计Trigg家族中的所有指令，以便在不依赖于外部设备中不同滞后的情况下，将所有信号设置在机械臂路径上的适当位置，从而获得尽可能好的过程结果。为此，在机械臂于程序停止（STOP）时保持静止，或登记紧急停止（QSTOP）之后，可延迟系统中0-80ms之间的I/O信号的设置。由于重启功能存在此种劣势，因此，预置值、后置值以及阴影两侧均通过重启数据引入。

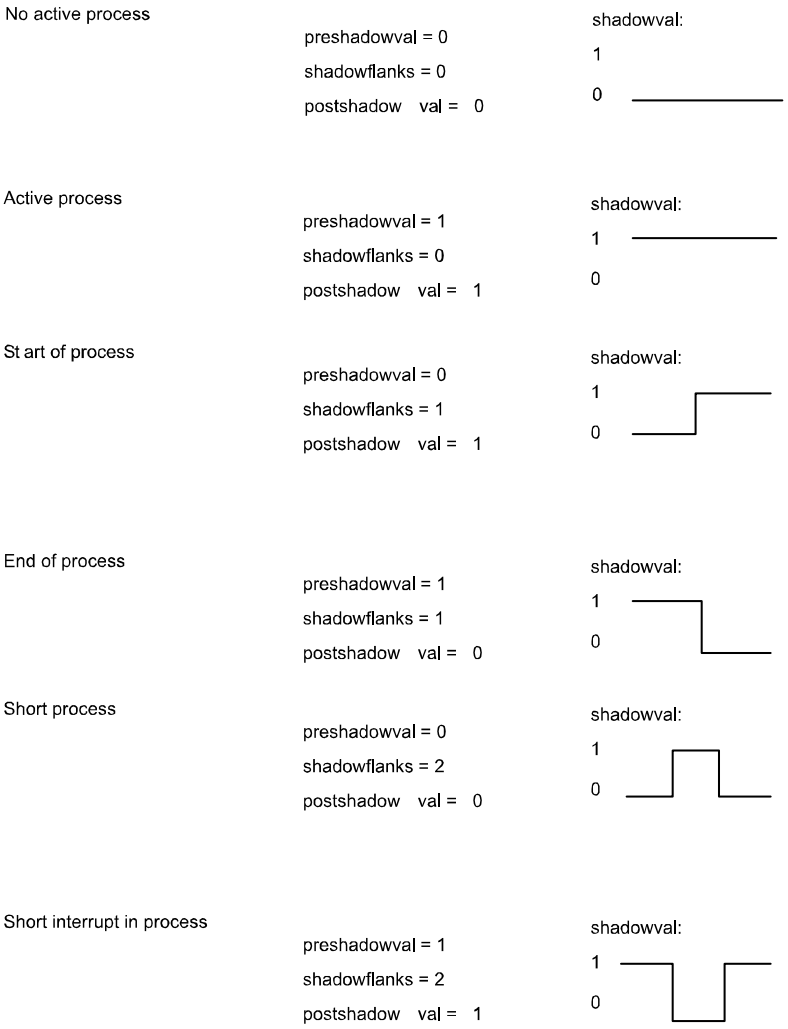
如果该临界时隙0-80ms符合以下应用过程情形，则难以实施较好的过程重启：

- 在应用过程开始时
- 在应用过程结束时
- 在短应用过程期间
- 在应用过程中的短中断期间

下一页继续

1.302 TriggStopProc - 产生关于停止时触发信号的重启数据
RobotWare - OS
续前页

下图阐明了临界时隙0-80ms内STOP或QSTOP下的各过程阶段



xx0500002326

实施重启

必须通过RESTART事件程序，重启沿机械臂路径的过程指令（NOSTEPIN程序）。
RESTART事件程序可包含以下步骤：

	操作
1.	QSTOP之后，在程序开始时恢复路径。
2.	根据最近的STOP或QSTOP，分析重启数据。

下一页继续

	操作
3.	根据分析结果，确定过程重启策略： • 过程有效，进行过程重启 • 过程无效，不进行过程重启 • 根据过程应用的类型，采取适当的行动： - 过程的开始 - 过程的结束 - 短过程 - 过程中的短中断
4.	沿路径后退。
5.	继续进行本程序，导致移动重启。

如果在任意STOP或QSTOP事件程序中等待，直至TriggStopProc过程准备好进行例如WaitUntil (myproc.restartstop=TRUE), \MaxTime:=2;，则用户必须始终通过RESTART事件程序以及例如myproc.restartstop:=FALSE，重置标记。此后，重置就绪。

错误处理

信号变量是在RAPID中声明的变量。其尚未与通过指令AliasIO而在I/O配置中确定的I/O信号相连。如果使用该信号，则将系统变量ERRNO设置为ERR_NO_ALIASIO_DEF，并用错误处理器继续执行。

如果没有同I/O单元接触，则将系统变量ERRNO设置为ERR_NORUNUNIT，并用错误处理器继续执行。

限制

电源故障后，不支持重启过程指令。

语法

```
TriggStopProc
  [ RestartRef ':=' ] < persistent (PERS) of restartdata>
  [ '\' DO1 ':=' < variable (VAR) of signaldo>
  [ '\' GO1 ':=' < variable (VAR) of signalgo> ]
  [ '\' GO2 ':=' < variable (VAR) of signalgo> ]
  [ '\' GO3 ':=' < variable (VAR) of signalgo> ]
  [ '\' GO4 ':=' < variable (VAR) of signalgo> ] ','
  [ ShadowDO ':=' ] < variable (VAR) of signaldo> ';'
```

相关信息

信息，关于	请参阅
过程指令	第*785*页的*TriggL - 关于事件的机械臂线性运动* 第*747*页的*TriggC - 关于事件的机械臂圆周移动*
重启数据	第*1459*页的*restartdata - 重启关于触发信号的数据*
沿路径后退。	第*674*页的*StepBwdPath - 在路径上向后移动一步*
Advanced RAPID	应用手册 - 控制器软件*IRC5*

1.303 TryInt - 测试数据对象是否为有效整数

手册用法

TryInt用于测试给定数据对象是否为有效整数。

基本示例

以下实例介绍了指令TryInt：

例 1

```
VAR num myint := 4;
...
TryInt myint;
```

将评估myint的值，且由于4为有效整数，因此，继续程序执行。

例 2

```
VAR dnum mydnum := 20000000;
...
TryInt mydnum;
```

将评估mydnum的值，且由于20000000为有效双数值整数，因此，继续程序执行。

例 3

```
VAR num myint := 5.2;
...
TryInt myint;
...
ERROR
  IF ERRNO = ERR_INT_NOTVAL THEN
    myint := Round(myint);
    RETRY;
  ENDIF
```

将评估myint的值，且由于5.2 并非有效的整数，因此，将引起错误。通过错误处理器，可将myint舍入为5，并再一次执行指令TryInt。

变元

```
TryInt DataObj | DataObj2
```

DataObj

Data Object

数据类型：num

测试数据对象是否为有效整数。

DataObj2

Data Object 2

数据类型：dnum

测试数据对象是否为有效整数。

程序执行

测试给定数据对象：

- 如果其为有效整数，则使用下一指令继续执行。

下一页继续

- 如果其并非有效整数，则通过实际无返回值程序中的错误处理器继续执行。

错误处理

如果DataObj 包含小数值，则将变量ERRNO设置为ERR_INT_NOTVAL。

如果DataObj的值大于或小于数据类型num的整数值范围，则将变量ERRNO设置为ERR_INT_MAXVAL。

如果DataObj2的值大于或小于数据类型dnum的整数值范围，则将变量ERRNO设置为ERR_INT_MAXVAL。

可用错误处理器来处理此类错误。

注意，由于可忽略.0，因此，将3.0值评估为整数。

语法

```
TryInt
  [ DataObj ‘:=’ ] < expression (IN) of num>
  | [ DataObj2 ‘:=’ ] < expression (IN) of dnum>’ ;’
```

相关信息

信息，关于	请参阅
数据类型num	第*1435*页的*num - 数值*

1.304 TRYNEXT - 跳转至引起错误的指令

手册用法

TRYNEXT指令用于在错误后恢复执行，以本指令开始，随后进行引起错误的指令。

基本示例

以下实例介绍了指令TryNext：

例 1

```
reg2 := reg3/reg4;
...
ERROR
  IF ERRNO = ERR_DIVZERO THEN
    reg2:=0;
    TRYNEXT;
  ENDIF
```

试图用reg3除以reg4。如果reg4等于0（除以零），则跳转至reg2分配为0的错误处理器。随后，TRYNEXT 指令用于继续下一指令。

程序执行

通过继引起错误的指令以后的指令，继续程序执行。

限制

本指令仅可存在于程序的错误处理器中。

语法

```
TRYNEXT';'
```

相关信息

信息，关于	请参阅
错误处理器	技术参考手册 - *RAPID*语言概览

1.305 TuneReset - 重置伺服调节

手册用法

TuneReset用于将所有机械臂轴和外部机械单元的动力学行为重置为其正常值。

本指令仅可用于主任务T_ROB1，或者如果在*MultiMove*系统中，则可用于运动任务中。

基本示例

以下实例介绍了指令TuneReset：

例 1

```
TuneReset;
```

将所有轴的调节值重置为100%。

程序执行

将所有轴的调节值重置为100%。

通过执行指令TuneReset，自动地设置所有轴的默认伺服调节值

- 重启时。
- 当载入一段新程序时。
- 当从起点开始执行程序时。

语法

```
TuneReset ';'
```

相关信息

信息，关于	请参阅
调节伺服	第826页的*TuneServo - 调节伺服*

1.306 TuneServo - 调节伺服

手册用法

TuneServo用于调节机械臂上单独轴的动力学行为。

对于绝大多数应用，没有必要使用TuneServo，但是对于一些应用，需要TuneServo，以获得期望精度。在许多情况下，可通过选择预定义*Motion Process Mode*，参见技术参考手册 - 系统参数，或通过修改预定义*Motion Process Mode*，替代TuneServo的使用。

对于外轴，TuneServo可用于负载适应。

当机械臂正在运动时，避免进行TuneServo命令。其可以产生瞬间高扭矩，由此引起错误指示和停止。

本指令仅可用于主任务T_ROB1，或者如果在*MultiMove*系统中，则可用于运动任务中。

注意

为获得最佳调节，有必要使用正确的负载数据。使用TuneServo前，进行上述检查。

警告

不正确地使用TuneServo会引起可对机械臂造成损坏的振荡移动或扭矩。当使用TuneServo时，必须谨记于此，并始终小心。

描述

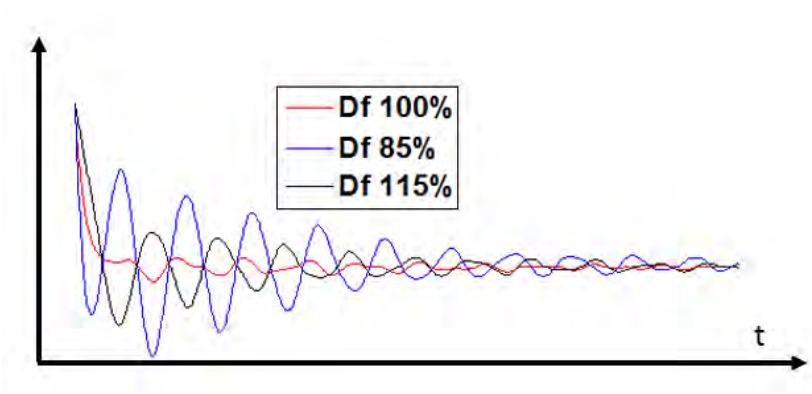
减少过度和振动-TUNE_DF

TUNE_DF可用于调整特定轴的预测机械谐振频率。95%的调节值将降低谐振频率5%。TUNE_DF最常见的用途是弥补基础的刚度不足，即挠性基础。在这种情况下，降低轴1和2的调节值，通常将其降低至80%与99%之间。

很少将TUNE_DF用于轴3 - 6，且通常不建议这么做。调节轴4 - 6，以补偿扩展灵活有效负载的谐振频率这一情况除外。

下一页继续

正确地调整（既不过高，也不过低）TUNE_DF，减少过度和振动。调整TUNE_DF时应小心，因为过高或过低的调节值会在很大程度上损害移动。下图显示了一个例子。在这种情况下，100%的调节值可获得最佳结果。



xx1400001280

通过使用推荐的TuneMaster，可自动优化调节值。

关于手动调节，有关调节轴1的RAPID代码片段实例如下：

```
MoveAbsJ [[0,0,0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]],
     v200, fine, myTool;
FOR DF FROM 80 TO 100 STEP 5 DO
  TuneServo ROB_1,1,DF\Type:=TUNE_DF;
  MoveAbsJ [[2,0,0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]],
       vmax, fine, myTool;
  WaitTime 1;
  MoveAbsJ [[0,0,0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]],
       vmax, fine, myTool;
  WaitTime 1;
ENDFOR
TuneReset;
```

此处，5%步骤中的调节值发生改变，2%步骤亦可予以使用。注意，移动应当较短，通常为2度。应当将机械臂定位在典型的工作区位置中。应当通过外观检查，选择可使过度和振动情形最小化的调节值。

亦可通过降低TUNE_DH的调节值，或通过使用AccSet来降低加速度，以此降低过度和振动。在许多情况中，上述操作是最佳解决方案。但是，如果可通过TUNE_DF来解决问题，则周期时间不受影响，因此，TUNE_DF的使用是最佳解决方案。

针对*Mounting Stiffness Factor*有效的机械臂，参见技术参考手册 - 系统参数中的*Motion Process Mode*，使用*Mounting Stiffness Factor*以补偿挠性基础，替代TUNE_DF的使用。

减少过度和振动-TUNE_DH

通过调整系统的有效带宽，TUNE_DH可用于增加机械臂路径的平滑度。仅可降低调节值，且高于100%的值将不会影响移动。小于100%的调节值会减小带宽，并增加平滑度，由此降低过度和振动。

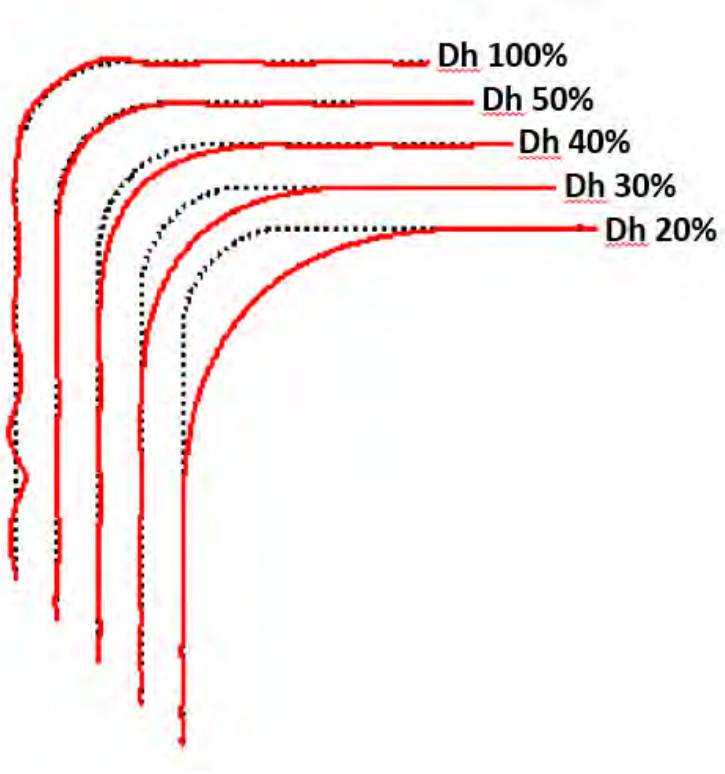
TUNE_DH仅增加精点中的周期时间，而加速度的降低会增加沿机械臂路径的周期时间。因此，与通过使用AccSet来降低加速度相比，使用TUNE_DH可成为一种用于减少振动和过度的极具周期时间效率的方法。在高速度下，当使用TUNE_DH时，将注意到大于编程水平的角区域。因此，通过在角区域中采取快捷方式，使用TUNE_DH会减

下一页继续

续前页

少振动所引起的路径错误，但是会增加高速度下的路径错误。快捷方式将增加已降低的调节值和已增加的速度。如果此类快捷方式不可接受，则建议用AccSet来替代TUNE_DH。

下图显示了已降低的调节值的影响，以及可通过适当的调节值来除去不希望有的振动。对于较小的调节值，角区域中的快捷方式变得显而易见。



xx1400001281

足以通过一个轴的参数\Type:=TUNE_DH，来执行指令TuneServo。相同机械单元中的所有轴将自动获取相同的调节值。

例子：

- 减少高达300 mm/s的TCP速度。50%的调节值可减少不希望有的振动。
  有时将与AccSet相结合，例如，AccSet 50,100;。
- 以高速度进行的材料处理。15%的调节值会减少不希望有的振动。

小心

当机械臂正在运动时，不得改变调节值，且当使用较小的调节值（小于30%）时，应当小心，因为机械臂将在角区域中采用快捷方式。

仅供ABB内部使用-TUNE_DK, TUNE_DL, TUNE_DG, TUNE_DI

警告

仅供ABB内部使用。不得使用此类调节类型。不正确的使用会引起振荡的移动或扭矩，这会对机械臂造成损坏。

下一页继续

调节外轴-TUNE_KP、TUNE_KV、TUNE_TI

此类调节类型会影响有关外轴的位置控制增益（kp）、速度控制增益（kv）和速度控制积分时间（ti）。其用于使外轴适应不同的负载惯量。通过使用此类调节类型，亦可简化外轴的基本调节。

调节机械臂轴-TUNE_KP, TUNE_KV, TUNE_TI

此类参数可用于改变伺服控制器的行为。TUNE_KP影响位置控制器的等效增益，TUNE_KV影响速度控制器的等效增益，且TUNE_TI影响控制器的积分作用。

TUNE_KV调节值的增加会提高机械臂的伺服刚度，并可用于接触应用，因为机器人系统的总体刚度取决于伺服刚度和机械刚度。TUNE_KV调节值的增加亦会减少低速度下的路径错误，并可用于速度低于100 mm/s的切割和焊接应用。典型调节值为150% - 200%。过高的调节值会引起电机振动，且必须予以避免。调整TUNE_KV时，始终小心并注意提高的电机噪声水平，且不得使用高于为满足应用要求所需要的调节值。由于机械共振，过高的调节值亦可增加振动。

关于TUNE_KP的增加的调节值以及关于TUNE_TI的减少的调节值，均会增加伺服刚度，并减少低频率区域中的低速路径错误。TUNE_KP的典型调节值为150% - 300%，TUNE_TI的典型调节值为20% - 50%。在大多数情况下，TUNE_KV为最重要的参数，且TUNE_KP和TUNE_TI无需调整。由于机械共振，关于TUNE_KP的过高调节值或关于TUNE_TI的过低调节值亦可增加振动。

例子：

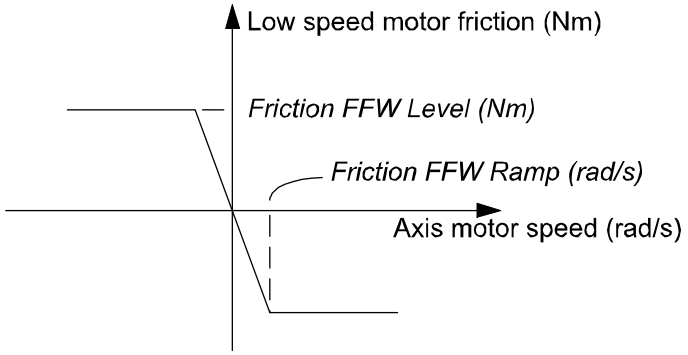
- 去毛刺应用中的机械臂需要更高的伺服刚度，以减少路径错误。TUNE_KV175%、TUNE_KP250%和TUNE_TI30%。

  常常将与AccSet相结合，例如，`AccSet 30,100;`。

摩擦力补偿-TUNE_FRIC_LEV、TUNE_FRIC_RAMP

此类调节类型可用于减少由低速（10 - 200 mm/s）下的摩擦和齿隙所引起的机械臂路径错误。当机械臂轴改变移动方向时，会出现此类路径错误。通过将系统参数Motion/Control Parameters/Friction FFW On设置为Yes，启用轴的摩擦力补偿。

本摩擦力模型为拥有轴速度方向相反符号的常量水平。*Friction FFW Level (Nm)*为（低）速下的绝对摩擦力水平，且大于*Friction FFW Ramp (rad/s)*。参见下图，其显示了摩擦力模型。



xx0500002188

TUNE_FRIC_LEV覆盖系统参数*Friction FFW Level*的值。

针对20 - 100 mm/s速度范围中、可改进机械臂路径精度的各机械臂轴，调节 *Friction FFW Level*（使用TUNE_FRIC_LEV）。针对更大的机械臂（尤其是IRB6400族），当其他跟踪错误源支配此类机械臂时，影响将最小。

下一页继续

1.306 TuneServo - 调节伺服
RobotWare - OS
续前页

TUNE_FRIC_RAMP覆盖系统参数*Friction FFW Ramp*的值。在绝大多数情况下，无需调节*Friction FFW Ramp*。默认设置将适当。

每次调节一个轴。一小步一小步地改变调节值，并寻找可最小化路径上各位置处（该特定轴于此改变移动方向）机械臂路径错误的水平。针对下一个轴等，重复相同的无返回值程序。

可将最终调节值转移到系统参数。例如：

Friction FFW Level = 1。最终调节值（TUNE_FRIC_LEV） = 150%。

设置Friction FFW Level = 1.5，且调节值 = 100%（默认值），两者等效。

变元

```
TuneServo MecUnit Axis TuneValue [\Type]
```

MecUnit

Mechanical Unit

数据类型：mecunit

机械单元名称

Axis

数据类型：num

机械单元当前轴的编号（1-6）。

TuneValue

数据类型：num

调节值以百分比（1-500）计。100%为正常值。

[\Type]

数据类型：tunetype

伺服调节的类型。可利用的类型为TUNE_DF, TUNE_KP, TUNE_KV, TUNE_TI, TUNE_FRIC_LEV, TUNE_FRIC_RAMP, TUNE_DG, TUNE_DH, TUNE_DI。类型TUNE_DK和TUNE_DL仅供ABB内部使用。

当使用调节类型TUNE_DF时，可省略该参数。

基本示例

以下实例介绍了指令TuneServo：

例 1

```
TuneServo MHA160R1, 1, 110 \Type:= TUNE_KP;
```

启用调节类型TUNE_KP，且机械单元MHA160R1中轴1上的调节值为110%。

程序执行

针对指定的轴，启用指定的调节类型和调节值。该值适用于所有移动，直至针对当前轴而编制新值，或通过使用指令TuneReset来重置所有轴的调节类型和值。

通过执行指令TuneReset，自动地设置所有轴的默认伺服调节值

- 重启时。
- 当载入一段新程序时。
- 当从起点开始执行程序时。

下一页继续

限制

上电失败时，始终将任意启用的伺服调节设置为默认值。

在电源故障后重启时，可通过用户程序来处理该限制。

语法

```
TuneServo
  [MecUnit ':=' ] < variable (VAR) of mecunit> ','
  [Axis ':=' ] < expression (IN) of num> ','
  [TuneValue ':=' ] < expression (IN) of num>
  ['\' Type ':=' <expression (IN) of tunetype>] ';'
```

相关信息

信息，关于	请参阅
其他运动设置	技术参考手册 - *RAPID*语言概览
伺服调节的类型	第*1520*页的*tunetype* - 伺服调节类型
重置所有伺服调节	第*825*页的*TuneReset* - 重置伺服调节
MotionProcessModeSet - 设置运动过程模式。	第*331*页的*MotionProcessModeSet* - 设置运动过程模式
外轴的调节	应用手册 - *Additional axes and stand alone controller*
摩擦力补偿	技术参考手册 - 系统参数

1.307 UIMsgBox - 用户消息对话框，基本类型

手册用法

UIMsgBox（*User Interaction Message Box*）用于同有关可用用户设备（例如，FlexPendant示教器）的机器人系统的用户进行沟通。写入消息，以供操作员使用，操作员通过选择按钮来进行回答。随后，将用户选择转移回程序。

基本示例

以下实例介绍了指令UIMsgBox：

另请参阅第*836*页的更多示例

例 1

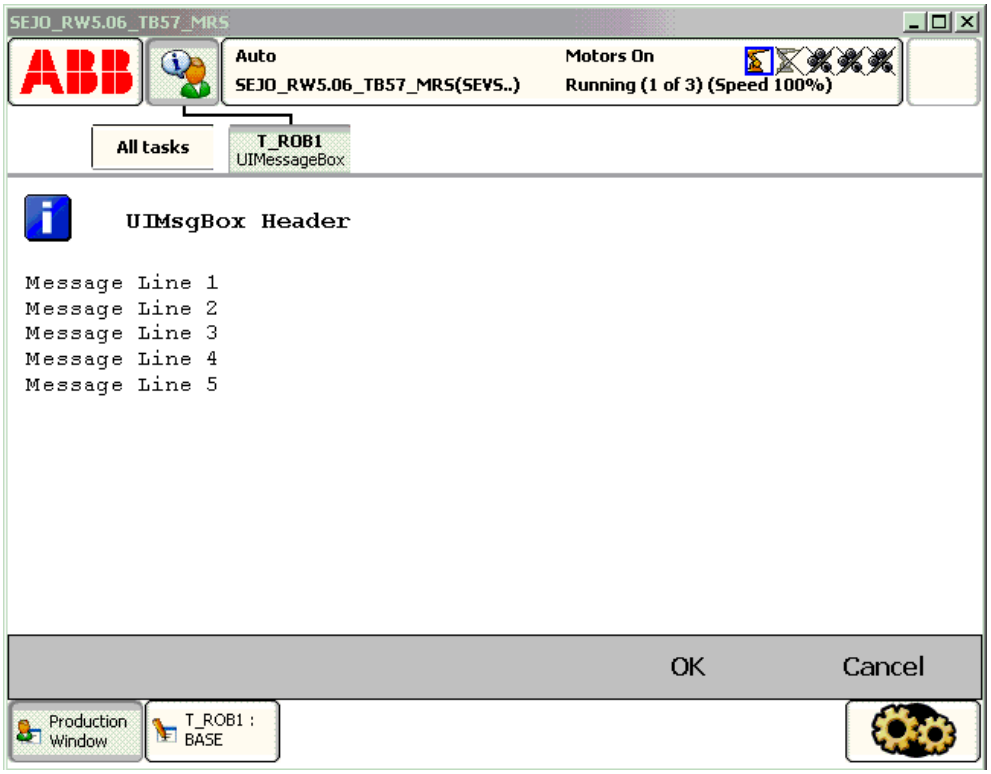
```
UIMsgBox "Continue the program ?";
```

显示消息"Continue the program ?"。当用户按下默认按钮OK时，继续本程序。

例 2

```
VAR btnres answer;
...
UIMsgBox
  \Header:="UIMsgBox Header",
  "Message Line 1"
  \MsgLine2:="Message Line 2"
  \MsgLine3:="Message Line 3"
  \MsgLine4:="Message Line 4"
  \MsgLine5:="Message Line 5"
  \Buttons:=btnOKCancel
  \Icon:=iconInfo
  \Result:=answer;
IF answer = resOK my_proc;
```

下一页继续



xx0500002432

在上文中，消息框以及图标、标题、消息行1到5和按钮得以写入FlexPendant示教器显示器。程序执行进入等待，直至按下确定（OK）或取消（Cancel）。换句话说，根据按下的按钮，将answer 分配为1（OK）或5（Cancel）。若回答为OK，则将调用my_proc。

注意，消息行1...消息行5在单独的行1到行5显示（未使用开关\Wrap）。

变元

UIMsgBox [\Header] MsgLine1 [\MsgLine2] [\MsgLine3] [\MsgLine4] [\MsgLine5] [\Wrap] [\Buttons] [\Icon] [\Image] [\Result] [\MaxTime] [\DIBreak] [\DIPassive] [\DOBreak] [\DOPassive] [\BreakFlag]

`[\Header]`

数据类型：string

将在消息框顶部写入的标题文本。最多40个字符。

`MsgLine1`

Message Line 1

数据类型：string

有待在显示器上写入的文本行1。最多55个字符。

`[\MsgLine2]`

Message Line 2

数据类型：string

有待在显示器上写入的附加文本行2。最多55个字符。

下一页继续

1.307 UIMsgBox - 用户消息对话框，基本类型
RobotWare - OS
续前页

[\MsgLine3]

Message Line 3

数据类型：string

有待在显示器上写入的附加文本行3。最多55个字符。

[\MsgLine4]

Message Line 4

数据类型：string

有待在显示器上写入的附加文本行4。最多55个字符。

[\MsgLine5]

Message Line 5

数据类型：string

有待在显示器上写入的附加文本行5。最多55个字符。

[\Wrap]

数据类型：switch

如果选择，则所有字符串MsgLine1 ... MsgLine5均将连接到一个字符串，各单独字符串之间仅存在单间距，并以尽可能少的行展开。

默认将在显示器上单独的行中显示各消息字符串*MsgLine1 ... MsgLine5*。

[\Buttons]

数据类型：buttondata

定义有待显示的按钮。仅可使用一种buttondata型预定义按钮组合。参见第836页的预定义数据。

系统默认显示OK按钮（\Buttons:=btn OK）。

[\Icon]

数据类型：icondata

定义有待显示的图标。仅可使用一种icondata型预定义图标。参见第836页的预定义数据。

默认没有图标。

[\Image]

数据类型：string

应当使用的图像的名称。为发射自己的图像，必须将图像置于有效系统中的HOME:路径中，或直接置于有效系统中。

建议将文件置于HOME:路径中，以便在完成备份和储存时将其保存。

需要重启，随后，FlexPendant示教器将加载图像。

对系统的要求是使用RobotWare附加功能*FlexPendant Interface*。

将显示的图像可拥有185个像素点的宽度和300个像素点的高度。如果图像较大，则将仅显示该图像左上方185*300像素点的图像。

无法指明关于图像尺寸或能够加载至FlexPendant示教器的图像数量的确切值。其取决于加载至FlexPendant示教器的其他文件的尺寸。如果使用的图像尚未加载至FlexPendant示教器，则将继续程序执行。

下一页继续

[\Result]

数据类型：btnres

关于返回数值0..7的变量，取决于按下哪一个按钮。仅可使用一个btnres型预定义常量，以测试用户的选择。参见第836页的预定义数据。

如果任意类型的系统中断，例如，\MaxTime、\DIBreak或\DOBreak，或者如果\Buttons:=btnNone，resUnkwn等于返回0。

[\MaxTime]

数据类型：num

程序执行等待的最长时间，以秒计。如果在该时间内未选择任何按钮，则用错误处理器继续执行程序，除非使用BreakFlag（参见下文）。常量ERR_TP_MAXTIME可用于测试是否已经过最长时间。

[\DIBreak]

Digital Input Break

数据类型：signaldi

可能中断运算符对话框的数字信号输入信号。如果将信号设置为1（或已经为1）时未选择任何按钮，则用错误处理器继续执行程序，除非使用BreakFlag（参见下文）。可使用常量ERR_TP_DIBREAK来测试是否已出现该情况。

[\DIPassive]

Digital Input Passive

数据类型：switch

当使用DIBreak可选参数时，该开关会覆盖默认行为。将信号设置为1（或已经为1）时，取代反应；将信号DIBreak设置为0（或已经为0）时，应当用错误处理器继续本指令（如果未使用BreakFlag）。常量ERR_TP_DIBREAK可用于测试是否已出现上述情况。

[\DOBreak]

Digital Output Break

数据类型：signaldo

可能中断运算符对话框的数字信号输出信号。如果将信号设置为1（或已经为1）时未选择任何按钮，则用错误处理器继续执行程序，除非使用BreakFlag（参见下文）。可使用常量ERR_TP_DOBREAK来测试是否已出现该情况。

[\DOPassive]

Digital Output Passive

数据类型：switch

当使用DOBreak可选参数时，该开关会覆盖默认行为。将信号设置为1（或已经为1）时，取代反应；将信号 DOBreak设置为0（或已经为0）时，应当用错误处理器继续本指令（如果未使用BreakFlag）。常量ERR_TP_DOBREAK可用于测试是否已出现上述情况。

[\BreakFlag]

数据类型：errnum

下一页继续

1.307 UIMsgBox - 用户消息对话框，基本类型
RobotWare - OS
续前页

使用 \MaxTime、\DIBreak或 \DOBreak时将保存错误代码的变量（使用前，由系统将其设置为0）。常量ERR_TP_MAXTIME、ERR_TP_DIBREAK和ERR_TP_DOBREAK可用于选择原因。如果省略该可选变量，则将执行错误处理器。

程序执行

根据编程参数，显示消息框以及图标、标题、消息行、图像和按钮。程序执行进入等待，直至用户选择一个按钮，或消息框被超时或信号行动打断。将用户选择和中断原因转移回程序。

从基础等级的消息框转而关注软中断等级的新消息框。

预定义数据

```
!Icons:
  CONST icondata iconNone := 0;
  CONST icondata iconInfo := 1;
  CONST icondata iconWarning := 2;
  CONST icondata iconError := 3;

!Buttons:
  CONST buttondata btnNone := -1;
  CONST buttondata btnOK := 0;
  CONST buttondata btnAbrtRtryIgn := 1;
  CONST buttondata btnOKCancel := 2;
  CONST buttondata btnRetryCancel := 3;
  CONST buttondata btnYesNo := 4;
  CONST buttondata btnYesNoCancel := 5;

!Results:
  CONST btnres resUnkwn := 0;
  CONST btnres resOK := 1;
  CONST btnres resAbort := 2;
  CONST btnres resRetry := 3;
  CONST btnres resIgnore := 4;
  CONST btnres resCancel := 5;
  CONST btnres resYes := 6;
  CONST btnres resNo := 7;
```

更多示例

有关于如何使用指令UIMsgBox的更多例子阐述如下。

例 1

```
VAR errnum err_var;
...
UIMsgBox \Header:= "Example 1", "Waiting for a break condition..."
     \Buttons:=btnNone \Icon:=iconInfo \MaxTime:=60 \DIBreak:=di5
     \BreakFlag:=err_var;

TEST err_var
CASE ERR_TP_MAXTIME:
  ! Time out break, max time 60 seconds has elapsed
```

下一页继续

```
CASE ERR_TP_DIBREAK:
  ! Input signal break, signal di5 has been set to 1
DEFAULT:
  ! Not such case defined
ENDTEST
```

显示消息框，直至中断条件成真。因为针对参数\Buttons而设置btnNone ，因此，操作员无法回答或删除消息框。当将di5设置为1或超时（60秒后）时，消息框得以删除。

例 2

```
VAR errnum err_var;
...
UIMsgBox \Header:= "Example 2", "Waiting for a break condition..."
     \Buttons:=btnNone \Icon:=iconInfo \MaxTime:=60 \DIBreak:=di5
     \DIPassive \BreakFlag:=err_var;

TEST err_var
CASE ERR_TP_MAXTIME:
  ! Time out break, max time 60 seconds has elapsed
CASE ERR_TP_DIBREAK:
  ! Input signal break, signal di5 has been set to 0
DEFAULT:
  ! Not such case defined
ENDTEST
```

显示消息框，直至中断条件成真。因为针对参数\Buttons而设置btnNone ，因此，操作员无法回答或删除消息框。当将di5设置为0或超时（60秒后）时，消息框得以删除。

错误处理

如果未使用参数\BreakFlag，则可通过错误处理器来处理此类情形：

- 如果在运算符输入之前出现超时（参数\MaxTime），则将系统变量ERRNO设置为ERR_TP_MAXTIME，并用错误处理器继续执行。
- 如果在运算符输入之前设置数字信号输入（参数\DIBreak），则将系统变量ERRNO设置为ERR_TP_DIBREAK，并用错误处理器继续执行。
- 如果在运算符输入之前设置数字信号输出（参数\DOBreak），则将系统变量ERRNO设置为ERR_TP_DOBREAK，并用错误处理器继续执行。

仅可通过错误处理器来处理该情形：

- 如果信号变量是在RAPID中声明的变量，且尚未与通过指令AliasIO而在I/O配置中确定的I/O信号相连，则将系统变量ERRNO设置为ERR_NO_ALIASIO_DEF，并用错误处理器继续执行。
- 如果没有客户端（例如，FlexPendant示教器）来关注指令，则将系统变量ERRNO设置为ERR_TP_NO_CLIENT，并用错误处理器继续执行。

限制

比如在回路中频繁执行UIMsgBox时，避免为超时参数 \MaxTime采用过小的值。这会造成系统性能出现无法预测的行为，比如，FlexPendant示教器响应缓慢。

下一页继续

1.307 UIMsgBox - 用户消息对话框，基本类型

RobotWare - OS

续前页

语法

```
UIMsgBox
  [´\´Header´:=´ <expression (IN) of string>´,´]
  [MsgLine1´:=´] <expression (IN) of string>
  [´\´MsgLine2´:=´<expression (IN) of string>]
  [´\´MsgLine3´:=´<expression (IN) of string>]
  [´\´MsgLine4´:=´<expression (IN) of string>]
  [´\´MsgLine5´:=´<expression (IN) of string>]
  [´\´Wrap]
  [´\´Buttons´:=´ <expression (IN) of buttondata>]
  [´\´Icon´:=´ <expression (IN) of icondata>]
  [‘\’Image´:=´<expression (IN) of string>]
  [´\´Result´:=´< var or pers (INOUT) of btnres>]
  [´\´MaxTime´:=´ <expression (IN) of num>]
  [´\´DIBreak´:=´ <variable (VAR) of signaldi>]
  [´\´DIPassive]
  [´\´DOBreak´:=´ <variable (VAR) of signaldo>]
  [´\´DOPassive]
  [´\´BreakFlag´:=´ <var or pers (INOUT) of errnum>]´;´
```

相关信息

信息，关于	请参阅
图标显示数据	第*1411*页的*icondata - 图标显示数据*
按钮数据	第*1349*页的*buttondata - 按钮数据*
按钮结果数据	第*1346*页的*btnres - 按钮结果数据*
先进型用户交互消息框	第*1317*页的*UIMessageBox - 先进型用户消息框*
用户交互数字条目	第*1324*页的*UINumEntry - 用户数字条目*
用户交互数字调节	第*1330*页的*UINumTune - 用户数字调节*
用户交互α条目	第*1292*页的*UIAlphaEntry - 用户α条目*
用户交互列表视图	第*1310*页的*UIListView - 用户列表视图*
与FlexPendant示教器等相连的系统。	第*1297*页的*UIClientExist - 现有客户端*
FlexPendant示教器界面	产品规格 - 控制器软件*IRC5*
清除运算符窗口	第*732*页的*TPErase - 擦除在FlexPendant示教器上印刷的文本*

1.308 UIShow - 用户界面显示

手册用法

UIShow（*User Interface Show*）用于同有效用户设备（例如，FlexPendant示教器）上的机器人系统的用户进行沟通。通过UIShow，可以从RAPID程序开始单独写入应用和标准应用。

基本示例

以下实例介绍了指令UIShow：

如果文件TpsViewMyAppl.dll 和TpsViewMyAppl.gtpu.dll 存在于HOME: 路径，且已经实施了重启，则仅例1和例2起作用。

例 1

```
CONST string Name:="TpsViewMyAppl.gtpu.dll";
CONST string Type:="ABB.Robotics.SDK.Views.TpsViewMyAppl";
CONST string Cmd1:="Init data string passed to the view";
CONST string Cmd2:="New init data string passed to the view";
PERS uishownum myinstance:=0;
VAR num mystatus:=0;
...
! Launch one view of my application MyAppl
UIShow Name, Type \InitCmd:=Cmd1 \InstanceID:=myinstance
     \Status:=mystatus;
! Update the view with new init command
UIShow Name, Type \InitCmd:=Cmd2 \InstanceID:=myinstance
     \Status:=mystatus;
```

上述代码将启动视图TpsViewMyAppl以及初始化命令Cmd1，随后，更新关于Cmd2的视图。

例 2

```
CONST string Name:="TpsViewMyAppl.gtpu.dll";
CONST string Type:="ABB.Robotics.SDK.Views.TpsViewMyAppl";
CONST string Cmd1:="Init data string passed to the view";
CONST string Cmd2:="New init data string passed to the view";
PERS uishownum myinstance:=0;
VAR num mystatus:=0;
...
! Launch one view of my application MyAppl
UIShow Name, Type \InitCmd:=Cmd1 \Status:=mystatus;
! Launch another view of the application MyAppl
UIShow Name, Type \InitCmd:=Cmd2 \InstanceID:=myinstance
     \Status:=mystatus;
```

上述代码将启动视图TpsViewMyAppl以及初始化命令Cmd1。随后，其将启动另一个有关初始化命令Cmd2的视图。

例 3

```
CONST string Name:="tpsviewbackupandrestore.dll";
CONST string Type:="ABB.Robotics.Tps.Views.TpsViewBackupAndRestore";
VAR num mystatus:=0;
...
```

下一页继续

1.308 UIShow - 用户界面显示
续前页

```
UIShow Name, Type \Status:=mystatus;
```

启动标准应用备份和储存。

例 4

```
CONST string Name:="TpsViewPanel.gtpu.dll";
CONST string Type:="ABB.Robotics.SDK.Views.MainScreen";
PERS uishownum myinstance:=0;
VAR num mystatus:=0;
...
UIShow Name, Type \InstanceID:=myinstance \Status:=mystatus;
```

启动通过ScreenMaker创建的应用。

变元

```
UIShow AssemblyName TypeName [\InitCmd] [\InstanceId] [\Status]
     [\NoCloseBtn]
```

AssemblyName

数据类型：string

包含视图的数据集的名称。

TypeName

数据类型：string

此为视图（待创建类型）的名称。此为本类型的全限定名，即包括其命名空间。

[\InitCmd]

Init Command

数据类型：string

传递给本视图的初始化数据字符串。

[\InstanceId]

数据类型：uishownum

代表用于确定视图的标记的参数。如果在调用 UIShow后显示视图，则传回用以确定视图的值。随后，该标记可用于其他UIShow调用，以启用已经在运行的视图。如果该值可确定现有（运行）视图，则将启用该视图。如果该值不存在，则将创建新实例。这意味着可使用该参数，以确定是否将启用新实例。如果其值可确定已经开始的视图，则将启用该视图，无论其他参数的值如何。建议针对将通过UIShow指令来启用的各新应用，使用唯一的InstanceId变量。

参数必须为永久变量，原因在于该变量应当维持其值，即使程序指针移动至main。如果执行与先前相同的UIShow，并使用相同的变量，则如果其仍然公开，则将启用相同的视图。如果已关闭该视图，则将启用新视图。

[\Status]

数据类型：num

Status表明操作是否成功。注意，如果使用该选项，则RAPID执行将进入等待，直至完成本指令，即启用本视图。

该可选参数主要用于调试目的（参见错误处理）。

Status	描述
0	确定

下一页继续

Status	描述
-1	FlexPendant示教器没有空间供新视图使用。在FlexPendant示教器，最多可同时打开6个视图。
-2	无法发现数据集，其并不存在
-3	发现文件，但是无法加载
-4	存在数据集，但是未能创建新实例
-5	typename对于该数据集无效
-6	InstanceID不匹配用于加载的数据集

[\NoCloseBtn]

No Close Button

数据类型：switch

NoCloseBtn禁用视图的关闭按钮。

程序执行

指令UIShow用于启用FlexPendant示教器上的单独应用。未启用单独的应用，必须将数据集置于有效系统中的HOME:路径中，或直接置于有效系统中，或置于附加选项中。建议将文件置于HOME:路径中，以便在完成备用和储存时进行保存。需要重启，随后，FlexPendant示教器加载新的数据集。对系统的要求是使用RobotWare附加功能*FlexPendant Interface*。

同时有可能启用标准应用，例如，备用和储存。随后，无需具有RobotWare附加功能*FlexPendant Interface*。

如果使用参数\Status，则程序执行将等待，直至开始本应用。如果未处理应用中的错误，则其仅为受监督启用的结果。在没有\Status参数的情况下，命令FlexPendant示教器开始本应用，但是并无检查，以确定是否可能开始。

错误处理

如果没有客户端（例如，FlexPendant示教器）来关注指令，则将系统变量ERRNO设置为ERR_TP_NO_CLIENT，并用错误处理器继续执行。

如果使用参数\Status，则可通过错误处理器来处理此类情形：

- 如果FlexPendant示教器上无剩余空间供数据集使用，则将系统变量ERRNO设置为ERR_UISHOW_FULL，并通过错误处理器继续执行。FlexPendant示教器可同时打开6个视图。
- 如果当视图启用视图时，其他方面出错，则将系统变量ERRNO设置为ERR_UISHOW_FATAL，并通过错误处理器继续执行。

限制

当使用UIShow指令来开始单独应用时，要求系统配备选项*FlexPendant Interface*。

已通过UIShow指令开始的应用，未能通过上电失败情形。POWER ON事件程序可用于再次设置应用。

语法

```
UIShow
  [AssemblyName ´:=´] < expression (IN) of string >’,’
  [TypeName ´:=´] < expression (IN) of string >’,’
```

下一页继续

1.308 UIShow - 用户界面显示

续前页

```
['\'InitCmd' :=' < expression (IN) of string> ]
['\'InstanceId ':=' < persistent (PERS) of uishownum> ]
['\'Status ':=' < variable (VAR) of num> ]
['\'NoCloseBtn ]';'
```

相关信息

信息，关于	请参阅
FlexPendant interface	产品规格 - 控制器软件*IRC5*
制定有关FlexPendant示教器的单独应用	http://developercenter.robotstudio.com/
`uishownum`	第*1521*页的*uishownum - UIShow*的实例标识符
清除运算符窗口	第*732*页的*TPErase - 擦除在FlexPendant*示教器上印刷的文本

1.309 UnLoad - 执行期间，卸载普通程序模块

手册用法

UnLoad用于在执行期间，从程序内存卸载普通程序模块。

必须通过使用指令Load或StartLoad - WaitLoad，于先前将普通程序模块加载到程序内存中。

基本示例

以下实例介绍了指令UnLoad：

同时参见下文更多例子。

例 1

```
UnLoad diskhome \File:="PART_A.MOD";
```

先前通过Load加载至程序内存中且来源于程序内存的UnLoad普通程序模块PART_A.MOD（参见指令Load）。diskhome为预定义字符串常量“HOME:”。

变元

```
UnLoad [\ErrIfChanged] | [\Save] FilePath [\File]
```

[\ErrIfChanged]

数据类型：switch

如果使用该参数，且模块自加载到系统中起便发生改变，则本指令将生成错误恢复代码ERR_NOTSAVED。

[\Save]

数据类型：switch

如果使用该参数，则在卸载开始前，保存普通程序模块。本普通程序模块将保存在Load或StartLoad指令中所指定的原始位置。

FilePath

数据类型：string

将从程序内存卸载的文件的路径和名称。文件路径和文件名称必须与先前执行的Load或StartLoad指令中的路径和名称相同。当使用参数\File 时，应当排除文件名称。

[\File]

数据类型：string

当在参数FilePath中排除文件名时，必须通过该参数对文件名进行定义。文件名必须与先前执行的Load或StartLoad 指令中的文件名相同。

程序执行

为了能够在程序中执行UnLoad指令，必须于早些时候在程序中执行具有相同文件路径和名称的Load或StartLoad - WaitLoad指令。

在通过下一指令继续执行前，程序执行等待普通程序模块完成卸载。

此后，卸载普通程序模块，并将连接剩余普通程序模块。

有关更多的信息，请参见指令Load或StartLoad-Waitload。

下一页继续

1.309 UnLoad - 执行期间，卸载普通程序模块

RobotWare - OS

续前页

更多示例

有关于如何使用指令UnLoad的更多例子阐述如下。

例 1

```
UnLoad "HOME:/DOORDIR/DOOR1.MOD";
```

先前加载到程序内存中，且来源于程序内存的UnLoad普通程序模块DOOR1.MOD。

例 2

```
UnLoad "HOME:" \File:="DOORDIR/DOOR1.MOD";
```

与上文例1相同，但是采用另一种语法。

例 3

```
Unload \Save, "HOME:" \File:="DOORDIR/DOOR1.MOD";
```

与上文例1和例2相同，但是在卸载前保存普通程序模块。

限制

不允许卸载正在执行中的普通程序模块（模块中的程序指针）。

卸载期间，无法执行软中断程序、系统I/O事件以及其他程序任务。

卸载期间，避免机械臂不间断地移动。

执行UnLoad指令期间，程序停止，会导致保护停止，且FlexPendant示教器上出现电机关闭和错误消息“20025停止指令超时”。

错误处理

如果由于在模块或错误路径内不间断地执行（不通过Load或StartLoad加载的模块），无法卸载UnLoad指令中的文件，则将系统变量ERRNO设置为ERR_UNLOAD。

如果使用参数\ErrIfChanged，且已改变模块，则该程序的执行将系统变量ERRNO设置为ERR_NOTSAVED。

随后，可用错误处理器来处理此类错误。

语法

```
UnLoad
  ['\'ErrIfChanged ',] | ['\'Save ',]
  [FilePath':=']<expression (IN) of string>
  ['\'File':=' <expression (IN) of string>]';'
```

相关信息

信息，关于	请参阅
检查程序参考	第97页的*CheckProgRef - 检查程序参考*
加载普通程序模块	第312页的*Load - 执行期间，加载普通程序模块* 第660页的*StartLoad - 执行期间，加载普通程序模块* 第881页的*WaitLoad - 将加载的模块与任务相连*

1.310 UnpackRawBytes - 打开来自原始数据字节数据的数据

手册用法

UnpackRawBytes用于将rawbytes型容器的内容解包至byte、num、dnum或string型变量。

基本示例

以下实例介绍了指令UnpackRawBytes：

例 1

```
VAR iodev io_device;
VAR rawbytes raw_data_out;
VAR rawbytes raw_data_in;
VAR num integer;
VAR dnum bigInt;
VAR num float;
VAR string string1;
VAR byte byte1;
VAR byte data1;

! Data packed in raw_data_out according to the protocol
...
Open "chan1:", io_device\Bin;
WriteRawBytes io_device, raw_data_out;
ReadRawBytes io_device, raw_data_in\Time := 1;
Close io_device;
```

根据程序员知晓的协议，将消息发送至设备"chan1:"。随后，从设备读取答案。答案包含，例如，以下内容：

字节数：	内容：
1-4	整数‘5’
5-8	浮点数‘234.6’
9-25	字符串“这真正有趣（This is real fun）！”
26	十六进制值‘4D’
27	ASCII代码122，即‘z’
28-36	整数‘4294967295’
37-40	整数‘4294967295’

```
UnpackRawBytes raw_data_in, 1, integer \IntX := DINT;
```

integer的内容将为5。

```
UnpackRawBytes raw_data_in, 5, float \Float4;
```

float的内容将为234.6小数。

```
UnpackRawBytes raw_data_in, 9, string1 \ASCII:=17;
```

string1的内容将为"This is real fun!"。

```
UnpackRawBytes raw_data_in, 26, byte1 \Hex1;
```

byte1的内容将为'4D'十六进制。

```
UnpackRawBytes raw_data_in, 27, data1 \ASCII:=1;
```

下一页继续

1.310 UnpackRawBytes - 打开来自原始数据字节数据的数据

RobotWare - OS

续前页

data1的内容将为122，有关“z”的ASCII代码。

```
UnpackRawBytes raw_data_in, 28, bigInt \IntX := LINT;
```

bigInt的内容将为4294967295。

```
UnpackRawBytes raw_data_in, 37, bigInt \IntX := UDINT;
```

bigInt的内容将为4294967295。

变元

```
UnpackRawBytes RawData [ \Network ] StartIndex Value [\Hex1 ]| [
    \IntX ] | [ \Float4 ] | [ \ASCII ]
```

RawData

数据类型：rawbytes

用于解包数据源的变量容器。

[\Network]

数据类型：switch

表明应当根据RawData中代表的大端法（网络顺序），解包整数和浮点数。ProfiBus和InterBus使用大端法。

在没有该开关的情况下，将通过来自RawData的小端法（非网络顺序）代表，解包整数和浮点数。DeviceNet使用小端法。

仅当连同选项参数 \IntX - UINT、UDINT、ULINT、INT、DINT、LINT和\Float4时相关。

StartIndex

数据类型：num

StartIndex介于1到1024之间，表明开始从RawData解包数据的位置。

Value

数据类型：anytype

包含从RawData解包的数据的变量。

容许的数据类型包括：byte、num、dnum或string。无法使用数组。

[\Hex1]

数据类型：switch

待解包并置于Value中的数据在1byte中具有十六进制格式，并将转换为byte变量中的小数格式。

[\IntX]

数据类型：inttypes

待解包数据拥有符合数据类型inttypes指定常量的格式。数据将转换成包含整数，并储存在Value中的num或dnum变量。

请参阅 第*847*页的预定义数据。

[\Float4]

数据类型：switch

待解包并置于值中的数据具有浮点4字节格式，并将转换为包含浮点的num变量。

下一页继续

[\ASCII]

数据类型：num

待解包并置于Value中的数据具有byte或string格式。

如果Value为byte型，则将数据解释为ASCII代码，并转换为byte格式（1个字符）。

如果Value为string型，则将数据储存为string（1...80字符）。字符串数据非空，并以rawbytes型数据终止。

必须编程参数\Hex1、\IntX、\Float4或\ASCII之一。

允许以下组合：

数值的数据类型：	容许的选项参数：
num *)	\IntX
dnum **)	\IntX
num	\Float4
string	\ASCII:=n with n between 1 and 80
byte	\Hex1 \ASCII:=1

*)必须为选定符号常量USINT、UINT、UDINT、SINT、INT或DINT数值范围内的整数。

**)必须为选定符号常量USINT、UINT、UDINT、ULINT、SINT、INT、DINT或LINT数值范围内的整数。

程序执行

程序执行期间，将数据从rawbytes型容器解包成anytype型变量。

上电失败重启时，将关闭系统中所有打开的文件或串行通道，并将重置iodev型变量中的I/O描述符。

预定义数据

预定义数据类型inttypes的以下符号常量，并可将其用于确定参数\IntX中的整数。

符号常量	常量值	整数格式	整数值域
USINT	1	无符号1字节整数	0 ... 255
UINT	2	无符号2字节整数	0 ... 65 535
UDINT	4	无符号4字节整数	0 ... 8 388 608 *) 0 ... 4 294 967 295 ****)
ULINT	8	无符号8字节整数	0 ... 4 503 599 627 370 496**)
SINT	- 1	有符号1字节整数	- 128... 127
INT	- 2	有符号2字节整数	- 32 768 ... 32 767
DINT	- 4	有符号4字节整数	- 8 388 607 ... 8 388 608 *) -2 147 483 648 ... 2 147 483 647 ***)
LINT	- 8	有符号8字节整数	- 4 503 599 627 370 496... 4 503 599 627 370 496 **)

*) 关于储存数据类型num中整数的RAPID限制。

**) 关于储存数据类型dnum中整数的RAPID限制。

下一页继续

1.310 UnpackRawBytes - 打开来自原始数据字节数据的数据

RobotWare - OS

续前页

***)使用双数值变量和inttypeDINT时的范围。

****)使用双数值变量和inttypeUDINT时的范围。

语法

```
UnpackRawBytes
  [RawData ´:=´ ] < variable (VAR) of rawbytes>
  [ ’\’ Network ] ´,´
  [StartIndex ´:=´ ] < expression (IN) of num> ´,´
  [Value ’:=’ ] < variable (VAR) of anytype>
  [ ’\’ Hex1 ]
  | [ ’\’ IntX’ :=’ < expression (IN) of inttypes>]
  | [’ \’ Float4 ]
  | [ ’\’ ASCII‘ :=’ < expression (IN) of num>] ’;’
```

相关信息

信息，关于	请参阅
rawbytes数据	第*1455*页的原始数据字节 - 原始数据
获取rawbytes数据的长度	第*1193*页的*RawBytesLen* - 获取原始数据字节数据的长度
清除rawbytes数据的内容	第*110*页的*ClearRawBytes* - 清除原始数据字节数据的内容
复制rawbytes数据的内容	第*128*页的*CopyRawBytes* - 复制原始数据字节数据的内容
将DeviceNet标题装入rawbytes数据	第*421*页的*PackDNHeader* - 将*DeviceNet*标题装入原始数据字节数据中。
将数据装入rawbytes数据	第*424*页的*PackRawBytes* - 将数据装入原始数据字节数据
写入rawbytes数据	第*923*页的*WriteRawBytes* - 写入原始数据字节数据
读取rawbytes数据	第*499*页的*ReadRawBytes* - 读取原始数据字节数据
打开来自rawbytes数据的数据	第*845*页的*UnpackRawBytes* - 打开来自原始数据字节数据的数据
位/字节功能	技术参考手册 - *RAPID*语言概览
字符串功能	技术参考手册 - *RAPID*语言概览
文件和串行通道处理	应用手册 - 控制器软件*IRC5*

1.311 VelSet - 改变编程速率

手册用法

VelSet用于增加或减少所有后续定位指令的编程速率。该执行同时用于使速率最大化。

本指令仅可用于主任务T_ROB1，或者如果在*MultiMove*系统中，则可用于运动任务中。

基本示例

以下实例介绍了指令VelSet：

另请参阅第*850*页的更多示例

例 1

```
VelSet 50, 800;
```

将所有的编程速率降至指令中值的50%。不允许TCP速率超过800 mm/s。

变元

```
VelSet Override Max
```

Override

数据类型：num

所需速率占编程速率的百分比。100%相当于编程速率。

Max

数据类型：num

最大TCP速率，以mm/s计。

程序执行

所有后续定位指令的编程速率受到影响，直至执行新的VelSet指令。

参数Override影响：

- speeddata中的所有速率分量（TCP、方位、旋转和线性外轴）。
- 通过定位指令进行的编程速度覆盖（参数\V）。
- 定时的移动。

参数Override不影响：

- welddata中的焊接速度。
- seamdata中的加热和填充速度。

参数Max仅影响TCP的速率。

Override和Max的默认值分别为100%和vmax.v_tcp mm/s *)。此类值自动设置

- 当使用重启模式重置RAPID时
- 当加载一则新程序或一个新模块时
- 当从起点开始执行程序时
- 当将程序指针移动到main时
- 当将程序指针移动到子程序时
- 移动程序指针造成执行顺序丢失时

下一页继续

1.311 VelSet - 改变编程速率
RobotWare - OS
续前页

*)关于已使用机械臂类型的TCP速度以及正常实际TCP值。RAPID函数MaxRobSpeed返回相同的值。

更多示例

有关于如何使用指令VelSet的更多例子阐述如下。

例 1

```
VelSet 50, 800;
MoveL p1, v1000, z10, tool1;
MoveL p2, v2000, z10, tool1;
MoveL p3, v1000\T:=5, z10, tool1;
```

点p1的速度为500mm/s，点p2的速度为800mm/s。从p2移动至p3需耗时10秒。

限制

当在定位指令中规定了时间时，将不考虑最大速度。

语法

```
VelSet
  [ Override ´:=´ ] < expression (IN) of num > ´,´
  [ Max ´:=´ ] < expression (IN) of num > ´;´
```

相关信息

信息，关于	请参阅
速度的定义	第*1480*页的*speeddata - 速度数据*
有关该机械臂的最大TCP速度	第*1140*页的*MaxRobSpeed - 最大机械臂速度*
定位器指令	技术参考手册 - *RAPID*语言概览
运动设置数据	第*1430*页的*motsetdata - 运动设置数据*

1.312 WaitAI - 等待直至已设置模拟信号输入信号值

手册用法

WaitAI（*Wait Analog Input*）用于等待，直至已设置模拟信号输入信号值。

基本示例

以下实例介绍了指令WaitAI：

例 1

```
WaitAI ai1, \GT, 5;
```

仅在ai1模拟信号输入具有大于5的值之后，方可继续程序执行。

例 2

```
WaitAI ai1, \LT, 5;
```

仅在ai1模拟信号输入具有小于5的值之后，方可继续程序执行。

变元

```
WaitAI Signal [\LT] | [\GT] Value [\MaxTime] [\ValueAtTimeout]
     [\Visualize] [\Header] [\Message] | [\MsgArray] [\Wrap]
     [\Icon] [\Image] [\VisualizeTime]
```

Signal

数据类型：signalai

模拟信号输入信号的名称。

[\LT]

Less Than

数据类型：switch

如果使用该参数，则WaitAI指令等待，直至模拟信号值小于Value中的值。

[\GT]

Greater Than

数据类型：switch

如果使用该参数，则WaitAI指令等待，直至模拟信号值大于Value中的值。

Value

数据类型：num

信号的期望值。

[\MaxTime]

Maximum Time

数据类型：num

允许的最长等待时间，以秒计。如果在满足条件之前耗尽该时间，则将调用错误处理器，如果存在这样的情况，则采用错误代码ERR_WAIT_MAXTIME。如果不存在错误处理器，则将停止执行。

[\ValueAtTimeout]

数据类型：num

下一页继续

1.312 WaitAI - 等待直至已设置模拟信号输入信号值
RobotWare - OS
续前页

如果指令超时，则将当前信号值储存在该变量中。仅将系统变量ERRNO设置为ERR_WAIT_MAXTIME时，方才设置本变量。

[\Visualize]

数据类型：switch

如被选中，将激活可视化。可视化包括未满足的条件的消息框、图标、页眉、消息行，将按编程参数来显示图像。

[\Header]

数据类型：string

要写在消息框顶部的页眉文字。最大40个字符。如果没有使用\Header 参数，则将显示默认消息。

[\Message]

数据类型：string

在显示器上可写入的一行文字。最多50字符。

[\MsgArray]

(Message Array)

数据类型：string

显示器上需编写的来自数组的若干文本行。同时只可采用参数\Message或\MsgArray中的其中一项。

最大布局空间是5 行，每行50字符。

[\Wrap]

数据类型：switch

如果选择，则参数\MsgArray中的所有指定字符串均将连接到一个字符串，各单独字符串之间仅存在单间距，并以尽可能少的行展开。

参数\MsgArray的各字符串将默认显示在显示器的单独各行。

[\Icon]

数据类型：icondata

定义有待显示的图标。仅可使用一种icondata型预定义图标。参见第*1411*页的预定义数据。

默认没有图标。

[\Image]

数据类型：string

应采用的图像的名称。为了开启您自己的图像，必须将图像放置在活跃系统的*HOME:*目录或直接放置在活跃系统中。

建议将文件放置在*HOME:*目录，从而在完成备份和恢复的情况下，可保存文件。

要求重启，然后，FlexPendant将加载图像。

系统命令是采用RobotWare附加功能*FlexPendant Interface*。

显示的图像可能为185像素宽度和300像素高度。如果图像更大，那么，只会从图像左上侧开始显示图像的185 * 300像素部分。

下一页继续

在图像可为多大尺寸或FlexPendant可加载多少图像方面，没有确切值。这取决于加载到FlexPendant的其他文件的大小。仅仅在采用未加载到FlexPendant的图像时，程序执行才会继续。

[\VisualizeTime]

数据类型：num

FlexPendant上应出现消息框前的等待时间。如果采用参数\VisualizeTime和\MaxTime，那么，参数\MaxTime采用的时间需大于参数\VisualizeTime采用的时间。

不采用参数\VisualizeTime情况下，可视化的默认时间为5s。最小值为1s。最大值没有限制。分辨率为0.001s。

程序执行

当执行本指令时，如果信号值正确，则本程序仅仅继续以下指令。

如果信号值不正确，则机械臂进入等待状态，且程序在信号改变为正确值时继续。通过中断来探测改变，其作出快速响应（非查询）。

当机械臂正在等待时，对时间进行监督。机械臂默认可永久地等待，但是，可通过可选参数\MaxTime来规定最长等待时间。如果超出该最长时间，则会引起错误。

如果停止程序执行，并随后重启，则本指令评估信号的当前值。否定程序停止期间的任意改变。

在手动模式下，在等待3 s以上时间后，警告框将弹出，询问是否想要模拟指令。如果不想警告框出现，那么您可将系统参数*SimulateMenu*设为NO，请参见技术参考手册 - 系统参数主题*Controller*类型*General RAPID*。

如果采用开关\Visualize，那么按编程参数，将在FlexPendant上显示消息框。如果未采用\Header参数，那么将显示默认页眉文本。当WaitAI指令的执行准备就绪时，消息框将从FlexPendant撤除。

TRAP层级的新消息框将焦点从基础层级的消息框移开。

更多示例

有关于指令WaitAI的更多例子阐述如下。

例 1

```
VAR num myvalattimeout:=0;
WaitAI ai1, \LT, 5 \MaxTime:=4 \ValueAtTimeout:=myvalattimeout;
ERROR
  IF ERRNO=ERR_WAIT_MAXTIME THEN
    TPWrite "Value of ai1 at timeout:" + ValToStr(myvalattimeout);
    TRYNEXT;
  ELSE
    ! No error recovery handling
  ENDIF
```

仅当ai1小于5或超时，方可继续程序执行。如果超时，则可录入超时情形下信号ai1的值，而无需再一次读取信号。

例 2

```
WaitAI ai1 \GT, 5 \Visualize \Header:="Waiting for signal"
     \MsgArray:=["Movement will not start until", "the condition
     below is TRUE"] \Icon:=iconError;
```

下一页继续

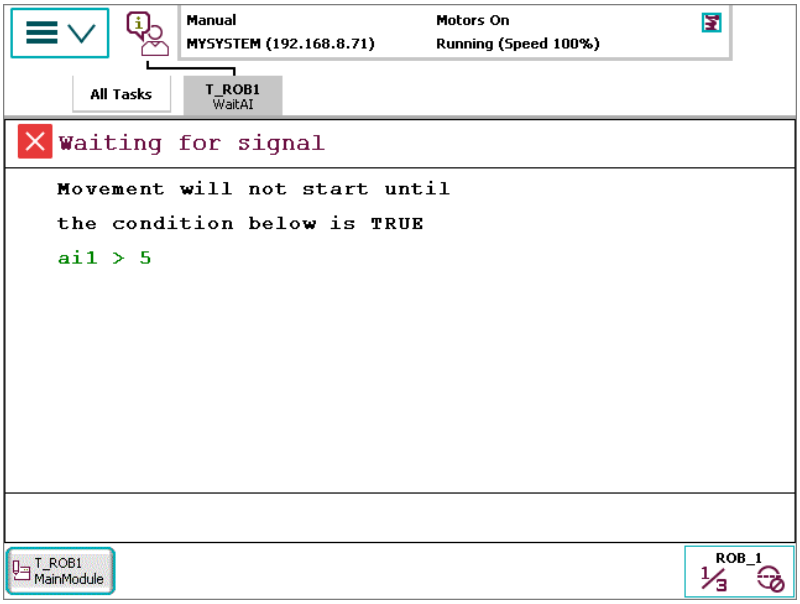
1.312 WaitAI - 等待直至已设置模拟信号输入信号值

RobotWare - OS

续前页

```
MoveL p40, v500, z20, L10tip;
..
```

如果条件不符，则在可选参数\Header和\MsgArray 将被写入FlexPendant示教器屏幕上，为满足条件的家族要随行。保持状态



xx1600000150

错误处理

系统会生成下列可恢复错误，并在错误处理器中处理这些错误。系统变量ERRNO将被设置成：

如果关于指定模拟信号输入信号Signal的编程Value参数超出限制，则ERR_AO_LIM。

如果信号变量是RAPID中声明的变量，则ERR_NO_ALIASIO_DEF。尚未同I/O配置以及指令AliasIO中确定的I/O信号相连。

如果与I/O单元无接触，则ERR_NORUNUNIT。

如果无法访问I/O信号（仅对ICI现场总线有效），则ERR_SIG_NOT_VALID。

如果在信号改变为正确值之前存在超时（参数\MaxTime），则ERR_WAIT_MAXTIME。

语法

```
WaitAI
  [ Signal ':=' ] < variable (VAR) of signalai> ´,´
  [ '\' LT] | [ '\' GT] ','
  [ Value ':=' ] < expression (IN) of num>
  ['\'MaxTime ´:='<expression (IN) of num>]
  [ '\'ValueAtTimeout' :=' < variable (VAR) of num >]
  ['\' Visualize]
  ['\' Header ':=' <expression (IN) of string>]]
  ['\' Message ':=' <expression (IN) of string>]
  | ['\' MsgArray ':=' <array {*} (IN) of string>]
  ['\' Wrap]
  ['\' Icon ':=' <expression (IN) of icondata>]
  ['\' Image ':=' <expression (IN) of string>]
```

下一页继续

续前页

```
['\' VisualizeTime ':=' <expression (IN) of num>] ';'
```

相关信息

信息，关于	请参阅
等待直至满足条件	第*897*页的*WaitUntil - 等待直至满足条件*
等待一段指定的时间	第*895*页的*WaitTime - 等待给定的时间*
等待直至已设置/重置模拟信号输出	第*856*页的*WaitAO - 等待直至已设置模拟信号输出信号值*

1.313 WaitAO - 等待直至已设置模拟信号输出信号值

手册用法

WaitAO（*Wait Analog Output*）用于等待，直至已设置模拟信号输出信号值。

基本示例

以下实例介绍了指令WaitAO：

例 1

```
WaitAO ao1, \GT, 5;
```

仅在ao1模拟信号输出具有大于5的值之后，方可继续程序执行。

例 2

```
WaitAO ao1, \LT, 5;
```

仅在ao1模拟信号输出具有小于5的值之后，方可继续程序执行。

变元

```
WaitAO Signal [\LT] | [\GT] Value [\MaxTime] [\ValueAtTimeout]
     [\Visualize] [\Header] [\Message] | [\MsgArray] [\Wrap]
     [\Icon] [\Image] [\VisualizeTime]
```

Signal

数据类型：signalao

模拟信号输出信号的名称。

[\LT]

Less Than

数据类型：switch

如果使用该参数，则WaitAO指令等待，直至模拟信号值小于Value中的值。

[\GT]

Greater Than

数据类型：switch

如果使用该参数，则WaitAO指令等待，直至模拟信号值大于Value中的值。

Value

数据类型：num

信号的期望值。

[\MaxTime]

Maximum Time

数据类型：num

允许的最长等待时间，以秒计。如果在满足条件之前耗尽该时间，则将调用错误处理器，如果存在这样的情况，则采用错误代码ERR_WAIT_MAXTIME。如果不存在错误处理器，则将停止执行。

[\ValueAtTimeout]

数据类型：num

下一页继续

如果指令超时，则将当前信号值储存在该变量中。仅将系统变量ERRNO设置为ERR_WAIT_MAXTIME时，方才设置本变量。

[\Visualize]

数据类型：switch

如被选中，将激活可视化。可视化包括未满足的条件的消息框、图标、页眉、消息行，将按编程参数来显示图像。

[\Header]

数据类型：string

要写在消息框顶部的页眉文字。最大40个字符。如果没有使用\Header 参数，则将显示默认消息。

[\Message]

数据类型：string

在显示器上可写入的一行文字。最多50字符。

[\MsgArray]

(Message Array)

数据类型：string

显示器上需编写的来自数组的若干文本行。同时只可采用参数\Message或\MsgArray中的其中一项。

最大布局空间是5 行，每行50字符。

[\Wrap]

数据类型：switch

如果选择，则参数\MsgArray中的所有指定字符串均将连接到一个字符串，各单独字符串之间仅存在单间距，并以尽可能少的行展开。

参数\MsgArray的各字符串将默认显示在显示器的单独各行。

[\Icon]

数据类型：icondata

定义有待显示的图标。仅可使用一种icondata型预定义图标。参见第*1411*页的预定义数据。

默认没有图标。

[\Image]

数据类型：string

应采用的图像的名称。为了开启您自己的图像，必须将图像放置在活跃系统的*HOME:*目录或直接放置在活跃系统中。

建议将文件放置在*HOME:*目录，从而在完成备份和恢复的情况下，可保存文件。

要求重启，然后，FlexPendant将加载图像。

系统命令是采用RobotWare附加功能*FlexPendant Interface*。

显示的图像可能为185像素宽度和300像素高度。如果图像更大，那么，只会从图像左上侧开始显示图像的185 * 300像素部分。

下一页继续

1.313 WaitAO - 等待直至已设置模拟信号输出信号值
RobotWare - OS
续前页

在图像可为多大尺寸或FlexPendant可加载多少图像方面，没有确切值。这取决于加载到FlexPendant的其他文件的大小。仅仅在采用未加载到FlexPendant的图像时，程序执行才会继续。

[\VisualizeTime]

数据类型：num

FlexPendant上应出现消息框前的等待时间。如果采用参数\VisualizeTime和\MaxTime，那么，参数\MaxTime采用的时间需大于参数\VisualizeTime采用的时间。

不采用参数\VisualizeTime情况下，可视化的默认时间为5s。最小值为1s。最大值没有限制。分辨率为0.001s。

程序执行

当执行本指令时，如果信号值正确，则本程序仅仅继续以下指令。

如果信号值不正确，则机械臂进入等待状态，且程序在信号改变为正确值时继续。通过中断来探测改变，其作出快速响应（非查询）。

当机械臂正在等待时，对时间进行监督。机械臂默认可永久地等待，但是，可通过可选参数 \MaxTime来规定最长等待时间。如果超出该最长时间，则会引起错误。

如果停止程序执行，并随后重启，则本指令评估信号的当前值。否定程序停止期间的任意改变。

在手动模式下，在等待3 s以上时间后，警告框将弹出，询问是否想要模拟指令。如果不想警告框出现，那么您可将系统参数*SimulateMenu*设为NO，请参见技术参考手册 - 系统参数主题*Controller*类型*General RAPID*。

如果采用开关\Visualize，那么按编程参数，将在FlexPendant上显示消息框。如果未采用\Header参数，那么将显示默认页眉文本。当WaitAO指令的执行准备就绪时，消息框将从FlexPendant撤除。

TRAP层级的新消息框将焦点从基础层级的消息框移开。

更多示例

有关于指令WaitAO的更多例子阐述如下。

例 1

```
VAR num myvalattimeout:=0;
WaitAO ao1, \LT, 5 \MaxTime:=4 \ValueAtTimeout:=myvalattimeout;
ERROR
  IF ERRNO=ERR_WAIT_MAXTIME THEN
    TPWrite "Value of ao1 at timeout:" + ValToStr(myvalattimeout);
    TRYNEXT;
  ELSE
    ! No error recovery handling
  ENDIF
```

仅当ao1小于5或超时，方可继续程序执行。如果超时，则可录入超时情形下信号ao1的值，而无需再一次读取信号。

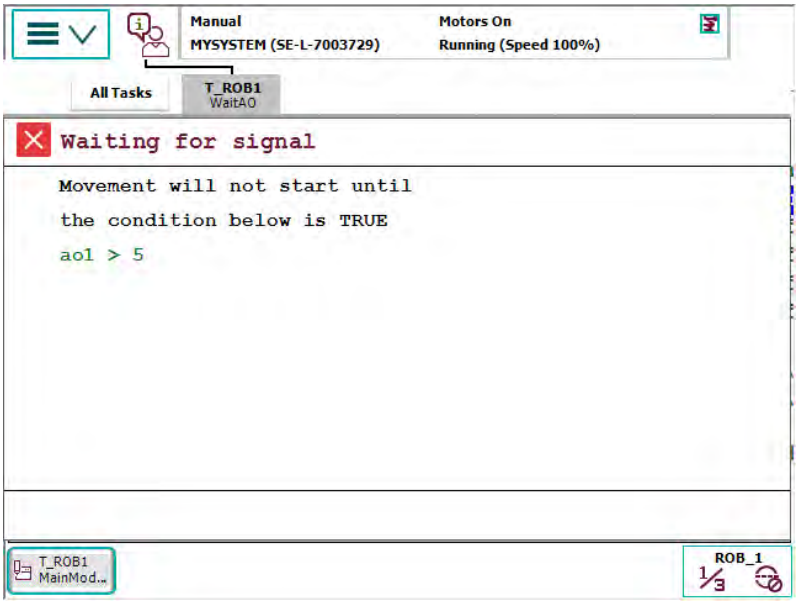
例 2

```
WaitAO ao1 \GT, 5 \Visualize \Header:="Waiting for signal"
     \MsgArray:=["Movement will not start until", "the condition
     below is TRUE"] \Icon:=iconError;
```

下一页继续

```
MoveL p40, v500, z20, L10tip;
..
```

如果条件不符，则在可选参数\Header和\MsgArray 将被写入FlexPendant示教器屏幕上，为满足条件的家族要随行。保持状态



xx1600000151

错误处理

系统会生成下列可恢复错误，并在错误处理器中处理这些错误。系统变量ERRNO将被设置成：

如果关于指定模拟信号输出信号Signal的编程Value参数超出限值，则ERR_AO_LIM。

如果信号变量是RAPID中声明的变量，则ERR_NO_ALIASIO_DEF。尚未同I/O配置以及指令AliasIO中确定的I/O信号相连。

如果与I/O单元无接触，则ERR_NORUNUNIT。

如果无法访问I/O信号（仅对ICI现场总线有效），则ERR_SIG_NOT_VALID。

如果在信号改变为正确值之前存在超时（参数\MaxTime），则ERR_WAIT_MAXTIME。

语法

```
WaitAO
  [ Signal ':=' ] < variable (VAR) of signalao> ´,'
  [ '\' LT] | [ '\' GT] ',’
  [ Value ':=' ] < expression (IN) of num>
  ['\'MaxTime ´:='<expression (IN) of num>]
  [ '\'ValueAtTimeout' :=' < variable (VAR) of num >]
  ['\' Visualize]
  ['\' Header ':=' <expression (IN) of string>]]
  ['\' Message ':=' <expression (IN) of string>]
  | ['\' MsgArray ':=' <array {*} (IN) of string>]
  ['\' Wrap]
  ['\' Icon ':=' <expression (IN) of icondata>]
  ['\' Image ':=' <expression (IN) of string>]
```

下一页继续

1.313 WaitAO - 等待直至已设置模拟信号输出信号值

RobotWare - OS

续前页

```
['\' VisualizeTime ':=' <expression (IN) of num>] ';'
```

相关信息

信息，关于	请参阅
等待直至满足条件	第*897*页的*WaitUntil* - 等待直至满足条件
等待一段指定的时间	第*895*页的*WaitTime* - 等待给定的时间
等待直至已设置/重置模拟信号输入	第*851*页的*WaitAI* - 等待直至已设置模拟信号输入信号值

1.314 WaitDI - 等待直至已设置数字信号输入信号

手册用法

`WaitDI`（*Wait Digital Input*）用于等待，直至已设置数字信号输入。

基本示例

以下实例介绍了指令`WaitDI`：

例 1

```
WaitDI di4, 1;
```

仅在已设置`di4`输入后，继续程序执行。

例 2

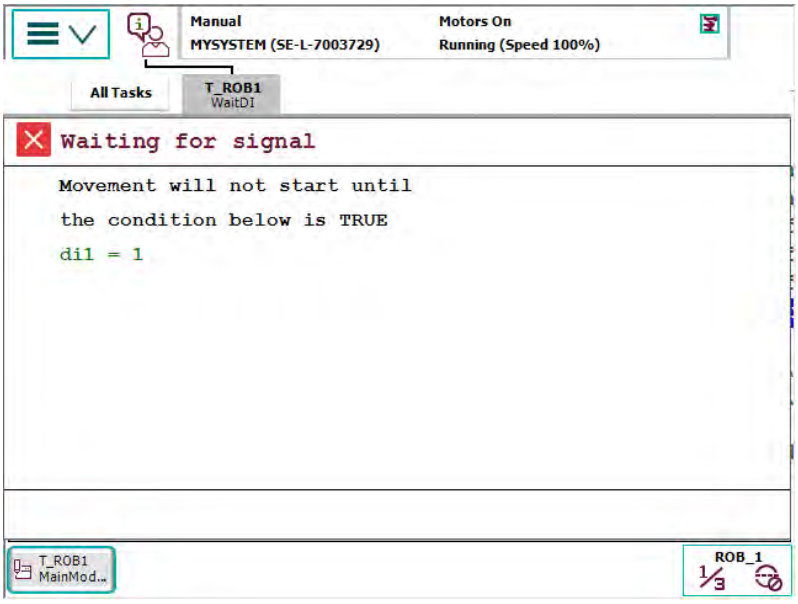
```
WaitDI grip_status, 0;
```

仅在已重置`grip_status`输入后，继续程序执行。

例 3

```
WaitDI di1, 1, \Visualize \Header:="Waiting for signal"
     \MsgArray:=["Movement will not start until", "the condition
     below is TRUE"] \Icon:=iconError;
MoveL p40, v500, z20, L10tip;
..
```

如果条件不符，则在可选参数`\Header`和`\MsgArray`将被写入FlexPendant示教器屏幕上，为满足条件的家族要随行。保持状态



xx1600000148

变元

```
WaitDI Signal Value [\MaxTime] [\TimeFlag] [\Visualize] [\Header]
     [\Message] | [\MsgArray] [\Wrap] [\Icon] [\Image]
     [\VisualizeTime]
```

`Signal`

数据类型：`signaldi`

下一页继续

1.314 WaitDI - 等待直至已设置数字信号输入信号
RobotWare - OS
续前页

信号的名称。

Value

数据类型：dionum

信号的期望值。

[\MaxTime]

Maximum Time

数据类型：num

允许的最长等待时间，以秒计。如果在满足条件之前耗尽该时间，则将调用错误处理器，如果存在这样的情况，则采用错误代码ERR_WAIT_MAXTIME。如果不存在错误处理器，则将停止执行。

[\TimeFlag]

Timeout Flag

数据类型：bool

如果在满足条件之前耗尽最长允许时间，则包含该值的输出参数为TRUE。如果该参数包含在本指令中，则不将其视为耗尽最长时间时的错误。如果MaxTime参数不包括在本指令中，则将忽略该参数。

[\Visualize]

数据类型：switch

如被选中，将激活可视化。可视化包括未满足的条件的消息框、图标、页眉、消息行，将按编程参数来显示图像。

[\Header]

数据类型：string

要写在消息框顶部的页眉文字。最大40个字符。如果没有使用\Header 参数，则将显示默认消息。

[\Message]

数据类型：string

在显示器上可写入的一行文字。最多50字符。

[\MsgArray]

(Message Array)

数据类型：string

显示器上需编写的来自数组的若干文本行。同时只可采用参数\Message或\MsgArray中的其中一项。

最大布局空间是5 行，每行50字符。

[\Wrap]

数据类型：switch

如果选择，则参数\MsgArray中的所有指定字符串均将连接到一个字符串，各单独字符串之间仅存在单间距，并以尽可能少的行展开。

参数\MsgArray的各字符串将默认显示在显示器的单独各行。

下一页继续

[\Icon]

数据类型：icondata

定义有待显示的图标。仅可使用一种icondata型预定义图标。参见第*1411*页的预定义数据。

默认没有图标。

[\Image]

数据类型：string

应采用的图像的名称。为了开启您自己的图像，必须将图像放置在活跃系统的*HOME:*目录或直接放置在活跃系统中。

建议将文件放置在*HOME:*目录，从而在完成备份和恢复的情况下，可保存文件。

要求重启，然后，FlexPendant将加载图像。

系统命令是采用RobotWare附加功能*FlexPendant Interface*。

显示的图像可能为185像素宽度和300像素高度。如果图像更大，那么，只会从图像左上侧开始显示图像的185 * 300像素部分。

在图像可为多大尺寸或FlexPendant可加载多少图像方面，没有确切值。这取决于加载到FlexPendant的其他文件的大小。仅仅在采用未加载到FlexPendant的图像时，程序执行才会继续。

[\VisualizeTime]

数据类型：num

FlexPendant上应出现消息框前的等待时间。如果采用参数\VisualizeTime和\MaxTime，那么，参数\MaxTime采用的时间需大于参数\VisualizeTime采用的时间。

不采用参数\VisualizeTime情况下，可视化的默认时间为5s。最小值为1s。最大值没有限制。分辨率为0.001s。

程序执行

当执行本指令时，如果信号值正确，则本程序仅仅继续以下指令。

如果信号值不正确，则机械臂进入等待状态，且当信号改变为正确值时，程序继续。通过中断来探测改变，其作出快速响应（非查询）。

当机械臂正在等待时，对时间进行监督，且如果其超出最长时间值，则程序将在指定TimeFlag时继续，否则将会引起错误。如果指定TimeFlag，则在超出时间时设置为TRUE。否则，其将设置为FALSE。

如果停止程序执行，并随后重启，则本指令评估信号的当前值。否定程序停止期间的任意改变。

在手动模式下，在等待3 s以上时间后，警告框将弹出，询问是否想要模拟指令。如果不想警告框出现，那么您可将系统参数*SimulateMenu*设为NO，请参见技术参考手册 - 系统参数主题*Controller*类型*General RAPID*。

如果采用开关\Visualize，那么按编程参数，将在FlexPendant上显示消息框。如果未采用\Header参数，那么将显示默认页眉文本。当WaitDI指令的执行准备就绪时，消息框将从FlexPendant撤除。

TRAP层级的新消息框将焦点从基础层级的消息框移开。

下一页继续

1.314 WaitDI - 等待直至已设置数字信号输入信号

RobotWare - OS

续前页

错误处理

系统会生成下列可恢复错误，并在错误处理器中处理这些错误。系统变量ERRNO将被设置成：

如果信号变量是RAPID中声明的变量，则ERR_NO_ALIASIO_DEF。尚未同I/O配置以及指令AliasIO中确定的I/O信号相连。

如果与I/O单元无接触，则ERR_NORUNUNIT。

如果无法访问I/O信号（仅对ICI现场总线有效），则ERR_SIG_NOT_VALID。

语法

```
WaitDI
  [ Signal ':=' ] < variable (VAR) of signaldi>' ,'
  [ Value ':=' ] < expression (IN) of dionum>
  ['\'MaxTime' :='<expression (IN) of num>]
  ['\'TimeFlag':='<variable (VAR) of bool>]
  ['\' Visualize]
  ['\' Header ':=' <expression (IN) of string>]]
  ['\' Message ':=' <expression (IN) of string>]
  | ['\' MsgArray ':=' <array {*} (IN) of string>]
  ['\' Wrap]
  ['\' Icon ':=' <expression (IN) of icondata>]
  ['\' Image ':=' <expression (IN) of string>]
  ['\' VisualizeTime ':=' <expression (IN) of num>] ';'
```

相关信息

信息，关于	请参阅
等待直至满足条件	第*897*页的*WaitUntil* - 等待直至满足条件
等待一段指定的时间	第*895*页的*WaitTime* - 等待给定的时间

1.315 WaitDO - 等待直至已设置数字信号输出信号

手册用法

`WaitDO`（*Wait Digital Output*）用于等待，直至已设置数字信号输出。

基本示例

以下实例介绍了指令`WaitDO`：

例 1

```
WaitDO do4, 1;
```

仅在已设置`do4`输出后，继续程序执行。

例 2

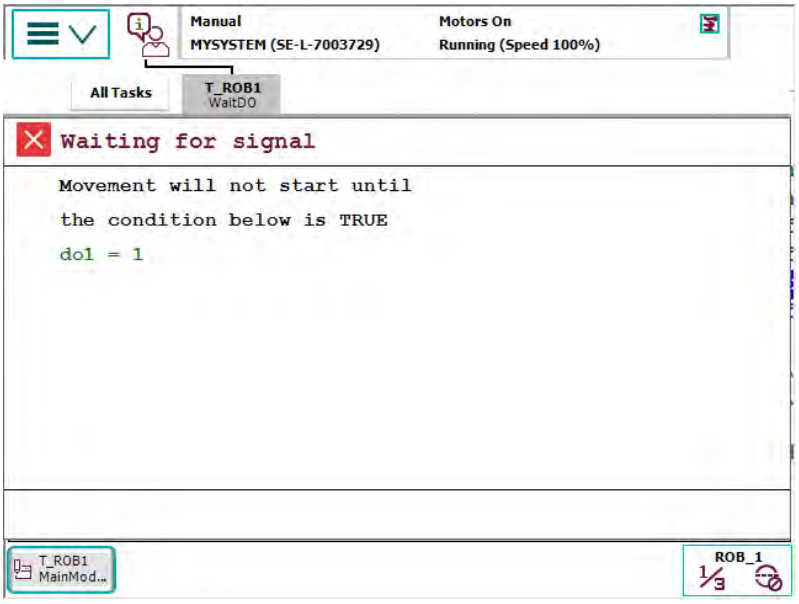
```
WaitDO grip_status, 0;
```

仅在已重置`grip_status`输出后，继续程序执行。

例 3

```
WaitDO do1, 1, \Visualize \Header:="Waiting for signal"
     \MsgArray:=["Movement will not start until", "the condition
     below is TRUE"] \Icon:=iconError;
MoveL p40, v500, z20, L10tip;
..
```

如果条件不符，则在可选参数`\Header`和`\MsgArray`将被写入FlexPendant示教器屏幕上，为满足条件的家族要随行。保持状态



xx1600000149

变元

```
WaitDO Signal Value [\MaxTime] [\TimeFlag] [\Visualize] [\Header]
     [\Message] | [\MsgArray] [\Wrap] [\Icon] [\Image]
     [\VisualizeTime]
```

`Signal`

数据类型：`signaldo`

下一页继续

1.315 WaitDO - 等待直至已设置数字信号输出信号
RobotWare - OS
续前页

信号的名称。

Value

数据类型：dionum

信号的期望值。

[\MaxTime]

Maximum Time

数据类型：num

允许的最长等待时间，以秒计。如果在满足条件之前耗尽该时间，且未使用TimeFlag参数，则可通过错误代码ERR_WAIT_MAXTIME调用错误处理器。如果不存在错误处理器，则将停止执行。

[\TimeFlag]

Timeout Flag

数据类型：bool

如果在满足条件之前耗尽最长允许时间，则包含该值的输出参数为TRUE。如果该参数包含在本指令中，则不将其视为耗尽最长时间时的错误。如果MaxTime参数不包括在本指令中，则将忽略该参数。

[\Visualize]

数据类型：switch

如被选中，将激活可视化。可视化包括未满足的条件的消息框、图标、页眉、消息行，将按编程参数来显示图像。

[\Header]

数据类型：string

要写在消息框顶部的页眉文字。最大40个字符。如果没有使用\Header 参数，则将显示默认消息。

[\Message]

数据类型：string

在显示器上可写入的一行文字。最多50字符。

[\MsgArray]

(Message Array)

数据类型：string

显示器上需编写的来自数组的若干文本行。同时只可采用参数\Message或\MsgArray中的其中一项。

最大布局空间是5 行，每行50字符。

[\Wrap]

数据类型：switch

如果选择，则参数\MsgArray中的所有指定字符串均将连接到一个字符串，各单独字符串之间仅存在单间距，并以尽可能少的行展开。

参数\MsgArray的各字符串将默认显示在显示器的单独各行。

下一页继续

[\Icon]

数据类型：icondata

定义有待显示的图标。仅可使用一种icondata型预定义图标。参见第*1411*页的预定义数据。

默认没有图标。

[\Image]

数据类型：string

应采用的图像的名称。为了开启您自己的图像，必须将图像放置在活跃系统的*HOME:*目录或直接放置在活跃系统中。

建议将文件放置在*HOME:*目录，从而在完成备份和恢复的情况下，可保存文件。

要求重启，然后，FlexPendant将加载图像。

系统命令是采用RobotWare附加功能*FlexPendant Interface*。

显示的图像可能为185像素宽度和300像素高度。如果图像更大，那么，只会从图像左上侧开始显示图像的185 * 300像素部分。

在图像可为多大尺寸或FlexPendant可加载多少图像方面，没有确切值。这取决于加载到FlexPendant的其他文件的大小。仅仅在采用未加载到FlexPendant的图像时，程序执行才会继续。

[\VisualizeTime]

数据类型：num

FlexPendant上应出现消息框前的等待时间。如果采用参数\VisualizeTime和\MaxTime，那么，参数\MaxTime采用的时间需大于参数\VisualizeTime采用的时间。

不采用参数\VisualizeTime情况下，可视化的默认时间为5s。最小值为1s。最大值没有限制。分辨率为0.001s。

程序执行

当执行本指令时，如果输出信号值正确，则本程序仅仅继续以下指令。

如果输出信号值不正确，则机械臂进入等待状态。当信号改变为正确值时，程序继续。通过中断来探测改变，其作出快速响应（非查询）。

当机械臂正在等待时，对时间进行监督，且如果其超出最长时间值，则程序将在指定TimeFlag时继续，否则将会引起错误。如果指定TimeFlag，则在超出时间时设置为TRUE。否则，其将设置为FALSE。

如果停止程序执行，并随后重启，则本指令评估信号的当前值。否定程序停止期间的任意改变。

在手动模式下，在等待3 s以上时间后，警告框将弹出，询问是否想要模拟指令。如果不想警告框出现，那么您可将系统参数*SimulateMenu*设为NO，请参见技术参考手册 - 系统参数主题*Controller*类型*General RAPID*。

如果采用开关\Visualize，那么按编程参数，将在FlexPendant上显示消息框。如果未采用\Header参数，那么将显示默认页眉文本。当WaitDO指令的执行准备就绪时，消息框将从FlexPendant撤除。

TRAP层级的新消息框将焦点从基础层级的消息框移开。

下一页继续

1.315 WaitDO - 等待直至已设置数字信号输出信号

RobotWare - OS

续前页

错误处理

系统会生成下列可恢复错误，并在错误处理器中处理这些错误。系统变量ERRNO将被设置成：

如果信号变量是RAPID中声明的变量，则ERR_NO_ALIASIO_DEF。尚未同I/O配置以及指令AliasIO中确定的I/O信号相连。

如果与I/O单元无接触，则ERR_NORUNUNIT。

如果无法访问I/O信号（仅对ICI现场总线有效），则ERR_SIG_NOT_VALID。

语法

```
WaitDO
  [ Signal ´:=´] < variable (VAR) of signaldo >´,´
  [ Value ’:=’ ] < expression (IN) of dionum>
  [’\’MaxTime’ :=’<expression (IN) of num>]
  [’\’TimeFlag’:=’<variable (VAR) of bool>]
  ['\' Visualize]
  ['\' Header ':=' <expression (IN) of string>]]
  ['\' Message ':=' <expression (IN) of string>]
  | ['\' MsgArray ':=' <array {*} (IN) of string>]
  ['\' Wrap]
  ['\' Icon ':=' <expression (IN) of icondata>]
  ['\' Image ':=' <expression (IN) of string>]
  ['\' VisualizeTime ':=' <expression (IN) of num>] ';'
```

相关信息

信息，关于	请参阅
等待直至满足条件	第*897*页的*WaitUntil* - 等待直至满足条件
等待一段指定的时间	第*895*页的*WaitTime* - 等待给定的时间
等待直至已设置/重置输入	第*861*页的*WaitDI* - 等待直至已设置数字信号输入信号

1.316 WaitGI - 等待直至已设置一组数字信号输入信号

手册用法

WaitGI（*Wait Group digital Input*）用于等待，直至将一组数字信号输入信号设置为指定值。

基本示例

以下实例介绍了指令WaitGI：

另请参阅第*872*页的更多示例

例 1

```
WaitGI gi4, 5;
```

仅在gi4输入已具有值5后，继续程序执行。

例 2

```
WaitGI grip_status, 0;
```

仅在已重置grip_status输入后，继续程序执行。

变元

```
WaitGI Signal [\NOTEQ] | [\LT] | [\GT] Value | Dvalue [\MaxTime]
     [\ValueAtTimeout] | [\DvalueAtTimeout] [\Visualize] [\Header]
     [\Message] | [\MsgArray] [\Wrap] [\Icon] [\Image]
     [\VisualizeTime]
```

Signal

数据类型：signalgi

数字组输入信号的名称。

[\NOTEQ]

NOT EQual

数据类型：switch

如果使用该参数，则WaitGI指令等待，直至数字组信号值除以Value中的值。

[\LT]

Less Than

数据类型：switch

如果使用该参数，则WaitGI指令等待，直至数字组信号值小于Value中的值。

[\GT]

Greater Than

数据类型：switch

如果使用该参数，则WaitGI指令等待，直至数字组信号值大于Value中的值。

Value

数据类型：num

信号的期望值。必须为所用数字组输入信号工作范围内的整数值。允许值取决于该组中的信号数量。可在Value参数中使用的最大值为8388608，该值为可作为最大值的一个23位数字信号值。

下一页继续

1.316 WaitGI - 等待直至已设置一组数字信号输入信号
RobotWare - OS
续前页

Dvalue

数据类型：dnum

所需的信号值。必须为所用数字组输入信号的工作范围内的整数。允许的值取决于组内信号的数量。一个数字组信号的最大信号位数量为 32。搭配 dnum 变量时，可以覆盖值范围 0-4294967295，这是 32 位数字组信号所能达到最大信号位数量。

[\MaxTime]

Maximum Time

数据类型：num

允许的最长等待时间，以秒计。如果在满足条件之前耗尽该时间，则将调用错误处理器（如果存在这样的情况），则采用错误代码ERR_WAIT_MAXTIME。如果不存在错误处理器，则将停止执行。

[\ValueAtTimeout]

数据类型：num

如果指令超时，则将当前信号值储存在该变量中。仅将系统变量ERRNO设置为ERR_WAIT_MAXTIME时，方才设置该变量。如果使用Dvalue参数，则使用参数DvalueAtTimeout，以储存关于信号的当前值（原因：关于num最大整数值的限制）。

始终将0和8388608之间的信号值储存为一个精确的整数。

[\DvalueAtTimeout]

数据类型：dnum

如果指令超时，则将当前信号值储存在该变量中。仅将系统变量ERRNO设置为ERR_WAIT_MAXTIME时，方才设置本变量。

始终将0和4294967295之间的信号值储存为一个精确的整数。

[\Visualize]

数据类型：switch

如被选中，将激活可视化。可视化包括未满足的条件的消息框、图标、页眉、消息行，将按编程参数来显示图像。

[\Header]

数据类型：string

要写在消息框顶部的页眉文字。最大40个字符。如果没有使用\Header 参数，则将显示默认消息。

[\Message]

数据类型：string

在显示器上可写入的一行文字。最多50字符。

[\MsgArray]

(Message Array)

数据类型：string

显示器上需编写的来自数组的若干文本行。同时只可采用参数\Message或\MsgArray中的其中一项。

最大布局空间是5 行，每行50字符。

下一页继续

[\Wrap]

数据类型：switch

如果选择，则参数\MsgArray中的所有指定字符串均将连接到一个字符串，各单独字符串之间仅存在单间距，并以尽可能少的行展开。

参数\MsgArray的各字符串将默认显示在显示器的单独各行。

[\Icon]

数据类型：icondata

定义有待显示的图标。仅可使用一种icondata型预定义图标。参见第*1411*页的预定义数据。

默认没有图标。

[\Image]

数据类型：string

应采用的图像的名称。为了开启您自己的图像，必须将图像放置在活跃系统的*HOME:*目录或直接放置在活跃系统中。

建议将文件放置在*HOME:*目录，从而在完成备份和恢复的情况下，可保存文件。

要求重启，然后，FlexPendant将加载图像。

系统命令是采用RobotWare附加功能*FlexPendant Interface*。

显示的图像可能为185像素宽度和300像素高度。如果图像更大，那么，只会从图像左上侧开始显示图像的185 * 300像素部分。

在图像可为多大尺寸或FlexPendant可加载多少图像方面，没有确切值。这取决于加载到FlexPendant的其他文件的大小。仅仅在采用未加载到FlexPendant的图像时，程序执行才会继续。

[\VisualizeTime]

数据类型：num

FlexPendant上应出现消息框前的等待时间。如果采用参数\VisualizeTime和\MaxTime，那么，参数\MaxTime采用的时间需大于参数\VisualizeTime采用的时间。

不采用参数\VisualizeTime情况下，可视化的默认时间为5s。最小值为1s。最大值没有限制。分辨率为0.001s。

程序执行

当执行本指令时，如果信号值正确，则本程序仅仅继续以下指令。

如果信号值不正确，则机械臂进入等待状态，且程序在信号改变为正确值时继续。通过中断来探测改变，其作出快速响应（非查询）。

当机械臂正在等待时，对时间进行监督。机械臂默认可永久地等待，但是，可通过可选参数\MaxTime来规定最长等待时间。如果超出该最长时间，则会引起错误。

如果停止程序执行，并随后重启，则本指令评估信号的当前值。否定程序停止期间的任意改变。

在手动模式下，在等待3 s以上时间后，警告框将弹出，询问是否想要模拟指令。如果不想警告框出现，那么您可将系统参数*SimulateMenu*设为NO，请参见技术参考手册 - 系统参数主题*Controller*类型*General RAPID*。

下一页继续

1.316 WaitGI - 等待直至已设置一组数字信号输入信号
RobotWare - OS
续前页

如果采用开关\Visualize，那么按编程参数，将在FlexPendant上显示消息框。如果未采用\Header参数，那么将显示默认页眉文本。当WaitGI指令的执行准备就绪时，消息框将从FlexPendant撤除。

TRAP层级的新消息框将焦点从基础层级的消息框移开。

更多示例

有关于指令WaitGI的更多例子阐述如下。

例 1

```
WaitGI gi1,\NOTEQ,0;
```

仅在不同于值0的gi1后继续程序执行。

例 2

```
WaitGI gi1,\LT,1;
```

仅在gi1小于1之后继续程序执行。

例 3

```
WaitGI gi1,\GT,0;
```

仅在gi1大于0之后继续程序执行。

例 4

```
VAR num myvalattimeout:=0;
WaitGI gi1, 5 \MaxTime:=4 \ValueAtTimeout:=myvalattimeout;
ERROR
  IF ERRNO=ERR_WAIT_MAXTIME THEN
    TPWrite "Value of gi1 at timeout:" + ValToStr(myvalattimeout);
    TRYNEXT;
  ELSE
    ! No error recovery handling
  ENDIF
```

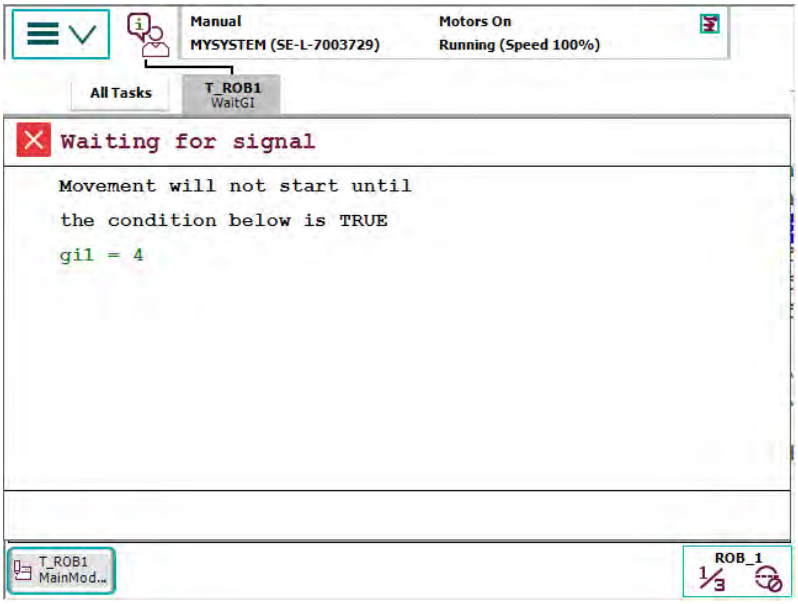
仅当gi1等于5或超时，方可继续程序执行。如果超时，则可录入超时情形下信号gi1的值，而无需再一次读取信号。

例 5

```
WaitGI gi1, 4, \Visualize \Header:="Waiting for signal"
     \MsgArray:=["Movement will not start until", "the condition
     below is TRUE"] \Icon:=iconError;
MoveL p40, v500, z20, L10tip;
..
```

下一页继续

如果条件不符，则在可选参数\Header和\MsgArray 将被写入FlexPendant示教器屏幕上，为满足条件的家族要随行。保持状态



xx1600000152

错误处理

系统会生成下列可恢复错误，并在错误处理器中处理这些错误。系统变量ERRNO将被设置成：

如果关于指定数字组输入信号Signal的已编程的Value或Dvalue参数超出限制，则ERR_GO_LIM。

如果信号变量是RAPID中声明的变量，则ERR_NO_ALIASIO_DEF。尚未同I/O配置以及指令AliasIO中确定的I/O信号相连。

如果与I/O单元无接触，则ERR_NORUNUNIT。

如果无法访问I/O信号（仅对ICI现场总线有效），则ERR_SIG_NOT_VALID。

如果在信号改变为正确值之前存在超时（参数\MaxTime），则ERR_WAIT_MAXTIME。

语法

```
WaitGI
  [ Signal ':=' ] < variable (VAR) of signalgi> ´,´
  ['\' NOTEQ] | [ '\' LT] | [ '\' GT] ','
  [ Value ':=' ] < expression (IN) of num>
  | [ Dvalue' :=' ] < expression (IN) of dnum>
  ['\'MaxTime ´:='<expression (IN) of num>]
  [ '\'ValueAtTimeout' :=' < variable (VAR) of num > ]
  | [ '\'DvalueAtTimeout' :=' < variable (VAR) of dnum > ]
  ['\' Visualize]
  ['\' Header ':=' <expression (IN) of string>]]
  ['\' Message ':=' <expression (IN) of string>]
  | ['\' MsgArray ':=' <array {*} (IN) of string>]
  ['\' Wrap]
  ['\' Icon ':=' <expression (IN) of icondata>]
```

下一页继续

1.316 WaitGI - 等待直至已设置一组数字信号输入信号

RobotWare - OS

续前页

```
['\' Image ':=' <expression (IN) of string>]
['\' VisualizeTime ':=' <expression (IN) of num>] ';'
```

相关信息

信息，关于	请参阅
等待直至满足条件	第*897*页的*WaitUntil* - 等待直至满足条件
等待一段指定的时间	第*895*页的*WaitTime* - 等待给定的时间
等待直至已设置/重置一组数字信号输出信号	第*875*页的*WaitGO* - 等待直至已设置一组数字信号输出信号

1.317 WaitGO - 等待直至已设置一组数字信号输出信号

手册用法

WaitGO（*Wait Group digital Output*）用于等待，直至将一组数字信号输出信号设置为指定值。

基本示例

以下实例介绍了指令WaitGO：

另请参阅第*878*页的更多示例

例 1

```
WaitGO go4, 5;
```

仅在go4输出具有值5之后继续程序执行。

例 2

```
WaitGO grip_status, 0;
```

仅在已重置grip_status输出之后继续程序执行。

变元

```
WaitGO Signal [\NOTEQ] | [\LT] | [\GT] Value | Dvalue [\MaxTime]
     [\ValueAtTimeout] | [\DvalueAtTimeout] [\Visualize] [\Header]
     [\Message] | [\MsgArray] [\Wrap] [\Icon] [\Image]
     [\VisualizeTime]
```

Signal

数据类型：signalgo

数字组输出信号的名称。

[\NOTEQ]

NOT EQual

数据类型：switch

如果使用该参数，则WaitGO指令等待，直至数字组信号值除以Value中的值。

[\LT]

Less Than

数据类型：switch

如果使用该参数，则WaitGO指令等待，直至数字组信号值小于Value中的值。

[\GT]

Greater Than

数据类型：switch

如果使用该参数，则WaitGO指令等待，直至数字组信号值大于Value中的值。

Value

数据类型：num

信号的期望值。必须为所用数字组输出信号工作范围内的整数值。允许值取决于该组中的信号数量。可在Value参数中使用的最大值为8388608，该值为可作为最大值的一个23位数字信号值。

下一页继续

1.317 WaitGO - 等待直至已设置一组数字信号输出信号
RobotWare - OS
续前页

Dvalue

数据类型：dnum

所需的信号值。必须为所用数字组输出信号的工作范围内的整数。允许的值取决于组内信号的数量。一个数字组信号的最大信号位数量为 32。搭配 dnum 变量时，可以覆盖值范围 0-4294967295，这是 32 位数字组信号所能达到最大信号位数量。

[\MaxTime]

Maximum Time

数据类型：num

允许的最长等待时间，以秒计。如果在满足条件之前耗尽该时间，则将调用错误处理器，如果存在这样的情况，则采用错误代码ERR_WAIT_MAXTIME。如果不存在错误处理器，则将停止执行。

[\ValueAtTimeout]

数据类型：num

如果指令超时，则将当前信号值储存在该变量中。仅将系统变量ERRNO设置为ERR_WAIT_MAXTIME时，方才设置该变量。如果使用Dvalue参数，则使用参数DvalueAtTimeout，以储存关于信号的当前值（原因：关于num最大整数值的限制）。

始终将0和8388608之间的信号值储存为一个精确的整数。

[\DvalueAtTimeout]

数据类型：dnum

如果指令超时，则将当前信号值储存在该变量中。仅将系统变量ERRNO设置为ERR_WAIT_MAXTIME时，方才设置本变量。

始终将0和4294967295之间的信号值储存为一个精确的整数。

[\Visualize]

数据类型：switch

如被选中，将激活可视化。可视化包括未满足的条件的消息框、图标、页眉、消息行，将按编程参数来显示图像。

[\Header]

数据类型：string

要写在消息框顶部的页眉文字。最大40个字符。如果没有使用\Header 参数，则将显示默认消息。

[\Message]

数据类型：string

在显示器上可写入的一行文字。最多50字符。

[\MsgArray]

(Message Array)

数据类型：string

显示器上需编写的来自数组的若干文本行。同时只可采用参数\Message或\MsgArray中的其中一项。

最大布局空间是5 行，每行50字符。

下一页继续

[\Wrap]

数据类型：switch

如果选择，则参数\MsgArray中的所有指定字符串均将连接到一个字符串，各单独字符串之间仅存在单间距，并以尽可能少的行展开。

参数\MsgArray的各字符串将默认显示在显示器的单独各行。

[\Icon]

数据类型：icondata

定义有待显示的图标。仅可使用一种icondata型预定义图标。参见第*1411*页的预定义数据。

默认没有图标。

[\Image]

数据类型：string

应采用的图像的名称。为了开启您自己的图像，必须将图像放置在活跃系统的*HOME:*目录或直接放置在活跃系统中。

建议将文件放置在*HOME:*目录，从而在完成备份和恢复的情况下，可保存文件。

要求重启，然后，FlexPendant将加载图像。

系统命令是采用RobotWare附加功能*FlexPendant Interface*。

显示的图像可能为185像素宽度和300像素高度。如果图像更大，那么，只会从图像左上侧开始显示图像的185 * 300像素部分。

在图像可为多大尺寸或FlexPendant可加载多少图像方面，没有确切值。这取决于加载到FlexPendant的其他文件的大小。仅仅在采用未加载到FlexPendant的图像时，程序执行才会继续。

[\VisualizeTime]

数据类型：num

FlexPendant上应出现消息框前的等待时间。如果采用参数\VisualizeTime和\MaxTime，那么，参数\MaxTime采用的时间需大于参数\VisualizeTime采用的时间。

不采用参数\VisualizeTime情况下，可视化的默认时间为5s。最小值为1s。最大值没有限制。分辨率为0.001s。

程序执行

当执行本指令时，如果信号值正确，则本程序仅仅继续以下指令。

如果信号值不正确，则机械臂进入等待状态，且程序在信号改变为正确值时继续。通过中断来探测改变，其作出快速响应（非查询）。

当机械臂正在等待时，对时间进行监督。机械臂默认可永久地等待，但是，可通过可选参数\MaxTime来规定最长等待时间。如果超出该最长时间，则会引起错误。

如果停止程序执行，并随后重启，则本指令评估信号的当前值。否定程序停止期间的任意改变。

在手动模式下，在等待3 s以上时间后，警告框将弹出，询问是否想要模拟指令。如果不想警告框出现，那么您可将系统参数*SimulateMenu*设为NO，请参见技术参考手册 - 系统参数主题*Controller*类型*General RAPID*。

下一页继续

1.317 WaitGO - 等待直至已设置一组数字信号输出信号
RobotWare - OS
续前页

如果采用开关\Visualize，那么按编程参数，将在FlexPendant上显示消息框。如果未采用\Header参数，那么将显示默认页眉文本。当WaitGO指令的执行准备就绪时，消息框将从FlexPendant撤除。

TRAP层级的新消息框将焦点从基础层级的消息框移开。

更多示例

有关于指令WaitGO的更多例子阐述如下。

例 1

```
WaitGO go1,\NOTEQ,0;
```

仅在不同于值0的go1后继续程序执行。

例 2

```
WaitGO go1,\LT,1;
```

仅在go1 小于1之后继续程序执行。

例 3

```
WaitGO go1,\GT,0;
```

仅在go1大于0之后继续程序执行。

例 4

```
VAR num myvalattimeout:=0;
WaitGO go1, 5 \MaxTime:=4 \ValueAtTimeout:=myvalattimeout;
ERROR
  IF ERRNO=ERR_WAIT_MAXTIME THEN
    TPWrite "Value of go1 at timeout:" + ValToStr(myvalattimeout);
    TRYNEXT;
  ELSE
    ! No error recovery handling
  ENDIF
```

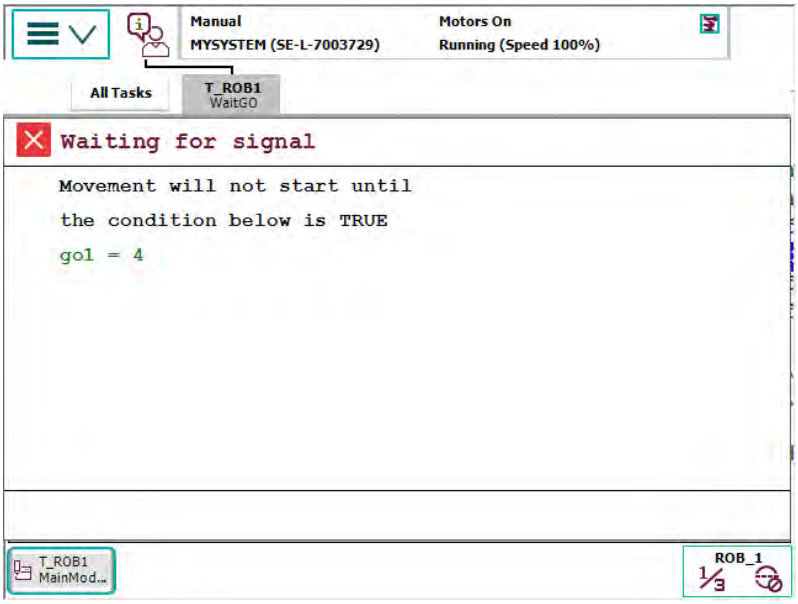
仅当go1等于5或超时，方可继续程序执行。如果超时，则可录入超时情形下信号go1的值，而无需再一次读取信号。

例 5

```
WaitGO go1, 4, \Visualize \Header:="Waiting for signal"
     \MsgArray:=["Movement will not start until", "the condition
     below is TRUE"] \Icon:=iconError;
MoveL p40, v500, z20, L10tip;
..
```

下一页继续

如果条件不符，则在可选参数\Header和\MsgArray 将被写入FlexPendant示教器屏幕上，为满足条件的家族要随行。保持状态



xx1600000153

错误处理

系统会生成下列可恢复错误，并在错误处理器中处理这些错误。系统变量ERRNO将被设置成：

如果关于指定数字组输出信号Signal的已编程的Value或Dvalue参数超出限制，则ERR_GO_LIM。

如果信号变量是RAPID中声明的变量，则ERR_NO_ALIASIO_DEF。尚未同I/O配置以及指令AliasIO中确定的I/O信号相连。

如果与I/O单元无接触，则ERR_NORUNUNIT。

如果无法访问I/O信号（仅对ICI现场总线有效），则ERR_SIG_NOT_VALID。

如果在信号改变为正确值之前存在超时（参数\MaxTime），则ERR_WAIT_MAXTIME。

语法

```
WaitGO
  [ Signal ':=' ] < variable (VAR) of signalgo> ','
  ['\' NOTEQ] | [ '\' LT] | [ '\' GT] ','
  [ Value ':=' ] < expression (IN) of num>
  | [ Dvalue ':=' ] < expression (IN) of dnum>
  ['\' MaxTime ':=' <expression (IN) of num>]
  ['\' ValueAtTimeout ':=' < variable (VAR) of num > ]
  | [ '\' DvalueAtTimeout ':=' < variable (VAR) of dnum > ]
  ['\' Visualize]
  ['\' Header ':=' <expression (IN) of string>]]
  ['\' Message ':=' <expression (IN) of string>]
  | ['\' MsgArray ':=' <array {*} (IN) of string>]
  ['\' Wrap]
  ['\' Icon ':=' <expression (IN) of icondata>]
```

下一页继续

1.317 WaitGO - 等待直至已设置一组数字信号输出信号

RobotWare - OS

续前页

```
['\' Image ':=' <expression (IN) of string>]
['\' VisualizeTime ':=' <expression (IN) of num>] ';'
```

相关信息

信息，关于	请参阅
等待直至满足条件	第*897*页的*WaitUntil* - 等待直至满足条件
等待一段指定的时间	第*895*页的*WaitTime* - 等待给定的时间
等待直至已设置/重置一组数字信号输入信号	第*869*页的*WaitGI* - 等待直至已设置一组数字信号输入信号

1.318 WaitLoad - 将加载的模块与任务相连

手册用法

WaitLoad用于同通过程序任务的指令StartLoad而加载的模块相连。

将加载普通程序模块添加至程序内存中已经存在的模块中。

在可以使用其任何符号或程序之前，必须通过指令WaitLoad，将通过 StartLoad 加载的模块与程序任务相连。

如果使用可选开关，则WaitLoad亦可卸载普通程序模块。这将尽可能地减少链接的数量（1代替2）。

如果使用可选开关 \CheckRef，则WaitLoad亦可检查任何未解决的引用。

基本示例

以下实例介绍了指令WaitLoad：

另请参阅第*882*页的更多示例

例 1

```
VAR loadsession load1;
...
StartLoad "HOME:/PART_A.MOD", load1;
MoveL p10, v1000, z50, tool1 \WObj:=wobj1;
MoveL p20, v1000, z50, tool1 \WObj:=wobj1;
MoveL p30, v1000, z50, tool1 \WObj:=wobj1;
MoveL p40, v1000, z50, tool1 \WObj:=wobj1;
WaitLoad load1;
%"routine_x"%;
UnLoad "HOME:/PART_A.MOD";
```

将普通程序模块PART_A.MOD从HOME:加载到程序内存中。与此同时，移动机械臂。随后，将新普通程序模块与程序任务相连，并调用模块PART_A中的程序routine_x。

变元

```
WaitLoad [\UnloadPath] [\UnloadFile] LoadNo [\CheckRef]
```

[\UnloadPath]

数据类型：string

将从程序内存卸载的文件的路径和名称。当使用参数\UnloadFile时，应当排除文件名称。

[\UnloadFile]

数据类型：string

当参数\UnloadPath中排除文件名称时，则必须通过该参数来进行定义。

LoadNo

数据类型：loadsession

此为由指令StartLoad创建的加载会话引用，其需要将已加载的普通程序模块与程序任务相连。

下一页继续

1.318 WaitLoad - 将加载的模块与任务相连

RobotWare - OS

续前页

[\CheckRef]

数据类型：switch

在加载模块后，检查程序任务中未解决的参考。如未使用，则不对未解决的参考进行检查。

程序执行

指令WaitLoad将首先等待完成加载，随后，模块将得以链接和初始化。已加载模块的初始化将所有模块等级的变量设置为其初始值。

如果未使用参数\CheckRef，则针对加载操作StartLoad - WaitLoad，将始终接受未解决的引用，但是在执行未解决的引用期间，其将成为运行时错误。

如作规定，本系统将以卸载操作开始。如果模块卸载失败，则将不会加载新的模块。

如果加载操作出现任何错误，包括使用开关\CheckRef时的未解决参考，则程序内存中已加载的模块将不再有效。

为取得易于理解和维持的良好程序结构，应当从主模块完成普通程序模块的所有加载和卸载，且执行期间，其始终存在于程序内存中。

关于将包含主要无返回值程序的程序加载至主要程序（通过另一个main无返回值程序），请参见指令Load。

更多示例

有关于指令WaitLoad的更多例子阐述如下。

例 1

```
StartLoad "HOME:/DOORDIR/DOOR2.MOD", load1;
...
WaitLoad \UnloadPath:="HOME:/DOORDIR/DOOR1.MOD", load1;
```

将来自路径DOORDIR中HOME:的普通程序模块DOOR2.MOD加载到程序内存中，并将新模块与任务相连。将从程序内存卸载普通程序模块DOOR1.MOD。

例 2

```
StartLoad "HOME:" \File:="DOORDIR/DOOR2.MOD", load1;
! The robot can do some other work
WaitLoad \UnloadPath:="HOME:" \File:= "DOORDIR/DOOR1.MOD", load1;
```

其与以下指令相同，但是在加载期间，机械臂可进行一些其他工作，且同时以更快的速度进行（仅一个链接代替以下两个链接）。

```
Load "HOME:" \File:="DOORDIR/DOOR2.MOD";
UnLoad "HOME:" \File:="DOORDIR/DOOR1.MOD";
```

错误处理

执行WaitLoad时，如果无法发现StartLoad指令中指定的文件，则将系统变量ERRNO设置为ERR_FILNOTFND。

如果读取待加载文件存在一些其他类型的问题，则将系统变量ERRNO设置为ERR_IOERROR。

如果参数LoadNo指代未知的加载会话，则将系统变量ERRNO设置为ERR_UNKPROC。

如果由于程序内存已满而无法加载模块，则将系统变量ERRNO设置为ERR_PRGMEMFULL。

如果已经将模块加载到程序内存中，则将系统变量ERRNO设置为ERR_LOADED。

下一页继续

如果已加载的模块包含语法错误，则将系统变量ERRNO设置为ERR_SYNTAX。

如果已加载的模块引起致命链接错误，则将系统变量 ERRNO设置为ERR_LINKREF。

如果将WaitLoad同开关\CheckRef一同使用，以检查任何引用错误，且程序内存包含未解决的引用，则将系统变量ERRNO设置为ERR_LINKREF。

仅当参数\UnloadPath用于指令WaitLoad中时，会出现以下错误：

- 如果由于模块内的持续执行而无法卸载参数\UnloadPath 中指定的模块，则将系统变量ERRNO设置为ERR_UNLOAD。
- 如果由于未通过来自RAPID程序的Load或StartLoad-WaitLoad来加载普通程序模块，导致无法卸载参数\UnloadPath中指定的模块，则亦将系统变量ERRNO设置为ERR_UNLOAD。

随后，可通过ERROR处理器来处理此类错误。如果出现一些此类错误，则实际模块将得以卸载，且将不可用于ERROR处理器。

注意

RETRY无法用于恢复来自WaitLoad的任何错误。

限制

通过给相同模块加载有关实际PERS变量的新初始化值，不可能改变一些PERS变量的当前值。

例子：

- 在系统中加载文件my_module.mod以及声明PERS num my_pers:=1;。
- 使用新的永久值，例如，PERS num my_pers:=3;，在磁盘上编辑文件my_module.mod
- 执行以下代码。
- 再次加载my_module.mod后，my_pers的值仍然为1而非3。

```
StartLoad \Dynamic, "HOME:/my_module.mod", load1;
...
WaitLoad \UnLoadPath:="HOME:/my_module.mod", load1;
```

该限制源于PERS变量特征。加载期间，如果PERS变量正在使用当中，则PERS 变量的当前值将不会为新加载的PERS初始化值所改变。

如果代而执行以下代码，将不会出现上述问题：

```
UnLoad "HOME:/my_module.mod";
StartLoad \Dynamic, "HOME:/my_module.mod", load1;
...
WaitLoad load1;
```

另一个选项是使用有关初始化值的CONST，并在新模块中开始执行时进行以下任务：

```
my_pers := my_const;
```

语法

WaitLoad

```
[ '\' UnloadPath ':=' <expression (IN) of string> ',']
[ '\' UnloadFile ':=' <expression (IN) of string> ',']
[ LoadNo ':=' ] <variable (VAR) of loadsession>
[ '\' CheckRef ] ';'
```

下一页继续

1.318 WaitLoad - 将加载的模块与任务相连
RobotWare - OS
续前页

相关信息

信息，关于	请参阅
执行期间，加载普通程序模块	第*660*页的*StartLoad* - 执行期间，加载普通程序模块
加载会话	第*1427*页的*loadsession* - 程序加载会话
加载普通程序模块	第*312*页的*Load* - 执行期间，加载普通程序模块
卸载普通程序模块	第*843*页的*UnLoad* - 执行期间，卸载普通程序模块
取消普通程序模块的加载	第*63*页的*CancelLoad* - 取消模块加载
检查程序参考	第*97*页的*CheckProgRef* - 检查程序参考
通过后期绑定进行过程调用	技术参考手册 - *RAPID*语言概览

1.319 WaitRob - 等待直至达到停止点或零速度

手册用法

WaitRob（*Wait Robot*）等待，直至机械臂和外轴已达到停止点或零速度。

基本示例

以下实例介绍了指令WaitRob：

另请参阅第*885*页的更多示例

例 1

```
WaitRob \InPos;
```

程序执行进入等待，直至机械臂和外轴已达到停止点。

变元

```
WaitRob [\InPos] | [\ZeroSpeed]
```

[\InPos]

In Position

数据类型：switch

如果使用该参数，则机械臂和外轴必须在继续执行之前达到停止点（当前移动指令的ToPoint）。

[\ZeroSpeed]

Zero Speed

数据类型：switch

如果使用该参数，则在可以继续执行前，机械臂和外轴必须拥有零速度。

如果未输入参数\InPos或\ZeroSpeed，则将显示错误消息。

更多示例

有关于如何使用指令WaitRob的更多例子阐述如下。

例 1

```
PROC stop_event()
  WaitRob \ZeroSpeed;
  SetDO rob_moving, 0;
ENDPROC
```

本例子显示了在程序停止时执行的事件程序。当机械臂和外轴已经在程序停止后停止移动时，只要机械臂正在移动并设置为0，则数字信号输出信号rob_moving为1。

语法

```
WaitRob
  [ '\' InPos ] | [ '\' ZeroSpeed ] ';'
```

相关信息

信息，关于	请参阅
一般动作	技术参考手册 - *RAPID*语言概览，运动和*I/O*原理一节

下一页继续

1.319 WaitRob - 等待直至达到停止点或零速度

RobotWare - OS

续前页

信息，关于	请参阅
其他定位指令	技术参考手册 - *RAPID*语言概览，*RAPID*概要 - 运动一节
停止点数据的定义	第*1483*页的*stoppointdata - 停止点数据*

1.320 WaitSensor - 等待传感器连接

手册用法

WaitSensor连接至传感器机械单元起动窗口中的一个对象。

基本示例

有关于指令WaitSensor的基本例子阐述如下。

另请参阅第*888*页的更多示例

例 1

```
WaitSensor Ssync1;
```

本程序连接传感器起动窗口内对象队列中的第一个对象。如果起动窗口内不存在任何对象，则执行停止，并等待一个对象。

变元

```
WaitSensor MechUnit [\RelDist ][\PredTime][\MaxTime][\TimeFlag]
```

MechUnit

机械装置

数据类型：mecunit

运动中的机械单元与指令中的机械臂位置相关。

[\RelDist]

相对距离

数据类型：num

等待一个对象进入起动窗口，并超出本参数指定的距离。如果已经连接工件，则停止执行，直至对象通过给定的距离。如果对象已超越相对距离，则继续执行。

[\PredTime]

预测时间

数据类型：num

等待一个对象进入起动窗口，并超出本参数指定的距离。如果已经连接工件，则停止执行，直至对象通过给定的距离。如果对象已超越预测时间，则继续执行。

[\MaxTime]

最大时间

数据类型：num

允许的最长等待时间，以秒计。如果在实现传感器连接或\RelDist之前耗尽该时间，则将调用错误处理器，如果存在这样的情况，则采用错误代码ERR_WAIT_MAXTIME。如果不存在错误处理器，则将停止执行。

[\TimeFlag]

超时标志

数据类型：bool

如果在实现传感器连接或\RelDist之前耗尽最长允许等待时间，则包含该值的输出参数为TRUE。如果该参数包含在本指令中，则不将其视为耗尽最长时间时的错误。

如果MaxTime参数包含在本指令中，则忽略该参数。

下一页继续

1.320 WaitSensor - 等待传感器连接
Machine Synchronization
续前页

程序执行

如果起动窗口中不存在任何对象，则程序停止执行。如果存在一个对象，则将对象与传感器相连，并继续执行。

如果在连接时发出第二个WaitSensor指令，则返回一个错误，除非使用\RelDist可选参数。

更多示例

有关于指令的更多例子阐述如下。

例 1

```
WaitSensor Ssync1\RelDist:=500.0;
```

如果未连接，则等待对象进入起动窗口，随后，等待对象通过传感器上的500 mm点。

如果已经同对象相连，则等待对象通过500 mm。

例 2

```
WaitSensor Ssync1\RelDist:=0.0;
```

如果未连接，则等待起始窗口中的对象。

如果已经连接，则当对象已经通过0.0 mm时，继续执行。

例 3

```
WaitSensor Ssync1;
WaitSensor Ssync1\RelDist:=0.0;
```

第一个WaitSensor连接起动窗口中的对象。如果对象仍热保持连接，则第二个WaitSensor将立即返回，但是，如果先前的对象已通过最大距离或得以下降，则其将等待下一个对象。

例 4

```
WaitSensor Ssync1\RelDist:=0.5\PredTime:=0.1;
```

如果对象已通过0.5米，则WaitSensor将立即返回，否则，将等待对象达到*=RelDist - C1speed * PredTime*。此处目标是在开始新的移动指令之前，对延迟进行预测。

例 5

```
WaitSensor Ssync1\RelDist:=0.5\MaxTime:=0.1\TimeFlag:=flag1;
```

如果对象已通过0.5米，WaitSensor将立即返回，否则，将等待一个对象0.1秒。如果在这0.1期间，无对象通过0.5米，则本指令将随flag1 = TRUE返回。

限制

与起动窗口中第一个对象相连需要50 ms。一旦连接，第二个WaitSensor以及\RelDist可选参数将仅采用普通RAPID指令执行时间。

错误处理

如果在执行WaitSensor指令期间出现以下错误，则将设置系统变量ERRNO。随后，可通过错误处理器来处理此类错误。

ERR_CNV_NOT_ACT	未启用传感器。
ERR_CNV_CONNECT	已经连接WaitSensor指令。
ERR_CNV_DROPPED	已通过另一个任务使指令WaitSensor正在等待的对象下降（DSQC 354 Revision 2：一个已通过起动窗口的对象）。
ERR_WAIT_MAXTIME	本对象未及时到来，且不存在TimeFlag。

下一页继续

语法

```
WaitSensor
  [ MechUnit ':=' ] < variable (VAR) of mecunit >
  [ '\' RelDist ':=' < expression (IN) of num > ]
  [ '\' PredTime ':=' < expression (IN) of num > ]
  [ '\' MaxTime ':=' < expression (IN) of num > ]
  [ '\' TimeFlag ':=' < variable (VAR) of bool > ] ';'
```

相关信息

信息，关于	请参阅
将对象丢到传感器上	第*147*页的*DropSensor - 使物体落于传感器上*
与传感器同步	第*721*页的*SyncToSensor - 同步至传感器*
与传感器同步	第*721*页的*SyncToSensor - 同步至传感器*
Machine Synchronization	应用手册 - 控制器软件*IRC5*

1.321 WaitSyncTask - 在同步点等待其他程序任务

手册用法

WaitSyncTask用于在各程序中一特殊点处同步若干程序任务。各程序任务进入等待，直至所有程序任务已达到命名的同步点。

> **注意**
>
> 指令WaitSyncTask仅同步程序执行。为同步程序执行和机械臂移动，则WaitSyncTask前的Move指令必须为所有相关程序任务中的停止点。通过在所有相关程序任务中使用WaitsyncTask \Inpos ...，亦有可能同步程序执行和机械臂移动。

> **警告**
>
> 为实现安全同步功能，交会点（参数SyncID）在各程序任务中必须拥有唯一的名称。名称必须同时与应当在交会点中满足的所有程序任务相同。

基本示例

以下实例介绍了指令WaitSyncTask：

另请参阅第892页的更多示例

例 1

位于任务T_ROB1中的程序实例

```
PERS tasks task_list{2} := [ ["T_ROB1"], ["T_ROB2"] ];
VAR syncident sync1;

...
WaitSyncTask sync1, task_list;
...
```

例 2

位于任务T_ROB2中的程序实例

```
PERS tasks task_list{2} := [ ["T_ROB1"], ["T_ROB2"] ];
VAR syncident sync1;

...
WaitSyncTask sync1, task_list;
...
```

首先达到WaitSyncTask以及识别号sync1的程序任务进入等待，直至其他程序任务达到其WaitSyncTask以及相同的识别号sync1。随后，程序任务T_ROB1和T_ROB2继续执行。

变元

```
WaitSyncTask [\InPos] SyncID TaskList [\TimeOut]
```

[\InPos]

In Position

下一页继续

数据类型：switch

如果使用该指令，则在该程序任务开始等待其他程序任务达到WaitSyncTask指令中指定的交会点之前，机械臂和外轴必须静止。

SyncID

Synchronization identity

数据类型：syncident

指定同步（交会）点名称的变量。数据类型syncident为非值类型，其仅作为用于命名同步点的标识符。

必须定义变量，使其在所有协作程序任务中拥有相同的名称。建议始终定义各程序任务中的变量全局（VAR syncident ...）。

TaskList

数据类型：tasks

位于任务列表（数组）中的永久变量，指定应当在同步点中满足的程序任务的名称（string），其名称符合参数SyncID。

必须定义永久变量，使其在所有协作程序任务中拥有相同的名称和内容。建议始终在系统中定义变量全局（PERS tasks ...）。

[\TimeOut]

数据类型：num

等待其他程序任务达到同步点的最长时间。超时以秒计（分辨率为0.001s）。如果未指定该参数，则程序任务将永远等待。

如果在所有程序任务已达到同步点之前耗尽时间，则将调用错误处理器。如果存在这样的情况，则其错误代码为ERR_WAITSYNCTASK。如果不存在错误处理器，则将停止执行。

程序执行

实际程序任务将在WaitSyncTask等待，直至TaskList中的其他程序任务已达到相同的SyncID点。此时，相应程序任务将继续执行其下一指令。

通过移动指令之间的角区域，可在移动指令之间编程WaitSyncTask。根据执行中程序任务之间的时序平衡，本系统可以：

- 在最佳时机，维持所有角区域。
- 在最糟糕时机，仅维持最终实现WaitSyncTask的程序任务的角区域。对于其他程序任务，其将产生停止点。

出于FlexPendant示教器-任务选择面板的测试目的而排除程序任务是可能的。

可使用以下原则：

- 原则1）：从main开始前（在设置PP到main之后），排除始终来自任务选择面板的程序任务周期。在整个程序周期内，该分离将有效。
- 原则2）：在程序周期中一些WaitSyncTask 指令之间，排除临时来自任务选择面板的程序任务。本系统将仅允许其他相关任务，但是，在出现错误消息时，会强制用户在通过合作的WaitSyncTask前，连接已排除的程序任务。
- 原则3）：如果根据原则2允许，则可能排除来自任务选择面板的一些程序任务的永久周期，从而通过执行服务程序SkipTaskExec，进一步根据原则1来运行。

下一页继续

1.321 WaitSyncTask - 在同步点等待其他程序任务

Multitasking

续前页

注意，当在同步移动中运行系统时，任务选择面板锁定。

更多示例

有关于指令WaitSyncTask的更多例子阐述如下。

例 1

位于任务T_ROB1中的程序实例

```
PERS tasks task_list{2} := [ ["T_ROB1"], ["T_ROB2"] ];
VAR syncident sync1;

...
WaitSyncTask \InPos, sync1, task_list \TimeOut := 60;
...
ERROR
  IF ERRNO = ERR_WAITSYNCTASK THEN
    RETRY;
  ENDIF
```

通过指令WaitSyncTask，程序任务T_ROB1进入等待，直至其机械单元就位，此后，其等待程序任务T_ROB2达到具有相同识别号的同步点。在等待60 s后，同于与ERR_WAITSYNCTASK等效的ERRNO来调用错误处理器。随后，再次调用指令WaitSyncTask，使其额外工作60 s。

错误处理

如果因为WaitSyncTask未及时就绪而出现超时，则将系统变量ERRNO设置为ERR_WAITSYNCTASK。

可用ERROR处理器来处理该错误。

限制

如果该指令位于移动指令之后，则必须使用停止点（zonedata fine）而非飞越点来编程移动指令，否则，将无法在电源故障后重启。

无法在与任意下列特殊系统事件关联的RAPID程序中执行WaitSyncTask \InPos：PowerOn、Stop、QStop,、Restart或者Step。

语法

```
WaitSyncTask
  ['\' InPos ',']
  [ SyncID ':=' ] < variable (VAR) of syncident> ','
  [ TaskList ':=' ] < persistent array {*} (PERS) of tasks>
  [ '\' TimeOut ':=' < expression (IN) of num > ] ';'
```

相关信息

信息，关于	请参阅
指定协作的程序任务	第*1500*页的*tasks - RAPID*程序任务
同步点的识别号	第*1496*页的*syncident* - 同步点的识别号

1.322 WaitTestAndSet - 等待，直至变量FALSE，随后设置

手册用法

WaitTestAndSet指令等待指定bool永久变量值成为FALSE。当变量值成为FALSE时，本指令将设置值为TRUE，并继续执行。永久变量可用作有关同步和互斥的二元信号。

该指令拥有与TestAndSet函数相同的基本功能，但是，只要TestAndSet指令立即终止时，bool为FALSE，则WaitTestAndSet进入等待。

不建议在软中断程序、撤销处理器或事件程序中使用WaitTestAndSet指令。

同时需要访问保护的资源实例：

- 并行执行时，一些RAPID程序的使用会产生函数问题。
- FlexPendant示教器的使用-运算符日志。

基本示例

以下实例介绍了指令WaitTestAndSet：

另请参阅第*894*页的更多示例

例 1

MAIN程序任务：

```
PERS bool tproutine_inuse := FALSE;
...
WaitTestAndSet tproutine_inuse;
TPWrite "First line from MAIN";
TPWrite "Second line from MAIN";
TPWrite "Third line from MAIN";
tproutine_inuse := FALSE;
```

BACK1程序任务：

```
PERS bool tproutine_inuse := FALSE;
...
WaitTestAndSet tproutine_inuse;
TPWrite "First line from BACK1";
TPWrite "Second line from BACK1";
TPWrite "Third line from BACK1";
tproutine_inuse := FALSE;
```

为避免运算符日志中的各行相混（一行来自MAIN，一行来自BACK1），WaitTestAndSet函数的使用可保证各任务的全部三行均不会分离。

如果程序任务MAIN首先采用信号WaitTestAndSet(tproutine_inuse)，则程序任务BACK1必须等待，直至程序任务MAIN已留下信号。

变元

```
WaitTestAndSet Object
```

Object

数据类型：bool

将用户定义的数据对象用作信号。数据对象必须为永久变量PERS。如果将WaitTestAndSet用于不同的程序任务之间，则对象必须为全局PERS。

下一页继续

1.322 WaitTestAndSet - 等待，直至变量FALSE，随后设置

RobotWare - OS

续前页

程序执行

该指令将通过一个不可分割的步骤来检查和设置用户定义的永久变量，如以下代码实例：

- 如果其拥有值FALSE，则将其设置为TRUE
- 如果其拥有值TRUE,，则等待，直至其成为FALSE，随后，将其设置为TRUE

```
IF Object = FALSE THEN
  Object := TRUE;
ELSE
  ! Wait until it become FALSE
  WaitUntil Object = FALSE;
  Object := TRUE;
ENDIF
```

此后，指令就绪。为避免出现问题，因为永久变量维持其值，因此，如果程序指针PP移动到main，则始终将信号对象设置为START事件程序中的FALSE。

更多示例

有关于指令WaitTestAndSet的更多例子阐述如下。

例 1

```
PERS bool semPers:= FALSE;
...
PROC doit(...)
  WaitTestAndSet semPers;
  ...
  semPers := FALSE;
ENDPROC
```

注意

如果在程序doit中停止程序执行，且程序指针移动至main，则将不会重置变量semPers。为避免这种情况，重置变量semPers为START事件程序中的FALSE。

语法

```
WaitTestAndSet
  [ Object ':=' ] < persistent (PERS) of bool> ';'
```

相关信息

信息，关于	请参阅
未设置时，测试变量并予以设置（通过等待时间来查询的类型）	第1272页的*TestAndSet - 未设置时，测试变量并予以设置*

1.323 WaitTime - 等待给定的时间

手册用法

WaitTime 用于等待给定的时间。该指令亦可用于等待，直至机械臂和外轴静止。

基本示例

以下实例介绍了指令WaitTime：

亦请参见下文中的第*895*页的更多示例。

例 1

```
WaitTime 0.5;
```

程序执行等待0.5秒。

变元

```
WaitTime [\InPos] Time
```

[\InPos]

In Position

数据类型：switch

如果使用该参数，则在开始统计等待时间之前，机械臂和外轴必须静止。如果本任务控制机械单元，则仅可使用该参数。

Time

数据类型：num

程序执行等待的最短时间（以秒计）为0 s。最长时间不受限制。分辨率为0.001 s。

程序执行

程序执行暂时停止规定的时间。中断处理和其他类似的函数，否则，其仍然有效。

在手动模式下，在等待3 s以上时间后，警告框将弹出，询问是否想要模拟指令。如果不想警告框出现，那么您可将系统参数*SimulateMenu*设为NO，请参见技术参考手册 - 系统参数主题*Controller*类型*General RAPID*。

更多示例

有关于如何使用指令WaitTime的更多例子阐述如下。

例 1

```
WaitTime \InPos,0;
```

程序执行进入等待，直至机械臂和外轴已静止。

限制

参数\Inpos不能与*SoftServo*一同使用。

如果在该指令之前采用Move指令，则必须通过停止点（zonedata fine）而非飞越点来编程Move指令。否则，不可能在电源故障后重启。

无法在与任意下列特殊系统事件关联的RAPID程序中执行WaitTime \Inpos：PowerOn、Stop、QStop、Restart或者Step。

下一页继续

1.323 WaitTime - 等待给定的时间

RobotWare - OS

续前页

语法

```
WaitTime
  ['\' InPos ',']
  [ Time ':='] <expression (IN) of num> ';'
```

相关信息

信息，关于	请参阅
等待直至满足条件	第*897*页的*WaitUntil - 等待直至满足条件*
等待直至已设置/重置I/O	第*861*页的*WaitDI - 等待直至已设置数字信号输入信号*

1.324 WaitUntil - 等待直至满足条件

手册用法

WaitUntil用于等待，直至满足逻辑条件；例如，其可以等待，直至已设置一个或多个输入。

基本示例

以下实例介绍了指令WaitUntil：

另请参阅第*899*页的更多示例

例 1

```
WaitUntil di4 = 1;
```

仅在已设置di4输入后，继续程序执行。

变元

```
WaitUntil [\InPos] Cond [\MaxTime] [\TimeFlag] [\PollRate]
     [\Visualize] [\Header] [\Message] | [\MsgArray] [\Wrap]
     [\Icon] [\Image] [\VisualizeTime]
```

[\InPos]

In Position

数据类型：switch

如果使用该参数，则机械臂和外轴必须在继续执行之前达到停止点（当前移动指令的ToPoint）。如果本任务控制机械单元，则仅可使用该参数。

Cond

数据类型：bool

将等待的逻辑表达式。

[\MaxTime]

数据类型：num

允许的最长等待时间，以秒计。如果在设置条件之前耗尽该时间，则将调用错误处理器，如果存在这样的情况，则采用错误代码ERR_WAIT_MAXTIME。如果不存在错误处理器，则将停止执行。

[\TimeFlag]

Timeout Flag

数据类型：bool

如果在满足条件之前耗尽最长允许时间，则包含该值的输出参数为TRUE。如果该参数包含在本指令中，则不将其视为耗尽最长时间时的错误。如果MaxTime参数不包括在本指令中，则将忽略该参数。

[\PollRate]

Polling Rate

数据类型：num

核实参数Cond的条件是否为TRUE时的回报率。这意味着WaitUntil首先马上核实条件，如果并非为TRUE，那么将在一段指定时间后，直至为TRUE。最低回报率值为0.004 s。如果未采用此参数，那么默认回报率设为0.1 s。

下一页继续

1.324 WaitUntil - 等待直至满足条件
RobotWare - OS
续前页

[\Visualize]

数据类型：switch

如被选中，将激活可视化。可视化包括未满足的逻辑条件的消息框、图标、页眉、消息行，将按编程参数来显示图像。

[\Header]

数据类型：string

要写在消息框顶部的页眉文字。最大40个字符。如果没有使用\Header 参数，则将显示默认消息。

[\Message]

数据类型：string

在显示器上可写入的一行文字。最多50字符。

[\MsgArray]

(Message Array)

数据类型：string

显示器上需编写的来自数组的若干文本行。同时只可采用参数\Message或\MsgArray中的其中一项。

最大布局空间是5 行，每行50字符。

[\Wrap]

数据类型：switch

如果选择，则参数\MsgArray中的所有指定字符串均将连接到一个字符串，各单独字符串之间仅存在单间距，并以尽可能少的行展开。

参数\MsgArray的各字符串将默认显示在显示器的单独各行。

[\Icon]

数据类型：icondata

定义有待显示的图标。仅可使用一种icondata型预定义图标。参见第*1411*页的预定义数据。

默认没有图标。

[\Image]

数据类型：string

应采用的图像的名称。为了开启您自己的图像，必须将图像放置在活跃系统的*HOME:*目录或直接放置在活跃系统中。

建议将文件放置在*HOME:*目录，从而在完成备份和恢复的情况下，可保存文件。

要求重启，然后，FlexPendant将加载图像。

系统命令是采用RobotWare附加功能*FlexPendant Interface*。

显示的图像可能为185像素宽度和300像素高度。如果图像更大，那么，只会从图像左上侧开始显示图像的185 * 300像素部分。

在图像可为多大尺寸或FlexPendant可加载多少图像方面，没有确切值。这取决于加载到FlexPendant的其他文件的大小。仅仅在采用未加载到FlexPendant的图像时，程序执行才会继续。

下一页继续

[\VisualizeTime]

数据类型：num

FlexPendant上应出现消息框前的等待时间。如果采用参数\VisualizeTime和\MaxTime，那么，参数\MaxTime采用的时间需大于参数\VisualizeTime采用的时间。

不采用参数\VisualizeTime情况下，可视化的默认时间为5s。最小值为1s。最大值没有限制。分辨率为0.001s。

程序执行

如果在执行WaitUntil指令时未满足编程条件，则每100 ms（或根据参数Pollrate中的规定值），再次对条件进行检查。

当机械臂正在等待时，对时间进行监督，且如果其超出最长时间值，则程序将在指定TimeFlag时继续，否则将会引起错误。如果指定TimeFlag，则在超出时间时设置为TRUE。否则，其将设置为假。

在手动模式下，在等待3 s以上时间后，警告框将弹出，询问是否想要模拟指令。如果不想警告框出现，那么您可将系统参数*SimulateMenu*设为NO，请参见技术参考手册 - 系统参数主题*Controller*类型*General RAPID*。

如果采用开关\Visualize，那么按编程参数，将在FlexPendant上显示消息框。如果未采用\Header参数，那么将显示默认页眉文本。当WaitUntil指令的执行准备就绪时，消息框将从FlexPendant撤除。

TRAP层级的新消息框将焦点从基础层级的消息框移开。

更多示例

有关于如何使用指令WaitUntil的更多例子阐述如下。

例 1

```
VAR bool timeout;
WaitUntil start_input = 1 AND grip_status = 1\MaxTime := 60
     \TimeFlag := timeout;
IF timeout THEN
  TPWrite "No start order received within expected time";
ELSE
  start_next_cycle;
ENDIF
```

如果未在60秒以内满足两个输入条件，则将在FlexPendant示教器的显示器上写入一条错误消息。

例 2

```
WaitUntil \Inpos, di4 = 1;
```

程序执行进入等待，直至机械臂已静止，且已设置di4输入。

例 3

```
WaitUntil di4 = 1 \MaxTime:=5.5;
..
ERROR
  IF ERRNO = ERR_NORUNUNIT THEN
    TPWrite "The I/O unit is not running";
    TRYNEXT;
```

下一页继续

1.324 WaitUntil - 等待直至满足条件

RobotWare - OS

续前页

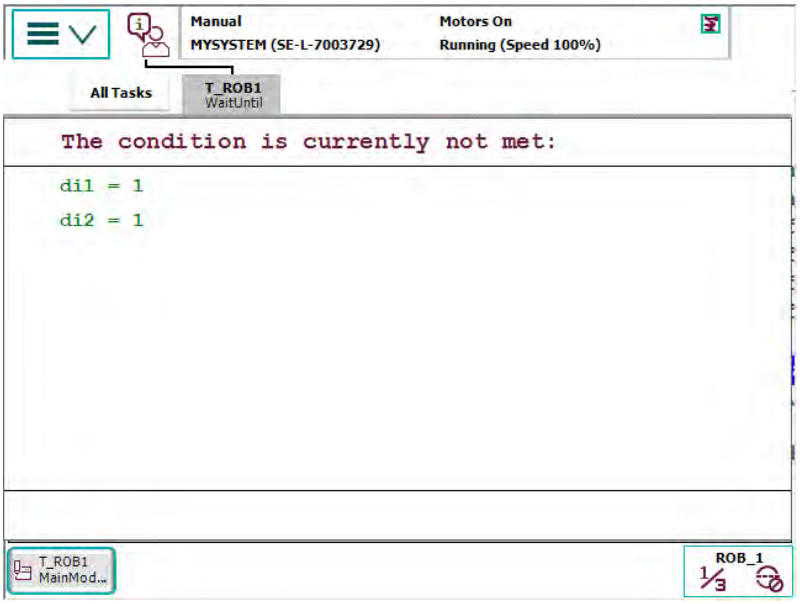
```
  ELSEIF ERRNO = ERR_WAIT_MAX THEN
    RAISE;
  ELSE
    Stop;
  ENDIF
```

程序执行进入等待，直至已设置di4输入。如果已禁用I/O单元，或等待时间到期，则通过错误处理器继续执行。

例 4

```
WaitUntil di1 = 1 AND di2 = 1 \MaxTime := 60 \Visualize;
..
  ERROR
    IF ERRNO = ERR_WAIT_MAX THEN
      RAISE;
    ELSE
      Stop;
    ENDIF
```

如果在5秒内未满足两个输入条件，那么将在FlexPendant示教器的显示器上编写消息。如果在60秒内未满足条件，那么将在错误处理器中继续执行。



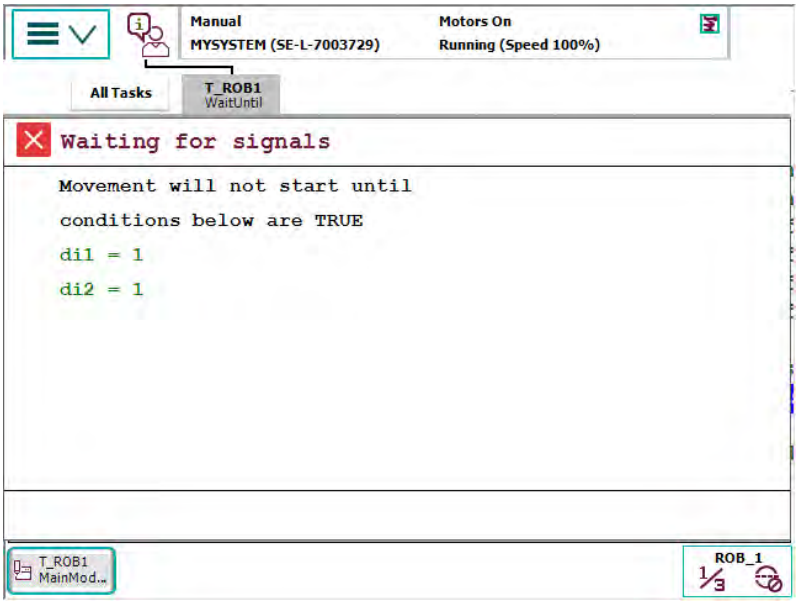
xx1600000146

例 5

```
WaitUntil di1 = 1 AND di2 = 1 \Visualize \Header:="Waiting for
     signals" \MsgArray:=["Movement will not start until",
     "conditions below are TRUE"] \Icon:=iconError;
MoveL p40, v500, z20, L10tip;
..
```

下一页继续

如果有两个输入条件不符，则在可选参数\Header和\MsgArray将被写入FlexPendant示教器屏幕上，未满足条件的家族要随行。



xx1600000147

错误处理

信号变量是在RAPID中声明的变量。其尚未与通过指令AliasIO而在I/O配置中确定的I/O信号相连。如果使用该信号，则将系统变量ERRNO设置为ERR_NO_ALIASIO_DEF，并用错误处理器继续执行。

如果在条件中使用了一个信号，且没有同I/O单元接触，则将系统变量ERRNO设置为ERR_NORUNUNIT，并用错误处理器继续执行。

随后，可通过错误处理器来处理此类情形。

如果在条件改变为正确值之前超时（参数\MaxTime），则将系统变量ERRNO设置为ERR_WAIT_MAXTIME，并用错误处理器继续执行。

随后，可通过错误处理器来处理此类情形。

限制

参数\Inpos不能与*SoftServo*一同使用。

如果在该指令之前采用Move指令，则必须通过停止点（zonedata fine）而非飞越点来编程Move指令。否则，不可能在电源故障后重启。

无法在与任意下列特殊系统事件关联的RAPID程序中执行WaitUntil \Inpos：PowerOn、Stop、QStop、Restart或者Step。

WaitUntil \Inpos无法与StopMove一同使用，以探测是否已停止移动。在这种情况下，可永远暂停WaitUntil指令。其并未探测到移动已停止，其探测到机械臂和外轴已达到最后编程的ToPoint（MoveX、SearchX、TriggX）。

语法

```
WaitUntil
  ['\' InPos ',']
  [Cond ':='] <expression (IN) of bool>
```

下一页继续

1.324 WaitUntil - 等待直至满足条件

RobotWare - OS

续前页

```
['\' MaxTime ':=' <expression (IN) of num>]
['\' TimeFlag ':=' <variable (VAR) of bool>]
['\' PollRate ':=' <expression (IN) of num>]
['\' Visualize]
['\' Header ':=' <expression (IN) of string>]]
['\' Message ':=' <expression (IN) of string>]
| ['\' MsgArray ':=' <array {*} (IN) of string>]
['\' Wrap]
['\' Icon ':=' <expression (IN) of icondata>]
['\' Image ':=' <expression (IN) of string>]
['\' VisualizeTime ':=' <expression (IN) of num>] ';'
```

相关信息

信息，关于	请参阅
等待直至已设置/重置输入	第*861*页的*WaitDI* - 等待直至已设置数字信号输入信号
等待给定的时间	第*895*页的*WaitTime* - 等待给定的时间
表达式	技术参考手册 - *RAPID*语言概览

1.325 WaitWObj - 等待传送带上的工件

手册用法

WaitWObj（*Wait Work Object*）连接至传送带机械单元起动窗口中的一个工件。

基本示例

以下实例介绍了指令WaitWObj：

另请参阅第*904*页的更多示例

例 1

```
WaitWObj wobj_on_cnv1;
```

本程序连接传送带起动窗口内对象队列中的第一个对象。如果起动窗口内不存在任何对象，则执行等待一个对象。

变元

```
WaitWObj WObj [ \RelDist ][\MaxTime][\TimeFlag]
```

WObj

Work Object

数据类型： wobjdata

运动中的工件（坐标系）与指令中的机械臂位置相关。通过工件中的ufmec，指定机械单元传送带。

[\RelDist]

Relative Distance

数据类型：num

等待一个对象进入起动窗口，并超出本参数指定的距离。如果已经连接工件，则执行等待，直至对象通过给定的距离。如果对象已通过\RelDist，则继续执行。

[\MaxTime]

Maximum Time

数据类型：num

允许的最长等待时间，以秒计。如果在实现对象连接或\Reldist之前耗尽该时间，则将调用错误处理器，如果存在这样的情况，则采用错误代码ERR_WAIT_MAXTIME。如果不存在错误处理器，则将停止执行。

[\TimeFlag]

Timeout Flag

数据类型：bool

如果在实现对象连接或\Reldist之前，耗尽最长允许等待时间，则包含该值的输出参数为TRUE。如果该参数包含在本指令中，则不将其视为耗尽最长时间时的错误。如果MaxTime参数不包括在本指令中，则将忽略该参数。

程序执行

如果起动窗口中不存在任何对象，则程序执行进入等待。如果存在一个工件，则将对象与传送带相连，并继续执行。

下一页继续

1.325 WaitWObj - 等待传送带上的工件

Conveyor Tracking

续前页

如果在连接时发出第二个WaitWObj指令，则返回一个错误，除非使用\RelDist可选参数。

更多示例

有关于指令WaitWObj的更多例子阐述如下。

例 1

```
WaitWObj wobj_on_cnv1\RelDist:=500.0;
```

如果未连接，则等待对象进入起动窗口，随后，等待对象通过传送带上的500 mm点。

如果已经同对象相连，则等待对象通过500 mm。

如果未连接，则等待起始窗口中的对象。

例 2

```
WaitWObj wobj_on_cnv1\RelDist:=0.0;
```

如果已经连接，则当对象已经通过0.0 mm时，继续执行。

例 3

```
WaitWObj wobj_on_cnv1;
WaitWObj wobj_on_cnv1\RelDist:=0.0;
```

第一个WaitWObj连接起动窗口中的对象。如果对象仍热保持连接，则第二个WaitWObj将立即返回，但是，如果先前的对象已通过最大距离或得以下降，则其将等待下一个对象。

例 4

```
WaitWObj wobj_on_cnv1\RelDist:=500.0\MaxTime:=0.1 \Timeflag:=flag1;
```

如果对象已通过500 mm，WaitWobj将立即返回，否则，将等待一个对象0.1秒。如果在这0.1期间，无对象通过500 mm，则当标志1 = TRUE时，本指令将返回。

限制

与起动窗口中第一个对象相连需要50 ms。一旦连接，第二个WaitWObj以及\RelDist可选参数将仅采用普通RAPID指令执行时间。

错误处理

如果在执行WaitWobj指令期间出现以下错误，则将设置系统变量ERRNO。随后，可通过错误处理器来处理此类错误。

ERR_CNV_NOT_ACT	未启用传送带。
ERR_CNV_CONNECT	已经连接WaitWobj指令。
ERR_CNV_DROPPED	已通过另一个任务使指令WaitWobj正在等待的对象下降（DSQC 354版本2：一个已通过起动窗口的对象）。
ERR_WAIT_MAXTIME	本对象未及时到来，且不存在Timeflag。

语法

```
WaitWObj
  [ WObj ':='] < persistent (PERS) of wobjdata> ';'
  [ '\' RelDist ':=' < expression (IN) of num > ]
  [ '\' MaxTime ':=' <expression (IN) of num>]
  [ '\' TimeFlag ':=' <variable (VAR) of bool>] ';
```

下一页继续

相关信息

信息，关于	请参阅
工件降至传送带上	第*148*页的*DropWObj* - 使工件落于传送带上
传送带跟踪	应用手册 - 传送带跟踪

1.326 WarmStart - 重启控制器

手册用法

WarmStart用于重启控制器。

通过指令WriteCfgData，可以从RAPID改变系统参数。必须重启控制器，从而进而改变，以影响一些系统参数。可通过该指令WarmStart，完成重启。

基本示例

以下实例介绍了指令WarmStart：

例 1

```
WriteCfgData "/MOC/MOTOR_CALIB/rob1_1","cal_offset",offset1;
WarmStart;
```

将num 变量offset1的值作为rob1上轴1的校准偏移量写入，并使控制器重启。

程序执行

Warmstart立即起效，并将系统指针设置到下一个指令。

语法

```
WarmStart ';'
```

相关信息

信息，关于	请参阅
写入系统参数的属性	第*919*页的*WriteCfgData - 写入系统参数的属性*
配置	技术参考手册 - 系统参数
Advanced RAPID	产品规格 - 控制器软件*IRC5*

1.327 WHILE - 只要...便重复

手册用法

只要给定条件表达式评估为TRUE值，当重复一些指令时，使用WHILE。

提示

如果可能确定重复的数量，则可以使用FOR指令。

基本示例

以下实例介绍了指令WHILE：

例 1

```
WHILE reg1 < reg2 DO
  ...
  reg1 := reg1 + 1;
ENDWHILE
```

只要reg1 < reg2，则重复WHILE块中的指令。

变元

```
WHILE Condition DO ... ENDWHILE
```

Condition

数据类型：bool

必须评估为TRUE的条件为用以满足待执行WHILE块中指令的值。

程序执行

1 评估条件表达式。如果表达式评估为TRUE值，则执行WHILE块中的指令。
2 随后，再次评估条件表达式，且如果该评估结果为TRUE，则再次执行WHILE块中的指令。
3 该过程继续，直至表达式评估结果成为FALSE。

随后，终止迭代，并在WHILE块后，根据本指令，继续程序执行。

如果表达式评估结果在开始时为FALSE，则不执行WHILE块中的指令，且程序控制立即转移至WHILE块后的指令。

语法

```
WHILE <conditional expression> DO
  <statement list>
ENDWHILE
```

相关信息

信息，关于	请参阅
表达式	技术参考手册 - *RAPID*语言概览，基本特征 - 表达式一节
重复给定的次数	第*205*页的*FOR* - 重复给定的次数

1.328 WorldAccLim - 控制世界坐标系中的加速度

手册用法

WorldAccLim（*World Acceleration Limitation*）用于限制世界坐标系中工具（和有效负载）的加速度/减速度。

将在实际工具、实际有效负载（如存在）的中心点以及机械臂安装法兰中一起达到限制。

基本示例

以下实例介绍了指令WorldAccLim：

例 1

```
WorldAccLim \On := 3.5;
```

将加速度限制为3.5 m/s^2。

例 2

```
WorldAccLim \Off;
```

将加速度重置为最大（默认）。

变元

```
WorldAccLim [\On]|[\Off]
```

[\On]

数据类型：num

加速度限制的绝对值，以m/s^2计。

[\Off]

数据类型：switch

最大加速度（默认）。

程序执行

加速度限制适用于下一个执行机械臂运动的命令，其始终有效，直至执行新的WorldAccLim指令。

自动设置最大加速度（WorldAccLim \Off）

- 当使用重启模式重置RAPID时
- 当加载一则新程序或一个新模块时
- 当从起点开始执行程序时
- 当将程序指针移动到main时
- 当将程序指针移动到子程序时
- 移动程序指针造成执行顺序丢失时

建议仅使用一种加速度限制。如果完成指令WorldAccLim、AccSet和PathAccLim的组合，则系统通过以下顺序降低加速度/减速度：

- 根据WorldAccLim
- 根据AccSet
- 根据PathAccLim

下一页继续

限制

本指令仅可用于主任务T_ROB1，或者如果在*MultiMove*系统中，则可用于运动任务中。

容许的最小加速度为0.1 m/s^2。

不支持以下机械臂模型，且无法使用WorldAccLim指令：

- IRB 340, IRB 360, IRB 540, IRB 1400, IRB 1410

错误处理

如果将参数On设置为一个过低的值，则将系统变量ERRNO设置为ERR_ACC_TOO_LOW。随后，可用错误处理器对该错误进行处理。

语法

```
WorldAccLim
  [ '\' On ':=' <expression (IN) of num>] | ['\' Off ] ';'
```

相关信息

信息，关于	请参阅
定位器指令	技术参考手册 - *RAPID*语言概览
运动设置数据	第*1430*页的*motsetdata - 运动设置数据*
加速度的降低	第*19*页的*AccSet – 降低加速度*
路径沿线的加速度限制	第*428*页的*PathAccLim - 降低路径沿线的TCP*加速度

1.329 Write - 写入到基于字符的文件或串行通道

手册用法

Write用于写入基于字符的文件或串行通道。可将特定数据的值同文本一样写入。

基本示例

以下实例介绍了指令Write：

另请参阅第*911*页的更多示例

例 1

```
Write logfile, "Execution started";
```

通过引用名称logfile，将文本Execution started写入文件。

例 2

```
VAR num reg1:=5;
...
Write logfile, "No of produced parts="\Num:=reg1;
```

通过引用名称 logfile，将文本No of produced parts=5写入文件。

变元

```
Write IODevice String [\Num] | [\Bool] | [\Pos] | [\Orient] |
     [\Dnum] [\NoNewLine]
```

IODevice

数据类型：iodev

当前文件或串行通道的名称（引用）。

String

数据类型：string

待写入的文本。

[\Num]

Numeric

数据类型：num

将在文本字符串后写入其数值的数据。

[\Bool]

Boolean

数据类型：bool

将在文本字符串后写入其逻辑值的数据。

[\Pos]

Position

数据类型：pos

将在文本字符串后写入其位置的数据。

[\Orient]

Orientation

下一页继续

数据类型：orient

将在文本字符串后写入其方位的数据。

[\Dnum]

Numeric

数据类型：dnum

将在文本字符串后写入其数值的数据。

[\NoNewLine]

数据类型：switch

省略通常指明文本结束的行进给字符，即将在相同的行上继续下一write指令。

程序执行

将文本字符串写入指定的文件或串行通道。同时写入行进给字符（LF），如果使用参数\NoNewLine，则可以省略。

如果使用参数\Num、\Bool、\Pos或\Orient之一，则在将其添加至第一个字符串之前，首先将其值转换为文本字符串。从值到文本字符串的转换如下所示：

变元	值	文本串
\Num	23	“23”
\Num	1.141367	“1.14137”
\Bool	TRUE	“TRUE”
\Pos	[1817.3,905.17,879.11]	"[1817.3,905.17,879.11]"
\Orient	[0.96593,0,0.25882,0]	“[0.96593,0,0.25882,0]"
\Dnum	4294967295	"4294967295"

通过标准RAPID格式，将值转换为字符串。这意味着，原则上为6个有效位。如果小数部分小于0.000005或大于0.999995，则将该数字四舍五入为整数。

上电失败重启时，将关闭系统中所有打开的文件或串行通道，并将重置iodev型变量中的I/O描述符。

更多示例

有关于指令Write的更多例子阐述如下。

例 1

```
VAR iodev printer;
VAR num reg1:=0;
VAR num stopprod_value:=0;
...
Open "com1:", printer\Write;
stopprod_value:=stopprod;
WHILE stopprod_value = 0 DO
  produce_part;
  reg1:=reg1+1;
  Write printer, "Produced part="\Num:=reg1\NoNewLine;
  Write printer, " "\NoNewLine;
  Write printer, CTime();
  stopprod_value:=stopprod;
```

下一页继续

1.329 Write - 写入到基于字符的文件或串行通道

RobotWare - OS

续前页

```
    ENDWHILE
    Close printer;
```

在各周期，将包含已生产部分和时间编号的一行输出到打印机。打印机与串行通道com1相连。打印出的消息如下所示：

Produced part=473	09:47:15

限制

参数\Num、\Dnum，\Bool、\Pos和\Orient互相排斥，因此，不可同时用于同一指令。

该指令仅可用于已为写入而打开的文件或串行通道。

错误处理

如果在写入期间出现错误，则将系统变量ERRNO设置为ERR_FILEACC。随后，可通过错误处理器来处理该错误。

语法

```
Write
  [ IODevice ':='] <variable (VAR) of iodev> ','
  [ String ':='] <expression (IN) of string>
  [ '\' Num ':=' <expression (IN) of num> ]
  | ['\' Bool ':=' <expression (IN) of bool> ]
  | ['\' Pos ':=' <expression (IN) of pos> ]
  | ['\' Orient ':=' <expression (IN) of orient> ]
  | ['\' Dnum ':=' <expression (IN) of dnum> ]
  [ '\' NoNewLine] ';'
```

相关信息

信息，关于	请参阅
打开文件或串行通道	技术参考手册 - *RAPID*语言概览
文件和串行通道处理	应用手册 - 控制器软件*IRC5*

1.330 WriteAnyBin - 将数据写入二进制串行通道或文件

手册用法

WriteAnyBin（*Write Any Binary*）用于将任意类型的数据写入二进制串行通道或文件。

基本示例

以下实例介绍了指令WriteAnyBin：

另请参阅第*914*页的更多示例

例 1

```
VAR iodev channel1;
VAR orient quat1 := [1, 0, 0, 0];
...
Open "com1:", channel1 \Bin;
WriteAnyBin channel1, quat1;
```

将orient数据quat1写入由channel1引用的通道。

变元

```
WriteAnyBin IODevice Data
```

IODevice

数据类型：iodev

为进行写入操作的二进制串行通道或文件的名称（引用）。

Data

数据类型：ANYTYPE

有待写入的数据。

程序执行

针对指定数据，将尽可能多的字节写入到指定的二进制串行通道或文件。

上电失败重启时，将关闭系统中所有打开的文件或串行通道，并将重置iodev型变量中的I/O描述符。

限制

该指令仅可用于为二进制写入而打开的串行通道或文件。

通过该指令WriteAnyBin而写入的数据，必须为数值数据类型，例如num、bool或string。亦可使用此类数值数据类型的记录、记录成分、数组或数组元素。无法使用半值或非值数据类型的整体数据集或部分数据集。

错误处理

如果在写入期间出现错误，则将系统变量ERRNO设置为ERR_FILEACC。随后，可通过错误处理器来处理该错误。

下一页继续

1.330 WriteAnyBin - 将数据写入二进制串行通道或文件

RobotWare - OS

续前页

更多示例

有关于指令WriteAnyBin的更多例子阐述如下。

例 1

```
VAR iodev channel;
VAR num input;
VAR robtarget cur_robt;

Open "com1:", channel\Bin;

! Send the control character enq
WriteStrBin channel, "\05";
! Wait for the control character ack
input := ReadBin (channel \Time:= 0.1);
IF input = 6 THEN
  ! Send current robot position
  cur_robt := CRobT(\Tool:= tool1\WObj:= wobj1);
  WriteAnyBin channel, cur_robt;
ENDIF

Close channel;
```

将机械臂的当前位置写入至二进制串行通道。

限制

因为WriteAnyBin-ReadAnyBin仅设计用于发送IRC5控制系统之间的内部控制器数据，未发布数据协议，且无法在任意 PC上解读数据。

> **注意**
>
> 控制软件开发可打破兼容性，因此，可能无法使用不同RobotWare软件版本之间的WriteAnyBin-ReadAnyBin。

语法

```
WriteAnyBin
  [ IODevice ':='] <variable (VAR) of iodev> ','
  [ Data ':='] <expression (IN) of ANYTYPE> ';'
```

相关信息

信息，关于	请参阅
打开串行通道或文件	技术参考手册 - *RAPID*语言概览
读取来自二进制串行通道或文件的数据	第*487*页的*ReadAnyBin* - 读取二进制串行通道或文件的数据
文件和串行通道处理	应用手册 - 控制器软件*IRC5*

1.331 WriteBin - 写入至一个二进制串行通道

手册用法

WriteBin用于将若干字节写入到二进制串行通道。

基本示例

以下实例介绍了指令WriteBin：

另请参阅第*915*页的更多示例

例 1

```
WriteBin channel2, text_buffer, 10;
```

将来自text_buffer列表的10个字符写入到由channel2引用的通道。

变元

```
WriteBin IODevice Buffer NChar
```

IODevice

数据类型：iodev

当前串行通道的名称（引用）。

Buffer

数据类型：array of num

包含待写入数字（字符）的列表（数组）。

NChar

Number of Characters

数据类型：num

从Buffer写入的字符数。

程序执行

列表中有待写入串行通道的数字（字符）的指定数量。

上电失败重启时，将关闭系统中所有打开的文件或串行通道，并将重置iodev型变量中的I/O描述符。

限制

该指令仅可用于为二进制写入而打开的串行通道。

错误处理

如果在写入期间出现错误，则将系统变量ERRNO设置为ERR_FILEACC。随后，可通过错误处理器来处理该错误。

更多示例

有关于如何使用指令WriteBin的更多例子阐述如下。

例 1

```
VAR iodev channel;
VAR num out_buffer{20};
VAR num input;
```

下一页继续

1.331 WriteBin - 写入至一个二进制串行通道

RobotWare - OS

续前页

```
VAR num nchar;
Open "com1:", channel\Bin;

out_buffer{1} := 5;!( enq )
WriteBin channel, out_buffer, 1;
input := ReadBin (channel \Time:= 0.1);

IF input = 6 THEN !( ack )
  out_buffer{1} := 2;!( stx )
  out_buffer{2} := 72;!( 'H' )
  out_buffer{3} := 101;!( 'e' )
  out_buffer{4} := 108;!( 'l' )
  out_buffer{5} := 108;!( 'l' )
  out_buffer{6} := 111;!( 'o' )
  out_buffer{7} := 32;!( ' ' )
  out_buffer{8} := StrToByte("w"\Char);!( 'w' )
  out_buffer{9} := StrToByte("o"\Char);!( 'o' )
  out_buffer{10} := StrToByte("r"\Char);!( 'r' )
  out_buffer{11} := StrToByte("l"\Char);!( 'l' )
  out_buffer{12} := StrToByte("d"\Char);!( 'd' )
  out_buffer{13} := 3;!( etx )
  WriteBin channel, out_buffer, 13;
ENDIF
```

信号交换（enq,ack）后，将文本字符串Hello world（以及相关的控制字符）写入到串行通道。在相同情况下使用函数StrToByte，以将字符串转换成byte（num）数据。

语法

```
WriteBin
  [ IODevice ':='] <variable (VAR) of iodev> ','
  [ Buffer ':='] <array {*} (IN) of num> ','
  [ NChar ':='] <expression (IN) of num> ';'
```

相关信息

信息，关于	请参阅
串行通道的打开等	技术参考手册 - *RAPID*语言概览
将一段字符串转换为一个字节数据	第*1262*页的*StrToByte - 将一段字符串转换为一个字节数据*
字节数据	第*1351*页的字节-整数值*0 - 255*
文件和串行通道处理	应用手册 - 控制器软件*IRC5*

1.332 WriteBlock - 将数据块写入设备

手册用法

WriteBlock用于将数据块写入同串行传感器接口相连的设备。从文件获得该数据。传感器接口与位于串行通道（使用RTP1传输层协议）上方的传感器进行通信。

此为关于传感器通道配置的实例。

```
COM_PHY_CHANNEL:
  Name "COM1:"
  Connector "COM1"
  Baudrate 19200
COM_TRP:
  Name "sen1:"
  Type "RTP1"
  PhyChannel "COM1"
```

基本示例

以下实例介绍了指令WriteBlock：

例 1

```
CONST string SensorPar := "flp1:senpar.cfg";
CONST num ParBlock:= 1;

! Connect to the sensor device "sen1:" (defined in sio.cfg).
SenDevice "sen1:";

! Write sensor parameters from flp1:senpar.cfg
! to sensor datablock 1.

WriteBlock "sen1:", ParBlock, SensorPar;
```

变元

```
WriteBlock device BlockNo FileName [ \TaskName ]
```

device

数据类型：string

在sio.cfg中配置了I/O设备名称，以供传感器使用。

BlockNo

数据类型：num

参数BlockNo用于在传感器中选择有待读取的数据块。

FileName

数据类型：string

参数FileName用于选择文件，将此类文件的数据写入到由BlockNo参数所选择的传感器中的数据块。

[\TaskName]

数据类型：string

参数TaskName使其可能访问其他RAPID任务中的设备。

下一页继续

1.332 WriteBlock - 将数据块写入设备

Sensor Interface

续前页

错误处理

错误常量 (ERRNO值)	描述
SEN_NO_MEAS	测量失败
SEN_NOREADY	传感器不能处理命令
SEN_GENERRO	一般性传感器错误
SEN_BUSY	传感器总线
SEN_UNKNOWN	未知的传感器
SEN_EXALARM	外部传感器错误
SEN_CAALARM	内部传感器错误
SEN_TEMP	传感器温度错误
SEN_VALUE	非法通信值
SEN_CAMCHECK	传感器检查失败
SEN_TIMEOUT	通信错误

语法

```
WriteBlock
  [ device ':=' ] < expression(IN) of string> ','
  [ BlockNo ':=' ] < expression (IN) of num > ','
  [ FileName ':=' ] < expression (IN) of string > ','
  [ '\' TaskName ':=' < expression (IN) of string > ] ';'
```

相关信息

信息，关于	请参阅
与传感器设备相连	第574页的*SenDevice - 与传感器设备相连*
写入传感器变量	第927页的*WriteVar - 写入变量*
读取传感器数据块	第490页的*ReadBlock - 读取设备的数据块*
传感器通信配置	技术参考手册 - 系统参数

1.333 WriteCfgData - 写入系统参数的属性

手册用法

WriteCfgData的作用是写入一个系统参数（配置数据）的一项属性。

除了写入已命名的参数外，同时有可能搜索和更新未命名的参数。

基本示例

以下例子阐明了指令WriteCfgData。此类例子均表明了如何写入已命名的参数数据。

例 1

```
VAR num offset1 := 1.2;
...
WriteCfgData "/MOC/MOTOR_CALIB/rob1_1","cal_offset",offset1;
```

在num中写入变量offset1，并写入有关rob_1.上轴1的校准偏移量。

例 2

```
VAR string io_device := "my_device";
...
WriteCfgData "/EIO/EIO_SIGNAL/process_error","Device",io_device;
```

在string中写入变量io_device,以及已确定信号process_error的I/O设备名称。

变元

```
WriteCfgData InstancePath Attribute CfgData [\ListNo]
```

InstancePath

数据类型： string

指定通往待访问参数的路径。

针对已命名的参数，该字符串的格式为/DOMAIN/TYPE/ParameterName.

针对未命名的参数，该字符串的格式为/DOMAIN/TYPE/Attribute/AttributeValue.

Attribute

数据类型：string

待写入参数的属性名称。

CfgData

数据类型：anytype

读取用于储存新数据的数据对象。根据属性类型，有些类型包括bool、num或string。

[\ListNo]

数据类型：num

包含待发现和更新Attribute +AttributeValue实例编号的变量。

首次出现的 Attribute+AttributeValue的实例编号为0。如果搜索更多的实例，则\ListNo中的返回值将递增1。否则，如果不再存在实例，则返回值将为-1。预定义常量END_OF_LIST可用于检查是否搜索更多的实例。

下一页继续

1.333 WriteCfgData - 写入系统参数的属性
RobotWare - OS
续前页

程序执行

根据由CfgData参数所指定的数据对象的值，设置由Attribute参数所指定的属性值。

如果使用InstancePath中的格式/DOMAIN/TYPE/ParameterName，则仅可访问已命名的参数，即第一属性为name、Name或NAME的参数。

针对未命名的参数，使用可选参数\ListNo，以指定写入属性值的实例。在每次成功写入后，将其更新为下一个可用于写入的实例。

更多示例

关于指令WriteCfgdata的更多例子，如下所示。此类例子均表明如何写入未命名的参数。

例 1

```
VAR num read_index;
VAR num write_index;
VAR string read_str;
...
read_index:=0;
write_index:=0;
ReadCfgData "/EIO/EIO_CROSS/Act1/do_13", "Res", read_str,
     \ListNo:=read_index;
WriteCfgData "/EIO/EIO_CROSS/Act1/do_13", "Res", "my"+read_str,
     \ListNo:=write_index;
```

读取有关未命名数字作用器信号do_13的总信号，并将名称置于字符串变量read_str中。随后，将名称更新为di_13，并采用前缀“my”。

在该例子中，域EIO拥有以下cfg代码：

EIO_CROSS:

-名称“Cross_di_1_do_2” -Res “di_1” -Act1 “do_2”

-名称“Cross_di_2_do_2” -Res “di_2” -Act1 “do_2”

-名称“Cross_di_13_do_13” -Res “di_13” -Act1 “do_13”

例 2

```
VAR num read_index;
VAR num write_index;
VAR string read_str;
...
read_index:=0;
write_index:=0;
WHILE read_index <> END_OF_LIST DO
  ReadCfgData "/EIO/EIO_SIGNAL/Device/USERIO", "Name", read_str,
       \ListNo:=read_index;
  IF read_index <> END_OF_LIST THEN
    WriteCfgData "/EIO/EIO_SIGNAL/Device/USERIO", "Name",
         "my"+read_str, \ListNo:=write_index;
  ENDIF
ENDWHILE
```

读取针对I/O设备USERIO而定义的所有信号的名称。改变信号的名称，以读取带有“my”前缀的名称。

下一页继续

在该例子中，域EIO拥有以下cfg代码：

```
EIO_SIGNAL:
  -Name "USERDO1" -SignalType "DO" -Device "USERIO" -DeviceMap "0"
  -Name "USERDO2" -SignalType "DO" -Device "USERIO" -DeviceMap "1"
  -Name "USERDO3" -SignalType "DO" -Device "USERIO" -DeviceMap "2"
```

错误处理

如果不可能发现由配置数据库中的“InstancePath + Attribute”所指定的数据，则将系统变量ERRNO设置为ERR_CFG_NOTFND。

如果参数CfgData的数据类型不等于配置数据库中“InstancePath + Attribute”所指定的已发现数据的实际数据类型，则将系统变量ERRNO设置为ERR_CFG_ILLTYPE.

如果有关参数CfgData的数据超出限制（最大值/最小值），则将系统变量ERRNO设置为ERR_CFG_LIMIT.

如果试图写入内部写保护的数据，则将系统变量ERRNO设置为ERR_CFG_INTERNAL.

如果在执行指令时，参数\ListNo中的变量具有超出可用实例（0 ... n）范围的值，则将ERRNO设置为ERR_CFG_OUTOFBOUNDS。

随后，可用错误处理器来处理此类错误。

限制

针对数据类型num的CfgData由RAPID程序单元（毫米、度、秒等）到系统参数单元（米、弧度、秒等）的转换，必须由RAPID程序中的用户来实施。

为了使改变生效，必须手动重启控制器，或执行指令WarmStart。

如果使用InstancePath中的格式/DOMAIN/TYPE/ParameterName，则仅可访问已命名的参数，即第一属性为name、Name或NAME的参数。

将RAPID字符串限制在80个字符。在某些情况下，理论上，该限制对于定义InstancePath、Attribute或CfgData而言过小。

预定义数据

当无法发现更多的实例时，值为-1的预定义常量END_OF_LIST可用于停止写入。

语法

```
WriteCfgData
  [ InstancePath ':=' ] < expression (IN) of string > ','
  [ Attribute ':=' ] < expression (IN) of string > ','
  [ CfgData ':=' ] < expression (IN) of anytype >
  [ '\' ListNo ':=' < variable (VAR) of num >] ';'
```

相关信息

信息，关于	请参阅
字符串的定义	第*1489*页的*string - 字符串*
读取系统参数的属性	第*492*页的*ReadCfgData - 读取系统参数的属性*
获取当前任务中的机械臂名称	第*1217*页的*RobName - 获取TCP机械臂名称*
配置	技术参考手册 - 系统参数
系统重启	第*906*页的*WarmStart - 重启控制器*

下一页继续

1.333 WriteCfgData - 写入系统参数的属性
RobotWare - OS
续前页

信息，关于	请参阅
Advanced RAPID	产品规格 - 控制器软件*IRC5*

1.334 WriteRawBytes - 写入原始数据字节数据

手册用法

WriteRawBytes用于将rawbytes型数据，写入通过Open\Bin而打开的设备。

基本示例

以下实例介绍了指令WriteRawBytes：

例 1

```
VAR iodev io_device;
VAR rawbytes raw_data_out;
VAR rawbytes raw_data_in;
VAR num float := 0.2;
VAR string answer;

ClearRawBytes raw_data_out;
PackDNHeader "10", "20 1D 24 01 30 64", raw_data_out;
PackRawBytes float, raw_data_out, (RawBytesLen(raw_data_out)+1)
     \Float4;

Open "/FCI1:/dsqc328_1", io_device \Bin;
WriteRawBytes io_device, raw_data_out;
ReadRawBytes io_device, raw_data_in \Time:=1;
Close io_device;

UnpackRawBytes raw_data_in, 1, answer \ASCII:=10;
```

在该例子中，清除raw_data_out，随后加载DeviceNet标题以及大小为0.2的浮动值。

打开设备"/FCI1/:dsqc328_1"，将当前raw_data_out中的有效数据写入到设备。随后，本程序最多等待1秒，以从设备进行读取，该程序储存在raw_data_in。

在关闭设备“/FCI1/:dsqc328_1”后，将读取的数据作为一连串10字符打开，并储存在回答中。

变元

```
WriteRawBytes IODevice RawData [\NoOfBytes]
```

IODevice

数据类型：iodev

IODevice为设备的标识符，应当将RawData写入该标识符。

RawData

数据类型：rawbytes

RawData为有待写入IODevice的数据容器。

[\NoOfBytes]

数据类型：num

\NoOfBytes告知：以索引1开始，应当将多少字节的RawData写入IODevice。

下一页继续

1.334 WriteRawBytes - 写入原始数据字节数据
RobotWare - OS
续前页

如果\NoOfBytes不存在，则将变量RawData中有效字节的当前长度写入设备IODevice。

程序执行

程序执行期间，将数据写入IODevice所指定的设备。

如果将WriteRawBytes用于现场总线命令，例如DeviceNet，则现场总线将始终发送答案。必须通过ReadRawBytes指令，用RAPID来处理答案。

未改变RawData变量中有效字节的当前长度。

上电失败重启时，将关闭系统中所有打开的文件或串行通道，并将重置iodev型变量中的I/O描述符。

错误处理

如果在写入期间出现错误，则将系统变量ERRNO设置为ERR_FILEACC。

随后，可通过错误处理器来处理该错误。

语法

```
WriteRawBytes
  [ IODevice ':=' ] < variable (VAR) of iodev> ','
  [ RawData ':=' ] < variable (VAR) of rawbytes>
  [ '\' NoOfBytes ':=' < expression (IN) of num>] ';'
```

相关信息

信息，关于	请参阅
rawbytes数据	第*1455*页的原始数据字节 - 原始数据
获取rawbytes数据的长度	第*1193*页的*RawBytesLen* - 获取原始数据字节数据的长度
清除rawbytes数据的内容	第*110*页的*ClearRawBytes* - 清除原始数据字节数据的内容
复制rawbytes数据的内容	第*128*页的*CopyRawBytes* - 复制原始数据字节数据的内容
将DeviceNet标题装入rawbytes数据	第*421*页的*PackDNHeader* - 将*DeviceNet*标题装入原始数据字节数据中。
将数据装入rawbytes数据	第*424*页的*PackRawBytes* - 将数据装入原始数据字节数据
读取rawbytes数据	第*499*页的*ReadRawBytes* - 读取原始数据字节数据
打开来自rawbytes数据的数据	第*845*页的*UnpackRawBytes* - 打开来自原始数据字节数据的数据
文件和串行通道处理	应用手册 - 控制器软件*IRC5*

1.335 WriteStrBin - 将字符串写入一个二进制串行通道

手册用法

WriteStrBin（*Write String Binary*）用于将一段字符串写入一条二进制串行通道或一份二进制文件。

基本示例

以下实例介绍了指令WriteStrBin：

另请参阅第*925*页的更多示例

例 1

```
WriteStrBin channel2, "Hello World\0A";
```

将The string "Hello World\0A"写入由channel2所引用的通道。在这种情况下，字符串以新行\0A结束。通过WriteStrBin而写入的所有字符和十六进制值将被本系统改变。

变元

```
WriteStrBin IODevice Str
```

IODevice

数据类型：iodev

当前串行通道的名称（引用）。

Str

String

数据类型：string

待写入的文本。

程序执行

将文本字符串写入到指定的串行通道或文件。

上电失败重启时，将关闭系统中所有打开的文件或串行通道，并将重置iodev型变量中的I/O描述符。

限制

该指令仅可用于为二进制读取和写入而打开的串行通道或文件。

错误处理

如果在写入期间出现错误，则将系统变量ERRNO设置为ERR_FILEACC。随后，可通过错误处理器来处理该错误。

更多示例

有关于如何使用指令WriteStrBin的更多例子阐述如下。

例 1

```
VAR iodev channel;
VAR num input;
Open "com1:", channel\Bin;
```

下一页继续

1.335 WriteStrBin - 将字符串写入一个二进制串行通道

RobotWare - OS

续前页

```
! Send the control character enq
WriteStrBin channel, "\05";
! Wait for the control character ack
input := ReadBin (channel \Time:= 0.1);
IF input = 6 THEN
  ! Send a text starting with control character stx and ending with
       etx
  WriteStrBin channel, "\02Hello world\03";
ENDIF

Close channel;
```

信号交换后，将文本字符串Hello world（以及相关的十六进制控制字符）写入到一个二进制串行通道。

语法

```
WriteStrBin
  [ IODevice ':='] <variable (VAR) of iodev> ','
  [ Str ':='] <expression (IN) of string> ';'
```

相关信息

信息，关于	请参阅
串行通道的打开等	技术参考手册 - *RAPID*语言概览
读取二进制字符串	第*1208*页的*ReadStrBin* - 从一条二进制串行通道或一份文件读取一段字符串
文件和串行通道处理	应用手册 - 控制器软件*IRC5*

1.336 WriteVar - 写入变量

手册用法

WriteVar用于将变量写入同串行传感器接口相连的设备。

传感器接口与位于串行通道（使用RTP1传输层协议）上方的传感器进行通信。

此为关于传感器通道配置的实例。

```
COM_PHY_CHANNEL:
  Name "COM1:"
  Connector "COM1"
  Baudrate 19200
COM_TRP:
  Name "sen1:"
  Type "RTP1"
  PhyChannel "COM1"
```

基本示例

以下实例介绍了指令WriteVar：

例 1

```
! Define variable numbers
CONST num SensorOn := 6;
CONST num XCoord := 8;
CONST num YCoord := 9;
CONST num ZCoord := 10;
VAR pos SensorPos;

! Connect to the sensor device“ sen1:” (defined in sio.cfg).
SenDevice "sen1:";

! Request start of sensor meassurements
WriteVar "sen1:", SensorOn, 1;

! Read a cartesian position from the sensor.
SensorPos.x := ReadVar "sen1:", XCoord;
SensorPos.y := ReadVar "sen1:", YCoord;
SensorPos.z := ReadVar "sen1:", ZCoord;

! Stop sensor
WriteVar "sen1:", SensorOn, 0;
```

变元

```
WriteVar device VarNo VarData [ \TaskName ]
```

device

数据类型：string

在sio.cfg中配置了I/O设备名称，以供传感器使用。

VarNo

数据类型：num

下一页继续

1.336 WriteVar - 写入变量
Sensor Interface
续前页

参数VarNo用于选择传感器变量。

VarData

数据类型：num

参数VarData规定了有待写入由VarNo参数所选定变量的数据。

[\TaskName]

数据类型：string

参数TaskName使其可能访问其他RAPID任务中的设备。

错误处理

错误常量 (ERRNO) 值	描述
SEN_NO_MEAS	测量失败
SEN_NOREADY	传感器不能处理命令
SEN_GENERRO	一般性传感器错误
SEN_BUSY	传感器繁忙
SEN_UNKNOWN	未知的传感器
SEN_EXALARM	外部传感器错误
SEN_CAALARM	内部传感器错误
SEN_TEMP	传感器温度错误
SEN_VALUE	非法通信值
SEN_CAMCHECK	传感器检查失败
SEN_TIMEOUT	通信错误

语法

```
WriteVar
  [ device ':=' ] < expression (IN) of string> ','
  [ VarNo ':=' ] < expression (IN) of num > ','
  [ VarData ':=' ] < expression (IN) of num > ','
  [ '\' TaskName ':=' < expression (IN) of string > ] ';'
```

相关信息

信息，关于	请参阅
与传感器设备相连	第574页的*SenDevice - 与传感器设备相连*
读取传感器变量	第1210页的*ReadVar - 从设备读取变量*
写入传感器数据块	第917页的*WriteBlock - 将数据块写入设备*
读取传感器数据块	第490页的*ReadBlock - 读取设备的数据块*
传感器通信配置	技术参考手册 - 系统参数

1.337 WZBoxDef - 定义一个箱形全局区域

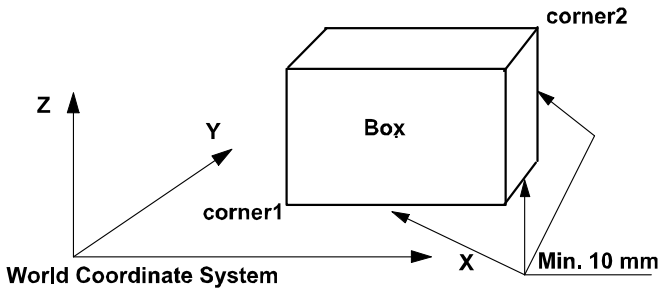
手册用法

WZBoxDef（*World Zone Box Definition*）用于定义拥有直线箱形状，且各侧均与世界坐标系各轴平行的全局区域。

基本示例

以下实例介绍了指令WZBoxDef：

例 1



xx0500002205

```
VAR shapedata volume;
CONST pos corner1:=[200,100,100];
CONST pos corner2:=[600,400,400];
...
WZBoxDef \Inside, volume, corner1, corner2;
```

定义坐标与世界坐标系各轴平行，且由相反角corner1和corner2所定义的直线箱。

变元

```
WZBoxDef [\Inside] | [\Outside] Shape LowPoint HighPoint
```

[\Inside]

数据类型：switch

定义箱内的体积。

[\Outside]

数据类型：switch

定义箱外的体积（相反体积）。

必须指定参数\Inside或\Outside之一。

Shape

数据类型：shapedata

用于储存指定体积的变量（系统专用数据）。

LowPoint

数据类型：pos

用于定义箱子的一个较低角的位置（x、y、z），以mm计。

下一页继续

1.337 WZBoxDef - 定义一个箱形全局区域

World Zones

续前页

`HighPoint`

数据类型：pos

用于定义与先前角相对的角的位置（x、y、z），以mm计。

程序执行

箱子的定义储存在shapedata型变量（参数Shape）中，以便将来在WZLimSup或WZDOSet指令中使用。

限制

LowPoint和HighPoint位置必须适用于相对的角（采用不同的x、y和z坐标值）。

如果机械臂用于指出LowPoint或HighPoint，则工件wobj0必须有效（分量trans在robtarget中的使用，例如，将p1.trans作为参数）。

语法

```
WZBoxDef
  [ ['\' Inside] | ['\' Outside] ',']
  [ LowPoint ':='] <expression (IN) of pos> ','
  [ Shape ':='] <variable (VAR) of shapedata> ','
  [ HighPoint ':='] <expression (IN) of pos> ';'
```

相关信息

信息，关于	请参阅
全局区域	技术参考手册 - *RAPID*语言概览
全局区域形状	第*1473*页的*shapedata* - 全局区域形状数据
定义球形全局区域	第*952*页的*WZSphDef* - 定义球形全局区域
定义圆柱形全局区域	第*931*页的*WZCylDef* - 定义圆柱形全局区域
定义内部接头的全局区域	第*943*页的*WZHomeJointDef* - 定义内部接头的全局区域
定义限制接头的全局区域	第*946*页的*WZLimJointDef* - 定义有关接头内限制的全局区域
启用全局区域限制监控	第*949*页的*WZLimSup* - 启用全局区域限制监控
启用全局区域数字信号输出设置	第*935*页的*WZDOSet* - 启用全局区域，设置数字信号输出

1.338 WZCylDef - 定义圆柱形全局区域

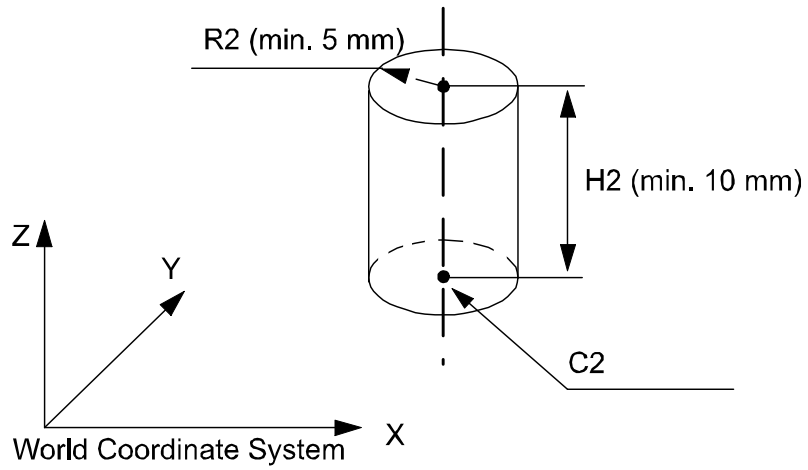
手册用法

WZCylDef（全局区域圆柱体定义)为，用于定义外形为圆柱形，且圆柱轴与世界坐标系z轴平行的全局区域。

基本示例

以下实例介绍了指令WZCylDef：

例 1



xx0500002206

```
VAR shapedata volume;
CONST pos C2:=[300,200,200];
CONST num R2:=100;
CONST num H2:=200;
...
WZCylDef \Inside, volume, C2, R2, H2;
```

定义底圆中心C2、半径R2且高度H2的圆柱体。

变元

```
WZCylDef [\Inside] | [\Outside] Shape CentrePoint Radius Height
```

[\Inside]

数据类型：switch

定义圆柱体内的体积。

[\Outside]

数据类型：switch

定义圆柱体外的体积（相反体积）。

必须指定参数\Inside或\Outside之一。

Shape

数据类型：shapedata

下一页继续

1.338 WZCylDef - 定义圆柱形全局区域

World Zones

续前页

用于储存指定体积的变量（系统专用数据）。

`CentrePoint`

数据类型：pos

用于定义圆柱体的一个圆周端中心的位置（x、y、z），以mm计。

`Radius`

数据类型：num

圆柱体的半径，以mm计。

`Height`

数据类型：num

圆柱体的高度，以mm计。如果其值为正（+z方向），则CentrePoint参数为圆柱体下端中心（如上述例子所示）。如果其值为负（-z方向），则CentrePoint参数为圆柱体上端中心。

程序执行

圆柱体的定义储存在shapedata型变量（参数Shape）中，以便将来在WZLimSup或WZDOSet指令中使用。

限制

如果机械臂用于指出CentrePoint，则工件wobj0必须有效（分量trans在robtarget中的使用，例如，将p1.trans作为参数）。

语法

```
WZCylDef
  [ '\' Inside] | [ '\' Outside] ','
  [ Shape ':='] <variable (VAR) of shapedata> ','
  [ CentrePoint ':='] <expression (IN) of pos> ','
  [ Radius ':='] <expression (IN) of num> ','
  [ Height ':='] <expression (IN) of num> ';'
```

相关信息

信息，关于	请参阅
全局区域	技术参考手册 - *RAPID*语言概览
全局区域形状	第*1473*页的*shapedata* - 全局区域形状数据
定义箱形全局区域	第*929*页的*WZBoxDef* - 定义一个箱形全局区域
定义球形全局区域	第*952*页的*WZSphDef* - 定义球形全局区域
定义内部接头的全局区域	第*943*页的*WZHomeJointDef* - 定义内部接头的全局区域
定义限制接头的全局区域	第*946*页的*WZLimJointDef* - 定义有关接头内限制的全局区域
启用全局区域限制监控	第*949*页的*WZLimSup* - 启用全局区域限制监控
启用全局区域数字信号输出设置	第*935*页的*WZDOSet* - 启用全局区域，设置数字信号输出

1.339 WZDisable - 停用临时全局区域监控

手册用法

WZDisable（*World Zone Disable*）用于停用对临时全局区域的监控，其预先定义以便停止移动或设置输出。

基本示例

以下实例介绍了指令WZDisable：

例 1

```
VAR wztemporary wzone;
...
PROC...
  WZLimSup \Temp, wzone, volume;
  MoveL p_pick, v500, z40, tool1;
  WZDisable wzone;
  MoveL p_place, v200, z30, tool1;
ENDPROC
```

当向p_pick移动时，检查机械臂TCP的位置，以便其不会进入指定体积wzone。当转到p_place时，不实施该监控。

变元

```
WZDisable WorldZone
```

WorldZone

数据类型：wztemporary

wztemporary型变量或永久变量，其包含待停用全局区域的识别号。

程序执行

停用临时全局区域。这意味着临时停止有关相应体积的机械臂TCP监控。通过WZEnable指令，可重新将其启用。

限制

仅可停用临时全局区域。固定式全局区域始终有效。

语法

```
WZDisable
  [ WorldZone ':='] <variable or persistent (INOUT) of wztemporary>
       ';'
```

相关信息

信息，关于	请参阅
全局区域	技术参考手册 - *RAPID*语言概览
全局区域形状	第*1473*页的*shapedata* - 全局区域形状数据
临时全局区域数据	第*1529*页的*wztemporary* - 临时全局区域数据
启用全局区域限制监控	第*949*页的*WZLimSup* - 启用全局区域限制监控

下一页继续

1.339 WZDisable - 停用临时全局区域监控

World Zones

续前页

信息，关于	请参阅
启用全局区域，设置数字信号输出	第*935*页的*WZDOSet* - 启用全局区域，设置数字信号输出
启用全局区域	第*939*页的*WZEnable* - 启用临时全局区域监控
擦除全局区域	第*941*页的*WZFree* - 擦除临时全局区域监控

1.340 WZDOSet - 启用全局区域，设置数字信号输出

手册用法

WZDOSet（*World Zone Digital Output Set*）用于定义行动，并启用全局区域，以监控机械臂移动。

在执行该指令后，当机械臂TCP或机械臂/外轴（接头中的区域）位于指定全局区域内，或正在接近全局区域时，将数字信号输出信号设置为指定值。

基本示例

以下实例介绍了指令WZDOSet：

另请参阅第*936*页的更多示例

例 1

```
VAR wztemporary service;

PROC zone_output()
  VAR shapedata volume;
  CONST pos p_service:=[500,500,700];
  ...
  WZSphDef \Inside, volume, p_service, 50;
  WZDOSet \Temp, service \Inside, volume, do_service, 1;
ENDPROC
```

机械臂TCP在程序执行期间位于指定球体内时，或出现点动时，有关在设置信号do_service的应用程序中的临时全局区域service的定义。

变元

```
WZDOSet [\Temp] | [\Stat] WorldZone [\Inside] | [\Before] Shape
     Signal SetValue
```

[\Temp]

Temporary

数据类型：switch

用于定义的全局区域为临时全局区域。

[\Stat]

Stationary

数据类型：switch

用于定义的全局区域为固定式全局区域。

必须指定参数\Temp 或\Stat之一。

WorldZone

数据类型：wztemporary或wzstationary

将通过全局区域的识别号（数值）来更新的变量或永久变量。

如果使用开关\Temp，则数据类型必须为wztemporary。如果使用开关\Stat，则数据类型必须为wzstationary。

[\Inside]

数据类型：switch

下一页继续

1.340 WZDOSet - 启用全局区域，设置数字信号输出
World Zones
续前页

当机械臂TCP或指定轴位于指定体积内时，将设置数字信号输出信号。

[\Before]

数据类型：switch

在机械臂TCP或指定轴达到规定体积前（尽可能在该体积前），将设置数字信号输出信号。

必须指定参数\Inside 或\Before之一。

Shape

数据类型：shapedata

用以定义全局区域体积的变量。

Signal

数据类型：signaldo

待改变数字信号输出信号的名称。

如果使用固定式全局区域，则必须写入信号，以作为用户访问保护（RAPID，FP）。针对系统参数或指定轴中的信号，设置Access Level。

SetValue

数据类型：dionum

当机械臂TCP位于体积内或刚好在进入体积之前的信号期望值（0或1）。

当位于体积外或恰好位于体积外时，将信号设置为相反值。

程序执行

启用规定的全局区域。从此刻起，监控机械臂TCP位置（或机械臂/外接头位置），且当机械臂TCP位置（或机械臂/外接头位置）位于体积（\Inside）内或接近体积（\Before）边界时，将设置输出。

如果使用WZHomeJointDef或WZLimJointDef连同WZDOSet，则仅当所有有效轴以及接头空间监控位于接头空间前或内部时，方才设置数字信号输出信号。

更多示例

有关于如何使用指令WZDOSet的更多例子阐述如下。

例 1

```
VAR wztemporary home;
VAR wztemporary service;
PERS wztemporary equip1:=[0];

PROC main()
  ...
  ! Definition of all temporary world zones
  zone_output;
  ...
  ! equip1 in robot work area
  WZEnable equip1;
  ...
  ! equip1 out of robot work area
  WZDisable equip1;
  ...
```

下一页继续

```
  ! No use for equip1 any more
  WZFree equip1;
  ...
ENDPROC

PROC zone_output()
  VAR shapedata volume;
  CONST pos p_home:=[800,0,800];
  CONST pos p_service:=[800,800,800];
  CONST pos p_equip1:=[-800,-800,0];
  ...
  WZSphDef \Inside, volume, p_home, 50;
  WZDOSet \Temp, home \Inside, volume, do_home, 1;
  WZSphDef \Inside, volume, p_service, 50;
  WZDOSet \Temp, service \Inside, volume, do_service, 1;
  WZCylDef \Inside, volume, p_equip1, 300, 1000;
  WZLimSup \Temp, equip1, volume;
  ! equip1 not in robot work area
  WZDisable equip1;
ENDPROC
```

有关应用程序中临时全局区域home和service的定义，当在程序执行或点动期间，机械臂分别位于球体home或service内部，则该定义可设置信号do_home和do_service。

与此同时，当equip1位于机械臂工作区内部时，临时全局区域equip1的定义仅适用于一部分机械臂程序。此时，在程序执行和手动点动期间，机械臂会在进入equip1体积之前停止。通过使用永久变量equip1值，可以从其他程序任务停用或启用equip1。

错误处理

系统会生成下列可恢复错误，并在错误处理器中处理这些错误。系统变量ERRNO将被设置成：

如果信号变量是RAPID中声明的变量，则ERR_NO_ALIASIO_DEF。尚未同I/O配置以及指令AliasIO中确定的I/O信号相连。

限制

通过使用参数WorldZone中的相同变量，无法重新定义全局区域。

无法在RAPID程序中停用、再次启用或擦除固定式全局区域。

可以在RAPID程序中停用（WZDisable）、再次启用（WZEnable）或擦除（WZFree））临时全局区域。

语法

```
WZDOSet
  [ ['\' Temp] | ['\' Stat] ',']
  [ WorldZone ':='] <variable or persistent (INOUT) of wztemporary>
  [ '\' Inside] | ['\' Before] ','
  [ Shape ':='] <variable (VAR) of shapedata> ','
  [ Signal ':='] <variable (VAR) of signaldo> ','
  [ SetValue ':='] <expression (IN) of dionum> ';'
```

下一页继续

1.340 WZDOSet - 启用全局区域，设置数字信号输出
World Zones
续前页

相关信息

信息，关于	请参阅
全局区域	技术参考手册 - *RAPID*语言概览
全局区域形状	第*1473*页的*shapedata* - 全局区域形状数据
临时全局区域	第*1529*页的*wztemporary* - 临时全局区域数据
固定式全局区域	第*1527*页的*wzstationary* - 固定式全局区域数据
定义直线箱形全局区域	第*929*页的*WZBoxDef* - 定义一个箱形全局区域
定义球形全局区域	第*952*页的*WZSphDef* - 定义球形全局区域
定义圆柱形全局区域	第*931*页的*WZCylDef* - 定义圆柱形全局区域
定义内部接头的全局区域	第*943*页的*WZHomeJointDef* - 定义内部接头的全局区域
启用全局区域限制监控	第*949*页的*WZLimSup* - 启用全局区域限制监控
信号访问级别	技术参考手册 - 系统参数

1.341 WZEnable - 启用临时全局区域监控

手册用法

WZEnable（*World Zone Enable*）用于重新启用对临时全局区域的监控，其预先定义，以便停止移动或设置输出。

基本示例

以下实例介绍了指令WZEnable：

例 1

```
VAR wztemporary wzone;
...
PROC ...
  WZLimSup \Temp, wzone, volume;
  MoveL p_pick, v500, z40, tool1;
  WZDisable wzone;
  MoveL p_place, v200, z30, tool1;
  WZEnable wzone;
  MoveL p_home, v200, z30, tool1;
ENDPROC
```

当向p_pick移动时，检查机械臂TCP的位置，以便其不会进入指定体积wzone。当转到p_place时，不实施该监控，但是在转到p_home前，重新启用该监控。

变元

```
WZEnable WorldZone
```

WorldZone

数据类型：wztemporary

wztemporary型变量或永久变量，其包含待启用全局区域的识别号。

程序执行

重新启用临时全局区域。注意，全局区域在创建时自动启用。当全局区域先前已由WZDisable停用时，其仅需重新启用。

限制

仅可停用和重新启用临时全局区域。固定式全局区域始终有效。

语法

```
WZEnable
  [ WorldZone ':='] <variable or persistent (INOUT) of wztemporary>
       ';'
```

相关信息

信息，关于	请参阅
全局区域	技术参考手册 - *RAPID*语言概览
全局区域形状	第*1473*页的*shapedata* - 全局区域形状数据
临时全局区域数据	第*1529*页的*wztemporary* - 临时全局区域数据

下一页继续

1.341 WZEnable - 启用临时全局区域监控

World Zones

续前页

信息，关于	请参阅
启用全局区域限制监控	第*949*页的*WZLimSup - 启用全局区域限制监控*
启用全局区域，设置数字信号输出	第*935*页的*WZDOSet - 启用全局区域，设置数字信号输出*
停用全局区域	第*933*页的*WZDisable - 停用临时全局区域监控*
擦除全局区域	第*941*页的*WZFree - 擦除临时全局区域监控*

1.342 WZFree - 擦除临时全局区域监控

手册用法

WZFree（*World Zone Free*）用于擦除临时全局区域的定义，其预先定义，以便停止移动或设置输出。

基本示例

以下实例介绍了指令WZFree：

例 1

```
VAR wztemporary wzone;
...
PROC ...
  WZLimSup \Temp, wzone, volume;
  MoveL p_pick, v500, z40, tool1;
  WZDisable wzone;
  MoveL p_place, v200, z30, tool1;
  WZEnable wzone;
  MoveL p_home, v200, z30, tool1;
  WZFree wzone;
ENDPROC
```

当向p_pick移动时，检查机械臂TCP的位置，以便其不会进入指定体积wzone。当转到p_place时，不实施该监控，但是在转到p_home前，重新启用该监控。当达到该位置时，擦除全局区域定义。

变元

```
WZFree WorldZone
```

WorldZone

数据类型：wztemporary

wztemporary型变量或永久变量，其包含待擦除全局区域的识别号。

程序执行

首先停用临时全局区域，随后，擦除其定义。

一旦擦除，将无法重新启用或停用临时全局区域。

限制

仅可停用、重新启用或擦除临时全局区域。固定式全局区域始终有效。

语法

```
WZFree
  [ WorldZone ':='] <variable or persistent (INOUT) of wztemporary>
       ';'
```

相关信息

信息，关于	请参阅
全局区域	技术参考手册 - *RAPID*语言概览
全局区域形状	第*1473*页的*shapedata* - 全局区域形状数据

下一页继续

1.342 WZFree - 擦除临时全局区域监控

World Zones

续前页

信息，关于	请参阅
临时全局区域数据	第*1529*页的*wztemporary - 临时全局区域数据*
启用全局区域限制监控	第*949*页的*WZLimSup - 启用全局区域限制监控*
启用全局区域，设置数字信号输出	第*935*页的*WZDOSet - 启用全局区域，设置数字信号输出*
停用全局区域	第*933*页的*WZDisable - 停用临时全局区域监控*
启用全局区域	第*939*页的*WZEnable - 启用临时全局区域监控*

1.343 WZHomeJointDef - 定义内部接头的全局区域

手册用法

WZHomeJointDef（*World Zone Home Joint Definition*）用于定义接头坐标系中的全局区域，以便将机械臂和外轴作为HOME或SERVICE位置。

基本示例

以下实例介绍了指令WZHomeJointDef：

例 1

```
VAR wzstationary home;
...
PROC power_on()
  VAR shapedata joint_space;
  CONST jointtarget home_pos := [ [ 0, 0, 0, 0, 0, -45], [ 0, 9E9,
       9E9, 9E9, 9E9, 9E9] ];
  CONST jointtarget delta_pos := [ [ 2, 2, 2, 2, 2, 2], [ 5, 9E9,
       9E9, 9E9, 9E9, 9E9] ];
  ...
  WZHomeJointDef \Inside, joint_space, home_pos, delta_pos;
  WZDOSet \Stat, home \Inside, joint_space, do_home, 1;
ENDPROC
```

程序执行和点动期间，当所有机械臂轴和外轴extax.eax_a 均位于接头位置home_pos（在各轴+/-delta_pos内）时，用以设置信号do_home to 1,的固定式全局区域home的定义和启用。数据类型shapedata的变量joint_space用于将数据从指令WZHomeJointDef转移至指令WZDOSet。

变元

```
WZHomeJointDef [\Inside] | [\Outside] Shape MiddleJointVal
     DeltaJointVal
```

[\Inside]

数据类型：switch

定义MiddleJointVal+/-DeltaJointVal内的接头空间。

[\Outside]

数据类型：switch

定义MiddleJointVal+/-DeltaJointVal外的接头空间（相反接头空间）。

Shape

数据类型：shapedata

用于储存指定接头空间的变量（系统专用数据）。

MiddleJointVal

数据类型：jointtarget

确定接头空间中心的接头坐标位置。针对各机械臂轴和外轴进行说明（旋转轴以度计，线性轴以mm计）。在绝对接头中进行说明（而非在有关外轴的偏移量坐标系EOffsSet-EOffsOn中）。有关某些轴的值9E9意味着不应该监控该轴。编程期间，无效外轴亦得出9E9。

下一页继续

1.343 WZHomeJointDef - 定义内部接头的全局区域

World Zones

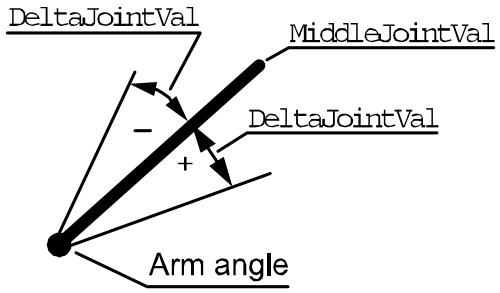
续前页

`DeltaJointVal`

数据类型：`jointtarget`

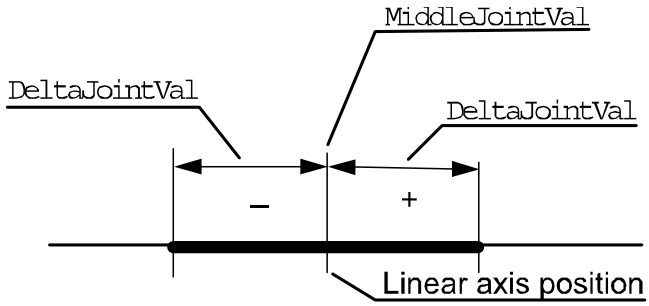
距离接头空间中心的接头坐标中的+/-δ位置。针对用于监控的所有轴，该值必须大于0。

下图显示了有关旋转轴接头空间的定义。



xx0500002208

下图显示了有关线性轴接头空间的定义。



xx0500002209

程序执行

接头空间的定义储存在`shapedata`型变量（参数`Shape`）中，以便将来在`WZLimSup`或`WZDOSet`指令中使用。

如果使用`WZHomeJointDef`连同`WZDOSet`，则仅当所有有效轴以及接头空间监控位于接头空间前或内部时，方才设置数字信号输出信号。

如果使用包含外部接头空间的`WZHomeJointDef`（参数`\Outside`）连同`WZLimSup`，则当一个包含接头空间监控的有效轴达到接头空间时，机械臂随即停止。

如果使用包含内部接头空间的`WZHomeJointDef`（参数`\Inside`）连同`WZLimSup`，则当包含接头空间监控的最后一个有效轴达到接头空间时，机械臂随即停止。这意味着接头空间内部可同时拥有一个或多个轴，但是并非所有轴都是有效并接受监控的轴。

在执行指令`ActUnit`或`DeactUnit`，以启用或停用机械单元期间，将更新有关HOME位置或工作区域限制的监控状态。

限制

xx0100000002

下一页继续

在全局区域启用期间（通过WZLimSup各自的指令WZDOSet），仅有效机械单元及其有效轴涵盖在工作区域限制的相关HOME位置的监控中。此外，在有待监控的程序移动或点动期间，机械单元及其轴必须始终有效。

例如，如果一个包含监控的轴位于其HOME 接头位置外，但是却予以停用，则当包含接头空间监控的所有其他有效轴位于HOME接头位置内时，其并未妨碍有关待设置HOME接头位置的数字信号输出信号。再次启用该轴时，其将涵盖在监控中，随后，机器人系统将位于HOME接头位置外，且数字信号输出将得以重置。

语法

```
WZHomeJointDef
  [ ['\' Inside] | ['\' Outside] ',']
  [ Shape ':='] <variable (VAR) of shapedata> ','
  [ MiddleJointVal ':='] <expression (IN) of jointtarget> ','
  [ DeltaJointVal ':='] <expression (IN) of jointtarget> ';'
```

相关信息

信息，关于	请参阅
全局区域	技术参考手册 - *RAPID*语言概览
全局区域形状	第*1473*页的*shapedata* - 全局区域形状数据
定义箱形全局区域	第*929*页的*WZBoxDef* - 定义一个箱形全局区域
定义圆柱形全局区域	第*931*页的*WZCylDef* - 定义圆柱形全局区域
定义球形全局区域	第*952*页的*WZSphDef* - 定义球形全局区域
定义限制接头的全局区域	第*946*页的*WZLimJointDef* - 定义有关接头内限制的全局区域
启用全局区域限制监控	第*949*页的*WZLimSup* - 启用全局区域限制监控
启用全局区域数字信号输出设置	第*935*页的*WZDOSet* - 启用全局区域，设置数字信号输出

1.344 WZLimJointDef - 定义有关接头内限制的全局区域

手册用法

WZLimJointDef（*World Zone Limit Joint Definition*）用于定义接头坐标系中的全局区域，以便将机械臂和外轴用于工作区域的限制。

通过WZLimJointDef，有可能限制RAPID程序中各机械臂和外轴的工作区域，以及可通过系统参数*Motion - Arm - robx_y - Upper Joint Bound ... Lower Joint Bound*完成的限制。

基本示例

以下实例介绍了指令WZLimJointDef：

例 1

```
VAR wzstationary work_limit;
  ...
  PROC power_on()
    VAR shapedata joint_space;
    CONST jointtarget low_pos:= [ [ -90, 9E9, 9E9, 9E9, 9E9, 9E9],
         [ -1000, 9E9, 9E9, 9E9, 9E9, 9E9]];
    CONST jointtarget high_pos := [ [ 90, 9E9, 9E9, 9E9,9E9, 9E9],
         [ 9E9, 9E9, 9E9, 9E9, 9E9, 9E9] ];
    ...
    WZLimJointDef \Outside, joint_space, low_pos, high_pos;
    WZLimSup \Stat, work_limit, joint_space;
  ENDPROC
```

程序执行和点动期间，固定式全局区域work_limit的定义和启用，将机械臂轴1的工作区域限制在-90到+90度，并将外轴extax.eax_a限制在-1000 mm。数据类型shapedata的变量joint_space用于将数据从指令WZLimJointDef转换至指令WZLimSup。

变元

```
WZLimJointDef [\Inside] | [\Outside] Shape LowJointVal HighJointVal
```

[\Inside]

数据类型：switch

定义LowJointVal ... HighJointVal内的接头空间。

[\Outside]

数据类型：switch

定义LowJointVal ... HighJointVal外的接头空间（相反接头空间）。

Shape

数据类型：shapedata

用于储存指定接头空间的变量（系统专用数据）。

LowJointVal

数据类型：jointtarget

确定有关接头空间下限的接头坐标位置。针对各机械臂轴和外轴进行说明（旋转轴以度计，线性轴以毫米计）。在绝对接头中进行说明（而非在针对外轴的偏移量坐标系

下一页继续

EOffsSet或EOffsOn中）。有关一些轴的值9E9意味着应当监控该轴的下限。编程期间，无效外轴同时得出9E9。

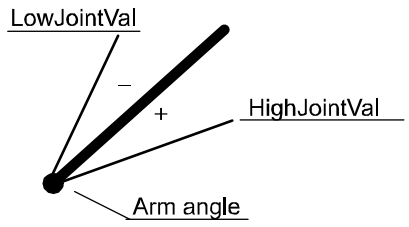
HighJointVal

数据类型：jointtarget

确定有关接头空间上限的接头坐标位置。针对各机械臂轴和外轴进行说明（旋转轴以度计，线性轴以毫米计）。在绝对接头中进行说明（而非在针对外轴的偏移量坐标系EOffsSet或EOffsOn中）。有关一个轴的值9E9意味着应当监控该轴的上限。编程期间，无效外轴同时得出9E9。

针对用于监控的所有轴，各轴的HighJointVal减去LowJointVal，所得结果必须大于0。

下图显示了有关旋转轴接头空间的定义。



xx0500002281

下图显示了有关线性轴接头空间的定义。

xx0100000002

程序执行

接头空间的定义储存在shapedata型变量（参数Shape）中，以便将来在WZLimSup或WZDOSet指令中使用。

如果使用WZLimJointDef连同WZDOSet，则仅当所有有效轴以及接头空间监控位于接头空间前或内部时，方才设置数字信号输出信号。

如果使用包含外部接头空间的WZLimJointDef（参数\Outside）连同WZLimSup，则当一个包含接头空间监控的有效轴达到接头空间时，机械臂随即停止。

如果使用包含内部接头空间的WZLimJointDef（参数 \Inside）连同WZLimSup，则当包含接头空间监控的最后一个有效轴达到接头空间时，机械臂随即停止。这意味着接头空间内部可同时拥有一个或多个轴，但是并非所有轴都是有效并接受监控的轴。

在执行指令ActUnit或DeactUnit期间，将更新监控状态。

限制

xx0100000002

警告！

下一页继续

1.344 WZLimJointDef - 定义有关接头内限制的全局区域

World Zones

续前页

在全局区域启用期间（通过WZLimSup各自的指令WZDOSet），仅有效机械单元及其有效轴涵盖在工作区域限制的相关HOME位置的监控中。此外，在有待监控的程序移动或点动期间，机械单元及其轴必须始终有效。

例如，如果一个包含监控的轴位于其HOME接头位置外，但是却予以停用，则当包含接头空间监控的所有其他有效轴位于HOME接头位置内时，其并未妨碍有关待设置HOME接头位置的数字信号输出信号。再次启用该轴时，其将涵盖在监控中，随后，机器人系统将位于HOME接头位置外，且数字信号输出将得以重置。

语法

```
WZLimJointDef
  [ ['\' Inside] | ['\' Outside] ',']
  [ Shape ':='] <variable (VAR) of shapedata> ','
  [ LowJointVal ':='] <expression (IN) of jointtarget> ','
  [ HighJointVal ':='] <expression (IN) of jointtarget> ';'
```

相关信息

信息，关于	请参阅
全局区域	技术参考手册 - *RAPID*语言概览
全局区域形状	第*1473*页的*shapedata* - 全局区域形状数据
定义箱形全局区域	第*929*页的*WZBoxDef* - 定义一个箱形全局区域
定义圆柱形全局区域	第*931*页的*WZCylDef* - 定义圆柱形全局区域
定义球形全局区域	第*952*页的*WZSphDef* - 定义球形全局区域
定义内部接头的全局区域	第*943*页的*WZHomeJointDef* - 定义内部接头的全局区域
启用全局区域限制监控	第*949*页的*WZLimSup* - 启用全局区域限制监控
启用全局区域数字信号输出设置	第*935*页的*WZDOSet* - 启用全局区域，设置数字信号输出

1.345 WZLimSup - 启用全局区域限制监控

手册用法

WZLimSup（*World Zone Limit Supervision*）用于定义行动，并启用全局区域，以监控机械臂或外轴的工作区域。

执行该指令后，在程序执行和点动期间，当机械臂TCP达到规定全局区域，或当机械臂/外轴达到接头中的规定全局区域时，移动得以停止。

基本示例

以下实例介绍了指令WZLimSup：

另请参阅第*950*页的更多示例

例 1

```
VAR wzstationary max_workarea;
...
PROC POWER_ON()
  VAR shapedata volume;
  ...
  WZBoxDef \Outside, volume, corner1, corner2;
  WZLimSup \Stat, max_workarea, volume;
ENDPROC
```

包含箱外区域（临时储存在volume中）以及行动工作区域监控的固定式全局区域max_workarea的定义和启用。在进入箱外区域之前，机械臂在出现错误消息时停止。

变元

```
WZLimSup [\Temp] | [\Stat] WorldZone Shape
```

[\Temp]

Temporary

数据类型：switch

用于定义的全局区域为临时全局区域。

[\Stat]

Stationary

数据类型：switch

用于定义的全局区域为固定式全局区域。

必须指定参数\Temp 或\Stat之一。

WorldZone

数据类型：wztemporary或wzstationary

将通过全局区域的识别号（数值）来更新的变量或永久变量。

如果使用开关\Temp，则数据类型必须为wztemporary。如果使用开关\Stat，则数据类型必须为wzstationary。

Shape

数据类型：shapedata

下一页继续

1.345 WZLimSup - 启用全局区域限制监控

World Zones

续前页

用以定义全局区域体积的变量。

程序执行

启用指定的全局区域。从此刻起，监控机械臂的TCP位置或机械臂/外轴接头位置。如果其达到指定区域，则停止移动。

如果使用包含外部接头空间的WZLimJointDef或WZHomeJointDef（参数\Outside）连同WZLimSup，则当一个包含接头空间监控的有效轴达到接头空间时，机械臂随即停止。

如果使用包含内部接头空间的WZLimJointDef或WZHomeJointDef（参数\Inside）连同WZLimSup，则当包含接头空间监控的最后一个有效轴达到接头空间时，机械臂随即停止。这意味着接头空间内部可同时拥有一个或多个轴，但是并非所有轴都是有效并接受监控的轴。

在执行指令ActUnit或DeactUnit期间，将更新监控状态。

更多示例

有关于如何使用指令WZLimSup的更多例子阐述如下。

例 1

```
VAR wzstationary box1_invers;
VAR wzstationary box2;

PROC wzone_power_on()
  VAR shapedata volume;
  CONST pos box1_c1:=[500,-500,0];
  CONST pos box1_c2:=[-500,500,500];
  CONST pos box2_c1:=[500,-500,0];
  CONST pos box2_c2:=[200,-200,300];
  ...
  WZBoxDef \Outside, volume, box1_c1, box1_c2;
  WZLimSup \Stat, box1_invers, volume;
  WZBoxDef \Inside, volume, box2_c1, box2_c2;
  WZLimSup \Stat, box2, volume;
ENDPROC
```

有关机械臂工作区域的限制，包含以下固定式全局区域：

- 位于box1_invers外时的外部工作区域
- 位于box2内时的外部工作区域

如果该程序与系统事件POWER ON相关，则此类全局区域将始终在系统中有效，其适用于程序移动和手动点动。

限制

通过使用参数WorldZone中的相同变量，无法重新定义全局区域。

无法在RAPID程序中停用、再次启用或擦除固定式全局区域。

可以在RAPID程序中停用（WZDisable）、再次启用（WZEnable）或擦除（WZFree)）临时全局区域。

下一页继续

语法

```
WZLimSup
  [ ['\' Temp] | ['\Stat] ',']
  [ WorldZone ':='] <variable or persistent (INOUT) of wztemporary>
       ','
  [ Shape ':='] <variable (VAR) of shapedata> ';'
```

相关信息

信息，关于	请参阅
全局区域	技术参考手册 - *RAPID*语言概览
全局区域形状	第*1473*页的*shapedata* - 全局区域形状数据
临时全局区域	第*1529*页的*wztemporary* - 临时全局区域数据
固定式全局区域	第*1527*页的*wzstationary* - 固定式全局区域数据
定义直线箱形全局区域	第*929*页的*WZBoxDef* - 定义一个箱形全局区域
定义球形全局区域	第*952*页的*WZSphDef* - 定义球形全局区域
定义圆柱形全局区域	第*931*页的*WZCylDef* - 定义圆柱形全局区域
定义内部接头的全局区域	第*943*页的*WZHomeJointDef* - 定义内部接头的全局区域
定义限制接头的全局区域	第*946*页的*WZLimJointDef* - 定义有关接头内限制的全局区域
启用全局区域数字信号输出设置	第*935*页的*WZDOSet* - 启用全局区域，设置数字信号输出

1.346 WZSphDef - 定义球形全局区域

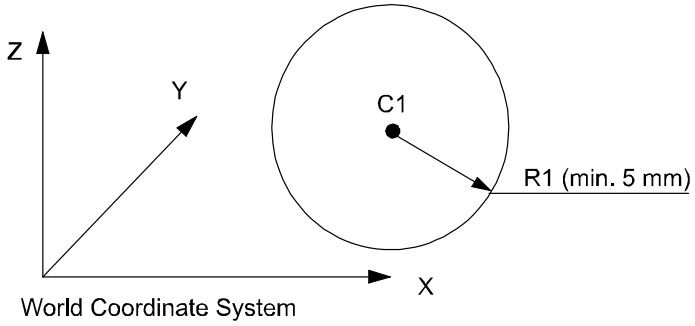
手册用法

WZSphDef（*World Zone Sphere Definition*）用于定义球形的全局区域。

基本示例

以下实例介绍了指令WZSphDef：

例 1



xx0500002207

```
VAR shapedata volume;
CONST pos C1:=[300,300,200];
CONST num R1:=200;
...
WZSphDef \Inside, volume, C1, R1;
```

根据其中心C1及其半径R1，定义命名为volume的球体。

变元

```
WZSphDef [\Inside] | [\Outside] Shape CentrePoint Radius
```

[\Inside]

数据类型：switch

定义球体内的体积。

[\Outside]

数据类型：switch

定义球体外的体积（相反体积）。

必须指定参数\Inside或\Outside之一。

Shape

数据类型：shapedata

用于储存指定体积的变量（系统专用数据）。

CentrePoint

数据类型：pos

用于球体中心的位置（x、y、z），以mm计。

下一页继续

Radius

数据类型：num

球体的半径，以mm计。

程序执行

球体的定义储存在shapedata型变量（参数Shape）中，以便将来在WZLimSup或WZDOSet指令中使用。

限制

如果机械臂用于指出CentrePoint，则工件wobj0必须有效（分量trans在robtarget中的使用，例如，将p1.trans作为参数）。

语法

```
WZSphDef
  [ ['\' Inside] | ['\' Outside] ',']
  [ Shape ':=’]<variable (VAR) of shapedata> ','
  [ CentrePoint ':='] <expression (IN) of pos> ','
  [ Radius ':='] <expression (IN) of num> ';'
```

相关信息

信息，关于	请参阅
全局区域	技术参考手册 - *RAPID*语言概览
全局区域形状	第*1473*页的*shapedata* - 全局区域形状数据
定义箱形全局区域	第*929*页的*WZBoxDef* - 定义一个箱形全局区域
定义圆柱形全局区域	第*931*页的*WZCylDef* - 定义圆柱形全局区域
定义内部接头的全局区域	第*943*页的*WZHomeJointDef* - 定义内部接头的全局区域
定义限制接头的全局区域	第*946*页的*WZLimJointDef* - 定义有关接头内限制的全局区域
启用全局区域限制监控	第*949*页的*WZLimSup* - 启用全局区域限制监控
启用全局区域数字信号输出设置	第*935*页的*WZDOSet* - 启用全局区域，设置数字信号输出

此页刻意留白

2 函数

2.1 Abs - 获得绝对值

手册用法

Abs用于获取绝对值，即数字数据的正值。

基本示例

以下示例介绍了函数Abs。

另请参阅第*955*页的更多示例

例 1

```
reg1 := Abs(reg2);
```

将Reg1指定为reg2的绝对值。

返回值

数据类型：num

绝对值，即正数值，例如：

输入值	返回值
3	3
-3	3
-2.53	2.53

变元

```
Abs (Value)
```

Value

数据类型：num

输入值。

更多示例

有关于函数Abs的更多例子阐述如下。

例 1

```
TPReadNum no_of_parts, "How many parts should be produced? ";
no_of_parts := Abs(no_of_parts);
```

要求操作员输入待生产零件的数量。确保该值大于零，并使操作员给出的值为正。

语法

```
Abs '('
  [ Value ':=' ] < expression (IN) of num >')'
```

返回值的数据类型是 num 的函数。

相关信息

信息，关于	请参阅
数学指令和函数	技术参考手册 - *RAPID*语言概览

2.2 AbsDnum - 获取一个双数值的绝对值

手册用法

AbsDnum用于获取绝对值，即dnum数值的正值。

基本示例

以下示例介绍了函数AbsDnum。

另请参阅第*956*页的更多示例

例 1

```
VAR dnum value1;
VAR dnum value2:=-20000000;
value1 := AbsDnum(value2);
```

将Value1指定为value2的绝对值。

返回值

数据类型：dnum

绝对值，即正数值，例如：

输入值	返回值
3	3
-3	-3
-2.53	2.53
-4503599627370496	4503599627370496

变元

```
AbsDnum (Value)
```

Value

数据类型：dnum

输入值。

更多示例

有关于函数AbsDnum的更多例子阐述如下。

例 1

```
TPReadDnum no_of_parts, "How many parts should be produced? ";
no_of_parts := AbsDnum(no_of_parts);
```

要求操作员输入待生产零件的数量。确保该值大于零，并使操作员给出的值为正。

语法

```
AbsDnum '('
  [ Value ':=' ] < expression (IN) of dnum > ')'
```

含数据类型dnum的返回值的函数。

下一页继续

相关信息

信息，关于	请参阅
数学指令和函数	技术参考手册 - *RAPID*语言概览

2.3 ACos - 计算反余弦值

手册用法

ACos（***Arc Cosine***）用于计算有关数据类型num的反余弦值。

基本示例

以下示例介绍了函数ACos。

例 1

```
VAR num angle;
VAR num value;
...
...
angle := ACos(value);
```

angle将获得value的反余弦值。

返回值

数据类型：num

反余弦值，以度表示，范围为【0，180】。

变元

```
ACos (Value)
```

Value

数据类型：num

参数值必须处于范围【-1，1】中。

限制

如果x处于范围【-1，1】外，则函数Acos(x)的执行将出错。

语法

```
Acos '('
  [Value ':='] <expression (IN) of num>')'
```

返回值的数据类型是 num 的函数。

相关信息

信息，关于	请参阅
数学指令和函数	技术参考手册 - *RAPID*语言概览

2.4 ACosDnum - 计算反余弦值

手册用法

ACosDnum（***Arc Cosine dnum***）用于计算有关数据类型dnum的反余弦值。

基本示例

以下示例介绍了函数ACosDnum。

例 1

```
VAR dnum angle;
VAR dnum value;
...
...
angle := ACosDnum(value);
```

angle将获得value的反余弦值。

返回值

数据类型：dnum

反余弦值，以度表示，范围为【0，180】。

变元

```
ACosDnum (Value)
```

Value

数据类型：dnum

参数值必须处于范围【-1，1】中。

限制

如果x处于范围【-1，1】外，则函数AcosDnum(x)的执行将出错。

语法

```
AcosDnum '('
  [Value ':='] <expression (IN) of dnum>')'
```

含数据类型dnum的返回值的函数。

相关信息

信息，关于	请参阅
数学指令和函数	技术参考手册 - *RAPID*语言概览

2.5 AInput - 读取模拟信号输入信号值

手册用法

AInput用于读取模拟信号输入信号的当前值。

> **注意**
>
> 注意，函数AInput为遗留函数，其无须再作使用。有关编程的替代方案和建议方法，请参见例子。

基本示例

以下示例介绍了函数AInput。

另请参阅第961页的更多示例

例 1

```
IF AInput(ai1) < 1.5 THEN ...
...
IF ai1 < 1.5 THEN ...
```

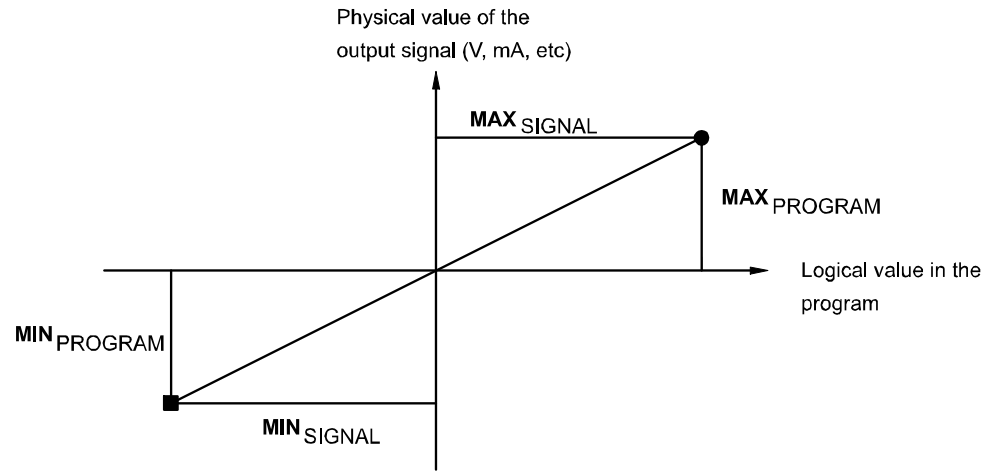
如果信号ai1的当前值小于1.5，则...

返回值

数据类型：num

信号的当前值。

在由RAPID程序读取当前值之前，对于当前值进行测量（按照系统参数）。下图显示了关于如何测量模拟信号值的图表。



xx0500002408

变元

```
AInput (Signal)
```

Signal

数据类型：signalai

待读取模拟信号输入的名称。

下一页继续

更多示例

有关于如何使用函数AInput的更多例子阐述如下。

例 1

```
WHILE AInput(current) > 35 DO ...
...
WHILE current > 35 DO ...
```

只要信号current的当前值大于35，则执行...

例 2

```
deviation := 3 * AInput(sensor) + 10;
...
deviation := 3 * sensor + 10;
```

根据信号sensor的值来计算偏差，并储存在变量deviation中。

语法

```
AInput '('
  [ Signal ':=' ] < variable (VAR) of signalai > ')'
```

返回值的数据类型是 num 的函数。

相关信息

信息，关于	请参阅
输入/输出指令	技术参考手册 - *RAPID*语言概览，*RAPID*概要 - 输入和输出信号一节
输入/输出功能性概述	技术参考手册 - *RAPID*语言概览，运动和*I/O*原则 - *I/O*原则一节
I/O配置	技术参考手册 - 系统参数

2.6 AND - 评估一个逻辑值

手册用法

AND为用于评估两个条件表达式（真/假）的函数。

基本示例

以下示例介绍了函数AND。

例 1

```
VAR num a;
VAR num b;
VAR bool c;
...
c := a>5 AND b=3;
```

如果a大于5，且b等于3，则c的返回值为TRUE。否则，返回值为FALSE。

例 2

```
VAR num mynum;
VAR string mystring;
VAR bool mybool;
VAR bool result;
...
result := mystring="Hello" AND mynum<15 OR mybool;
```

如果mystring为"Hello"，且mynum小于15，则result的返回值为TRUE。或者如果mybool为TRUE。否则，返回值为FALSE。

首先评估AND声明，随后评估OR声明。由以下行中的插入成分对此进行说明。

```
result := (mystring="Hello" AND mynum<15) OR mybool;
```

返回值

数据类型：bool

如果条件表达式均正确，则返回值为TRUE，否侧，返回值为FALSE。

语法

```
<expression of bool> AND <expression of bool>
```

含数据类型bool的返回值的函数。

相关信息

信息，关于	请参阅
byte数据上的逻辑逐位AND运算	第*974*页的*BitAnd - 字节数据上的逻辑逐位AND运算*
dnum数据上的逻辑逐位AND运算	第*976*页的*BitAndDnum - 双数值数据上的逻辑逐位AND运算*
OR	第*1159*页的*OR - 评估一个逻辑值*
XOR	第*1341*页的*XOR - 评估一个逻辑值*
NOT	第*1150*页的*NOT - 转化一个逻辑值*
表达式	技术参考手册 - *RAPID*语言概览

2.7 AOutput - 读取模拟信号输出信号值

手册用法

AOutput用于读取模拟信号输出信号的当前值。

基本示例

以下示例介绍了函数AOutput。

例 1

```
IF AOutput(ao4) > 5 THEN ...
```

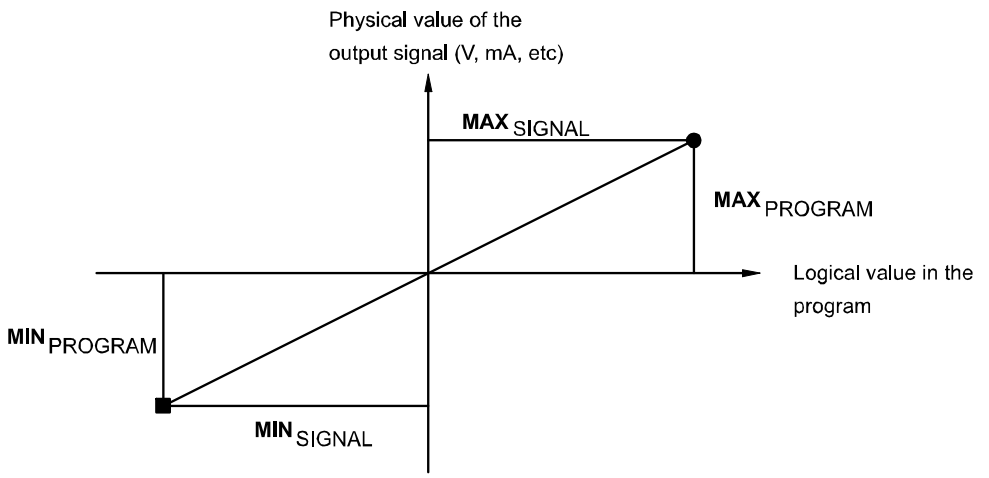
如果信号ao4的当前值大于5，则 ...

返回值

数据类型：num

信号的当前值。

在由RAPID程序读取当前值之前，对于当前值进行测量（按照系统参数）。下图显示了关于如何测量模拟信号值的图表。



xx0500002408

变元

```
AOutput (Signal)
```

Signal

数据类型：signalao

待读取模拟信号输出的名称。

错误处理

系统会生成下列可恢复错误，并在错误处理器中处理这些错误。系统变量ERRNO将被设置成：

如果信号变量是RAPID中声明的变量，则ERR_NO_ALIASIO_DEF。尚未同I/O配置以及指令AliasIO中确定的I/O信号相连。

如果与I/O单元无接触，则ERR_NORUNUNIT。

如果无法访问I/O信号（仅对ICI现场总线有效），则ERR_SIG_NOT_VALID。

下一页继续

2.7 AOutput - 读取模拟信号输出信号值

RobotWare - OS

续前页

语法

```
AOutput '('
  [ Signal ':=' ] < variable (VAR) of signalao > ')'
```

含数据类型num的返回值的函数。

相关信息

信息，关于	请参阅
设置模拟信号输出信号	第*580*页的*SetAO - 改变模拟信号输出信号的值*
输入/输出指令	技术参考手册 - *RAPID*语言概览，*RAPID*概要 - 输入和输出信号一节
输入/输出功能性概述	技术参考手册 - *RAPID*语言概览，运动和*I/O*原则 - *I/O*原则一节
I/O配置	技术参考手册 - 系统参数

2.8 ArgName - 获取参数名

手册用法

ArgName（*Argument Name*）的作用是是获取当前自变数或当前数据所需的原始数据对象的名称。

基本示例

以下示例介绍了函数ArgName。

另请参阅第*966*页的更多示例

例 1

```
VAR num chales :=5;
...
proc1 chales;
PROC proc1 (num par1)
  VAR string name;
  ...
  name:=ArgName(par1);
  TPWrite "Argument name "+name+" with value "\Num:=par1;
ENDPROC
```

为变量name分配字符串值“chales”，并在FlexPendant示教器上写入以下字符串：“Argument name chales with value 5”。

返回值

数据类型：string

原始数据对象名称。

变元

```
ArgName (Parameter [\ErrorNumber])
```

Parameter

数据类型：anytype

正式参数标识符（适用于包含ArgName的程序）或数据标识符。

可以使用有关结构原子、记录、记录成分、数组或数组元素的各类数据。

ErrorNumber

数据类型：errnum

当参数为表达式值时，将保存错误代码的变量（使用前，由系统将其设置为0），参数不存在，或参数为事件类型开关。如果省略该可选变量，则将执行错误处理器。

程序执行

针对有关类型常量、变量或永久数据对象的整个对象，函数会返回原始数据对象名称。原始数据对象可以为程序模块中的全局、局部数据对象，或者为程序中的局部数据对象（普通RAPID范围规则）。

如果其为数据对象的一部分，则返回整个数据对象的名称。

下一页继续

更多示例

有关于函数ArgName的更多例子阐述如下。

从标识符转换为字符串

通过指定有关模块中全局、局部数据对象，或程序范围中局部数据对象的参数Parameter中的标识符，该函数亦可用于从标识符转换为string。

```
VAR num chales :=5;
...
proc1;

PROC proc1 ()
  VAR string name;
  ...
  name:=ArgName(chales);
  TPWrite "Global data object "+name+" has value "\Num:=chales;
ENDPROC
```

为变量name 分配字符串值“chales”，并在FlexPendant示教器上写入以下字符串：“全局数据对象chales拥有值5”。

通过若干步骤进行程序调用

注意，函数会返回原始数据对象名称：

```
VAR num chales :=5;
...
proc1 chales;
...
PROC proc1 (num parameter1)
  ...
  proc2 parameter1;
  ...
ENDPROC

PROC proc2 (num par1)
  VAR string name;
  ...
  name:=ArgName(par1);
  TPWrite "Original data object name "+name+" with value"
       \Num:=par1;
ENDPROC
```

为变量name 分配字符串值“chales”，并在FlexPendant示教器上写入以下字符串：“包含值5的原始数据对象名称chales”。

抑制错误处理器中的执行

```
PROC main()
  VAR string mystring:="DUMMY";
  proc1 mystring;
  proc1 "This is a test";
  ...
ENDPROC
```

下一页继续

```
PROC proc1 (string par1)
  VAR string name;
  VAR errnum myerrnum;

  name := ArgName(par1 \ErrorNumber:=myerrnum);
  IF myerrnum=ERR_ARGNAME THEN
    TPWrite "The argument par1 is an expression value";
    TPWrite "The name of the argument can not be evaluated";
  ELSE
    TPWrite "The name on the argument is "+name;
  ENDIF
ENDPROC
```

当首次调用proc1时，为变量name分配字符串值“mystring”。当第二次调用proc1时，为空字符串分配名称。在FlexPendant示教器上写入以下字符串：“The argument par1 is an expression value”和“The name of the argument can not be evaluated”。

错误处理

如果出现以下错误之一，则将系统变量ERRNO设置为ERR_ARGNAME：

- 参数为表达式值
- 参数不存在
- 参数为事件类型开关

随后，可用错误处理器来处理该错误。

语法

```
ArgName '('
  [ Parameter ':=' ] < reference (REF) of any type>
  ['\' ErrorNumber ':=' <var or pers (INOUT) of errnum>] ')'
```

返回值的数据类型是 string 的函数。

相关信息

信息，关于	请参阅
字符串功能	技术参考手册 - *RAPID*语言概览
字符串的定义	第*1489*页的*string - 字符串*
字符串值	技术参考手册 - *RAPID*语言概览
Advanced RAPID	*Advanced RAPID*

2.9 ASin - 计算反正弦值

手册用法

ASin（*Arc Sine*）用于计算有关数据类型num的反正弦值。

基本示例

以下示例介绍了函数ASin

例 1

```
VAR num angle;
VAR num value;
...
...
angle := ASin(value);
```

angle将获得value的反正弦值

返回值

数据类型：num

反正弦值，以度表示，范围为【-90，90】。

变元

```
ASin (Value)
```

Value

数据类型：num

参数值必须处于范围【-1，1】中。

限制

如果x处于范围【1，-1】外，则函数ASin(x)的执行将出错。

语法

```
ASin '('
  [Value ':='] <expression (IN) of num> ')'
```

返回值的数据类型是 num 的函数。

相关信息

信息，关于	请参阅
数学指令和函数	技术参考手册 - *RAPID*语言概览

2.10 ASinDnum - 计算反正弦值

手册用法

ASinDnum（***Arc Sine dnum***）用于计算有关数据类型dnum的反正弦值。

基本示例

以下示例介绍了函数ASinDnum

例 1

```
VAR dnum angle;
VAR dnum value;
...
...
angle := ASinDnum(value);
```

angle将获得value的反正弦值

返回值

数据类型：dnum

反正弦值，以度表示，范围为【-90，90】。

变元

```
ASinDnum (Value)
```

Value

数据类型：dnum

参数值必须处于范围【-1，1】中。

限制

如果x处于范围【1，-1】外，则函数ASinDnum(x)的执行将出错。

语法

```
ASinDnum '('
  [Value ':='] <expression (IN) of dnum> ')'
```

含数据类型dnum的返回值的函数。

相关信息

信息，关于	请参阅
数学指令和函数	技术参考手册 - *RAPID*语言概览

2.11 ATan - 计算反正切值

手册用法

ATan（*Arc Tangent*）用于计算有关数据类型num的反正切值。

基本示例

以下示例介绍了函数ATan。

例 1

```
VAR num angle;
VAR num value;
...
...
angle := ATan(value);
```

angle将获得value的反正切值。

返回值

数据类型：num

反正切值，以度表示，范围为【-90，90】。

变元

```
ATan (Value)
```

Value

数据类型：num

参数值。

语法

```
ATan '('
  [Value ':='] <expression (IN) of num> ')'
```

返回值的数据类型是 num 的函数。

相关信息

信息，关于	请参阅
数学指令和函数	技术参考手册 - *RAPID*语言概览
包含范围【-180，180】中返回值的反正切	第*972*页的*ATan2* - 计算反正切*2*的值

2.12 ATanDnum - 计算反正切值

手册用法

ATanDnum（*Arc Tangent dnum*）用于计算有关数据类型dnum的反正切值。

基本示例

以下示例介绍了函数ATanDnum。

例 1

```
VAR dnum angle;
VAR dnum value;
...
...
angle := ATanDnum(value);
```

angle将获得value的反正切值。

返回值

数据类型：dnum

反正切值，以度表示，范围为【-90，90】。

变元

```
ATanDnum (Value)
```

Value

数据类型：dnum

参数值。

语法

```
ATanDnum '('
  [Value ':='] <expression (IN) of dnum> ')'
```

含数据类型dnum的返回值的函数。

相关信息

信息，关于	请参阅
数学指令和函数	技术参考手册 - *RAPID*语言概览
包含范围【-180，180】中返回值的反正切	第*972*页的*ATan2 - 计算反正切2*的值

2.13 ATan2 - 计算反正切2的值

手册用法

ATan2（*Arc Tangent2*）用于计算有关数据类型num的反正切2的值。

基本示例

以下示例介绍了函数ATan2。

例 1

```
VAR num angle;
VAR num x_value;
VAR num y_value;
...
...
angle := ATan2(y_value, x_value);
```

angle将获得y_value/x_value的反正切值。

返回值

数据类型：num

反正切值，以度表示，范围为【-180，180】。但是在范围【-180，180】中，该值将等于ATan(y/x)，因为函数使用两种参数的标签，以确定返回值的象限。

变元

```
ATan2 (Y X)
```

Y

数据类型：num

分子参数值。

X

数据类型：num

分母参数值。

语法

```
ATan2 '('
  [Y ':='] <expression (IN) of num> ','
  [X ':='] <expression (IN) of num> ')'
```

返回值的数据类型是 num 的函数。

相关信息

信息，关于	请参阅
数学指令和函数	技术参考手册 - *RAPID*语言概览
仅含一个参数的反正切	第*970*页的*ATan* - 计算反正切值

2.14 ATan2Dnum - 计算反正切2的值

手册用法

ATan2Dnum（*Arc Tangent2 dnum*）用于计算有关数据类型dnum的反正切2的值。

基本示例

以下示例介绍了函数ATan2Dnum。

例 1

```
VAR dnum angle;
VAR dnum x_value;
VAR dnum y_value;
...
...
angle := ATan2Dnum(y_value, x_value);
```

angle将获得y_value/x_value的反正切值。

返回值

数据类型：dnum

反正切值，以度表示，范围为【-180，180】。但是在范围【-180，180】中，该值将等于ATanDnum(y/x)，因为函数使用两种参数的标签，以确定返回值的象限。

变元

```
ATan2Dnum (Y X)
```

Y

数据类型：dnum

分子参数值。

X

数据类型：dnum

分母参数值。

语法

```
ATan2Dnum '('
  [Y ':='] <expression (IN) of dnum> ','
  [X ':='] <expression (IN) of dnum> ')'
```

含数据类型dnum的返回值的函数。

相关信息

信息，关于	请参阅
数学指令和函数	技术参考手册 - *RAPID*语言概览
仅含一个参数的反正切	第*970*页的*ATan* - 计算反正切值

2.15 BitAnd - 字节数据上的逻辑逐位AND运算

手册用法

BitAnd用于在数据类型字节上执行一次逻辑逐位AND运算。

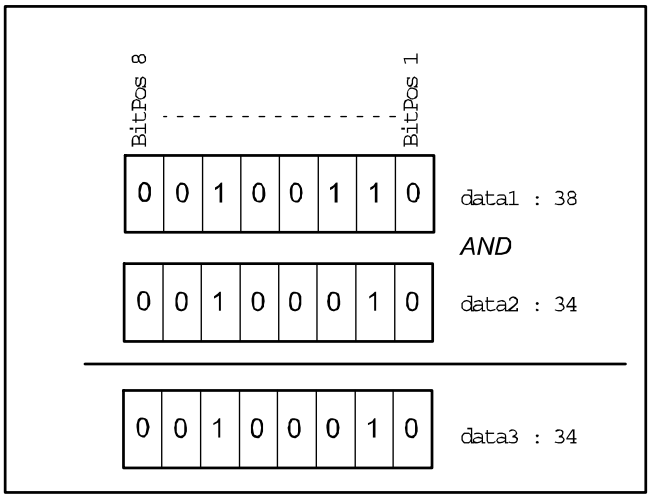
基本示例

以下示例介绍了函数BitAnd。

例 1

```
VAR byte data1 := 38;
VAR byte data2 := 34;
VAR byte data3;

data3 := BitAnd(data1, data2);
```

在data1和data2上执行逻辑逐位AND运算（参见下图）。将结果返回data3（整数表示）。



xx0500002454

返回值

数据类型：byte

用整数表示逻辑逐位AND运算的结果。

变元

```
BitAnd (BitData1 BitData2)
```

BitData1

数据类型：byte

位数据1，以整数表示。

BitData2

数据类型：byte

位数据2，以整数表示。

下一页继续

限制

数据类型byte的范围为0 - 255。

语法

```
BitAnd '('
  [BitData1 ':='] <expression (IN) of byte> ','
  [BitData2 ':='] <expression (IN) of byte> ')'
```

含数据类型byte的返回值的函数。

相关信息

信息，关于	请参阅
字节数据上的逻辑逐位OR运算	第*990*页的*BitOr - 字节数据上的逻辑逐位OR运算*
字节数据上的逻辑逐位XOR运算	第*998*页的*BitXOr - 字节数据上的逻辑逐位XOR运算*
字节数据上的逻辑逐位NEGATION运算	第*986*页的*BitNeg - 字节数据上的逻辑逐位NEGATION运算*
其他位函数	技术参考手册 - *RAPID*语言概览
Advanced RAPID	产品规格 - 控制器软件*IRC5*

2.16 BitAndDnum - 双数值数据上的逻辑逐位AND运算

手册用法

BitAndDnum用于在数据类型dnum上执行一次逻辑逐位AND运算。

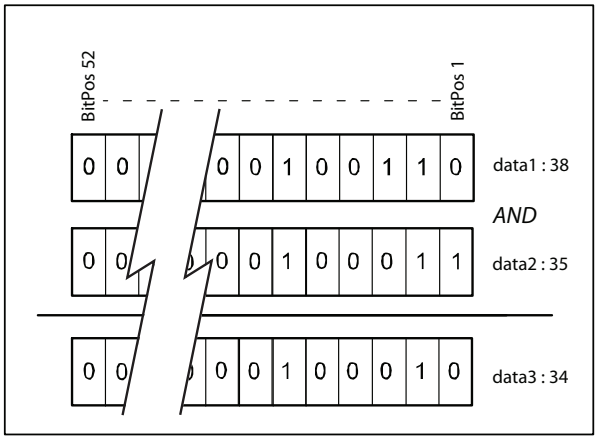
基本示例

以下示例介绍了函数BitAndDnum。

例 1

```
VAR dnum data1 := 38;
VAR dnum data2 := 35;
VAR dnum data3;

data3 := BitAndDnum(data1, data2);
```

将在data1和data2上执行逻辑逐位AND运算（参见下图）。将结果返回data3（整数表示）。



xx1200000007

返回值

数据类型：dnum

用整数表示逻辑逐位AND运算的结果。

变元

```
BitAndDnum (Value1 Value2)
```

Value1

数据类型：dnum

第一个位数据值，以整数表示。

Value2

数据类型：dnum

第二个位数据值，以整数表示。

限制

数据类型dnum的范围为0 - 4503599627370495。

下一页继续

语法

```
BitAndDnum '('
  [Value1 ':='] <expression (IN) of dnum> ','
  [Value2 ':='] <expression (IN) of dnum> ')'
```

含数据类型dnum的返回值的函数。

相关信息

信息，关于	请参阅
byte数据上的逻辑逐位AND运算	第974页的*BitAnd - 字节数据上的逻辑逐位AND运算*
数据类型dnum	第1385页的*dnum - 双数值*
dnum数据上的逻辑逐位OR运算	第992页的*BitOrDnum - 双数值数据上的逻辑逐位OR运算*
dnum数据上的逻辑逐位XOR运算	第1000页的*BitXOrDnum - 双数值数据上的逻辑逐位XOR运算*
dnum数据上的逻辑逐位NEGATION运算	第988页的*BitNegDnum - 双数值数据上的逻辑逐位NEGATION运算*
其他位函数	技术参考手册 - *RAPID*语言概览

2.17 BitCheck - 检查字节数据中的特定位是否设置完毕

手册用法

BitCheck用于检查是否将指定byte数据中的指定位设置为1。

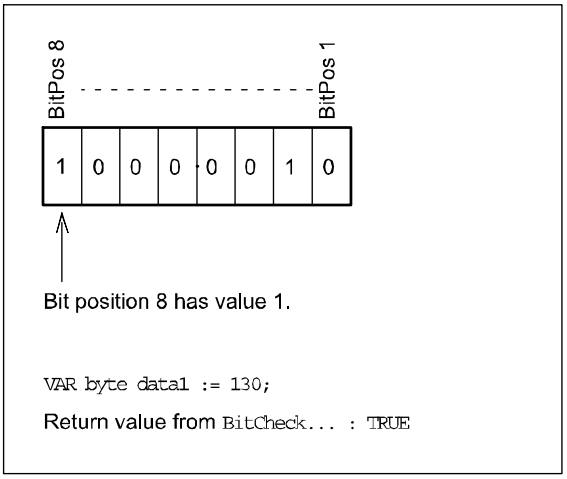
基本示例

以下示例介绍了函数BitCheck。

例 1

```
CONST num parity_bit := 8;
VAR byte data1 := 130;

IF BitCheck(data1, parity_bit) = TRUE THEN
  ...
ELSE
  ...
ENDIF
```

检查变量data1中的位号8（parity_bit），例如，如果在变量data1中将指定位设置为1，则该函数将返回至TRUE。在下图中阐明有关数据类型byte的位检查。



xx0500002442

返回值

数据类型：bool

如果将指定位设置为1，则TRUE；如果将指定位设置为0，则FALSE。

变元

```
BitCheck (BitData BitPos)
```

BitData

数据类型：byte

有待检查的位数据，以整数表示。

BitPos

Bit Position

下一页继续

数据类型：num

位于BitData中的有待检查的数位位置（1-8）。

限制

数据类型byte的范围为0 - 255位小数。

数位位置1-8有效。

语法

```
BitCheck '('
  [BitData ':='] <expression (IN) of byte> ','
  [BitPos ':='] <expression (IN) of num> ')'
```

含数据类型bool的返回值的函数。

相关信息

信息，关于	请参阅
设置字节数据中的指定位	第*37*页的*BitSet* - 在一个字节或者双数值数据中设置一个特定位
清除字节数据中的指定位	第*34*页的*BitClear* - 在一个字节或双数值数据中清除一个特定位
其他位函数	技术参考手册 - *RAPID*语言概览
Advanced RAPID	产品规格 - 控制器软件*IRC5*

2.18 BitCheckDnum - 检查双数值数据中的特定位是否设置完毕

手册用法

BitCheckDnum用于检查是否将指定dnum数据中的指定位设置为1。

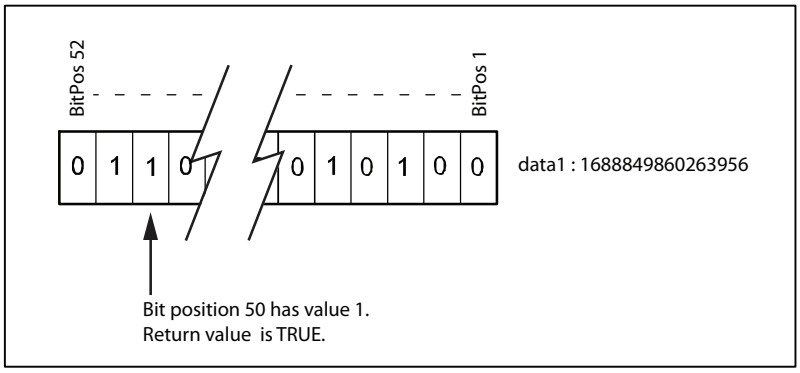
基本示例

以下示例介绍了函数BitCheckDnum。

例 1

```
CONST num check_bit := 50;
VAR dnum data1 := 1688849860263956;

IF BitCheckDnum(data1, check_bit) = TRUE THEN
  ...
ELSE
  ...
ENDIF
```

将检查变量data1中的位号50（check_bit），例如，如果在变量data1中的指定位1，则该函数将返回至TRUE。在下图中阐明有关数据类型dnum的位检查。



xx1200000016

返回值

数据类型：bool

如果将指定位设置为1，则TRUE；如果将指定位设置为0，则FALSE。

变元

```
BitCheckDnum (Value BitPos)
```

Value

数据类型：dnum

有待检查的位数据，以整数表示。

BitPos

Bit Position

数据类型：num

位于Value中的有待检查的数位位置（1-52）。

下一页继续

限制

数据类型dnum的范围为0 - 4503599627370495位小数。

数位位置1-52有效。

语法

```
BitCheckDnum '('
  [Value ':='] <expression (IN) of dnum> ','
  [BitPos ':='] <expression (IN) of num> ')'
```

含数据类型bool的返回值的函数。

相关信息

信息，关于	请参阅
检查是否已设置byte数据中的指定位	第*978*页的*BitCheck* - 检查字节数据中的特定位是否设置完毕
数据类型dnum	第*1385*页的*dnum* - 双数值
设置byte或dnum数据中的指定位	第*37*页的*BitSet* - 在一个字节或者双数值数据中设置一个特定位
清除byte或dnum数据中的指定位	第*34*页的*BitClear* - 在一个字节或双数值数据中清除一个特定位
其他位函数	技术参考手册 - *RAPID*语言概览

2.19 BitLSh - 字节上的逻辑逐位LEFT SHIFT运算

手册用法

BitLSh（*Bit Left Shift*）用于执行数据类型byte上的逻辑逐位LEFT SHIFT运算。

基本示例

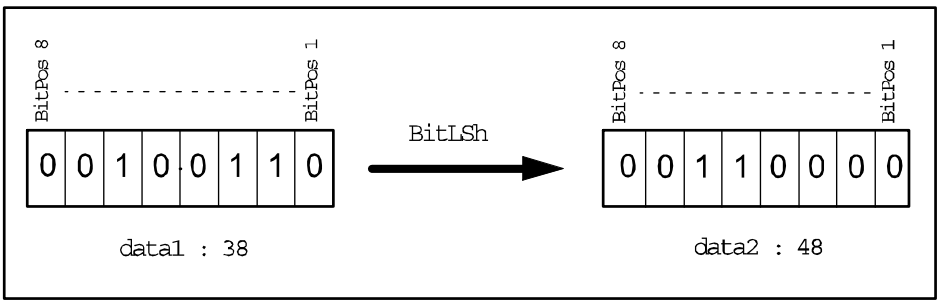
以下示例介绍了函数BitLSh。

例 1

```
VAR num left_shift := 3;
VAR byte data1 := 38;
VAR byte data2;

data2 := BitLSh(data1, left_shift);
```

将通过左移3（left_shift）步，在data1上执行逻辑逐位LEFT SHIFT运算，并将结果返回至data2（整数表示）。

下图显示了逻辑逐位LEFT SHIFT运算。



xx0500002457

返回值

数据类型：byte

用整数表示逻辑逐位LEFT SHIFT运算的结果。

将用0位来填充右数位围笼。

变元

```
BitLSh (BitData ShiftSteps)
```

BitData

数据类型：byte

有待移位的位数据，以整数表示。

ShiftSteps

数据类型：num

有待执行的逻辑移位的编号（1 - 8）。

限制

数据类型byte的范围为0 - 255。

根据一个字节，ShiftSteps参数1 - 8有效。

下一页继续

语法

```
BitLSh '('
  [BitData ':='] <expression (IN) of byte> ','
  [ShiftSteps ':='] <expression (IN) of num> ')'
```

含数据类型byte的返回值的函数。

相关信息

信息，关于	请参阅
字节数据上的逻辑逐位RIGHT SHIFT运算	第*994*页的*BitRSh - 字节上的逻辑逐位RIGHT SHIFT运算*
其他位函数	技术参考手册 - *RAPID*语言概览，*RAPID*概要 - 数学 - 位函数一节
Advanced RAPID	产品规格 - 控制器软件*IRC5*

2.20 BitLShDnum - 双数值上的逻辑逐位LEFT SHIFT运算

手册用法

BitLShDnum（*Bit Left Shift dnum*）用于执行数据类型dnum上的逻辑逐位LEFT SHIFT运算。

基本示例

以下示例介绍了函数BitLShDnum。

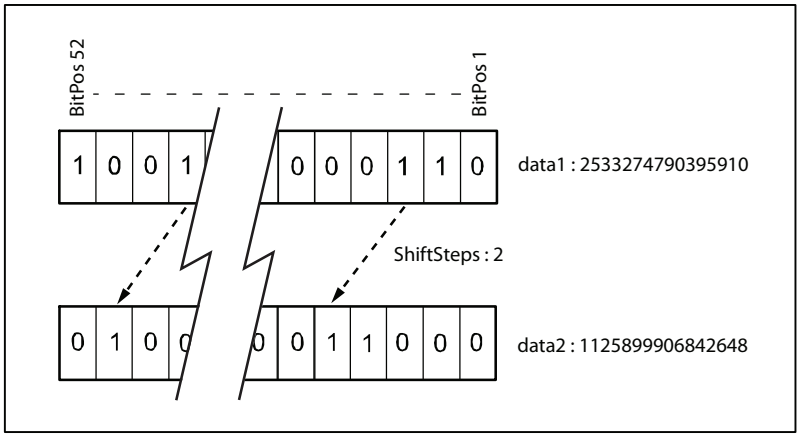
另请参阅第*985*页的更多示例

例 1

```
VAR num left_shift := 2;
VAR dnum data1 := 2533274790395910;
VAR dnum data2;

data2 := BitLShDnum(data1, left_shift);
```

将通过左移2（left_shift）步，在data1上执行逻辑逐位LEFT SHIFT运算，并将结果返回至data2（整数表示）。

下图显示了逻辑逐位LEFT SHIFT运算。



xx1200000008

返回值

数据类型：dnum

用整数表示逻辑逐位LEFT SHIFT运算的结果。

将用0位来填充右数位围笼。

变元

```
BitLShDnum (Value ShiftSteps [\Size])
```

Value

数据类型：dnum

有待移位的位数据，以整数表示。

ShiftSteps

数据类型：num

下一页继续

有待执行的逻辑移位的编号（1 - 52）。

Size

数据类型：num

在参数Value上实施逻辑逐位LEFT SHIFT运算时所应考虑的规模（数位数量）。有效规模为1 - 52。

限制

数据类型dnum的范围为0 - 4503599627370495。

由于一个dnum为52位，因此，ShiftSteps参数1 - 52有效。

更多示例

有关于函数BitLshDnum的更多例子阐述如下。

例 1

```
VAR dnum result;
VAR dnum data1:=221;
! Only consider the 8 lowest bits
result := BitLshDnum(data1, 4 \Size:=8);
TPWrite "" \Dnum:=result;
! Consider all 52 bits in the dnum datatype
result := BitLshDnum(data1, 4);
TPWrite "" \Dnum:=result;
```

将在data1上执行逻辑逐位LEFT SHIFT运算，并将结果返回至result（整数表示）。第一个将在FlexPendant示教器上写入的值为208。第二个将在FlexPendant示教器上写入的值为3536。

语法

```
BitLShDnum '('
  [Value ':='] <expression (IN) of dnum> ','
  [ShiftSteps ':='] <expression (IN) of num>
  ['\' Size ':=' < expression (IN) of num>]
  ')'
```

含数据类型dnum的返回值的函数。

相关信息

信息，关于	请参阅
byte数据上的逻辑逐位LEFT SHIFT运算	第*982*页的*BitLSh - 字节上的逻辑逐位LEFT SHIFT运算*
数据类型dnum	第*1385*页的*dnum - 双数值*
双数值数据上的逻辑逐位RIGHT SHIFT运算	第*996*页的*BitRShDnum - 双数值上的逻辑逐位RIGHT SHIFT运算*
其他位函数	技术参考手册 - *RAPID*语言概览，*RAPID*概要 - 数学 - 位函数一节

2.21 BitNeg - 字节数据上的逻辑逐位NEGATION运算

手册用法

BitNeg（*Bit Negation*）用于在数据类型byte上执行逻辑逐位NEGATION运算（一的补数）。

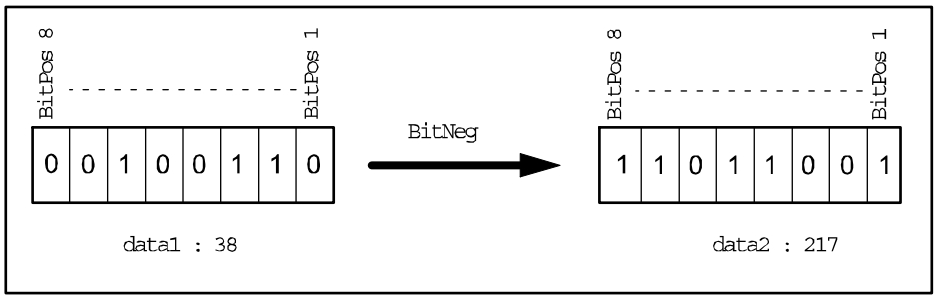
基本示例

以下示例介绍了函数BitNeg。

例 1

```
VAR byte data1 := 38;
VAR byte data2;

data2 := BitNeg(data1);
```

将在data1上执行逻辑逐位NEGATION运算（参见下图），并将结果返回data2（整数表示）。



xx0500002456

返回值

数据类型：byte

用整数表示逻辑逐位NEGATION运算的结果。

变元

```
BitNeg (BitData)
```

BitData

数据类型：byte

位数据，以整数表示。

限制

数据类型byte的范围为0 - 255。

语法

```
BitNeg '('
  [BitData ':='] <expression (IN) of byte>
  ')'
```

含数据类型byte的返回值的函数。

下一页继续

相关信息

信息，关于	请参阅
字节数据上的逻辑逐位AND运算	第*974*页的*BitAnd - 字节数据上的逻辑逐位AND运算*
字节数据上的逻辑逐位OR运算	第*990*页的*BitOr - 字节数据上的逻辑逐位OR运算*
字节数据上的逻辑逐位XOR运算	第*998*页的*BitXOr - 字节数据上的逻辑逐位XOR运算*
其他位函数	技术参考手册 - *RAPID*语言概览
Advanced RAPID	产品规格 - 控制器软件*IRC5*

2.22 BitNegDnum - 双数值数据上的逻辑逐位NEGATION运算

手册用法

BitNegDnum（*Bit Negation dnum*）用于在数据类型dnum上执行逻辑逐位NEGATION运算（一的补数）。

基本示例

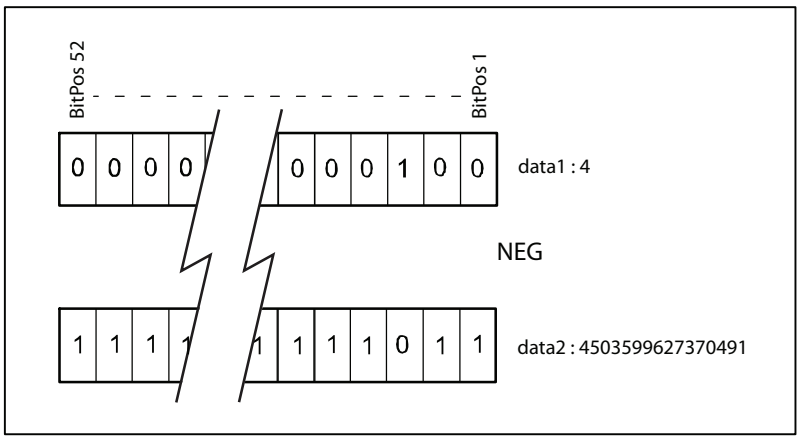
以下示例介绍了函数BitNegDnum。

另请参阅第989页的更多示例

例 1

```
VAR dnum data1 := 4;
VAR dnum data2;

data2 := BitNegDnum(data1);
```

将在data1上执行逻辑逐位NEGATION运算（参见下图），并将结果返回data2（整数表示）。



xx1200000012

返回值

数据类型：dnum

用整数表示逻辑逐位NEGATION运算的结果。

变元

```
BitNegDnum (Value [\Size])
```

Value

数据类型：dnum

双数值数据，以整数表示。

Size

数据类型：num

在参数Value上实施逻辑逐位NEGATION运算时所应考虑的规模（数位数量）。有效规模为1 - 52。

下一页继续

限制

数据类型dnum的范围为0 - 4503599627370495。

更多示例

有关于函数BitNegDnum的更多例子阐述如下。

例 1

```
VAR dnum result;
VAR dnum data1:=38;
! Only consider the 16 lowest bits
result := BitNegDnum(data1 \Size:=16);
TPWrite "" \Dnum:=result;
! Consider all 52 bits in the dnum datatype
result := BitNegDnum(data1);
TPWrite "" \Dnum:=result;
```

将在data1上执行逻辑逐位NEGATION运算，并将结果返回至result（整数表示）。第一个将在FlexPendant示教器上写入的值为65497。第二个将在FlexPendant示教器上写入的值为4503599627370457。

语法

```
BitNegDnum '('
  [Value ':='] <expression (IN) of dnum>
  ['\'Size ':=' < expression (IN) of num>]
  ')'
```

含数据类型dnum的返回值的函数。

相关信息

信息，关于	请参阅
字节数据上的逻辑逐位NEGATION运算	第*986*页的*BitNeg - 字节数据上的逻辑逐位NEGATION*运算
数据类型dnum	第*1385*页的*dnum - 双数值*
双数值数据上的逻辑逐位AND运算	第*976*页的*BitAndDnum - 双数值数据上的逻辑逐位AND*运算
双数值数据上的逻辑逐位OR运算	第*992*页的*BitOrDnum - 双数值数据上的逻辑逐位OR*运算
双数值数据上的逻辑逐位XOR运算	第*1000*页的*BitXOrDnum - 双数值数据上的逻辑逐位XOR*运算
其他位函数	技术参考手册 - *RAPID*语言概览

2.23 BitOr - 字节数据上的逻辑逐位OR运算

手册用法

BitOr（*Bit inclusive Or*）用于在数据类型byte上执行一次逻辑逐位OR运算。

基本示例

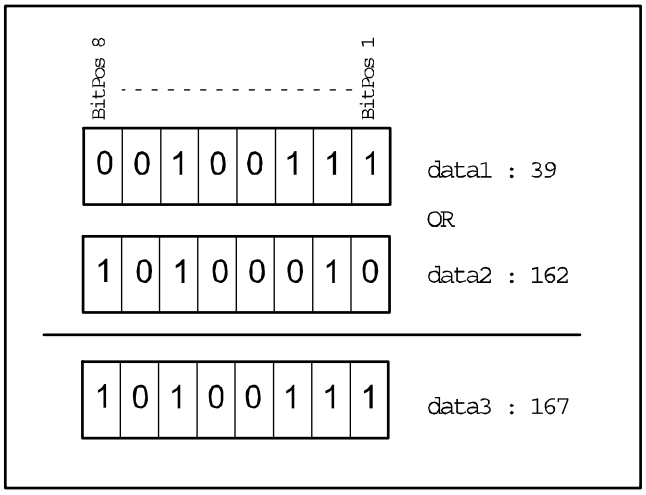
以下示例介绍了函数BitOr。

例 1

```
VAR byte data1 := 39;
VAR byte data2 := 162;
VAR byte data3;

data3 := BitOr(data1, data2);
```

将在data1和data2上执行逻辑逐位OR运算，并将结果返回data3（整数表示）。

下图显示了逻辑逐位OR运算。



xx0500002458

返回值

数据类型：byte

用整数表示逻辑逐位OR运算的结果。

变元

```
BitOr (BitData1 BitData2)
```

BitData1

数据类型：byte

位数据1，以整数表示。

BitData2

数据类型：byte

位数据2，以整数表示。

下一页继续

限制

数据类型byte的范围为0 - 255。

语法

```
BitOr '('
  [BitData1 ':='] <expression (IN) of byte> ','
  [BitData2 ':='] <expression (IN) of byte>
  ')'
```

含数据类型byte的返回值的函数。

相关信息

信息，关于	请参阅
字节数据上的逻辑逐位AND运算	第*974*页的*BitAnd* - 字节数据上的逻辑逐位*AND*运算
字节数据上的逻辑逐位XOR运算	第*998*页的*BitXOr* - 字节数据上的逻辑逐位*XOR*运算
字节数据上的逻辑逐位NEGATION运算	第*986*页的*BitNeg* - 字节数据上的逻辑逐位*NEGATION*运算
其他位函数	技术参考手册 - *RAPID*语言概览
Advanced RAPID	产品规格 - 控制器软件*IRC5*

2.24 BitOrDnum - 双数值数据上的逻辑逐位OR运算

手册用法

BitOrDnum（*Bit inclusive Or dnum*）用于在数据类型dnum上执行一次逻辑逐位OR运算。

基本示例

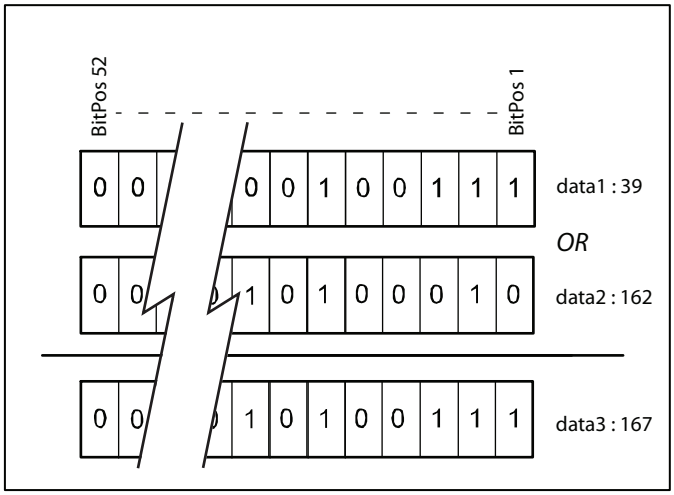
以下示例介绍了函数BitOrDnum。

例 1

```
VAR dnum data1 := 39;
VAR dnum data2 := 162;
VAR dnum data3;

data3 := BitOrDnum(data1, data2);
```

将在data1和data2上执行逻辑逐位OR运算，并将结果返回data3（整数表示）。

下图显示了逻辑逐位OR运算。



xx1200000011

返回值

数据类型：dnum

用整数表示逻辑逐位OR运算的结果。

变元

```
BitOrDnum (Value1 Value2)
```

Value1

数据类型：dnum

第一个位数据值，以整数表示。

Value2

数据类型：dnum

第二个位数据值，以整数表示。

下一页继续

限制

数据类型dnum的范围为0 - 4503599627370495。

语法

```
BitOrDnum '('
  [Value1 ':='] <expression (IN) of dnum> ','
  [Value2 ':='] <expression (IN) of dnum>
  ')'
```

含数据类型dnum的返回值的函数。

相关信息

信息，关于	请参阅
字节数据上的逻辑逐位OR运算	第*990*页的*BitOr - 字节数据上的逻辑逐位OR运算*
数据类型dnum	第*1385*页的*dnum - 双数值*
dnum数据上的逻辑逐位AND运算	第*976*页的*BitAndDnum - 双数值数据上的逻辑逐位AND运算*
dnum数据上的逻辑逐位XOR运算	第*1000*页的*BitXOrDnum - 双数值数据上的逻辑逐位XOR运算*
dnum数据上的逻辑逐位NEGATION运算	第*988*页的*BitNegDnum - 双数值数据上的逻辑逐位NEGATION运算*
其他位函数	技术参考手册 - *RAPID*语言概览

2.25 BitRSh - 字节上的逻辑逐位RIGHT SHIFT运算

手册用法

BitRSh（位右移）用于在数据类型byte上执行一次逻辑逐位RIGHT SHIFT运算。

基本示例

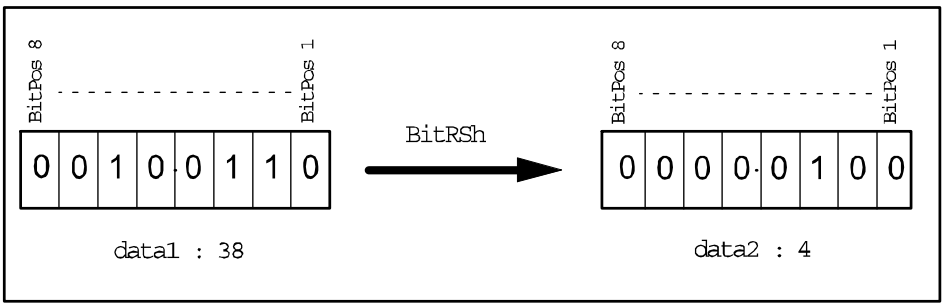
以下示例介绍了函数BitRSh。

例 1

```
VAR num right_shift := 3;
VAR byte data1 := 38;
VAR byte data2;

data2 := BitRSh(data1, right_shift);
```

将通过右移3（right_shift）步，在data1上执行逻辑逐位RIGHT SHIFT运算，并将结果返回至data2（整数表示）。

下图显示了逻辑逐位RIGHT SHIFT运算。



xx0500002455

返回值

数据类型：byte

用整数表示逻辑逐位RIGHT SHIFT运算的结果。

将用0位来填充左数位围笼。

变元

```
BitRSh (BitData ShiftSteps)
```

BitData

数据类型：byte

有待移位的位数据，以整数表示。

ShiftSteps

数据类型：num

有待执行的逻辑移位的编号（1 - 8）。

限制

数据类型byte的范围为0 - 255。

根据一个字节，ShiftSteps参数1 - 8有效。

下一页继续

语法

```
BitRSh '('
  [BitData ':='] <expression (IN) of byte> ','
  [ShiftSteps ':='] <expression (IN) of num>
  ')'
```

含数据类型字节的返回值的函数。

相关信息

信息，关于	请参阅
字节数据上的逻辑逐位LEFT SHIFT运算	第*982*页的*BitLSh - 字节上的逻辑逐位LEFT SHIFT运算*
其他位函数	技术参考手册 - *RAPID*语言概览
Advanced RAPID	产品规格 - 控制器软件*IRC5*

2.26 BitRShDnum - 双数值上的逻辑逐位RIGHT SHIFT运算

手册用法

BitRShDnum（位右移双数值）用于在数据类型dnum上执行一次逻辑逐位RIGHT SHIFT运算。

基本示例

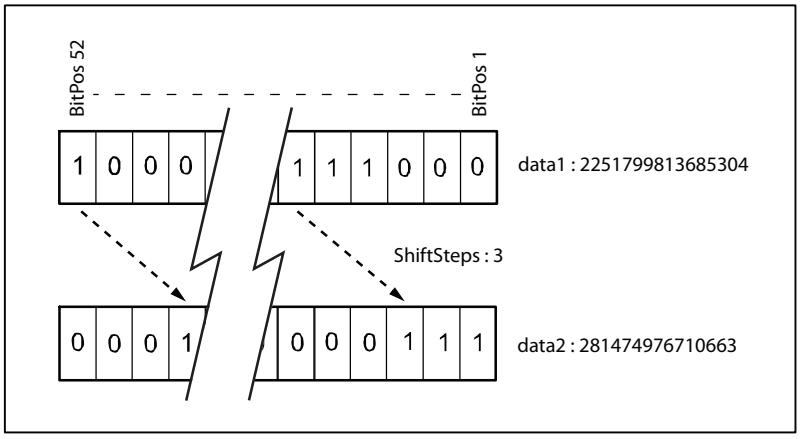
以下示例介绍了函数BitRShDnum。

例 1

```
VAR num right_shift := 3;
VAR dnum data1 := 2251799813685304;
VAR dnum data2;

data2 := BitRShDnum(data1, right_shift);
```

将通过右移3（right_shift）步，在data1上执行逻辑逐位RIGHT SHIFT运算，并将结果返回至data2（整数表示）。

下图显示了逻辑逐位RIGHT SHIFT运算。



xx1200000009

返回值

数据类型：dnum

用整数表示逻辑逐位RIGHT SHIFT运算的结果。

将用0位来填充左数位围笼。

变元

```
BitRShDnum (Value ShiftSteps)
```

Value

数据类型：dnum

有待移位的位数据，以整数表示。

ShiftSteps

数据类型：num

有待执行的逻辑移位的编号（1 - 52）。

下一页继续

限制

数据类型dnum的范围为0 - 4503599627370495。

由于一个双数值为52位，因此，ShiftSteps参数1 - 52有效。

语法

```
BitRShDnum '('
  [Value ':='] <expression (IN) of dnum> ','
  [ShiftSteps ':='] <expression (IN) of num>
  ')'
```

含数据类型dnum的返回值的函数。

相关信息

信息，关于	请参阅
字节数据上的逻辑逐位RIGHT SHIFT运算	第*994*页的*BitRSh - 字节上的逻辑逐位RIGHT SHIFT*运算
数据类型dnum	第*1385*页的*dnum - 双数值*
双数值数据上的逻辑逐位LEFT SHIFT运算	第*984*页的*BitLShDnum - 双数值上的逻辑逐位LEFT SHIFT*运算
其他位函数	技术参考手册 - *RAPID*语言概览

2.27 BitXOr - 字节数据上的逻辑逐位XOR运算

手册用法

BitXOr（*Bit eXclusive Or*）用于执行数据类型byte上的逻辑逐位XOR运算。

基本示例

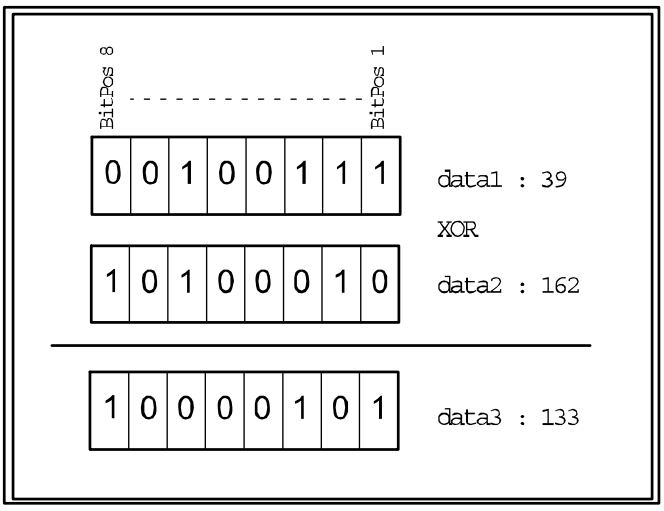
以下示例介绍了函数BitXOr。

例 1

```
VAR byte data1 := 39;
VAR byte data2 := 162;
VAR byte data3;

data3 := BitXOr(data1, data2);
```

将在data1和data2上执行逻辑逐位XOR运算，并将结果返回data3（整数表示）。

下图显示了逻辑逐位XOR运算。



xx0500002459

返回值

数据类型：byte

用整数表示逻辑逐位XOR运算的结果。

变元

```
BitXOr (BitData1 BitData2)
```

BitData1

数据类型：byte

位数据1，以整数表示。

BitData2

数据类型：byte

位数据2，以整数表示。

下一页继续

限制

数据类型byte的范围为0 - 255。

语法

```
BitXOr '('
  [BitData1 ':='] <expression (IN) of byte> ','
  [BitData2 ':='] <expression (IN) of byte>
  ')'
```

含数据类型byte的返回值的函数。

相关信息

信息，关于	请参阅
字节数据上的逻辑逐位AND运算	第*974*页的*BitAnd - 字节数据上的逻辑逐位AND运算*
字节数据上的逻辑逐位OR运算	第*990*页的*BitOr - 字节数据上的逻辑逐位OR运算*
字节数据上的逻辑逐位NEGATION运算	第*986*页的*BitNeg - 字节数据上的逻辑逐位NEGATION运算*
其他位函数	技术参考手册 - *RAPID*语言概览
Advanced RAPID	产品规格 - 控制器软件*IRC5*

2.28 BitXOrDnum - 双数值数据上的逻辑逐位XOR运算

手册用法

BitXOrDnum（*Bit eXclusive Or dnum*）用于执行数据类型dnum上的逻辑逐位XOR运算。

基本示例

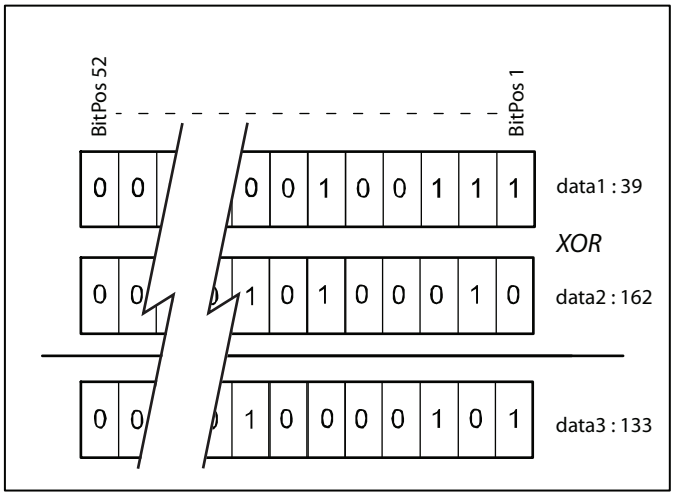
以下示例介绍了函数BitXOrDnum。

例 1

```
VAR dnum data1 := 39;
VAR dnum data2 := 162;
VAR dnum data3;

data3 := BitXOrDnum(data1, data2);
```

将在data1和data2上执行逻辑逐位XOR运算，并将结果返回data3（整数表示）。

下图显示了逻辑逐位XOR运算。



xx1200000010

返回值

数据类型：dnum

用整数表示逻辑逐位XOR运算的结果。

变元

```
BitXOrDnum (Value1 Value2)
```

Value1

数据类型：dnum

第一个位数据值，以整数表示。

Value2

数据类型：dnum

第二个位数据值，以整数表示。

下一页继续

限制

数据类型dnum的范围为0 - 4503599627370495。

语法

```
BitXOrDnum '('
  [Value1 ':='] <expression (IN) of dnum> ','
  [Value2 ':='] <expression (IN) of dnum>
  ')'
```

含数据类型dnum的返回值的函数。

相关信息

信息，关于	请参阅
字节数据上的逻辑逐位XOR运算	第*998*页的*BitXOr - 字节数据上的逻辑逐位XOR运算*
数据类型dnum	第*1385*页的*dnum - 双数值*
双数值数据上的逻辑逐位AND运算	第*976*页的*BitAndDnum - 双数值数据上的逻辑逐位AND运算*
双数值数据上的逻辑逐位OR运算	第*992*页的*BitOrDnum - 双数值数据上的逻辑逐位OR运算*
双数值数据上的逻辑逐位NEGATION运算	第*988*页的*BitNegDnum - 双数值数据上的逻辑逐位NEGATION运算*
其他位函数	技术参考手册 - *RAPID*语言概览

2.29 ByteToStr - 将字节转换为字符串数据

手册用法

ByteToStr（*Byte To String*）用于通过规定字节数据格式，将字节转换成字符串数据。

基本示例

以下示例介绍了函数ByteToStr。

例 1

```
VAR string con_data_buffer{5};
VAR byte data1 := 122;

con_data_buffer{1} := ByteToStr(data1);
```

在ByteToStr ...函数后，数组分量con_data_buffer{1}的容量将为“122”。

```
con_data_buffer{2} := ByteToStr(data1\Hex);
```

在ByteToStr ...函数后，数组分量 con_data_buffer{2}的容量将为“7A”。

```
con_data_buffer{3} := ByteToStr(data1\Okt);
```

在ByteToStr ...函数后，数组分量con_data_buffer{3}的容量将为“172”。

```
con_data_buffer{4} := ByteToStr(data1\Bin);
```

在ByteToStr ...函数后，数组分量con_data_buffer{4}的容量将为“01111010”。

```
con_data_buffer{5} := ByteToStr(data1\Char);
```

在ByteToStr ...函数后，数组分量con_data_buffer{5}的容量将为“z”。

返回值

数据类型：string

采用以下格式的转换运算结果：

格式	字符	字符串长度	范围
Dec .....:	‘0’-‘9’	1-3	‘0’-‘255’
Hex .....:	‘0’-‘9’,‘A’-‘F’	2	"00" - "FF"
Okt .....:	‘0’-‘7’	3	‘000’-‘377’
Bin .....:	‘0’-‘1’	8	‘00000000’-‘11111111’
Char ....:	任意ASCII字符（*）	1	一个ASCII字符

(*)如果其为不可写入的ASCII字符，则返回格式将为RAPID字符代码格式（例如，有关BEL控制字符的“\07”）。

变元

```
ByteToStr (BitData [\Hex] | [\Okt] | [\Bin] | [\Char])
```

BitData

数据类型：byte

有待转换的位数据。

如果省略可选开关参数，则将以decimal（Dec）格式来转换数据。

下一页继续

[\Hex]

Hexadecimal

数据类型：switch

将以hexadecimal格式来转换数据。

[\Okt]

Octal

数据类型：switch

将以octal格式来转换数据。

[\Bin]

Binary

数据类型：switch

将以 binary格式来转换数据。

[\Char]

Character

数据类型：switch

将以ASCII字符格式来转换数据。

限制

数据类型byte的范围为0到255十进制。

语法

```
ByteToStr '('
  [BitData ':='] <expression (IN) of byte>
  ['\' Hex ] | ['\' Okt] | ['\' Bin] | ['\' Char]
  ')'
```

返回值的数据类型是 string 的函数。

相关信息

信息，关于	请参阅
将一段字符串转换为一个字节数据	第*1262*页的*StrToByte* - 将一段字符串转换为一个字节数据
其他位（字节）函数	技术参考手册 - *RAPID*语言概览
其他字符串函数	技术参考手册 - *RAPID*语言概览

2.30 CalcJointT - 计算机器人位置的接头角

手册用法

CalcJointT（*Calculate Joint Target*）用于计算来自指定 robtarget数据的机械臂轴和外轴的接头角。

按照Tool、WObj相关参数中的规定，在有效程序位移（ProgDisp）和外轴偏移量(EOffs)执行期间，应当用相同的坐标系来指定输入robtarget数据。用校准坐标系来表示已返回的jointtarget数据。

如果MultiMove应用类型与协调工件形成半协调或同步协调模式，并通过位于另一程序任务中的一些机械单元来移动，则在以下情况下，可使用函数CalcJointT：

- 在计算（当前用户坐标系）中使用由机械臂单元所移动的已协调工件的当前位置是适当的。将从RAPID程序获取所有其他数据。
- 位于另一程序任务中的机械单元静止不动。
- 使用参数\UseCurWObjPos。

基本示例

以下示例介绍了函数CalcJointT。

例 1

```
VAR jointtarget jointpos1;
CONST robtarget p1 := [...];
jointpos1 := CalcJointT(p1, tool1 \WObj:=wobj1);
```

将符合robtarget 值p1 的jointtarget 值储存在 jointpos1中。工具tool1和工件wobj1用于计算接头角jointpos1。

例 2

```
VAR jointtarget jointpos2;
CONST robtarget p2 := [...];
jointpos2 := CalcJointT(\UseCurWObjPos, p2, tool2 \WObj:=orb1);
```

将符合robtarget 值p2的jointtarget 值储存在jointpos2中。工具tool2和工件orb1用于计算接头角jointpos2。静止机械臂orb1的当前位置并非同TCP机械臂一样位于相同的程序任务中，而是用于计算。

例 3

```
VAR jointtarget jointpos3;
CONST robtarget p3 := [...];
VAR errnum myerrnum;
jointpos3 := CalcJointT(p3, tool2 \WObj:=orb1
     \ErrorNumber:=myerrnum);
IF myerrnum = ERR_ROBLIMIT THEN
  TPWrite "Joint jointpos3 can not be reached.";
  TPWrite "jointpos3.robax.rax_1: "+ValToStr(jointpos3.robax.rax_1);
  ..
  ..
  TPWrite "jointpos3.extax.eax_f"+ValToStr(jointpos3.extax.eax_f);
ELSEIF myerrnum = ERR_OUTSIDE_REACH THEN
  TPWrite "Joint jointpos3 is outside reach.";
  TPWrite "jointpos3.robax.rax_1: "+ValToStr(jointpos3.robax.rax_1);
```

下一页继续

```
  ..
  ..
  TPWrite "jointpos3.extax.eax_f"+ValToStr(jointpos3.extax.eax_f);
ELSE
  MoveAbsJ jointpos3, v100, fine, tool2 \WObj:=orb1;
ENDIF
```

将符合robtarget值p3的jointtarget值储存在jointpos3中。如果可达到该位置，则予以使用，否则，将jointtarget值写入到FlexPendant示教器上。

返回值

数据类型：jointtarget

有关臂侧上机械臂各轴的角度，以度计。

有关外轴的值，线性轴以mm计，旋转轴以度计。

返回值始终与校准位置相关。

变元

```
CalcJointT ( [\UseCurWObjPos] Rob_target Tool [\WObj]
     [\ErrorNumber])
```

[\UseCurWObjPos]

数据类型：switch

将另一个任务中的机械单元所移动的协调工件的当前位置用于计算（当前用户坐标系）。将从RAPID程序获取所有其他数据。

Rob_target

数据类型： robtarget

最外面坐标系中的机械臂和外轴的位置与有效程序位移(ProgDisp) 和/或外轴偏移量(EOffs)执行期间的指定工具和工件相关。

Tool

数据类型： tooldata

用于计算机械臂接头角的工具。

[\WObj]

Work Object

数据类型： wobjdata

同机械臂位置相关的工件（坐标系）。

如果省略该参数，则使用工件wobj0。当使用固定工具、协调外轴或传送带时，必须详细说明该参数。

[\ErrorNumber]

Error number

数据类型：errnum

如果至少一个轴在关节限制区外或如果超出限制至少一个耦合关节，则为保存错误常量 ERR_ROBLIMIT 的变量（VAR 或 PERS），如果位置 (robtarget) 在机器人的工作区域外，则为保存 ERR_OUTSIDE_REACH。如果使用了此可选变元且执行了功能后变

下一页继续

2.30 CalcJointT - 计算机器人位置的接头角
RobotWare - OS
续前页

量设置为 ERR_ROBLIMIT 或 ERR_OUTSIDE_REACH，则返回值将为对应所用的 robtarget 的 jointtarget 值。

如果省略该可选变量，则将执行错误处理器，且如果一个轴位于工作区域外或超出限制，则将不会更新返回的jointarget。

程序执行

根据输入robtarget，计算返回的jointtarget。如果使用参数\UseCurWObjPos，则所用位置来自用以控制用户坐标系的机械单元的当前位置。为计算机械臂接头角，应考虑指定的Tool、WObj（包括协调的用户坐标系）以及在执行期间有效的ProgDisp。为计算执行期间的外轴位置，应考虑有效的EOffs。

计算始终根据输入robtarget数据中的指定配置数据来选择机械臂配置。指令ConfL和ConfJ不会影响该计算原则。当使用腕奇异点时，将机械臂轴4设置为0度。

如果在机器人位置得以储存时，存在任何有效的程序位移（ProgDisp）和/或外轴偏移量（EOffs），则在执行CalcJointT时，相同的程序位移和/或外轴偏移量必须有效。

限制

如果使用协调坐标系，则在使用CalcJointT前，必须启用协调的单元。

在与执行CalcJointT的TCP机械臂相同的任务中，通常必须可利用用于控制工件中用户坐标系的机械单元。

CalcJointT通常使用来自RAPID程序的robtarget、tooldata和wobjdata，以计算jointtarget。有关协调的工件，将机械单元位置作为robtarget中的外轴位置。如果机械单元由另一个程序任务（MultiMove系统）控制，或者机械单元并非由控制系统（传送带）控制，则除外。针对除传送带外的MultiMove系统，如果机械单元在CalCJointT执行期间静止不动，则使用参数\UseCurWObjPos是可能的。

错误处理

如果位置可以到达打，但是至少一个轴在关节限制外或超出限制至少一个耦合关节，则系统变量 ERRNO 设置为 ERR_ROBLIMIT，执行在错误处理程序中继续。

如果位置（机器人位置）位于机械臂工作范围外，则将系统变量ERRNO设置为 ERR_OUTSIDE_REACH，并用错误处理器继续执行。

如果用于控制工件（用户坐标系）机械单元，未在执行CalJointT \UseCurWobjPos期间静止不动，则将系统变量ERRNO设置为ERR_WOBJ_MOVING，并用错误处理器继续执行。

随后，错误处理器可处理各种情形。

语法

```
CalcJointT '('
  [ '\'UseCurWObjPos ',']
  [ Rob_target ':='] <expression (IN) of robtarget> ','
  [ Tool ':=' ] <persistent (PERS) of tooldata>
  [ '\' WObj ':=' <persistent (PERS) of wobjdata>]
  [ '\' ErrorNumber ':=' <variable or persistent (INOUT) of errnum>]
  ')'
```

含数据类型jointtarget的返回值的函数。

下一页继续

相关信息

信息，关于	请参阅
根据接头位置，计算机器人位置	第*1008*页的*CalcRobT - 根据接头位置，计算机器人位置*
位置的定义	第*1467*页的*robtarget - 位置数据*
接头位置的定义	第*1418*页的*jointtarget - 接头位置数据*
工具的定义	第*1502*页的*tooldata - 工具数据*
工件的定义	第*1523*页的*wobjdata - 工件数据*
坐标系	技术参考手册 - *RAPID*语言概览
程序位移坐标系	第*448*页的*PDispOn - 启用程序位移*
外轴偏移量坐标系	第*187*页的*EOffsOn - 启用附加轴的偏移量*

2.31 CalcRobT - 根据接头位置，计算机器人位置

手册用法

CalcRobT（*Calculate Robot Target*）用于计算来自给定jointtarget数据的robtarget数据。

该函数返回robtarget值以及位置（x，y，z）、方位（q1 ... q4）、机械臂轴配置和外轴位置。

应当用校准坐标系指定输入jointtarget数据。

用最外面的坐标系表示返回的robtarget数据。其将指定工具、工件以及执行期间的有效程序位移（ProgDisp）和外轴偏移量（EOffs）纳入考虑。

基本示例

以下示例介绍了函数CalcRobT。

例 1

```
VAR robtarget p1;
CONST jointtarget jointpos1 := [...];

p1 := CalcRobT(jointpos1, tool1 \WObj:=wobj1);
```

将符合jointtarget值jointpos1的robtarget值储存在 p1中。工具tool1和工件wobj1用于计算p1的位置。

返回值

数据类型： robtarget

用数据类型robtarget返回机械臂和外轴位置，并用最外面的坐标系来表示机械臂和外轴位置。其将指定工具、工件以及执行期间的有效程序位移（ProgDisp）和外轴偏移量（EOffs）纳入考虑。

如果不存在有效的ProgDisp，则用工件坐标系来表示机械臂位置。如果不存在有效的EOffs，则用校准坐标系来表示外轴位置。

变元

```
CalcRobT( Joint_target Tool [\WObj] )
```

Joint_target

数据类型：jointtarget

同校准坐标系相关的机械臂轴和外轴的接头位置。

Tool

数据类型： tooldata

用于计算机械臂位置的工具。

[\WObj]

Work Object

数据类型： wobjdata

同函数所返回的机械臂位置相关的工件（坐标系）。

下一页继续

如果省略该参数，则使用工件wobj0。当使用固定工具、协调外轴或传送带时，必须详细说明该参数。

程序执行

根据输入jointtarget，计算返回的robtarget。为计算笛卡尔机械臂位置，应将指定的Tool、WObj（包括协调的用户坐标系）以及执行期间的有效ProgDisp纳入考虑。

为计算外轴位置，亦将执行期间的有效EOffs纳入考虑。

限制

如果使用协调坐标系，则在使用CalcRobT之前，必须启用协调单元。同时必须将协调单元用于与机械臂相同的任务中。

语法

```
CalcRobT '('
  [Joint_target ':=' ] <expression (IN) of jointtarget> ','
  [Tool ':=' ] <persistent (PERS) of tooldata>
  ['\' WObj ':=' <persistent (PERS) of wobjdata>] ')'
```

含数据类型robtarget的返回值的函数。

相关信息

信息，关于	请参阅
根据机器人位置，计算接头位置	第*1004*页的*CalcJointT - 计算机器人位置的接头角*
位置的定义	第*1467*页的*robtarget - 位置数据*
接头位置的定义	第*1418*页的*jointtarget - 接头位置数据*
工具的定义	第*1502*页的*tooldata - 工具数据*
工件的定义	第*1523*页的*wobjdata - 工件数据*
坐标系	技术参考手册 - *RAPID*语言概览
程序位移坐标系	第*448*页的*PDispOn - 启用程序位移*
外轴偏移量坐标系	第*187*页的*EOffsOn - 启用附加轴的偏移量*

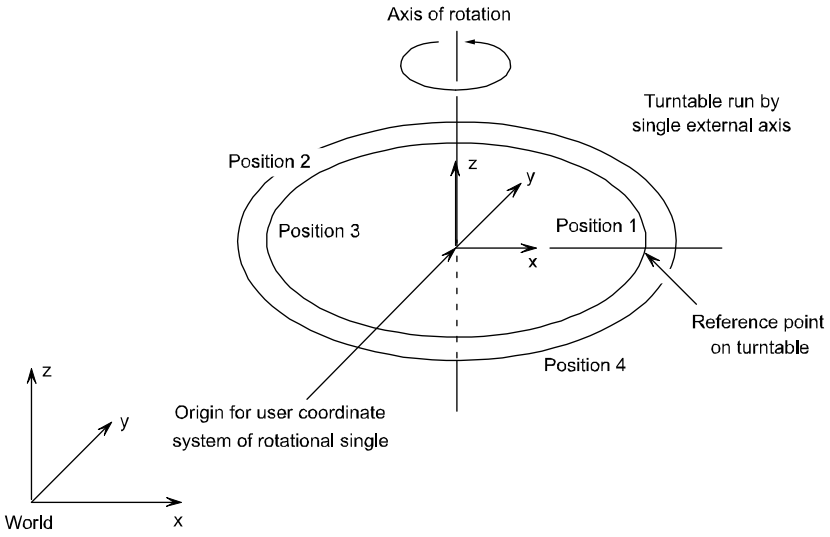
2.32 CalcRotAxFrameZ - 计算一个旋转轴坐标系

手册用法

`CalcRotAxFrameZ`（***Calculate Rotational Axis Frame with positive Z-point***）用于计算旋转轴式机械单元的用户坐标系。当主机械臂和附加轴位于不同的RAPID任务中时，将使用该函数。如果他们位于相同的任务，则应当使用函数`CalcRotAxisFrame`。

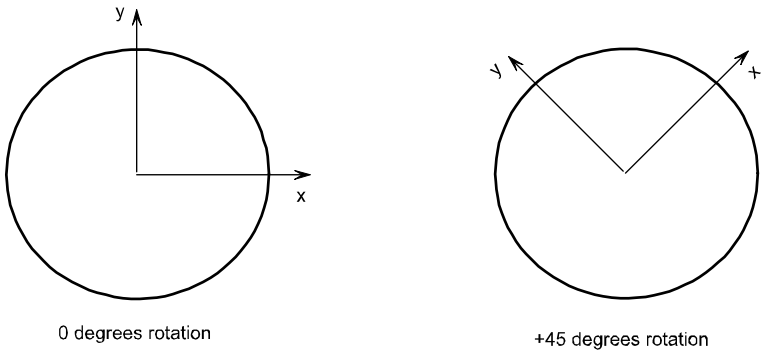
描述

旋转外轴的用户坐标系的定义要求位于外轴上的转盘（或类似的机械结构）标记有参考点。此外，必须校准TCP机械臂的基准坐标系和TCP。当转盘旋转至不同角度时，校准无返回值程序包含有关参考点上机械臂TCP的许多位置。同时需要对正z方向上的机械臂TCP进行定位。有关旋转轴各点的定义，请参见下图。



xx0500002468

有关旋转轴的用户坐标系源于转盘中心。z方向同轴的旋转一致，且x轴经过参考点。下图显示了有关转盘的两个不同位置的坐标系（从上方观察转盘）。



xx0500002469

下一页继续

基本示例

以下示例介绍了函数CalcRotAxFrameZ。

例 1

```
CONST robtarget pos1 := [...];
CONST robtarget pos2 := [...];
CONST robtarget pos3 := [...];
CONST robtarget pos4 := [...];
CONST robtarget zpos;
VAR robtarget targetlist{10};
VAR num max_err := 0;
VAR num mean_err := 0;
VAR pose resFr:=[...];
PERS tooldata tMyTool:= [...];

! Instructions for creating/ModPos pos1 - pos4 with TCP pointing
     at the turntable.
MoveJ pos1, v10, fine, tMyTool;
MoveJ pos2, v10, fine, tMyTool;
MoveJ pos3, v10, fine, tMyTool;
MoveJ pos4, v10, fine, tMyTool;

!Instruction for creating/ModPos zpos with TCP pointing at a point
     in positive z direction
MoveJ zpos, v10, fine, tMyTool;

! Add the targets to the array
targetlist{1}:= pos1;
targetlist{2}:= pos2;
targetlist{3}:= pos3;
targetlist{4}:= pos4;

resFr:=CalcRotAxFrameZ(targetlist, 4, zpos, max_err, mean_err);

! Update the system parameters.
IF (max_err < 1.0) AND (mean_err < 0.5) THEN
  WriteCfgData "/MOC/SINGLE/STN_1", "base_frame_pos_x",
       resFr.trans.x/1000;
  WriteCfgData "/MOC/SINGLE/STN_1", "base_frame_pos_y",
       resFr.trans.y/1000;
  WriteCfgData "/MOC/SINGLE/STN_1", "base_frame_pos_z",
       resFr.trans.z/1000;
  WriteCfgData "/MOC/SINGLE/STN_1", "base_frame_orient_u0",
       resFr.rot.q1;
  WriteCfgData "/MOC/SINGLE/STN_1", "base_frame_orient_u1",
       resFr.rot.q2;
  WriteCfgData "/MOC/SINGLE/STN_1", "base_frame_orient_u2",
       resFr.rot.q3;
  WriteCfgData "/MOC/SINGLE/STN_1", "base_frame_orient_u3",
       resFr.rot.q4;
  TPReadFK reg1,"Warmstart required for calibration to take
       effect.", stEmpty, stEmpty, stEmpty, stEmpty,"OK";
```

下一页继续

2.32 CalcRotAxFrameZ - 计算一个旋转轴坐标系

RobotWare - OS

续前页

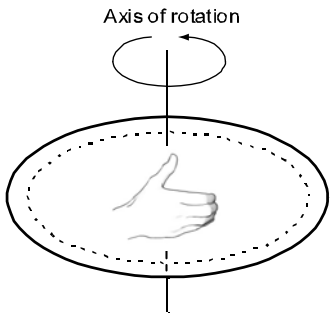
```
    WarmStart;
ENDIF
```

创建/修改四个位置，pos1 - pos4，以便机械臂工具tMyTool点与外轴STN_1上的参考点相同，但是拥有不同的外轴旋转。创建/修改位置zpos，以便正z方向的机械臂工具tMyTool点符合外部旋转机械单元正z方向的定义。使用外部旋转机械单元正z方向的定义，请参见第1010页的描述。随后，各点用于计算与世界坐标系相关的外轴基准坐标系resFr。最后，将坐标系写入配置文件，并执行WarmStart指令，从而使改变生效。

注意

外部旋转机械单元正z方向的定义：

使右手手指与旋转轴的正旋转轴一致。随后，拇指方向规定正z方向。参见下图。



xx0500002472

返回值

数据类型：pose

计算出的坐标系。

变元

```
CalcRotAxFrameZ (TargetList TargetsInList PositiveZPoint
    MaxErrMeanErr)
```

TargetList

数据类型： robtarget

机器人位置的数组通过指出转盘，以此保存确定的位置。机器人位置的最小数量为4，最大为10。

TargetsInList

数据类型：num

一个数组中的机器人位置的数量。

PositiveZPoint

数据类型： robtarget

robtarget通过指出正z方向中的一个点，以此保存确定的位置。使用外部旋转机械单元正z方向的定义，参见第1010页的描述。

下一页继续

MaxErr

Maximum Error

数据类型：num

预估最大误差，以mm计。

MeanErr

Mean Error

数据类型：num

预估平均误差，以mm计。

错误处理

如果位置不具有所需关系，或者未通过足够精度来指定，则将系统变量ERRNO设置为ERR_FRAME。随后，可通过错误处理器来处理该错误。

语法

```
CalcRotAxFrameZ '('
  [TargetList ':='] <array {*} (IN) of robtarget> ','
  [TargetsInList ':='] <expression (IN) of num> ','
  [PositiveZPoint ':='] <expression (IN) of robtarget> ','
  [MaxErr ':='] <variable (VAR) of num> ','
  [MeanErr ':='] <variable (VAR) of num> ')'
```

含数据类型pose的返回值的函数。

相关信息

信息，关于	请参阅
数学指令和函数	技术参考手册 - *RAPID*语言概览

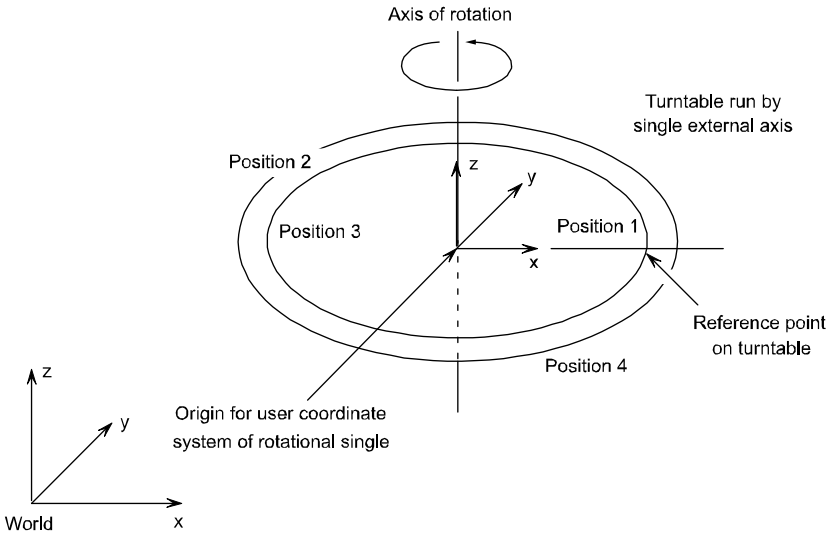
2.33 CalcRotAxisFrame - 计算一个旋转轴坐标系

手册用法

`CalcRotAxisFrame`（*Calculate Rotational Axis Frame*）用于计算旋转轴式机械单元的用户坐标系。当主机械臂和附加轴位于相同RAPID任务中时，使用该函数。如果他们位于不同的任务中，则应当使用函数`CalcRotAxFrameZ`。

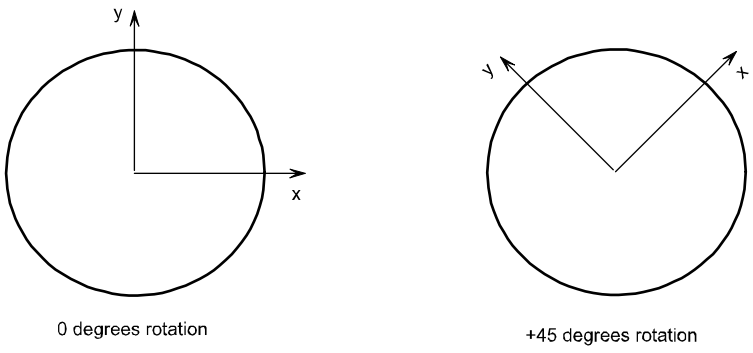
描述

旋转外轴的用户坐标系的定义要求位于外轴上的转盘（或类似的机械结构）标记有参考点。此外，必须校准主机械臂的基准坐标系和TCP。当转盘旋转至不同角度时，校准无返回值程序包含有关参考点上机械臂TCP的许多位置。下图阐明了有关旋转轴各点的定义。



xx0500002468

有关旋转轴的用户坐标系源于转盘中心。z方向同轴的旋转一致，且x轴经过参考点。下图显示了有关转盘的两个不同位置的坐标系（从上方观察转盘）。



xx0500002469

下一页继续

基本示例

以下示例介绍了函数CalcRotAxisFrame。

例 1

```
CONST robtarget pos1 := [...];
CONST robtarget pos2 := [...];
CONST robtarget pos3 := [...];
CONST robtarget pos4 := [...];
VAR robtarget targetlist{10};
VAR num max_err := 0;
VAR num mean_err := 0;
VAR pose resFr:=[...];
PERS tooldata tMyTool:= [...];

! Instructions needed for creating/ModPos pos1 - pos4 with TCP
     pointing at the turntable.
MoveJ pos1, v10, fine, tMyTool;
MoveJ pos2, v10, fine, tMyTool;
MoveJ pos3, v10, fine, tMyTool;
MoveJ pos4, v10, fine, tMyTool;

! Add the targets to the array
targetlist{1}:= pos1;
targetlist{2}:= pos2;
targetlist{3}:= pos3;
targetlist{4}:= pos4;

resFr:=CalcRotAxisFrame(STN_1 , targetlist, 4, max_err, mean_err);

! Update the system parameters.
IF (max_err < 1.0) AND (mean_err < 0.5) THEN
  WriteCfgData "/MOC/SINGLE/STN_1", "base_frame_pos_x",
       resFr.trans.x/1000;
  WriteCfgData "/MOC/SINGLE/STN_1", "base_frame_pos_y",
       resFr.trans.y/1000;
  WriteCfgData "/MOC/SINGLE/STN_1", "base_frame_pos_z",
       resFr.trans.z/1000;
  WriteCfgData "/MOC/SINGLE/STN_1", "base_frame_orient_u0",
       resFr.rot.q1;
  WriteCfgData "/MOC/SINGLE/STN_1", "base_frame_orient_u1",
       resFr.rot.q2;
  WriteCfgData "/MOC/SINGLE/STN_1", "base_frame_orient_u2",
       resFr.rot.q3;
  WriteCfgData "/MOC/SINGLE/STN_1", "base_frame_orient_u3",
       resFr.rot.q4;
  TPReadFK reg1,"Warmstart required for calibration to take
       effect.", stEmpty, stEmpty, stEmpty, stEmpty, "OK";
  WarmStart;
ENDIF
```

创建/修改四个位置，pos1 - pos4，以便机械臂工具tMyTool指向外轴STN_1上的相同参考点，但是其具有不同的外轴旋转。随后，各点用于计算同世界坐标系相关的外

下一页继续

2.33 CalcRotAxisFrame - 计算一个旋转轴坐标系

RobotWare - OS

续前页

轴基准坐标系resFr。最后，将坐标系写入配置文件，并执行WarmStart指令，从而使改变起效。

返回值

数据类型：pose

计算出的坐标系。

变元

```
CalcRotAxisFrame (MechUnit [\AxisNo] TargetList TargetsInList MaxErr
    MeanErr)
```

MechUnit

Mechanical Unit

数据类型：mecunit

待校准机械单元的名称。

[\AxisNo]

数据类型：num

应当确定有关规定轴编号的可选参数的坐标系。默认值为1，适用于单一旋转轴。针对含多个轴的机械单元，轴编号应当包括该参数。

TargetList

数据类型： robtarget

机器人位置的数组通过指出转盘，以此保存确定的位置。机器人位置的最小数量为4，最大为10。

TargetsInList

数据类型：num

一个数组中的机器人位置的数量。

MaxErr

Maximum Error

数据类型：num

预估最大误差，以mm计。

MeanErr

Mean Error

数据类型：num

预估平均误差，以mm计。

错误处理

如果位置不具有所需关系，或者未通过足够精度来指定，则将系统变量ERRNO设置为ERR_FRAME。随后，可通过错误处理器来处理该错误。

语法

```
CalcRotAxisFrame '('
  [MechUnit ':='] <variable (VAR) of mecunit>
  [\AxisNo ':=' <expression (IN) of num> ] ','
```

下一页继续

```
[TargetList ':='] <array {*} (IN) of robtarget> ','
[TargetsInList ':='] <expression (IN) of num> ','
[MaxErr ':='] <variable (VAR) of num> ','
[MeanErr ':='] <variable (VAR) of num> ')'
```

含数据类型pose的返回值的函数。

相关信息

信息，关于	请参阅
数学指令和函数	技术参考手册 - *RAPID*语言概览

2.34 CamGetExposure - 获取具体摄像头的数据

手册用法

CamGetExposure (Camera Get Exposure) 是一个读取当前摄像头设置的函数。使用此函数与 CamSetExposure 指令可以在运行时根据环境调整摄像头图像。

基本示例

以下示例介绍了 CamGetExposure 函数。

例 1

```
VAR num exposuretime;
...
exposuretime:=CamGetExposure(mycamera \ExposureTime);
IF exposuretime = 10 THEN
  CamSetExposure mycamera \ExposureTime:=9.5;
ENDIF
```

如果当前曝光时间设置为 10 ms，则命令摄像头 mycamera 将曝光时间改为 9.5 ms。

返回值

数据类型：num

从摄像头以数值方式返回的曝光时间、亮度或对比度中的某个设置。

变元

```
CamGetExposure (Camera [\ExposureTime] | [\Brightness] |
     [\Contrast])
```

Camera

数据类型：cameradev

摄像头名称。

[\ExposureTime]

数据类型：num

返回摄像头曝光时间。其值以毫秒 (ms) 为单位。

[\Brightness]

数据类型：num

返回摄像头的亮度设置。

[\Contrast]

数据类型：num

返回摄像头的对比度设置。

错误处理

可能会产生下列可恢复错误。错误可以由错误处理程序处理。系统变量 ERRNO 将设置为：

名称	错误原因
ERR_CAM_BUSY	摄像头正忙于处理其他请求，无法执行当前命令。

下一页继续

名称	错误原因
ERR_CAM_COM_TIMEOUT	与摄像头通信错误。摄像头可能已断开。

语法

```
CamGetExposure '('
  [ Camera ':=' ] < variable (VAR) of cameradev >
  ['\'ExposureTime]
  | ['\'Brightness]
  | ['\'Contrast] ')'
```

返回值的数据类型是 num 的函数。

2.35 CamGetLoadedJob - 获取所加载摄像头任务的名称

手册用法

CamGetLoadedJob *(*摄像头获取加载作业*)* 是一个从摄像头读取当前所加载作业名称并以字符串形式返回的函数。

基本示例

以下示例介绍了 CamGetLoadedJob 函数。

例 1

```
VAR string currentjob;
...
currentjob:=CamGetLoadedJob(mycamera);
IF CurrentJob = "" THEN
  TPWrite "No job loaded in camera "+CamGetName(mycamera);
ELSE
  TPWrite "Job "+CurrentJob+" is loaded in camera "
       "+CamGetName(mycamera);
ENDIF
```

在 FlexPendant 上写入加载的作业名称。

返回值

数据类型：string

指定摄像头当前加载的作业名称。

变元

```
CamGetLoadedJob (Camera)
```

Camera

数据类型：cameradev

摄像头名称。

程序执行

CamGetLoadedJob 函数从摄像头获取当前加载的作业名称。如果没有作业加载摄像头，则返回空字符串。

错误处理

可能会产生下列可恢复错误。错误可以由错误处理程序处理。系统变量 ERRNO 将设置为：

名称	错误原因
ERR_CAM_BUSY	摄像头正忙于处理其他请求，无法执行当前命令。
ERR_CAM_COM_TIMEOUT	与摄像头通信错误。摄像头可能已断开。

语法

```
CamGetLoadedJob '('
  [ Camera ':=' ] < variable (VAR) of cameradev > ')'
```

返回值的数据类型是 string 的函数。

2.36 CamGetName - 获取所使用摄像头的名称

手册用法

CamGetName (Camera Get Name) 用于获取摄像头的配置名称。

基本示例

以下示例介绍了 CamGetName 函数。

例 1

```
  ...
  logcameraname camera1;
  CamReqImage camera1;
  ...
  logcameraname camera2;
  CamReqImage camera2;
  ...
PROC logcameraname(VAR cameradev camdev)
  TPWrite "Now using camera: "+CamGetName(camdev);
ENDPROC
```

此步骤记录当前使用的摄像头的名称到 FlexPendant。

返回值

数据类型：string

当前使用的摄像头的名称以字符串返回。

变元

```
CamGetName(Camera)
```

Camera

数据类型：cameradev

摄像头名称。

语法

```
CamGetName( '('
  [ Camera ':=' ] < variable (VAR) of cameradev > ')'
```

返回值的数据类型是 string 的函数。

2.37 CamNumberOfResults - 获取可用结果的数量

手册用法

CamNumberOfResults (Camera Number of Results) 是读取获取可用结果数量，并以数值形式返回的函数。

基本示例

以下示例介绍了 CamNumberOfResults 函数。

例 1

```
VAR num foundparts;
...
CamReqImage mycamera;
WaitTime 1;
FoundParts := CamNumberOfResults(mycamera);
TPWrite "Number of identified parts in the camera image:
     "\Num:=foundparts;
```

采集图像。等待图像处理完成，在本例中为 1 秒。读取识别的部件并将其写入 FlexPendant。

返回值

数据类型：num

返回指定摄像头集合中结果的数量。

变元

```
CamNumberOfResults (Camera [\SceneId])
```

Camera

数据类型：cameradev

摄像头名称。

[\SceneId]

场景识别

数据类型：num

SceneId 是一个标识符，指定从哪个图像读取识别的部件的编号。

程序执行

CamNumberOfResults 是读取可用图像结果数量，并将其以数值方式返回的函数。可以用于遍历所有可用结果。

此功能被执行时直接返回队列级别。如果在请求图片后直接执行功能，则结果为 0，因为摄像头尚未完成图片处理。

错误处理

可能会产生下列可恢复错误。错误可以由错误处理程序处理。系统变量 ERRNO 将设置为：

名称	错误原因
ERR_CAM_BUSY	摄像头正忙于处理其他请求，无法执行当前命令。

下一页继续

语法

```
CamNumberOfResults '('
  [ Camera ':=' ] < variable (VAR) of cameradev >
  [ '\'SceneId ':=' < expression (IN) of num > ] ')'
```

返回值的数据类型是 num **的函数。**

2.38 CapGetFailSigs - 获取失败的I/O信号

手册用法

CapGetFailSigs将用于返回监控CapL或CapC期间失败的信号或信号名称。

如果过程中一个或若干信号监控失败，那么将从CapL/CapC指令返回可恢复错误。为了确定哪些信号失败，可针对所有监控错误情况，在错误处理器中采用CapGetFailSigs。

基本示例

```
Stringcopied := CapGetFailSigs(Failstring);
```

如果拷贝成功，那么将Stringcopied赋值为真，否则赋值为假。

Failstring包含以文本形式失败的信号。如果不可拷贝任何字符串，那么将返回字符串EMPTY。

返回值

数据类型：bool

是真还是假取决于失败字符串是否被修改。

变元

```
CapGetFailSigs (ErrorNames)
```

ErrorNames

数据类型：string

CapGetFailSigs需要输入参数形式的字符串变量。

限制

如果监控列表中的多个信号同时失败，那么只以CapGetFailSigs报告其中三个信号。

语法

```
CapGetFailSigs '('
[ErrorNames ':='] < variable (INOUT) of string >')'
```

含数据类型bool的返回值的函数。

相关信息

信息，关于	请参阅
Continuous Application Platform	应用手册 - *Continuous Application Platform*
InitSuperv指令	第*258*页的*InitSuperv - 重置CAP的所有监控*
SetupSuperv指令	第*599*页的*SetupSuperv - 设置CAP信号监控条件*
RemoveSuperv指令	第*505*页的*RemoveSuperv - 撤除一个信号的条件*

2.39 CDate - 将当前日期作为字符串来读取

手册用法

CDate（*Current Date*）用于读取当前系统日期。

该函数可用于将当前日期呈现在FlexPendant示教器显示器上的运算符，或者将当前日期粘贴到程序写入的文本文件中。

基本示例

以下示例介绍了函数CDate。

另请参阅第*1025*页的更多示例

例 1

```
VAR string date;
date := CDate();
```

将当前日期储存在变量date中。

返回值

数据类型：string

位于一个字符串中的当前日期。

标准日期格式为“年-月-日”，例如，“1998-01-29”。

更多示例

有关于函数CDate的更多例子阐述如下。

例 1

```
VAR string date;
date := CDate();
TPWrite "The current date is: "+date;
Write logfile, date;
```

将当前日期写入到FlexPendant示教器显示器和文本文件中。

语法

```
CDate '(' ')'
```

含string型返回值的函数。

相关信息

信息，关于	请参阅
时间指令	技术参考手册 - *RAPID*语言概览
设置系统时钟	操作员手册 - 带 *FlexPendant* 的 *IRC5*

2.40 CJointT - 读取当前接头的角度

手册用法

CJointT（*Current Joint Target*）用于读取机械臂轴和外轴的当前角度。

基本示例

以下示例介绍了函数CJointT。

另请参阅第*1026*页的更多示例

例 1

```
VAR jointtarget joints;
joints := CJointT();
```

将机械臂轴和外轴的当前角度储存在joints中。

返回值

数据类型：jointtarget

有关臂侧上机械臂各轴的当前角度，以度计。

有关外轴的当前值，线性轴以mm计，旋转轴以度计。

返回值与校准位置相关。

变元

```
CJointT ([\TaskRef]|[\TaskName])
```

[\TaskRef]

Task Reference

数据类型：taskid

应当从程序任务识别号读取jointtarget。

对于系统中的所有程序任务，数据类型taskid的预定义变量将有效。可变识别号将为“任务名”+“Id”，例如，针对T_ROB1任务，可变识别号将为T_ROB1Id。

[\TaskName]

数据类型：string

应当从程序任务名称读取jointtarget。

如果未指定自变量\TaskRef或\TaskName，则使用当前任务。

更多示例

有关于函数CJointT的更多例子阐述如下。

例 1

```
! In task T_ROB1
VAR jointtarget joints;
joints := CJointT(\TaskRef:=T_ROB2Id);
```

将任务T_ROB2中的机械臂和外轴的当前位置储存在任务T_ROB1的joints中。

注意，读取位置时，任务T_ROB2 中的机械臂可能正在移动。为确保机械臂静止不动，可编程任务T_ROB2的先前移动指令中的停止点fine，并可使用指令WaitSyncTask，以同步任务T_ROB1中的指令。

下一页继续

例 2

```
! In task T_ROB1
VAR jointtarget joints;
joints := CJointT(\TaskName:="T_ROB2");
```

与上文例1的作用相同。

错误处理

如果参数\TaskRef或\TaskName指定一些非运动任务，则将系统ERRNO设置为ERR_NOT_MOVETASK。可用错误处理器来处理该错误。

但是，如果参数\TaskRef或\TaskName指定用以执行该函数CJointT的非运动任务（自身非运动任务的引用），则将不会产生错误。随后，将从相关运动任务获得该位置。

语法

```
CJointT '('
  ['\' TaskRef ':=' <variable (VAR) of taskid>]
  |['\' TaskName ':=' <expression (IN) of string>] ')'
```

含数据类型jointtarget的返回值的函数。

相关信息

信息，关于	请参阅
接头的定义	第*1418*页的*jointtarget - 接头位置数据*
读取当前电机的角度	第*1200*页的*ReadMotor - 读取当前电机的角度*

2.41 ClkRead - 读取用于定时的时钟

手册用法

ClkRead用于读取作为定时用秒表的时钟。

基本示例

以下示例介绍了函数ClkRead。

例 1

```
reg1:=ClkRead(clock1);
```

读取时钟clock1，并将时间（以秒计）储存在变量reg1中。

例 2

```
reg1:=ClkRead(clock1 \HighRes);
```

读取时钟clock1，并以高分辨率，将时间（以秒计）储存在变量reg1中。

返回值

数据类型：num

将时间（以秒计）储存在时钟中。分辨率通常为0.001秒。如果使用HighRes开关，则可能获得0.000001秒的分辨率。

变元

```
ClkRead (Clock \HighRes)
```

Clock

数据类型：clock

用于读取的时钟的名称。

[\HighRes]

High Resolution

数据类型：switch

规定应当以更高的分辨率来读取时间。如果使用该开关，则以分辨率0.000001来读取时间是可能的。

由于数据类型数字的精度，只要读取值小于1秒，则仅可获得微秒的分辨率。

程序执行

停止或运行时可以读取的时钟。

一旦读取一个时钟，则可以再次读取、再次启动、停止或重置。

错误处理

如果时钟运行4,294,967秒（49天17小时2分钟47秒），则其会溢出，且系统变量ERRNO得以设置为ERR_OVERFLOW。

可以用错误处理器来处理错误。

如果使用HighRes开关，则不会出现错误ERR_OVERFLOW，但是，在大约49700天后，时钟将回绕。

下一页继续

语法

```
ClkRead '('
  [ Clock ':=' ] < variable (VAR) of clock >
  [ '\' HighRes] ')'
```

含num型返回值的函数。

相关信息

信息，关于	请参阅
时钟指令	技术参考手册 - *RAPID*语言概览
更多示例	第*113*页的*ClkStart* - 启动用于定时的时钟

2.42 CorrRead - 读取当前的总偏移量

手册用法

CorrRead 读取所有连接的校正发生器传送的所有连接。

CorrRead可用于：

- 发现当前路径与原始路径的差异程度。
- 采取行动，以减小差异。

基本示例

以下示例介绍了函数CorrRead。

另请参阅第*1030*页的更多示例

例 1

```
VAR pos offset;
...
offset := CorrRead();
```

在变量偏移量中可利用所有连接的校正发生器传送的当前偏移量。

返回值

数据类型：pos

迄今从所有连接的校正发生器传送的总绝对偏移量。

更多示例

关于函数CorrRead的更多例子，参见指令CorrCon。

语法

```
CorrRead '(' ')'
```

含数据类型pos的返回值的函数。

相关信息

信息，关于	请参阅
与修正发电机相连	第*131*页的*CorrCon - 与修正发电机相连*
与修正发电机断开	第*136*页的*CorrDiscon - 与修正发电机断开*
写入修正发电机	第*137*页的*CorrWrite - 写入修正发电机*
移除所有修正发电机	第*130*页的*CorrClear - 移除所有修正发电机*
修正描述符	第*1381*页的*corrdescr - 修正发电机描述符*

2.43 Cos - 计算余弦值

手册用法

Cos（*Cosine*）用于根据有关数据类型num的角度值，计算余弦值。

基本示例

以下示例介绍了函数Cos。

例 1

```
VAR num angle;
VAR num value;
...
...
value := Cos(angle);
```

value将获得angle的余弦值。

返回值

数据类型：num

余弦值，范围 = 【-1，1】。

变元

```
Cos (Angle)
```

Angle

数据类型：num

角度值，以度表示。

语法

```
Cos '('
  [Angle ':='] <expression (IN) of num> ')'
```

返回值的数据类型是 num 的函数。

相关信息

信息，关于	请参阅
数学指令和函数	技术参考手册 - *RAPID*语言概览

2.44 CosDnum - 计算余弦值

手册用法

CosDnum（***Cosine dnum***）用于根据有关数据类型dnum的角度值，计算余弦值。

基本示例

以下示例介绍了函数CosDnum。

例 1

```
VAR dnum angle;
VAR dnum value;
...
...
value := CosDnum(angle);
```

value将获得angle的余弦值。

返回值

数据类型：dnum

余弦值，范围 = 【-1，1】。

变元

```
CosDnum (Angle)
```

Angle

数据类型：dnum

角度值，以度表示。

语法

```
CosDnum '('
  [Angle ':='] <expression (IN) of dnum> ')'
```

含数据类型dnum的返回值的函数。

相关信息

信息，关于	请参阅
数学指令和函数	技术参考手册 - *RAPID*语言概览

2.45 CPos - 读取当前位置（pos）数据

手册用法

CPos *(Current Position)* 用于读取机械臂的当前位置。

该函数返回机械臂TCP的x、y和z值，以作为位置数据。如果有待读取完整的机械臂位置（robtarget），则转而使用函数CRobT。

基本示例

以下示例介绍了函数CPos。

另请参阅第*1034*页的更多示例

例 1

```
VAR pos pos1;

MoveL *, v500, fine \Inpos := inpos50, tool1;
pos1 := CPos(\Tool:=tool1 \WObj:=wobj0);
```

将机械臂TCP的当前位置储存在变量pos1中。工具tool1和工件wobj0用于计算位置。

注意，在读取和计算位置前，机械臂静止不动。通过使用先前移动指令中位置精度inpos50内的停止点fine，实现上述操作。

返回值

数据类型：pos

机械臂以及最外面坐标系中x、y和z的当前位置（pos），其将指定工具、工件和有效ProgDisp坐标系纳入考虑。

变元

```
CPos([\Tool] [\WObj])
```

[\Tool]

数据类型： tooldata

用于计算当前机械臂位置的工具。

如果省略该参数，则使用当前的有效工具。

[\WObj]

Work Object

数据类型： wobjdata

同函数所返回的当前机械臂位置相关的工件（坐标系）。

如果省略该参数，则使用当前的有效工件。

> **警告**
>
> 编程期间，建议始终指定参数\Tool和\WObj。随后，函数将始终返回所需位置，即使已启用另一个工具或工件。

下一页继续

2.45 CPos - 读取当前位置（pos）数据

RobotWare - OS

续前页

程序执行

返回的坐标代表TCP在ProgDisp坐标系中的位置。

更多示例

有关于函数CPos的更多例子阐述如下。

```
VAR pos pos2;
VAR pos pos3;
VAR pos pos4;

pos2 := CPos(\Tool:=grip3 \WObj:=fixture);
...
pos3 := CPos(\Tool:=grip3 \WObj:=fixture);
pos4 := pos3-pos2;
```

在使用CPos函数的程序内的两处位置，捕获机械臂的x、y和z位置。工具grip3和工件fixture用于计算位置。随后，计算此类位置之间的x、y和z距离，并将其储存在变量pos4中。

语法

```
CPos '('
  ['\' Tool ':=' <persistent (PERS) of tooldata>]
  ['\' WObj ':=' <persistent (PERS) of wobjdata>] ')'
```

含数据类型pos的返回值的函数。

相关信息

信息，关于	请参阅
位置的定义	第*1450*页的*pos - 位置（仅X、Y和Z）*
工具的定义	第*1502*页的*tooldata - 工具数据*
工件的定义	第*1523*页的*wobjdata - 工件数据*
ProgDisp坐标系	第*448*页的*PDispOn - 启用程序位移*
坐标系	技术参考手册 - *RAPID*语言概览
读取当前的robtarget	第*1035*页的*CRobT - 读取当前位置（机器人位置）数据*

2.46 CRobT - 读取当前位置（机器人位置）数据

手册用法

CRobT*(Current Robot Target)* 用于读取机械臂和外轴的当前位置。

该函数返回robtarget值以及位置（x、y、z）、方位（q1 ... q4）、机械臂轴配置和外轴位置。如果仅读取机械臂TCP（pos）的x、y和z值，则转而使用函数CPos。

基本示例

以下示例介绍了函数CRobT。

另请参阅第*1036*页的更多示例

例 1

```
VAR robtarget p1;
MoveL *, v500, fine \Inpos := inpos50, tool1;
p1 := CRobT(\Tool:=tool1 \WObj:=wobj0);
```

将机械臂和外轴的当前位置储存在p1中。工具tool1和工件wobj0用于计算位置。

注意，在读取和计算位置前，机械臂静止不动。通过使用先前移动指令中位置精度inpos50内的停止点fine，实现上述操作。

返回值

数据类型： robtarget

最外面坐标系中的机械臂和外轴的当前位置，其将指定工具、工件和有效ProgDisp/ExtOffs坐标系纳入考虑。

变元

```
CRobT ([\TaskRef]|[\TaskName] [\Tool] [\WObj])
```

[\TaskRef]

Task Reference

数据类型：taskid

应当从程序任务识别号读取robtarget。

对于系统中的所有程序任务，数据类型taskid的预定义变量将有效。可变识别号将为“任务名”+“Id”，例如，针对T_ROB1任务，可变识别号将为T_ROB1Id。

[\TaskName]

数据类型：string

应当从程序任务名称读取robtarget。

如果未指定自变量\TaskRef或\TaskName，则使用当前任务。

[\Tool]

数据类型： tooldata

有关用于计算当前机械臂位置的工具的永久变量。

如果省略该参数，则使用当前的有效工具。

[\WObj]

Work Object

下一页继续

2.46 CRobT - 读取当前位置（机器人位置）数据

RobotWare - OS

续前页

数据类型： wobjdata

同函数所返回的当前机械臂位置相关的工件（坐标系）的永久变量。

如果省略该参数，则使用当前的有效工件。

> **警告**
>
> 编程期间，建议始终指定参数\Tool和\WObj。随后，函数将始终返回所需位置，即使已启用另一个工具或工件。

程序执行

返回的坐标代表TCP在 ProgDisp坐标系中的位置。用 ExtOffs坐标系来代表外轴。

如果使用参数\TaskRef或\TaskName之一，但是未使用参数Tool和WObj，则将使用指定任务中的当前工具和工件。

更多示例

有关于函数CRobT的更多例子阐述如下。

例 1

```
VAR robtarget p2;
p2 := ORobT( CRobT(\Tool:=grip3 \WObj:=fixture) );
```

将机械臂和外轴的工件坐标系（不含任何ProgDisp或ExtOffs）中的当前位置储存在p2中。工具grip3和工件fixture用于计算位置。

例 2

```
! In task T_ROB1
VAR robtarget p3;
p3 := CRobT(\TaskRef:=T_ROB2Id \Tool:=tool1 \WObj:=wobj0);
```

将任务T_ROB2中机械臂和外轴的当前位置储存在任务T_ROB1中的 p3。工具tool1和工件wobj0用于计算位置。

注意，读取和计算位置时，任务T_ROB2中的机械臂可能正在移动。为确保机械臂静止不动，可编程任务T_ROB2的先前移动指令中的停止点fine，并可使用指令WaitSyncTask，以同步任务T_ROB1中的指令。

例 3

```
! In task T_ROB1
VAR robtarget p4;
p4 := CRobT(\TaskName:="T_ROB2");
```

将任务T_ROB2中机械臂和外轴的当前位置储存在任务T_ROB1中的 p4。任务T_ROB2中的当前工具和工件用于计算位置。

错误处理

如果参数\TaskRef或\TaskName指定一些非运动任务，则将系统ERRNO设置为ERR_NOT_MOVETASK。可用错误处理器来处理该错误。

但是，如果参数\TaskRef或\TaskName指定用以执行该函数CRobT的非运动任务（自身非运动任务的引用），则将不会产生错误。随后，将从相关运动任务获得该位置。

下一页继续

语法

```
CRobT '('
  ['\' TaskRef ':=' <variable (VAR) of taskid>]
  |['\' TaskName ':=' <expression (IN) of string>]
  ['\'Tool ':=' <persistent (PERS) of tooldata>]
  ['\' WObj ':=' <persistent (PERS) of wobjdata>] ')'
```

含数据类型robtarget的返回值的函数。

相关信息

信息，关于	请参阅
位置的定义	第*1467*页的*robtarget - 位置数据*
工具的定义	第*1502*页的*tooldata - 工具数据*
工件的定义	第*1523*页的*wobjdata - 工件数据*
坐标系	技术参考手册 - *RAPID*语言概览
ProgDisp坐标系	第*448*页的*PDispOn - 启用程序位移*
ExtOffs坐标系	第*187*页的*EOffsOn - 启用附加轴的偏移量*
读取当前的pos（仅限x、y、z）	第*1033*页的*CPos - 读取当前位置（pos）数据*

2.47 CSpeedOverride - 读取当前的覆盖速度

手册用法

CSpeedOverride用于读取由来自FlexPendant示教器的运算符所设置的速度覆盖。返回值显示为一个百分比，其中，100%相当于编程速度。

在涉及指令SpeedRefresh的应用中，该函数亦可用于读取有关该运动程序任务或相关运动程序任务的当前速度覆盖值。

注意！不得与RAPID指令VelSet中的参数Override相混淆。

基本示例

以下示例介绍了函数CSpeedOverride。

例 1

```
VAR num myspeed;
myspeed := CSpeedOverride();
```

将当前覆盖速度储存在变量myspeed中。例如，如果该值为100，则其相当于100%。

返回值

数据类型：num

覆盖速度值占编程速度的百分比。其将为0 - 100范围中的一个数值。

变元

```
CSpeedOverride ( [\CTask] )
```

[\CTask]

数据类型：switch

获取有关该运动程序任务或相关运动程序任务的当前速度覆盖值。与指令SpeedRefresh一同使用。

如果未使用该参数，则函数返回有关整个系统（所有运动程序任务）的当前速度覆盖。这意味着根据FlexPendant示教器来设置手动速度覆盖。

语法

```
CSpeedOverride '('
  ['\' CTask ] ')'
```

返回值的数据类型是 num 的函数。

相关信息

信息，关于	请参阅
改变覆盖速度	操作员手册 - 带 *FlexPendant* 的 *IRC5*，编程与测试生产运行 - *QuickSet*菜单，速度一节
更新RAPID语言的速度覆盖	第655页的*SpeedRefresh* - 更新持续运动速度覆盖

2.48 CTime - 将当前时间作为字符串来读取

手册用法

CTime用于读取当前系统日期。

该函数可用于将当前时间呈现在FlexPendant示教器显示器上的运算符，或者将当前时间粘贴到程序写入的文本文件中。

基本示例

以下示例介绍了函数CTime。

另请参阅第*1039*页的更多示例

例 1

```
VAR string time;
time := CTime();
```

将当前时间储存在变量time中。

返回值

数据类型：string

位于一个字符串中的当前时间。

标准时间格式为“小时：分钟：秒”，例如，“18:20:46”。

更多示例

有关于函数CTime的更多例子阐述如下。

例 1

```
VAR string time;
time := CTime();
TPWrite "The current time is: "+time;
Write logfile, time;
```

将当前时间写入到FlexPendant示教器显示器和文本文件中。

语法

```
CTime '(' ')'
```

含string型返回值的函数。

相关信息

信息，关于	请参阅
时间和日期指令	技术参考手册 - *RAPID*语言概览，*RAPID*概要 - 系统&时间一节
设置系统时钟	操作员手册 - 带 *FlexPendant* 的 *IRC5*，改变*FlexPendant*示教器设置一节

2.49 CTool - 读取当前的工具数据

手册用法

CTool（*Current Tool*）用于读取当前工具的数据。

基本示例

以下示例介绍了函数CTool。

例 1

```
PERS tooldata temp_tool:= [ TRUE, [ [0, 0, 0], [1, 0, 0 ,0] ],
     [0.001, [0, 0, 0.001], [1, 0, 0, 0], 0, 0, 0] ];
temp_tool := CTool();
```

将当前工具的值储存在变量temp_tool中。

返回值

数据类型： tooldata

该函数返回用于保存当前工具（即最后用于移动指令中的工具）值的tooldata值。

返回的值代表TCP在腕中心坐标系中的位置和方位。参见tooldata。

变元

```
CTool ([\TaskRef]|[\TaskName])
```

[\TaskRef]

Task Reference

数据类型：taskid

用以读取当前工具数据的程序任务识别号。

对于系统中的所有程序任务，数据类型taskid的预定义变量将有效。可变识别号将为“任务名”+“Id”，例如，针对T_ROB1任务，可变识别号将为T_ROB1Id。

[\TaskName]

数据类型：string

用以读取当前工具数据的程序任务名称。

如果未指定自变量\TaskRef或\TaskName，则使用当前任务。

错误处理

如果参数\TaskRef或\TaskName指定一些非运动任务，则将系统ERRNO设置为ERR_NOT_MOVETASK。可用错误处理器来处理该错误。

但是，如果参数\TaskRef或\TaskName指定用以执行该函数CTool的非运动任务（自身非运动任务的引用），则将不会产生错误。随后，将从相关运动任务获得该工具数据。

语法

```
CTool '('
  ['\' TaskRef ':=' <variable (VAR) of taskid>]
  |['\' TaskName ':=' <expression (IN) of string>] ')'
```

含数据类型tooldata的返回值的函数。

下一页继续

相关信息

信息，关于	请参阅
工具的定义	第*1502*页的*tooldata - 工具数据*
坐标系	技术参考手册 - *RAPID*语言概览，运动和*I/O*原则 - 坐标系一节

2.50 CWObj - 读取当前工件数据

手册用法

CWObj（*Current Work Object*）用于读取当前工件的数据。

基本示例

以下示例介绍了函数CWObj。

例 1

```
PERS wobjdata temp_wobj:= [FALSE, TRUE, "", [[0,0,0], [1,0,0,0]],
     [[0,0,0], [1,0,0,0]]];
temp_wobj := CWObj();
```

将当前工件的值储存在变量temp_wobj中。

返回值

数据类型： wobjdata

该函数返回用于保存当前工件（即最后用于移动指令中的工件）值的wobjdata值。

返回的值代表工件在世界坐标系中的位置和方位。参见wobjdata。

变元

```
CWobj ([\TaskRef]|[\TaskName])
```

[\TaskRef]

Task Reference

数据类型：taskid

用于读取当前工件数据的程序任务识别号。

对于系统中的所有程序任务，数据类型taskid的预定义变量将有效。可变识别号将为“任务名”+“Id”，例如，针对T_ROB1任务，可变识别号将为T_ROB1Id。

[\TaskName]

数据类型：string

用于读取当前工件数据的程序任务名称。

如果未指定自变量\TaskRef或\TaskName，则使用当前任务。

错误处理

如果参数\TaskRef或\TaskName指定一些非运动任务，则将系统ERRNO设置为ERR_NOT_MOVETASK。可用错误处理器来处理该错误。

但是，如果参数\TaskRef或\TaskName指定用以执行该函数CWobj的非运动任务（自身非运动任务的引用），则将不会产生错误。随后，将从相关运动任务获得该工件数据。

语法

```
CWobj '('
  ['\' TaskRef ':=' <variable (VAR) of taskid>]
  |['\' TaskName ':=' <expression (IN) of string>] ')'
```

含数据类型wobjdata的返回值的函数。

下一页继续

相关信息

信息，关于	请参阅
工件的定义	第*1523*页的*wobjdata - 工件数据*
坐标系	技术参考手册 - *RAPID*语言概览，运动和*I/O*原则 - 坐标系一节

2.51 DecToHex - 从十进制转换为十六进制

手册用法

DecToHex用于将基座10中可读字符串中指定的数字转换为基座16中可读字符串中指定的数字。

根据字符集【0-9，A-F，a-f】，构建所得字符串。

程序处理数字从0多达9223372036854775807十进制或7FFFFFFFFFFFFFFF十六进制。

基本示例

以下示例介绍了函数DecToHex。

例 1

```
VAR string str;

str := DecToHex("99999999");
```

变量str被赋予值"5F5E0FF"。

返回值

数据类型：string

将字符串转换成代表非参数字符串中给定数字的十六进制。

变元

```
DecToHex ( Str )
```

Str

String

数据类型：string

用于转换的字符串。

语法

```
DecToHex '('
  [ Str ':=' ] <expression (IN) of string> ')'
```

返回值的数据类型是 string 的函数。

相关信息

信息，关于	请参阅
字符串功能	技术参考手册 - *RAPID*语言概览，*RAPID*概要 - 字符串函数一节
字符串的定义	第*1489*页的*string* - 字符串
字符串值	技术参考手册 - *RAPID*语言概览，基本特征 - 基本元素一节

2.52 DefAccFrame - 定义一个准确的坐标系

手册用法

`DefAccFrame`（***Define Accurate Frame***）用于定义一个从原始位置三到十的坐标系以及相同数量的位移位置。

描述

当已知一组目标点位于两个不同的位置时，可定义一个坐标系。因此，使用相同的物理位置，但是以不同的方式表示。

将其视为采用两种不同的方法：

1 通过不同的坐标系，表达相同的物理位置。例如，从CAD图纸检索了许多位置，因此，用CAD局部坐标系来表示此类位置。随后，用机械臂世界坐标系来表示相同的位置。根据这两组位置，计算CAD坐标系与机械臂世界坐标系之间的坐标系。

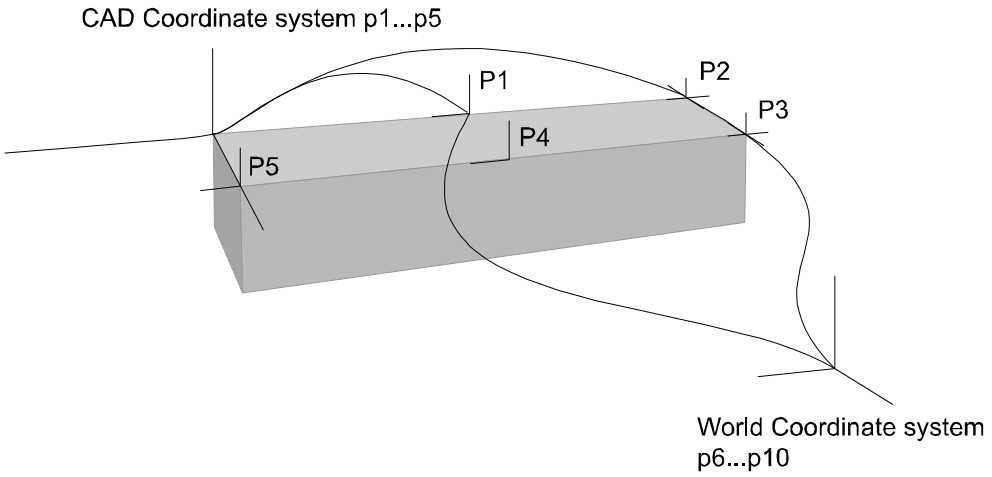
2 许多位置与原始位置中的对象相关。在对象位移后，再次确定各位置（通常进行搜索）。根据这两组位置（旧位置、新位置），计算位移坐标系。

三个目标点足以定义一个坐标系，但是，为提高准确性，应当使用多个点。

基本示例

以下示例介绍了函数`DefAccFrame`。

例 1



xx0500002179

```
CONST robtarget p1 := [...];
CONST robtarget p2 := [...];
CONST robtarget p3 := [...];
CONST robtarget p4 := [...];
CONST robtarget p5 := [...];

VAR robtarget p6 := [...];
VAR robtarget p7 := [...];
VAR robtarget p8 := [...];
VAR robtarget p9 := [...];
VAR robtarget p10 := [...];
```

下一页继续

2.52 DefAccFrame - 定义一个准确的坐标系

RobotWare - OS

续前页

```
VAR robtarget pWCS{5};
VAR robtarget pCAD{5};

VAR pose frame1;
VAR num max_err;
VAR num mean_err;

! Add positions to robtarget arrays
pCAD{1}:=p1;
...
pCAD{5}:=p5;

pWCS{1}:=p6;
...
pWCS{5}:=p10;
frame1 := DefAccFrame (pCAD, pWCS, 5, max_err, mean_err);
```

已经储存了同一个对象相关的五个位置p1- p5。同时将这五个位置储存在世界坐标系中，作为p6-p10。根据这10个位置，计算对象与世界坐标系之间的坐标系frame1。该坐标系将成为用世界坐标系表示的CAD坐标系。如果互换目标点列表的输入顺序，即DefAccFrame（pWCS, pCAD....），则将用CAD坐标系来表示世界坐标系。

返回值

数据类型：pose

用TargetListTwo坐标系来表示计算出的TargetListOne坐标系。

变元

```
DefAccFrame (TargetListOne TargetListTwo TargetsInList
     MaxErrMeanErr)
```

TargetListOne

数据类型： robtarget

用于保存坐标系一中确定位置的机器人位置数组。机器人位置的最小数量为3，最大数量为10。

TargetListTwo

数据类型： robtarget

用于保存坐标系二中确定位置的机器人位置数组。机器人位置的最小数量为3，最大数量为10。

TargetsInList

数据类型：num

一个数组中的机器人位置的数量。

MaxErr

数据类型：num

预估最大误差，以mm计。

MeanErr

数据类型：num

下一页继续

预估平均误差，以mm计。

错误处理

如果位置不具有所需关系，或者未通过足够精度来指定，则将系统变量ERRNO设置为ERR_FRAME。随后，可通过错误处理器来处理该错误。

语法

```
DefAccFrame '('
  [TargetListOne ':='] <array {*} (IN) of robtarget> ','
  [TargetListTwo ':='] <array {*} (IN) of robtarget> ','
  [TargetsInList ':='] <expression (IN) of num> ','
  [MaxErr ':='] <variable (VAR) of num> ','
  [MeanErr ':='] <variable (VAR) of num> ')'
```

含数据类型pose的返回值的函数。

相关信息

信息，关于	请参阅
根据三个位置，计算一个坐标系	第*1051*页的*DefFrame - 定义一个坐标系*
根据6个位置，计算一个坐标系	第*1048*页的*DefDFrame - 定义一个位移坐标系*

2.53 DefDFrame - 定义一个位移坐标系

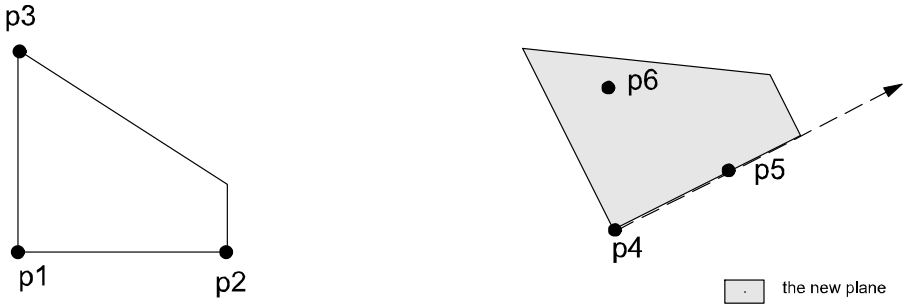
手册用法

DefDFrame（*Define Displacement Frame*）为，用于根据三个原始位置和三个位移位置，计算位移坐标系。

基本示例

以下示例介绍了函数DefDFrame。

例 1



xx0500002177

```
CONST robtarget p1 := [...];
CONST robtarget p2 := [...];
CONST robtarget p3 := [...];
VAR robtarget p4;
VAR robtarget p5;
VAR robtarget p6;
VAR pose frame1;
...
!Search for the new positions
SearchL sen1, p4, *, v50, tool1;
...
SearchL sen1, p5, *, v50, tool1;
...
SearchL sen1, p6, *, v50, tool1;
frame1 := DefDframe (p1, p2, p3, p4, p5, p6);
...
!Activation of the displacement defined by frame1
PDispSet frame1;
```

已经储存了同原始位置中的一个对象相关的三个位置p1-p3。在对象位移后，搜索三个新位置，并储存为 p4-p6。根据这六个位置，计算位移坐标系。随后，计算出的坐标系用于取代储存在程序中的所有位置。

返回值

数据类型：pose

位移坐标系。

下一页继续

变元

```
DefDFrame (OldP1 OldP2 OldP3 NewP1 NewP2 NewP3)
```

`OldP1`

数据类型：`robtarget`

第一个原始位置。

`OldP2`

数据类型：`robtarget`

第二个原始位置。

`OldP3`

数据类型：`robtarget`

第三个原始位置。

`NewP1`

数据类型：`robtarget`

第一个位移位置。`OldP1`与`NewP1`之间的差异将确定坐标系的平移部分，且必须以极大的准确性来测量和确定。

`NewP2`

数据类型：`robtarget`

第二个位移位置。生产线`NewP1 ... NewP2`将确定旧生产线`OldP1 ... OldP2`的旋转。

`NewP3`

数据类型：`robtarget`

第三个位移位置。该位置将确定平面的旋转，例如，应当将其置于`NewP1`、`NewP2`和`NewP3`的新平面上。

错误处理

如果由于各位置中的精确度过差而无法计算坐标系，则将系统变量`ERRNO`设置为`ERR_FRAME`。随后，可用错误处理器来处理该错误。

语法

```
DefDFrame '('
  [OldP1 ':='] <expression (IN) of robtarget> ','
  [OldP2 ':='] <expression (IN) of robtarget> ','
  [OldP3 ':='] <expression (IN) of robtarget> ','
  [NewP1 ':='] <expression (IN) of robtarget> ','
  [NewP2 ':='] <expression (IN) of robtarget> ','
  [NewP3 ':='] <expression (IN) of robtarget> ')'
```

含数据类型`pose`的返回值的函数。

相关信息

信息，关于	请参阅
位移坐标系的启用	第*452*页的*PDispSet* - 启用使用已知坐标系的程序位移

下一页继续

2.53 DefDFrame - 定义一个位移坐标系

RobotWare - OS

续前页

信息，关于	请参阅
位移坐标系的手动定义	操作员手册 - 带 *FlexPendant* 的 *IRC5*，计算一节

2.54 DefFrame - 定义一个坐标系

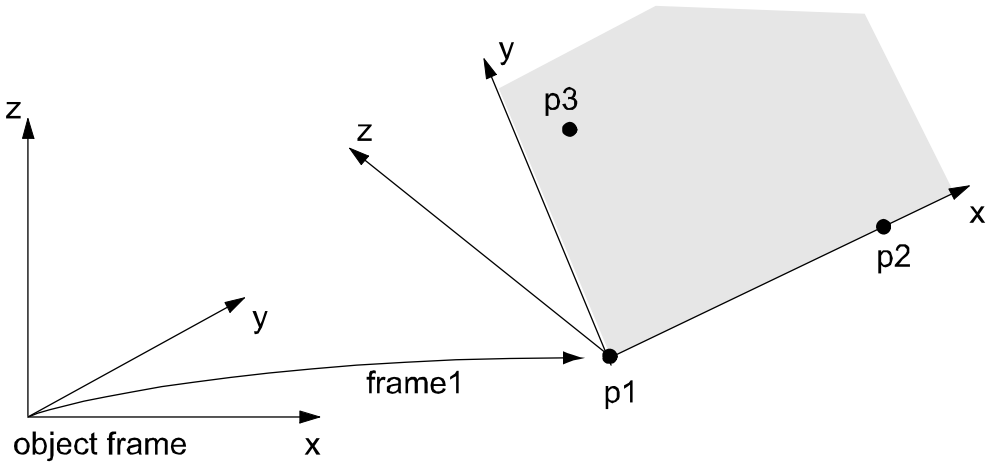
手册用法

DefFrame（*Define Frame*）为，用于计算一个坐标系，并从三个位置来定义该坐标系。

基本示例

以下示例介绍了函数DefFrame。

例 1



xx0500002181

与工件坐标系相关的三个位置p1- p3，用于定义新的坐标系frame1。第一个位置p1，定义新坐标系的原点。第二个位置p2，定义x轴的方向。第三个位置p3，定义xy平面的位置。可将已确定的frame1用作位移坐标系，如以下例子所示：

```
CONST robtarget p1 := [...];
CONST robtarget p2 := [...];
CONST robtarget p3 := [...];
VAR pose frame1;
...
...
frame1 := DefFrame (p1, p2, p3);
...
...
!Activation of the displacement defined by frame1
PDispSet frame1;
```

返回值

数据类型：pose

计算出的坐标系。

计算与有效工件坐标系相关。

变元

```
DefFrame (NewP1 NewP2 NewP3 [\Origin])
```

下一页继续

2.54 DefFrame - 定义一个坐标系

RobotWare - OS

续前页

NewP1

数据类型： robtarget

第一个位置将定义新坐标系的原点。

NewP2

数据类型： robtarget

第二个位置将定义新坐标系x轴的方向。

NewP3

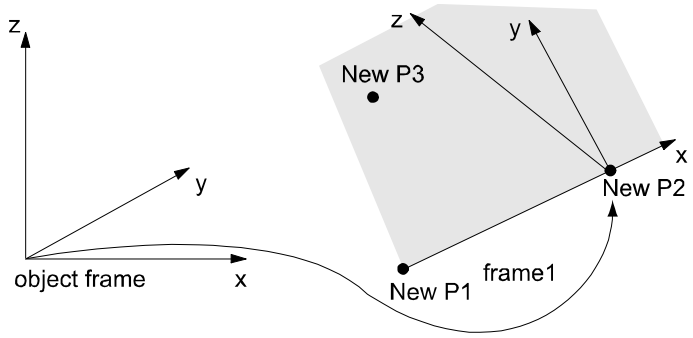
数据类型： robtarget

第三个位置将定义新坐标系的xy平面。点3将位于正y侧，参见上图。

[\Origin]

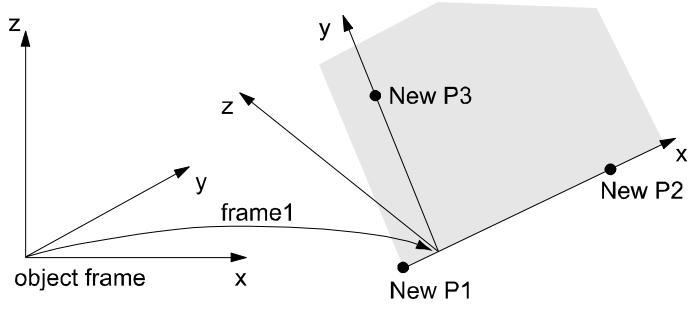
数据类型：num

可选参数将规定如何确定新坐标系的原点。Origin = 1，意味着原点位于NewP1，即如同省略该参数。Origin = 2，意味着原点位于NewP2。参见下图。



xx0500002178

Origin = 3，意味着原点位于通过NewP1和NewP2的直线上，因此，NewP3将位于y轴。参见下图。



xx0500002180

其他值，或者如果省略Origin，将位于NewP1原点。

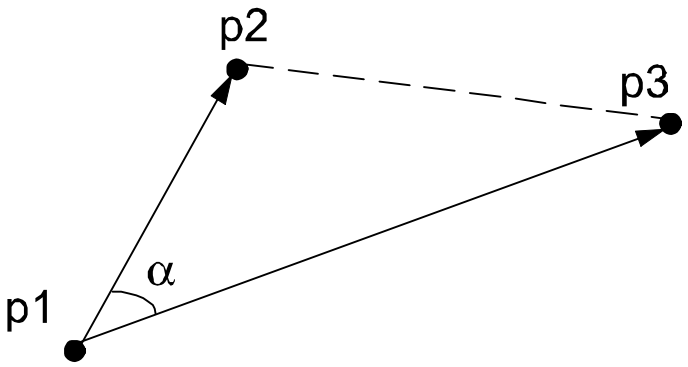
错误处理

如果由于低于限制而无法计算坐标系，则将系统变量ERRNO设置为ERR_FRAME。随后，可用错误处理器对该错误进行处理。

下一页继续

限制

用于定义坐标系的三个位置p1 - p3，必须定义一个匀称的三角形。最匀称的三角形各边等长。



xx0500002182

如果角α过小，则不认为三角形匀称。角α在以下情况下过小：

$|\cos \alpha| < 1 - 10^{-4}$

三角形p1、p2、p3不得过小，即各位置不得过于接近。位置 p1 - p2与 p1 - p3之间的距离不得小于0.1 mm。

语法

```
DefFrame '('
  [NewP1 ':='] <expression (IN) of robtarget> ','
  [NewP2 ':='] <expression (IN) of robtarget> ','
  [NewP3 ':='] <expression (IN) of robtarget>
  ['\' Origin ':=' <expression (IN) of num >] ')'
```

含数据类型pose的返回值的函数。

相关信息

信息，关于	请参阅
数学指令和函数	技术参考手册 - *RAPID*语言概览，*RAPID*概要 - 数学一节
位移坐标系的启用	第*452*页的*PDispSet* - 启用使用已知坐标系的程序位移

2.55 Dim - 获取数组的规模

手册用法

Dim （*Dimension*）用于获取一个数组中的元素数量。

基本示例

以下示例介绍了函数Dim。

另请参阅第*1054*页的更多示例

例 1

```
PROC arrmul(VAR num array{*}, num factor)
  FOR index FROM 1 TO Dim(array, 1) DO
    array{index} := array{index} * factor;
  ENDFOR
ENDPROC
```

使一个数字数组的所有元素乘以一个系数。该无返回值程序可将数据类型为num的任意一维数组作为输入。

返回值

数据类型：num

指定维度数组元素的数量。

变元

```
Dim (ArrPar DimNo)
```

ArrPar

Array Parameter

数据类型：任意类型

数组名称。

DimNo

Dimension Number

数据类型：num

所需数组维度：

1 = 第一个维度

2 = 第二个维度

3 = 第三个维度

更多示例

有关于如何使用函数Dim的更多例子阐述如下。

例 1

```
PROC add_matrix(VAR num array1{*,*,*}, num array2{*,*,*})

  IF Dim(array1,1) <> Dim(array2,1) OR Dim(array1,2) <>
       Dim(array2,2) OR Dim(array1,3) <> Dim(array2,3) THEN
    TPWrite "The size of the matrices are not the same";
    Stop;
```

下一页继续

```
  ELSE
    FOR i1 FROM 1 TO Dim(array1, 1) DO
      FOR i2 FROM 1 TO Dim(array1, 2) DO
        FOR i3 FROM 1 TO Dim(array1, 3) DO
          array1{i1,i2,i3} := array1{i1,i2,i3} + array2{i1,i2,i3};
        ENDFOR
      ENDFOR
    ENDFOR
  ENDIF
  RETURN;

ENDPROC
```

增加两个矩阵。如果矩阵的规模有所不同，则程序停止，并会出现错误消息。

该无返回值程序可将数据类型为num的任意三维数组作为输入。

语法

```
Dim '('
  [ArrPar ':='] <reference (REF) of any type> ','
  [DimNo ':='] <expression (IN) of num> ')'
```

REF参数需要相关参数为常量、变量或整个永久数据对象。该参数亦可为IN参数、VAR参数或整个PERS参数。

返回值的数据类型是 num 的函数。

相关信息

信息，关于	请参阅
数组参数	技术参考手册 - *RAPID*语言概览，基本特征 - 程序一节
数组声明	技术参考手册 - *RAPID*语言概览，基本特征 - 数据一节

2.56 DInput - 读取数字信号输入信号值

手册用法

DInput用于读取数字信号输入信号的当前值。

> **注意**
>
> 注意，函数DInput为遗留函数，其无须再作使用。有关编程的替代方案和建议方法，请参见例子。

基本示例

以下示例介绍了函数DInput。

另请参阅第*1056*页的更多示例

例 1

```
IF DInput(di2) = 1 THEN ...
...
IF di2 = 1 THEN ...
```

如果信号di2的当前值等于1，则...

返回值

数据类型：num

信号的当前值（0或1）。

变元

```
DInput (Signal)
```

Signal

数据类型：signaldi

待读取数字信号输入的名称。

程序执行

读取的值取决于信号配置。如果信号在系统参数中反转，该函数返回的值与物理通道的值相反。

更多示例

有关于如何使用函数DInput的更多例子阐述如下。

例 1

```
weld_flag := DInput(weld);
...
weld_flag := weld;
```

将变量weld_flag设置为与信号weld当前值相同的值。

> **注意**
>
> 注意，在这种情况下，weld_flag会反映信号的当前值。因此，如果随后在程序中使用weld_flag，则无法确定其是否将反映信号的当前值。

下一页继续

语法

```
DInput '('
  [ Signal ':=' ] < variable (VAR) of signaladi > ')'
```

含数据类型dionum的返回值的函数。

相关信息

信息，关于	请参阅
输入/输出指令	技术参考手册 - *RAPID*语言概览，*RAPID*概要 - 输入和输出信号一节
输入/输出功能性概述	技术参考手册 - *RAPID*语言概览，运动和*I/O*原则 - *I/O*原则一节
I/O配置	技术参考手册 - 系统参数

2.57 Distance - 两点之间的距离

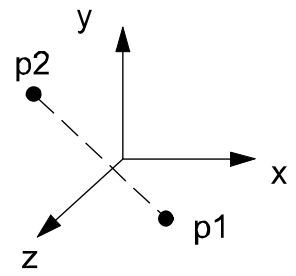
手册用法

Distance用于计算空间中两点之间的距离。

基本示例

以下示例介绍了函数Distance。

例 1



xx0500002321

```
VAR num dist;
CONST pos p1 := [4,0,4];
CONST pos p2 := [-4,4,4];
...
dist := Distance(p1, p2);
```

计算点p1 与p2 之间的空间距离，并将其储存在变量dist中。

返回值

数据类型：num

各点之间的距离（始终为正），以mm计。

变元

```
Distance (Point1 Point2)
```

Point1

数据类型：pos

由pos数据类型描述的第一个点。

Point2

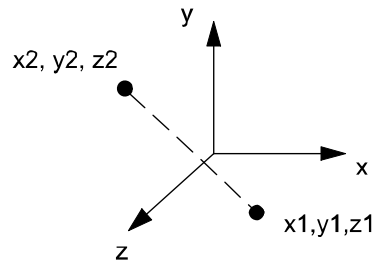
数据类型：pos

由pos数据类型描述的第二个点。

下一页继续

程序执行

计算两个点之间的距离：



xx0500002322

$$\text{distance} = \sqrt{\left((x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2\right)}$$

xx0500002323

语法

```
Distance '('
[Point1 ':='] <expression (IN) of pos> ','
[Point2 ':='] <expression (IN) of pos> ')'
```

返回值的数据类型是 num 的函数。

相关信息

信息，关于	请参阅
数学指令和函数	技术参考手册 - *RAPID*语言概览，*RAPID*概要 - 数学一节
位置的定义	第*1450*页的*pos* - 位置（仅*X*、*Y*和*Z*）

2.58 DIV - 评估一个整数除法

手册用法

DIV为用于评估整数除法的条件表达式。

基本示例

以下示例介绍了函数DIV。

例 1

```
reg1 := 14 DIV 4;
```

因为14可以除以4达3次，因此，返回值为3。

例 2

```
VAR dnum mydnum1 := 10;
VAR dnum mydnum2 := 5;
VAR dnum mydnum3;
...
mydnum3 := mydnum1 DIV mydnum2;
```

因为10可以除以5达2次，因此，返回值为2。

返回值

数据类型：num、dnum

从整数的除法得到整数。

语法

```
<expression of num> DIV <expression of num>
```

含数据类型num的返回值的函数。

```
<expression of dnum> DIV <expression of dnum>
```

含数据类型dnum的返回值的函数。

相关信息

信息，关于	请参阅
num - 数值	第*1435*页的*num - 数值*
dnum - 双数值	第*1385*页的*dnum - 双数值*
MOD	第*1143*页的*MOD - 评估一个整数模数*
表达式	技术参考手册 - *RAPID*语言概览

2.59 DnumToNum - 将双数值转换为数值

手册用法

如有可能，DnumToNum将dnum转换为num，否则，其会产生可恢复的错误。

基本示例

以下示例介绍了函数DnumToNum。

例 1

```
VAR num mynum:=0;
VAR dnum mydnum:=8388607;
VAR dnum testFloat:=8388609;
VAR dnum anotherdnum:=4294967295;
! Works OK
mynum:=DnumToNum(mydnum);
! Accept floating point value
mynum:=DnumToNum(testFloat);
! Cause error recovery error
mynum:=DnumToNum(anotherdnum \Integer);
```

由函数返回如数值8388607的双数值8388607。

由函数返回如数值8.38861E+06的双数值8388609。

双数值4294967295会产生可恢复的错误ERR_ARGVALERR。

返回值

数据类型：num

输入双数值可处于范围-8388607到8388608中，并返回如num的相同值。如果未使用\Integer开关，则输入双数值可处于范围-3.40282347E+38到3.40282347E+38中，且返回值可能成为一个浮动点值。

变元

```
DnumToNum (Value [\Integer])
```

Value

数据类型：dnum

有待转换的数值。

[\Integer]

数据类型：switch

仅整数值

如果未使用开关\Integer，则向下计算，即使该值成为一个浮动点值。如果未使用，则检查该值是否为一个介于-8388607到8388608之间的值。如果不是，则会产生可恢复错误。

下一页继续

2.59 DnumToNum - 将双数值转换为数值

RobotWare - OS

续前页

错误处理

系统会生成下列可恢复错误，并在错误处理器中处理这些错误。系统变量ERRNO将被设置成：

错误代码	描述
ERR_ARGVALERR	该值高于8388608，或低于-8388607，或并非为整数（如果使用可选参数Integer）
ERR_NUM_LIMIT	该值高于3.40282347E+38，或低于-3.40282347E+38
ERR_INT_NOTVAL	该值并非一个整数

语法

```
DnumToNum '('
  [ Value ':=' ] < expression (IN) of dnum >
  ['\' Integer ')'
```

返回值的数据类型是 num 的函数。

相关信息

信息，关于	请参阅
Dnum数据类型	第*1385*页的*dnum - 双数值*。
Num数据类型	第*1435*页的*num - 数值*。

2.60 DnumToStr - 将数值转换为字符串

手册用法

DnumToStr（*Numeric To String*）用于转换数值为字符串。

基本示例

以下示例介绍了函数DnumToStr。

例 1

```
VAR string str;
str := DnumToStr(0.3852138754655357,3);
```

变量str被赋予值"0.385"。

例 2

```
VAR dnum val;
val:= 0.3852138754655357;
str := DnumToStr(val, 2\Exp);
```

变量str被赋予值"3.85E-01"。

例 3

```
VAR dnum val;
val := 0.3852138754655357;
str := DnumToStr(val, 15);
```

变量str被赋予值"0.385213875465536"。

例 4

```
VAR dnum val;
val:=4294967295.385215;
str := DnumToStr(val, 4);
```

变量str被赋予值"4294967295.3852"。

返回值

数据类型：str

通过指定的小数位数，如有要求，可通过指数，将数值转换为字符串。如有必要，将数值四舍五入。如果不包括任何小数，则抑制小数点。

变元

```
DnumToStr (Val Dec [\Exp])
```

Val

Value

数据类型：dnum

有待转换的数值。

Dec

Decimals

数据类型：num

小数位数。小数位数不得为负或大于数值的可用精度。

可以使用的最多小数位数为15。

下一页继续

2.60 DnumToStr - 将数值转换为字符串

RobotWare - OS

续前页

`[\Exp]`

Exponent

数据类型：`switch`

为了在返回值中使用指数。

语法

```
DnumToStr '('
  [ Val ':=' ] <expression (IN) of dnum>
  [ Dec ':=' ] <expression (IN) of num>
  ['\' Exp ')'
```

返回值的数据类型是 `string` 的函数。

相关信息

信息，关于	请参阅
字符串功能	技术参考手册 - *RAPID*语言概览，*RAPID*概要 - 字符串函数一节
字符串的定义	第*1489*页的*string - 字符串*
字符串值	技术参考手册 - *RAPID*语言概览，基本特征 - 基本元素一节
将一个数值转换为一段字符串	第*1154*页的*NumToStr - 将数值转换为字符串*

2.61 DotProd - 两个位置矢量的点积

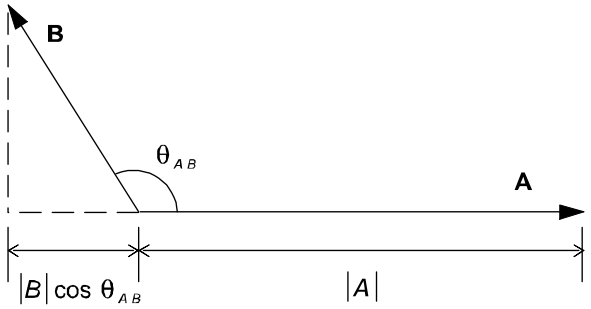
手册用法

DotProd（*Dot Product*）用于计算两个位置矢量的点（或标量）积。典型用途是计算矢量之间的相互投射，或计算两个矢量之间的角度。

基本示例

以下示例介绍了函数DotProd。

例 1



xx0500002449

两个矢量A和B的点或标量积为一个标量，其等于A和B量级的结果及其夹角的余弦。

$$A \cdot B = |A||B| \cos\theta_{AB}$$

点积：

- 小于或等于其量级结果。
- 根据其夹角是否小于或大于90度，可以为正值或负值。
- 等于一个矢量量级的结果以及其他矢量在第一个矢量上的投射。
- 当矢量相互垂直时，为零。

通过数据类型pos来描述矢量，并通过数据类型num来描述点积。

```
VAR num dotprod;
VAR pos vector1;
VAR pos vector2;
...
...
vector1 := [1,1,1];
vector2 := [1,2,3];
dotprod := DotProd(vector1, vector2);
```

返回值

数据类型：num

两个矢量的点积的值。

变元

```
DotProd (Vector1 Vector2)
```

下一页继续

2.61 DotProd - 两个位置矢量的点积

RobotWare - OS

续前页

Vector1

数据类型：pos

由pos数据类型描述的第一个矢量。

Vector2

数据类型：pos

由pos数据类型描述的第二个矢量。

语法

```
DotProd '('
  [Vector1 ':='] <expression (IN) of pos>','
  [Vector2 ':='] <expression (IN) of pos>
  ')'
```

返回值的数据类型是 num 的函数。

相关信息

信息，关于	请参阅
数学指令和函数	技术参考手册 - *RAPID*语言概览，*RAPID*概要 - 数学一节

2.62 DOutput - 读取数字信号输出信号值

手册用法

DOutput用于读取数字信号输出信号的当前值。

> **注意**
>
> 需注意函数DOutput是一个不再必须使用的遗留函数。有关备用和推荐的编程方式，请参见示例。

基本示例

以下示例介绍了函数DOutput。

另请参阅第*1068*页的更多示例

例 1

```
IF DOutput(do2) = 1 THEN ...
...
IF do2 = 1 THEN ...
```

如果信号do2的当前值等于1，则...

返回值

数据类型：dionum

信号的当前值（0或1）。

变元

```
DOutput (Signal)
```

Signal

数据类型：signaldo

待读取信号的名称。

程序执行

读取的值取决于信号配置。如果信号在系统参数中反转，该函数返回的值与物理通道的真值相反。

错误处理

可能会产生下列可恢复错误。错误可以由错误处理程序处理。系统变量ERRNO将设置为：

名称	错误原因
ERR_NO_ALIASIO_DEF	该信号变量是在RAPID中声明的一个变量，与用指令AliasIO在I / O配置中定义的I / O信号无关。
ERR_NORUNUNIT	如果与I/O单元无接触
ERR_SIG_NOT_VALID	无法访问I/O信号。原因可能在于I/O单元未运行，或者配置中出现错误（仅对ICI现场总线有效）。

下一页继续

2.62 DOutput - 读取数字信号输出信号值

RobotWare - OS

续前页

更多示例

有关于函数DOutput的更多例子阐述如下。

例 1

```
IF DOutput(auto_on) <> active THEN ...
...
IF auto_on <> active THEN ...
```

如果系统信号auto_on 的当前值为not active，则...，即如果机械臂采用手动运行模式，则...

> **注意**
>
> 必须首先将信号定义为系统参数中的系统输出。

语法

```
DOutput '('
  [ Signal ':=' ] < variable (VAR) of signaldo > ')'
```

含数据类型dionum的返回值的函数。

相关信息

信息，关于	请参阅
设置数字信号输出信号	第589页的*SetDO - 改变数字信号输出信号值*
输入/输出指令	技术参考手册 - *RAPID*语言概览，*RAPID*概要 - 输入和输出信号一节
输入/输出功能性概述	技术参考手册 - *RAPID*语言概览，运动和*I/O*原则 - *I/O*原则一节
I/O配置	技术参考手册 - 系统参数

2.63 EGMGetState – 获取当前的EGM状态

手册用法

EGMGetState会检索一项EGM进程（EGMid）的状态。

基本示例

```
VAR egmident egmID1;
VAR egmstate egmState1:= EGM_STATE_DISCONNECTED;

EGMGetId egmID1;
egmState1 := EGMGetState(egmID1);
```

返回值

数据类型：egmstate

该自变数中指定的EGM标识所识别出的EGM进程的当前状态。

变元

```
EGMGetState (EGMid)
```

EGMid

数据类型：egmident

EGM标识。

限制

- EGMGetState只能用在RAPID运动任务中。
- 该机械单元必须是一台机器人。

语法

```
EGMGetState '('
  [EGMid ':='] < variable (VAR) of egmident >')'
```

相关信息

信息，关于	请参阅
Externally Guided Motion	应用手册 - 控制器软件*IRC5*

2.64 EulerZYX - 根据定向，获取欧拉角

手册用法

EulerZYX（*Euler ZYX rotations*）用于从orient类型变量获取欧拉角分量。

基本示例

以下示例介绍了函数EulerZYX。

例 1

```
VAR num anglex;
VAR num angley;
VAR num anglez;
VAR pose object;
...
...
anglex := EulerZYX(\X, object.rot);
angley := EulerZYX(\Y, object.rot);
anglez := EulerZYX(\Z, object.rot);
```

返回值

数据类型：num

相关的欧拉角，以度表示，范围为【-180，180】。

变元

```
EulerZYX ([\X] | [\Y] | [\Z] Rotation)
```

[\X]

数据类型：switch

获取X轴周围的旋转。

[\Y]

数据类型：switch

获取Y轴周围的旋转。

[\Z]

数据类型：switch

获取Z轴周围的旋转。

注意!

参数\X、\Y和\Z互相排斥。如果未指定此类参数，则会产生运行时错误。

Rotation

数据类型：orient

用其四元数来表示旋转。

语法

```
EulerZYX '('
  ['\' X ','] | ['\' Y ','] | ['\' Z ','] 
  [Rotation ':='] <expression (IN) of orient> ')'
```

下一页继续

返回值的数据类型是 num 的函数。

相关信息

信息，关于	请参阅
数学指令和函数	技术参考手册 - *RAPID*语言概览，*RAPID*概要 - 数学一节

2.65 EventType - 获取任意事件程序内的当前事件类型

手册用法

可在任意事件程序中使用EventType，随后，返回当前执行的事件类型。

如果从任意程序任务程序调用EventType，则EventType始终返回0，意味着EVENT_NONE。

基本示例

以下示例介绍了函数EventType。

例 1

```
TEST EventType()
CASE EVENT_NONE:
  ! Not executing any event
CASE EVENT_POWERON:
  ! Executing POWER ON event
CASE EVENT_START:
  ! Executing START event
CASE EVENT_STOP:
  ! Executing STOP event
CASE EVENT_QSTOP:
  ! Executing QSTOP event
CASE EVENT_RESTART:
  ! Executing RESTART event
CASE EVENT_RESET:
  ! Executing RESET event
CASE EVENT_STEP:
  ! Executing STEP event
ENDTEST
```

在任意事件程序中使用函数EventType，以发现当前正在执行的系统事件（如有）。

返回值

数据类型： event_type

当前执行的事件类型为1 ... 7，或者如果未执行事件程序时，为0。

预定义数据

以下event_type型预定义符号常量可用于检查返回值：

```
CONST event_type EVENT_NONE := 0;
CONST event_type EVENT_POWERON := 1;
CONST event_type EVENT_START := 2;
CONST event_type EVENT_STOP := 3;
CONST event_type EVENT_QSTOP:= 4;
CONST event_type EVENT_RESTART := 5;
CONST event_type EVENT_RESET := 6;
CONST event_type EVENT_STEP := 7;
```

语法

```
EventType '(' ')'
```

下一页继续

含数据类型event_type的返回值的函数。

相关信息

信息，关于	请参阅
一般的事件程序	技术参考手册 - 系统参数，*Controller - Event Routine*一节
数据类型event_type，预定义常量	第*1403*页的*event_type - 事件程序类型*

2.66 ExecHandler - 获取执行处理器的类型

手册用法

ExecHandler可用于发现是否用任意RAPID程序处理器来执行实际RAPID代码。

基本示例

以下示例介绍了函数ExecHandler。

例 1

```
TEST ExecHandler()
CASE HANDLER_NONE:
  ! Not executing in any routine handler
CASE HANDLER_BWD:
  ! Executing in routine BACKWARD handler
CASE HANDLER_ERR:
  ! Executing in routine ERROR handler
CASE HANDLER_UNDO:
  ! Executing in routine UNDO handler
ENDTEST
```

使用函数ExecHandler ，以发现是否用某种类型的程序处理器来执行代码。

HANDLER_ERR 将得以返回，即使用错误处理器子方法来执行调用。

返回值

数据类型： handler_type

当前执行的处理器类型为1 ... 3，或者如果未用任何程序处理器来执行时，为0。

预定义数据

以下handler_type型预定义符号常量可用于检查返回值：

```
CONST handler_type HANDLER_NONE := 0;
CONST handler_type HANDLER_BWD := 1;
CONST handler_type HANDLER_ERR := 2;
CONST handler_type HANDLER_UNDO := 3;
```

语法

```
ExecHandler '(' ')'
```

含数据类型handler_type的返回值的函数。

相关信息

信息，关于	请参阅
执行处理器的类型	第*1410*页的*handler_type - 执行处理器的类型*

2.67 ExecLevel - 获取执行等级

手册用法

ExecLevel可用于发现有关当前所执行的RAPID代码的当前执行等级。

基本示例

以下示例介绍了函数ExecLevel。

例 1

```
TEST ExecLevel()
CASE LEVEL_NORMAL:
  ! Execute on base level
CASE LEVEL_TRAP:
  ! Execute in TRAP routine
CASE LEVEL_SERVICE:
  ! Execute in service, event or system input interrupt routine
ENDTEST
```

使用函数ExecLevel，以发现当前的执行等级。

返回值

数据类型： exec_level

当前执行等级0...2。

预定义数据

以下event_level型预定义符号常量可用于检查返回值：

```
CONST exec_level LEVEL_NORMAL := 0;
CONST exec_level LEVEL_TRAP := 1;
CONST exec_level LEVEL_SERVICE := 2;
```

语法

```
ExecLevel '(' ')'
```

含数据类型exec_level的返回值的函数。

相关信息

信息，关于	请参阅
有关执行等级的数据类型	第*1404*页的*exec_level - 执行等级*

2.68 Exp - 计算指数值

手册用法

Exp（*Exponential*）用于计算指数值，e^x。

基本示例

以下示例介绍了函数Exp。

例 1

```
VAR num x;
VAR num value;
...
value:= Exp( x);
```

将获得x的指数值的值。

返回值

数据类型：num

指数值e^x。

变元

```
Exp (Exponent)
```

Exponent

数据类型：num

指数参数值。

语法

```
Exp '('
  [Exponent ':='] <expression (IN) of num> ')'
```

返回值的数据类型是 num 的函数。

相关信息

信息，关于	请参阅
数学指令和函数	技术参考手册 - *RAPID*语言概览，*RAPID*概要 - 数学一节

2.69 FileSize - 检索文件的大小

手册用法

FileSize的作用是检索指定文件的大小。

基本示例

以下示例介绍了函数FileSize。

另请参阅第*1077*页的更多示例

例 1

```
PROC listfile(string filename)
  VAR num size;
  size := FileSize(filename);
  TPWrite filename+" size: "+NumToStr(size,0)+" Bytes";
ENDPROC
```

该无返回值程序打印出了指定文件的名称连同尺寸规格。

返回值

数据类型：num

尺寸以字节计。

变元

```
FileSize (Path)
```

Path

数据类型：string

通过完整或相关的路径，指定文件名称。

程序执行

该函数返回一个数字，其通过指定文件的字节数来规定尺寸。

其同时可能获得关于路径的相同信息。

更多示例

有关于函数的基本例子阐述如下。

例 1

该例子列出了“HOME:”路径结构（包括所有子路径）下所有大于1千字节的文件。

```
PROC searchdir(string dirname, string actionproc)
  VAR dir directory;
  VAR string filename;
  IF IsFile(dirname \Directory) THEN
    OpenDir directory, dirname;
    WHILE ReadDir(directory, filename) DO
        ! .. and . is the parent and resp. this directory
        IF filename <> ".." AND filename <> "." THEN
          searchdir dirname+"/"+filename, actionproc;
        ENDIF
    ENDWHILE
```

下一页继续

2.69 FileSize - 检索文件的大小

RobotWare - OS

续前页

```
    CloseDir directory;
  ELSE
    %actionproc% dirname;
  ENDIF
ERROR
  RAISE;
ENDPROC

PROC listfile(string filename)
  IF FileSize(filename) > 1024 THEN
      TPWrite filename;
  ENDIF
ENDPROC

PROC main()
  ! Execute the listfile routine for all files found under the
  ! tree of HOME:
  searchdir "HOME:","listfile";
ENDPROC
```

该程序横穿“HOME:”下的路径结构，并针对各个文件而调用listfile无返回值程序。searchdir为通用部分，其不知晓有关搜索的开始，或者不知晓应当针对各文件调用哪一个程序。其使用IsFile，以检查是否已发现子路径或文件，且其使用后期绑定机制，以调用有关所发现文件的actionproc中指定的无返回值程序。actionproc程序listfile检查文件是否大于1千字节。

错误处理

如果不存在该文件，则将系统变量ERRNO设置为ERR_FILEACC。随后，可用错误处理器对该错误进行处理。

语法

```
FileSize '('
  [ Path ':=' ] < expression (IN) of string> ')'
```

返回值的数据类型是 num 的函数。

相关信息

信息，关于	请参阅
建立路径	第322页的*MakeDir - 创建新路径*
删除路径	第503页的*RemoveDir - 删除路径*
重命名文件	第507页的*RenameFile - 重命名文件*
删除文件	第504页的*RemoveFile - 删除文件*
复制文件	第126页的*CopyFile - 复制文件*
检查文件类型	第1126页的*IsFile - 检查文件的类型*
检查文件系统大小	第1082页的*FSSize - 检索文件系统的大小*
文件和串行通道处理	应用手册 - 控制器软件*IRC5*

2.70 FileTimeDnum - 检索有关文件的时间信息

手册用法

FileTimeDnum将用于检索最后一次文件修改、访问或文件状态变更时间。该时间自1970年1月1日00:00:00（格林威治标准时间）起进行测量，以秒计。该时间以dnum形式返回，或也可选择在stringdig中返回。

基本示例

下列示例说明了函数FileTimeDnum。

另请参阅第*1080*页的更多示例

例 1

```
IF FileTimeDnum ("HOME:/mymod.mod" \ModifyTime) > ModTimeDnum
     ("mymod") THEN
  UnLoad "HOME:/mymod.mod";
  Load \Dynamic, "HOME:/mymod.mod";
ENDIF
```

如果源文件更新一些，此程序将重新加载一个模块。其采用ModTimeDnum来检索指定模块的最后一次修改时间，并将之与源处的FileTimeDnum ("HOME:/mymod.mod" \ModifyTime)相比较。如果源更新一些，那么程序将卸载并再次加载此模块。

返回值

数据类型：dnum

测得的时间以秒计，始于1970年1月1日格林威治标准时间00:00:00。

变元

```
FileTimeDnum ( Path [\ModifyTime] | [\AccessTime] | [\StatCTime]
     [\StrDig])
```

Path

数据类型：string

通过完整或相关的路径而指定的文件。

[\ModifyTime]

数据类型：switch

最后修改时间。

[\AccessTime]

数据类型：switch

最后访问（读取、执行修改）的时间。

[\StatCTime]

数据类型：switch

最后文件状态（访问资格）变更时间。

[\StrDig]

String Digit

数据类型：stringdig

下一页继续

2.70 FileTimeDnum - 检索有关文件的时间信息

续前页

为获得stringdig表达中的文件时间。

程序执行

该函数返回一个数字，其规定了自上次起的时间：

- 修改
- 访问
- 文件状态变更

指定文件的。

其同时可能获得关于路径的相同信息。

更多示例

下文给出了函数FileTimeDnum的更多示例。

此为一个完整的例子，其实施有关最多10个文件的预警服务。

```
LOCAL RECORD falert
  string filename;
  dnum ftime;
ENDRECORD

LOCAL VAR falert myfiles[10];
LOCAL VAR num currentpos:=0;
LOCAL VAR intnum timeint;

PROC alertInit(num freq)
  currentpos:=0;
  CONNECT timeint WITH mytrap;
  ITimer freq,timeint;
ENDPROC

LOCAL TRAP mytrap
  VAR num pos:=1;
  WHILE pos <= currentpos DO
    IF FileTimeDnum(myfiles{pos}.filename \ModifyTime) >
         myfiles{pos}.ftime THEN
      TPWrite "The file "+myfiles{pos}.filename+" is changed";
    ENDIF
    pos := pos+1;
  ENDWHILE
ENDTRAP

PROC alertNew(string filename)
  currentpos := currentpos+1;
  IF currentpos <= 10 THEN
    myfiles{currentpos}.filename := filename;
    myfiles{currentpos}.ftime := FileTimeDnum (filename
         \ModifyTime);
  ENDIF
ENDPROC
```

下一页继续

```
PROC alertFree()
  IDelete timeint;
ENDPROC
```

错误处理

如果不存在该文件，则将系统变量ERRNO设置为ERR_FILEACC。随后，可用错误处理器对该错误进行处理。

语法

```
FileTimeDnum '('
[ Path ':=' ] < expression (IN) of string>
[ '\' ModifyTime] |
[ '\' AccessTime] |
[ '\' StatCTime]
[ '\' StrDig ':=' < variable (VAR) of stringdig> ] ')'
```

含数据类型dnum的返回值的函数。

相关信息

信息，关于	请参阅
已加载模块的最后修改时间	第*1145*页的*ModTimeDnum* - 获取加载模块的文件修改时间
只含数字的字符串	第*1491*页的*stringdig* - 只含数字的字符串
将仅含数字的两个字符串进行比较	第*1247*页的*StrDigCmp* - 将仅含数字的两个字符串进行比较

2.71 FSSize - 检索文件系统的大小

手册用法

FSSize（*File System Size*）用于检索包含指定文件的文件系统的大小。大小以字节、千字节或兆字节计，并作为num返回。

基本示例

以下示例介绍了函数FSSize。

另请参阅第*1083*页的更多示例

例 1

```
PROC main()
  VAR num totalfsyssize;
  VAR num freefsyssize;
  freefsyssize := FSSize("HOME:/spy.log" \Free);
  totalfsyssize := FSSize("HOME:/spy.log" \Total);
  TPWrite NumToStr(((totalfsyssize -
       freefsyssize)/totalfsyssize)*100,0) +" percent used";
ENDPROC
```

该无返回值程序打印出在HOME上使用的磁盘空间：占文件系统（闪存盘/hd0a/）的百分比。

返回值

数据类型：num

尺寸以字节计。

变元

```
FSSize (Name [\Total] | [\Free] [\Kbyte] [\Mbyte])
```

Name

数据类型：string

通过完整或相关的路径，指定文件系统中的文件的名称。

[\Total]

数据类型：switch

检索文件系统中的空间总量。

[\Free]

数据类型：switch

检索文件系统中的空闲空间量。

[\Kbyte]

数据类型：switch

将读取的字节数转换为千字节，例如，将大小除以1024。

[\Mbyte]

数据类型：switch

将读取的字节数转换为兆字节，例如，将大小除以1048576（1024*1024）。

下一页继续

程序执行

该函数返回一个数字，其规定包含指定文件的文件系统的大小。

更多示例

有关于函数FSSize的更多例子阐述如下。

例 1

```
LOCAL VAR intnum timeint;

LOCAL TRAP mytrap
  IF FSSize("HOME:/spy.log" \Free)/FSSize("HOME:/spy.log" \Total)
       <= 0.1 THEN
    TPWrite "The disk is almost full";
    alertFree;
  ENDIF
ENDTRAP

PROC alertInit(num freq)
  CONNECT timeint WITH mytrap;
  ITimer freq,timeint;
ENDPROC

PROC alertFree()
  IDelete timeint;
ENDPROC
```

此为有关实施预警服务的完整例子，当“HOME:”文件系统中的剩余空闲空间小于10%时，其将在FlexPendant示教器上显示警告。

错误处理

系统会生成下列可恢复错误，并在错误处理器中处理这些错误。系统变量ERRNO将被设置成：

ERR_FILEACC	文件系统不存在
ERR_FILESIZE	大小超过一个数字的最大整数值，即8388608

语法

```
FSSize '('
  [ Name ':=' ] < expression (IN) of string>
  [ '\' Total ] | [ '\' Free ]
  [ '\' Kbyte ]
  [ '\' Mbyte ] ')'
```

返回值的数据类型是 num 的函数。

相关信息

信息，关于	请参阅
建立路径	第322页的*MakeDir - 创建新路径*
删除路径	第503页的*RemoveDir - 删除路径*
重命名文件	第507页的*RenameFile - 重命名文件*

下一页继续

2.71 FSSize - 检索文件系统的大小

RobotWare - OS

续前页

信息，关于	请参阅
删除文件	第*504*页的*RemoveFile - 删除文件*
复制文件	第*126*页的*CopyFile - 复制文件*
检查文件类型	第*1126*页的*IsFile - 检查文件的类型*
检查文件大小	第*1077*页的*FileSize - 检索文件的大小*
文件和串行通道处理	应用手册 - 控制器软件*IRC5*

2.72 GetMaxNumberOfCyclicBool - 获取进行循环求值的逻辑条件的最大数

手册用法

GetMaxNumberOfCyclicBool将用于检索可同时关联的进行了循环求值的逻辑条件的最大数。

基本示例

下列示例说明了函数GetMaxNumberOfCyclicBool。

例 1

```
VAR num maxno := 0;
maxno := GetMaxNumberOfCyclicBool();
TpWrite "Maximum cyclic bool: " \Num:=maxno;
```

进行循环求值的关联逻辑条件的最大数应显示在FlexPendant示教器上。

返回值

数据类型：num

语法

```
GetMaxNumberOfCyclicBool '(' ')'
```

返回值的数据类型是 num 的函数。

相关信息

信息，关于	请参阅
设置进行循环求值的逻辑条件	第596页的*SetupCyclicBool - 设置进行循环求值的逻辑条件*
撤销进行循环求值的逻辑条件	第502页的*RemoveCyclicBool - 撤销进行循环求值的逻辑条件*
撤除所有进行循环求值的逻辑条件	第501页的*RemoveAllCyclicBool - 撤除所有进行循环求值的逻辑条件*
进行循环求值的逻辑条件，*Cyclic bool*。	应用手册 - 控制器软件*IRC5*

2.73 GetMecUnitName - 获取机械单元的名称

手册用法

GetMecUnitName用于获取机械单元的名称，并将安装的机械单元之一作为参数。该函数返回机械单元名称，以作为一个string。

基本示例

以下示例介绍了函数GetMecUnitName。

例 1

```
VAR string mecname;
mecname:= GetMecUnitName(ROB1);
```

向mecname分配值"ROB1"，以作为一个string。所有机械单元（数据类型 mecunit），例如，ROB1，均应当在系统中进行预定义。

返回值

数据类型：string

返回值将为作为一个string的机械单元名称。

变元

```
GetMecUnitName ( MechUnit )
```

MechUnit

Mechanical Unit

数据类型：mecunit

MechUnit采用在配置中发现的预定义机械单元之一。

语法

```
GetMecUnitName '('
  [ MechUnit ':=' ] < variable (VAR) of mecunit > ')'
```

返回值的数据类型是 string 的函数。

相关信息

信息，关于	请参阅
机械单元	第*1428*页的*mecunit - 机械单元*

2.74 GetModalPayLoadMode - 获取ModalPayLoadMode值。

手册用法

GetModalPayLoadMode用于获取ModalPayLoadMode。

基本示例

以下示例介绍了函数GetModalPayLoadMode。

例 1

```
IF GetModalPayloadMode() = 1 THEN
  GripLoad piece1;
  MoveL p1, v1000, fine, gripper;
ELSE
  MoveL p1, v1000, fine, tool2 \TLoad:=gripperpiece1;
ENDIF
```

从系统读取ModalPayLoadMode值，并根据值，使用不同的代码，以规定在移动指令中使用的负载。

返回值

数据类型：num

返回值将为设置为一个num的ModalPayLoadMode。

语法

```
GetModalPayloadMode '(' ')'
```

返回值的数据类型是 num 的函数。

相关信息

信息，关于	请参阅
激活和停用载荷的系统参数 *ModalPayLoadMode*。 （主题 Controller，类型 System Misc，行动值，*ModalPayLoadMode*）	技术参考手册 - 系统参数
使用运动指令中的有效负载。	第*388*页的*MoveL - 使机械臂沿直线移动*

2.75 GetMotorTorque - 读取当前的电机扭矩

手册用法

GetMotorTorque用于读取机械臂和外轴电机的当前扭矩。

GetMotorTorque主要用于探测伺服夹具是否保存一个负载。

基本示例

以下示例介绍了函数GetMotorTorque。

另请参阅第*1089*页的更多示例

例 1

```
VAR num motor_torque2;
motor_torque2 := GetMotorTorque(2);
```

将机械臂第二个轴的当前电机扭矩储存在motor_torque2中。

返回值

数据类型：num

机械臂或外轴的指定轴的当前电机扭矩，以牛顿米（Nm）计。

变元

```
GetMotorTorque [\MecUnit ] AxisNo
```

MecUnit

Mechanical Unit

数据类型：mecunit

包含待读取轴的机械单元的名称。如果省略该参数，则读取有关相连机械臂的轴。

AxisNo

数据类型：num

待读取轴的编号（1 - 6）。

程序执行

本函数读取在机械臂和外轴的电机上应用的当前滤过的电机扭矩。

当使用*Test Signal Viewer*或*Tune Master*时，亦可将电机扭矩值看作测试信号编号2000。

限制

根据齿轮摩擦、电机温度等，GetMotorTorque的结果将会有所不同。在同一位置的两次测量可能有所不同。例如，齿轮箱温度可改变摩擦，从而改变结果。

上述限制使其不可能探测到扭矩中的极小变化。

其仅可能读取有关由当前程序任务控制的机械单元的当前扭矩。对于非运动任务，读取有关由相关运动任务控制的机械单元的扭矩是可能的。

下一页继续

更多示例

以下示例介绍了函数GetMotorTorque。

例 1

```
VAR num torque_value;
torque_value := GetMotorTorque(\MecUnit:=STN1, 1);
```

将STN1第一个轴的当前电机扭矩储存在torque_value。

例 2

```
VAR num pre_grip_torque;
VAR num post_grip_torque;
..
MoveJ p10, v1000, fine, Gripper;
! Read the torque for axis 5 before gripping the piece
pre_grip_torque:=GetMotorTorque(5);
! Grip the piece
grip_piece;
! Read the torque for axis 5 after gripping the piece
post_grip_torque:=GetMotorTorque(5);
! Compare torque for axis 5 before and after gripping the piece
piece_gripped:=check_gripped_piece(pre_grip_torque,
     post_grip_torque);
IF piece_gripped = TRUE THEN
  GripLoad piece1;
ELSE
  TPWrite "Failed to grip the piece";
  Stop;
ENDIF
..
```

握紧工件前，读取机械臂轴5的当前电机扭矩。随后，握紧该工件。再次读取扭矩，并将扭矩进行比较，以探测夹具中是否存在额外负载。

错误处理

系统会生成下列可恢复错误，并在错误处理器中处理这些错误。系统变量ERRNO将被设置成：

ERR_AXIS_PAR	指令中的参数轴错误。

语法

```
GetMotorTorque '('
  ['\' MecUnit ':=' < variable (VAR) of mecunit> ',']
  [AxisNo ':=' ] < expression (IN) of num> ')'
```

返回值的数据类型是 num 的函数。

相关信息

信息，关于	请参阅
读取当前电机的角度	第*1200*页的*ReadMotor* - 读取当前电机的角度

2.76 GetNextCyclicBool - 获取进行循环求值的逻辑条件的名称

手册用法

GetNextCyclicBool将用于检索进行循环求值的关联逻辑条件的名称。

基本示例

下列示例说明了函数GetNextCyclicBool。

例 1

```
VAR num listno := 0;
WHILE GetNextCyclicBool(listno, name) DO
TpWrite "Cyclic bool: "+name;
! listno := listno + 1 is done by GetNextCyclicBool
ENDWHILE
```

系统中所有进行循环求值的关联逻辑条件的名称将显示在FlexPendant示教器上。

返回值

数据类型：bool

如果发现进行循环求值的逻辑条件，那么返回值为TRUE，否则为FALSE。

变元

```
GetNextCyclicBool(ListNumber Name)
```

ListNumber

数据类型：num

这指定了将检索哪些进行循环求值的关联逻辑条件项。在返回时，系统通常会将此变量递增一来方便访问列表中的下一条件。列表中第一个进行循环求值的逻辑条件为索引0。

Name

数据类型：string

进行循环求值的逻辑条件的名称

语法

```
GetNextCyclicBool '('
  [ ListNumber ':=' ] < variable (VAR) of num> ','
  [ Name ':=' ] < variable (VAR) of string> ','
  ')'
```

含数据类型bool的返回值的函数。

相关信息

信息，关于	请参阅
设置进行循环求值的逻辑条件	第596页的*SetupCyclicBool - 设置进行循环求值的逻辑条件*
撤销进行循环求值的逻辑条件	第502页的*RemoveCyclicBool - 撤销进行循环求值的逻辑条件*
撤除所有进行循环求值的逻辑条件	第501页的*RemoveAllCyclicBool - 撤除所有进行循环求值的逻辑条件*

下一页继续

信息，关于	请参阅
进行循环求值的逻辑条件，*Cyclic bool*。	应用手册 - 控制器软件*IRC5*

2.77 GetNextMechUnit - 获取有关机械单元的名称和数据

手册用法

GetNextMechUnit（*Get Next Mechanical Unit*）用于在机器人系统中检索机械单元的名称。除机械单元名称外，还可检索机械单元的若干可选属性。

基本示例

下列示例说明了函数GetNextMechUnit。

另请参阅第*1093*页的更多示例

例 1

```
VAR num listno := 0;
VAR string name := "";

TPWrite "List of mechanical units:";
WHILE GetNextMechUnit(listno, name) DO
  TPWrite name;
  ! listno := listno + 1 is done by GetNextMechUnit
ENDWHILE
```

系统中可利用的所有机械单元的名称，将于FlexPendant示教器上显示。

返回值

数据类型：bool

如果发现机械单元，则TRUE，否则，FALSE。

变元

```
GetNextMechUnit( ListNumber UnitName [\MecRef] [\TCPRob] [\NoOfAxes]
     [\MecTaskNo] [\MotPlanNo] [\Active] [\DriveModule]
     [\OKToDeact])
```

ListNumber

数据类型：num

这规定了系统内构件机械单元列表中有待检索的各项。返回时，该变量始终由系统递增一，从而使其便于访问列表中的下一个单元。列表中的第一个机械单元拥有指数0。

UnitName

数据类型：string

机械单元名称

[\MecRef]

数据类型：mecunit

机械单元的系统引用。

[\TCPRob]

数据类型：bool

如果机械单元为TCP机械臂，则TRUE，否则，FALSE。

[\NoOfAxes]

数据类型：num

下一页继续

有关机械单元的轴的数量。整数值。

[\MecTaskNo]

数据类型：num

用于控制机械单元的程序任务编号。位于范围1-20中的整数值。如果不由任何程序任务控制，则返回-1。

用系统参数域控制器定义该实际连接（可在运行时的某些应用中重新定义）。

[\MotPlanNo]

数据类型：num

控制机械单元的运动规划器编号。位于范围1-6中的整数值。如果不由任何运动规划器控制，则返回-1。

用系统参数域控制器来确定该连接。

[\Active]

数据类型：bool

如果机械单元有效，则TRUE，否则，FALSE。

[\DriveModule]

数据类型：num

由该机械单元使用的驱动模块编号1 - 4。

[\OKToDeact]

数据类型：bool

如果允许停用来自RAPID程序的机械单元，则返回TRUE。

更多示例

下文给出了更多指令GetNextMechUnit的示例。

例 1

```
VAR num listno := 4;
VAR string name := "";
VAR bool found := FALSE;

found := GetNextMechUnit (listno, name);
```

如果将found设置为TRUE，则机械单元编号4的名称将位于变量name中，否则，name仅包含一个空字符串。

语法

```
GetNextMechUnit '('
  [ ListNumber ':=' ] < variable (VAR) of num> ','
  [ UnitName ':=' ] < variable (VAR) of string> ','
  [ '\' MecRef ':=' < variable (VAR) of mecunit> ]
  [ '\' TCPRob ':=' < variable (VAR) of bool> ]
  [ '\' NoOfAxes ':=' < variable (VAR) of num> ]
  [ '\' MecTaskNo ':=' < variable (VAR) of num> ]
  [ '\' MotPlanNo ':=' < variable (VAR) of num> ]
  [ '\' Active ':=' < variable (VAR) of bool>]
  [ '\' DriveModule ':=' < variable (VAR) of num>]
```

下一页继续

2.77 GetNextMechUnit - 获取有关机械单元的名称和数据

RobotWare - OS

续前页

```
[ '\' OKToDeact ':=' < variable (VAR) of bool>]
')'
```

含数据类型bool的返回值的函数。

相关信息

信息，关于	请参阅
机械单元	第*1428*页的*mecunit - 机械单元*
启用/停用机械单元	第*24*页的*ActUnit - 启用机械单元* 第*140*页的*DeactUnit - 停用机械单元*
非值数据类型的特征	技术参考手册 - *RAPID*语言概览，基本特征 - 数据类型一节

2.78 GetNextSym - 获取下一个匹配的符号

手册用法

GetNextSym（*Get Next Symbol*）用于同SetDataSearch一同使用，以从系统检索数据对象。

基本示例

以下示例介绍了函数GetNextSym。

例 1

```
VAR datapos block;
VAR string name;
VAR bool truevar:=TRUE;
...
SetDataSearch "bool" \Object:="my.*" \InMod:="mymod"\LocalSym;
WHILE GetNextSym(name,block) DO
  SetDataVal name\Block:=block,truevar;
ENDWHILE
```

该会话将模块mymod中以my开始的所有局部bool数据对象设置为TRUE。

返回值

数据类型：bool

如果已检索一个新对象，则TRUE。随后，对象名称及其封闭块在其参数中返回。

如果没有更多的对象匹配，则FALSE。

变元

```
GetNextSym (Object Block [\Recursive])
```

Object

数据类型：string

用以储存待检索数据对象名称的变量（VAR或PERS）。

Block

数据类型：datapos

对象的封闭块。

[\Recursive]

数据类型：switch

这将强制研究进入以下封闭块，例如，如果搜索过程已经从任务等级开始，则其亦将搜索任务下的模块和程序。

语法

```
GetNextSym '('
  [ Object ':=' ] < variable or persistent (INOUT) of string > ','
  [ Block ':='] <variable (VAR) of datapos>
  ['\' Recursive ] ')'
```

含数据类型bool的返回值的函数。

下一页继续

2.78 GetNextSym - 获取下一个匹配的符号

RobotWare - OS

续前页

相关信息

信息，关于	请参阅
定义在搜索会话中设置的符号	第*582*页的*SetDataSearch - 定义在搜索序列中设置的符号*
获取数据对象的值	第*213*页的*GetDataVal - 获得数据对象的值*
设置数据对象的值	第*586*页的*SetDataVal - 设置数据对象的值*
设置许多数据对象的值	第*578*页的*SetAllDataVal - 在定义设置下，设置所有数据对象的值*
相关的数据类型`datapos`	第*1382*页的*datapos - 数据类型的封闭块*
Advanced RAPID	产品规格 - 控制器软件*IRC5*

2.79 GetNumberOfCyclicBool - 获取进行循环求值的逻辑条件的编号

手册用法

GetNumberOfCyclicBool将用于检索进行循环求值的关联逻辑条件的编号。

基本示例

下列示例说明了函数GetNumberOfCyclicBool。

例 1

```
VAR num listno := 0;
listno := GetNumberOfCyclicBool();
TpWrite "Connected cyclic bool: " \Num:=listno;
```

进行循环求值的关联逻辑条件的编号应显示在FlexPendant示教器上。

返回值

数据类型：num

语法

```
GetNumberOfCyclicBool '(' ')'
```

返回值的数据类型是 num 的函数。

相关信息

信息，关于	请参阅
设置进行循环求值的逻辑条件	第596页的*SetupCyclicBool - 设置进行循环求值的逻辑条件*
撤销进行循环求值的逻辑条件	第502页的*RemoveCyclicBool - 撤销进行循环求值的逻辑条件*
撤除所有进行循环求值的逻辑条件	第501页的*RemoveAllCyclicBool - 撤除所有进行循环求值的逻辑条件*
进行循环求值的逻辑条件，*Cyclic bool*。	应用手册 - 控制器软件*IRC5*

2.80 GetServiceInfo - 从系统获取服务信息

手册用法

GetServiceInfo用于从系统读取服务信息。该函数作为一个string，返回服务信息，

基本示例

以下示例介绍了函数GetServiceInfo。

另请参阅第*1099*页的更多示例

例 1

```
VAR string mystring;
VAR num mynum;
IF TaskRunRob() THEN
  mystring:=GetServiceInfo(ROB_ID \DutyTimeCnt);
  IF StrToVal(mystring, mynum) = FALSE THEN
    TPWrite "Conversion failed!";
    Stop;
  ENDIF
ENDIF
```

如果任务控制一个机械臂，则使用预定义变量ROB_ID来读取运行计时器。随后，将字符串值转换为一个数值。

返回值

数据类型：string

有关指定机械单元的服务信息值。在以下参数中，读取更多有关返回值的信息。

变元

```
GetServiceInfo (MechUnit [\DutyTimeCnt])
```

MechUnit

Mechanical Unit

数据类型：mecunit

用于获取信息的机械单元的名称。

[\DutyTimeCnt]

Duty Time Counter

数据类型：switch

返回有关在参数MechUnit中使用的机械单元的运行计时器。如果在虚拟控制器中使用该选项，则返回含“0”的字符串。

运行计时器记录了机械单元电机开启以及停止制动的小时数。

程序执行

读取有关使用过的可选参数的服务信息。

下一页继续

更多示例

有关于如何使用函数GetServiceInfo的更多例子阐述如下。

例 1

```
VAR string mystring;
mystring:=GetServiceInfo(ROB_1 \DutyTimeCnt);
TPWrite "DutyTimeCnt for ROB_1: " + mystring;
mystring:=GetServiceInfo(ROB_2 \DutyTimeCnt);
TPWrite "DutyTimeCnt for ROB_2: " + mystring;
mystring:=GetServiceInfo(INTERCH \DutyTimeCnt);
TPWrite "DutyTimeCnt for INTERCH: " + mystring;
mystring:=GetServiceInfo(STN_1 \DutyTimeCnt);
TPWrite "DutyTimeCnt for STN_1: " + mystring;
mystring:=GetServiceInfo(STN_2 \DutyTimeCnt);
TPWrite "DutyTimeCnt for STN_2: " + mystring;
```

获取有关一个multimove系统中所有机械单元的运行计时器的信息，并将各值写入到FlexPendant示教器上。

语法

```
GetServiceInfo '('
  [MechUnit ':=' ] <variable (VAR) of mecunit> ','
  ['\' DutyTimeCnt] ')'
```

返回值的数据类型是 string 的函数。

相关信息

信息，关于	请参阅
机械单元	第*1428*页的*mecunit* - 机械单元。

2.81 GetSignalOrigin - 获得有关I/O信号来源的信息

手册用法

GetSignalOrigin用于获得有关I/O信号来源的信息。

基本示例

以下示例介绍了函数GetSignalOrigin：

例 1

```
VAR signalorigin myorig;
VAR string signalname;
...
myorig:=GetSignalOrigin(mysignal, signalname);
IF myorig = SIGORIG_NONE THEN
  TPWrite "Signal cannot be used. AliasIO needed.";
ELSEIF myorig = SIGORIG_CFG THEN
  TPWrite "Signal "+signalname+" is defined in I/O configuration.";
ELSEIF myorig = SIGORIG_ALIAS THEN
  TPWrite "Signal is declared in RAPID.";
  TPWrite "Name according to the I/O configuration: "+signalname;
ENDIF
```

上述代码可用于确定名为mysignal的信号的来源。

返回值

数据类型：signalorigin

signalorigin如下表所述。

返回值	符号常量	备注
0	SIGORIG_NONE	在RAPID中声明了I/O信号变量，且没有别名耦合。
1	SIGORIG_CFG	在I/O配置中配置信号。
2	SIGORIG_ALIAS	在RAPID中声明了I/O信号变量，且拥有一个同I/O配置中所配置的I/O信号耦合的别名。

变元

```
GetSignalOrigin Signal SignalName
```

Signal

数据类型：signalxx

信号名称。必须为数据类型signaldo、signaldi、signalgo、signalgi、signalao或signalai.

SignalName

数据类型：string

符合I/O配置的信号名称，或空字符串。

程序执行

函数返回以下预定义信号来源之一：SIGORIG_NONE、SIGORIG_CFG或SIGORIG_ALIAS。

下一页继续

如果返回SIGORIG_NONE，则SignalName包含一个空字符串。

如果返回SIGORIG_CFG或SIGORIG_ALIAS，则参数SignalName包含符合I/O配置的I/O信号名称。

可以在通用程序中使用GetSignalOrigin，以检查信号是否拥有别名耦合，且是否为正确物理I/O信号的耦合。

语法

```
GetSignalOrigin
  [ Signal':=' ] < variable (VAR) of anytype > ','
  [ SignalName ':=' ] < variable (VAR) of string > ';'
```

相关信息

信息，关于	请参阅
输入/输出指令	技术参考手册 - *RAPID*语言概览
输入/输出功能性概述	技术参考手册 - *RAPID*语言概览
I/O配置	技术参考手册 - 系统参数
确定I/O信号以及别名	第*28*页的*AliasIO - 确定I/O信号以及别名*
数据类型signalorigin	第*1475*页的*signalorigin - 介绍I/O信号来源*

2.82 GetSysInfo - 获取系统相关信息

手册用法

GetSysInfo将用于读取有关系统的信息。可取得的信息包括序列号、软件版本、软件版本名称、机器人类型、控制柜ID、WAN ip地址、控制柜语言或系统名称。

基本示例

以下示例介绍了函数GetSysInfo。

例 1

```
VAR string serial;
VAR string version;
VAR string versionname;
VAR string rtype;
VAR string cid;
VAR string lanip;
VAR string clang;
VAR string sysname;
serial := GetSysInfo(\SerialNo);
version := GetSysInfo(\SWVersion);
versionname := GetSysInfo(\SWVersionName);
rtype := GetSysInfo(\RobotType);
cid := GetSysInfo(\CtrlId);
lanip := GetSysInfo(\LanIp);
clang := GetSysInfo(\CtrlLang);
sysname := GetSysInfo(\SystemName);
```

序列号将保存在变量serial中，RobotWare版本将保存在变量version中，RobotWare版本名称将保存在versionname中，机器人编号将保存在变量rtype，中，控制柜ID号将保存在变量cid中，WAN ip地址将保存在变量lanip中，控制柜语言将保存在变量clang中，当前活跃系统的名称将保存在sysname中。

返回的字符串的示例：

序列号：24-12345

软件版本：ROBOTWARE_6.03.xxxx

软件版本名称：6.03.00.00

机器人类型：IRB 2400-16/1.5 Type A

控制柜ID：MyRobot

WAN IP地址：192.168.8.103

语言：en

系统名称：MySystem

返回值

数据类型：string

序列号、软件版本、软件版本名称、机器人类型、控制柜ID、WAN IP地址、控制柜语言或系统名称中的一项。请阅读有关下文第*1103*页的变元中的返回值的更多信息。

下一页继续

变元

```
GetSysInfo ([\SerialNo] | [\SWVersion] | [\SWVersionName] |
     [\RobotType] | [\CtrlId] | [\LanIp] | [\CtrlLang] |
     [\SystemName])
```

必须存在参数SerialNo、SWVersion、SWVersionName、RobotType、CtrlId、LanIp、CtrlLang或SystemName中的一项。

[\SerialNo]

Serial Number

数据类型：switch

返回序列号。

[\SWVersion]

Software Version

数据类型：switch

按*PRODUCTS*文件夹中安装的形式，返回RobotWare媒体版本。

[\SWVersionName]

Software Version Name

数据类型：switch

返回RobotWare媒体版本显示名称。

[\RobotType]

数据类型：switch

在当前或相关任务中返回机械臂类型。如果机械单元并非一个TCP机械臂，则返回“-”。

[\CtrlId]

Controller ID

数据类型：switch

返回控制器ID。如果未指定控制器ID，则返回一个空的字符串。如果将该选项用于虚拟控制器中，则返回含“VC”的字符串。

[\LanIp]

Lan Ip address

数据类型：switch

返回有关控制器的广域网ip地址。如果将该选项用于虚拟控制器中，则返回含“VC”的字符串。如果未在系统中配置广域网ip地址，则返回一个空字符串。

[\CtrlLang]

Controller Language

数据类型：switch

返回在控制器上使用的语言。

返回值	语言
cs	捷克语
zh	汉语（简体中文，大陆中文）
da	丹麦语

下一页继续

2.82 GetSysInfo - 获取系统相关信息

RobotWare - OS

续前页

返回值	语言
nl	荷兰语
en	英语
fi	芬兰语
fr	法语
de	德国
hu	匈牙利语
it	意大利语
ja	日语
ko	韩语
pl	波兰语
pt	葡萄牙语（巴西葡萄牙语）
ro	罗马尼亚语
ru	俄语
sl	斯洛维尼亚语
es	西班牙语
sv	瑞典语
tr	土耳其语

`[\SystemName]`

数据类型：`switch`

返回有效系统名称。

语法

```
GetSysInfo '('
  ['\'SerialNo]
  | ['\' SWVersion]
  | ['\' SWVersionName]
  | ['\' RobotType]
  | ['\' CtrlId]
  | ['\' LanIp]
  | ['\' CtrlLang]
  | ['\' SystemName] ')'
```

返回值的数据类型是 `string` **的函数。**

相关信息

信息，关于	请参阅
测试系统的识别号	*第1138页的IsSysId - 测试系统识别号*

2.83 GetTaskName - 获取当前任务的名称和编号

手册用法

GetTaskName用于获取当前程序任务的识别号及其名称和编号。

同时有可能从某些非运动任务获取相关运动任务的名称和编号。针对*MultiMove*系统，系统参数*Controller/Tasks/Use Mechanical Unit Group*定义相关的运动任务，且在基座系统中，主要任务始终为同任意其他任务相关的运动任务。

基本示例

以下示例介绍了函数GetTaskName。

例 1

```
VAR string taskname;
...
taskname := GetTaskName();
```

在变量taskname中返回当前的任务名称。

例 2

```
VAR string taskname;
VAR num taskno;
...
taskname := GetTaskName(\TaskNo:=taskno);
```

在变量taskname中返回当前的任务名称。将任务的整数识别号储存在变量taskno中。

例 3

```
VAR string taskname;
VAR num taskno;
...
taskname := GetTaskName(\MecTaskNo:=taskno);
```

如果当前任务为非运动任务，则在变量taskname中返回相关运动任务的名称。将相关运动任务的数字识别号储存在变量taskno中。

如果当前任务控制一些机械单元，则在变量taskname中返回当前任务名称。将任务的数字识别号储存在变量taskno中。

返回值

数据类型：string

执行函数的任务的名称或相关运动任务的名称。

变元

```
GetTaskName ( [\TaskNo] | [\MecTaskNo] )
```

[\TaskNo]

数据类型：num

返回当前的任务名称（与不使用任何开关 \TaskNo或\MecTaskNo时的功能相同）。同时获取当前表示为一个整数值的任务的识别号。返回的数字将位于范围1-20中。

[\MecTaskNo]

数据类型：num

下一页继续

2.83 GetTaskName - 获取当前任务的名称和编号

RobotWare - OS

续前页

返回相关运动任务的名称或当前运动任务的名称。同时获得表示为一个整数值的相关或当前运动任务的识别号。返回的数字将位于范围1-20中。

语法

```
GetTaskName '('
  [ \TaskNo ':=' ] < variable (VAR) of num >
  [ \MecTaskNo ':=' ] < variable (VAR) of num > ')'
```

返回值的数据类型是 `string` 的函数。

相关信息

信息，关于	请参阅
多任务处理	技术参考手册 - *RAPID*语言概览，*RAPID*概述 - *RAPID*概要-多任务处理一节 技术参考手册 - *RAPID*语言概览，基本特征 - 多任务处理一节

2.84 GetTime - 将当前时间作为一个数值来读取

手册用法

GetTime用于将当前系统时间的指定分量作为一个数值来读取。

GetTime可用于：

- 使程序在特定时间采取行动
- 在一个工作日实施特定活动
- 放弃在周末实施特定活动
- 根据当日时间，对错误作出不同响应。

基本示例

以下示例介绍了函数GetTime。

另请参阅第*1107*页的更多示例

例 1

```
hour := GetTime(\Hour);
```

将当前小时数储存在变量hour中。

返回值

数据类型：num

下面规定了四个时间分量之一。

变元

```
GetTime ( [\WDay] | [\Hour] | [\Min] | [\Sec] )
```

[\WDay]

数据类型：switch

返回当前的工作日。范围：1到7（周一到周日）。

【\小时】

数据类型：switch

返回当前的小时数。范围：0到23。

[\Min]

数据类型：switch

返回当前的分钟数。范围：0到59。

[\Sec]

数据类型：switch

返回当前的秒数。范围：0到59。

必须指定其中一个参数，否则，在出现错误消息时，停止程序执行。

更多示例

有关于函数GetTime的更多例子阐述如下。

例 1

```
weekday := GetTime(\WDay);
```

下一页继续

2.84 GetTime - 将当前时间作为一个数值来读取

RobotWare - OS

续前页

```
hour := GetTime(\Hour);
IF weekday < 6 AND hour >6 AND hour < 16 THEN
  production;
ELSE
  maintenance;
ENDIF
```

如果为工作日，且时间介于7:00到15:59之间，则机械臂进行生产。在任何其他时间，机械臂处于维护模式。

语法

```
GetTime '('
['\' WDay ]
| [ '\' Hour ]
| [ '\' Min ]
| [ '\' Sec ] ')'
```

含num型返回值的函数。

相关信息

信息，关于	请参阅
时间和日期指令	技术参考手册 - *RAPID*语言概览，*RAPID*概要 - 系统&时间一节
设置系统时钟	操作员手册 - 带 *FlexPendant* 的 *IRC5*，改变*FlexPendant*示教器设置一节

2.85 GInput - 读取组输入信号的值

手册用法

GInput用于读取一组数字信号输入信号的当前值。

> **注意**
>
> 注意，函数GInput为遗留函数，其无须再作使用。有关编程的替代方案和建议方法，请参见例子。

基本示例

以下示例介绍了函数GInput。

例 1

```
IF GInput(gi2) = 5 THEN ...
...
IF gi2 = 5 THEN ...
```

如果信号gi2的当前值等于5，则 ...

返回值

数据类型：num

信号的当前值（正整数）。

读取该组中各信号的值，并解释为无符号二进制数字。随后，将该二进制数字转换成一个整数。

返回的值位于范围内，其取决于该组中信号的数量。

信号数目	允许值
1	0-1
2	0-3
3	0-7
4	0-15
5	0-31
6	0-63
7	0-127
8	0-255
9	0-511
10	0-1023
11	0-2047
12	0-4095
13	0-8191
14	0-16383
15	0-32767
16	0-65535
17	0-131071

下一页继续

2.85 GInput - 读取组输入信号的值

续前页

信号数目	允许值
18	0-262143
19	0-524287
20	0-1048575
21	0-2097151
22	0-4194303
23	0-8388607

变元

```
GInput (Signal)
```

`Signal`

数据类型：`signalgi`

待读取信号组的名称。

语法

```
GInput '('
  [ Signal ':=' ] < variable (VAR) of signalgi > ')'
```

含数据类型num的返回值的函数。

相关信息

信息，关于	请参阅
23位以上的组输入信号的读取值	第*1111*页的*GInputDnum - 读取组输入信号的值*
输入/输出指令	技术参考手册 - *RAPID*语言概览，*RAPID*概要 - 输入和输出信号一节
输入/输出功能性概述	技术参考手册 - *RAPID*语言概览，运动和*I/O*原则 - *I/O*原则一节
I/O配置	技术参考手册 - 系统参数

2.86 GInputDnum - 读取组输入信号的值

手册用法

GInputDnum将用于读取一组大于23位的数字输入信号组的当前值。

基本示例

以下示例介绍了函数GInputDnum。

例 1

```
IF GInputDnum(gi2) = 55 THEN ...
```

如果信号gi2的当前值等于55，则 ...

例 2

```
IF GInputDnum(gi2) = 4294967295 THEN ...
```

如果信号gi2的当前值等于4294967295，则 ...

返回值

数据类型：dnum

信号的当前值（正整数）。

读取该组中各信号的值，并解释为无符号二进制数字。随后，将该二进制数字转换成一个整数。

返回的值位于范围内，其取决于该组中信号的数量。

信号数目	允许值
1	0-1
2	0-3
3	0-7
4	0-15
5	0-31
6	0-63
7	0-127
8	0-255
9	0-511
10	0-1023
11	0-2047
12	0-4095
13	0-8191
14	0-16383
15	0-32767
16	0-65535
17	0-131071
18	0-262143
19	0-524287

下一页继续

2.86 GInputDnum - 读取组输入信号的值

RobotWare - OS

续前页

信号数目	允许值
20	0-1048575
21	0-2097151
22	0-4194303
23	0-8388607
24	0-16777215
25	0-33554431
26	0-67108863
27	0-134217727
28	0-268435455
29	0-536870911
30	0-1073741823
31	0-2147483647
32	0-4294967295

变元

```
GInputDnum (Signal)
```

`Signal`

数据类型：`signalgi`

待读取信号组的名称。

错误处理

系统会生成下列可恢复错误，并在错误处理器中处理这些错误。系统变量ERRNO将被设置成：

如果信号变量是RAPID中声明的变量，则`ERR_NO_ALIASIO_DEF`。尚未同I/O配置以及指令`AliasIO`中确定的I/O信号相连。

如果与I/O单元无接触，则`ERR_NORUNUNIT`。

如果无法访问I/O信号（仅对ICI现场总线有效），则`ERR_SIG_NOT_VALID`。

语法

```
GInputDnum '('
  [ Signal ':=' ] < variable (VAR) of signalgi > ')'
```

含数据类型`dnum`的返回值的函数。

相关信息

信息，关于	请参阅
读取组输入信号的值	第*1109*页的*GInput - 读取组输入信号的值*
输入/输出指令	技术参考手册 - *RAPID*语言概览，*RAPID*概要 - 输入和输出信号一节
输入/输出功能性概述	技术参考手册 - *RAPID*语言概览，运动和*I/O*原则 - *I/O*原则一节
I/O配置	技术参考手册 - 系统参数

2.87 GOutput - 读取一组数字信号输出信号的值

手册用法

GOutput用于读取一组数字信号输出信号的当前值。

基本示例

以下示例介绍了函数GOutput。

例 1

```
IF GOutput(go2) = 5 THEN ...
```

如果信号go2的当前值等于5，则 ...

返回值

数据类型：num

信号的当前值（正整数）。

读取该组中各信号的值，并解释为无符号二进制数字。随后，将该二进制数字转换成一个整数。

返回的值位于范围内，其取决于该组中信号的数量。

信号数量	允许值
1	0-1
2	0-3
3	0-7
4	0-15
5	0-31
6	0-63
7	0-127
8	0-255
9	0-511
10	0-1023
11	0-2047
12	0-4095
13	0-8191
14	0-16383
15	0-32767
16	0-65535
17	0-131071
18	0-262143
19	0-524287
20	0-1048575
21	0-2097151
22	0-4194303

下一页继续

2.87 GOutput - 读取一组数字信号输出信号的值

RobotWare - OS

续前页

信号数量	允许值
23	0-8388607

变元

```
GOutput (Signal)
```

`Signal`

数据类型：`signalgo`

待读取信号组的名称。

错误处理

系统会生成下列可恢复错误，并在错误处理器中处理这些错误。系统变量ERRNO将被设置成：

如果信号变量是RAPID中声明的变量，则ERR_NO_ALIASIO_DEF。尚未同I/O配置以及指令AliasIO中确定的I/O信号相连。

如果与I/O单元无接触，则ERR_NORUNUNIT。

如果无法访问I/O信号（仅对ICI现场总线有效），则ERR_SIG_NOT_VALID。

语法

```
GOutput '('
  [ Signal ':=' ] < variable (VAR) of signalgo > ')'
```

含数据类型num的返回值的函数。

相关信息

信息，关于	请参阅
设置一个输出信号组	第*591*页的*SetGO - 改变一组数字信号输出信号的值*
读取一组输出信号	第*1115*页的*GOutputDnum - 读取组输出信号的值*
读取一组输入信号	第*1111*页的*GInputDnum - 读取组输入信号的值*
输入/输出指令	技术参考手册 - *RAPID*语言概览，*RAPID*概要 - 输入和输出信号一节
输入/输出功能性概述	技术参考手册 - *RAPID*语言概览，运动和*I/O*原则 - *I/O*原则一节
I/O配置	技术参考手册 - 系统参数

2.88 GOutputDnum - 读取组输出信号的值

手册用法

GOutputDnum将用于读取一组大于23位的数字输出信号组的当前值。

基本示例

以下示例介绍了函数GOutputDnum。

例 1

```
IF GOutputDnum(go2) = 55 THEN ...
```

如果信号go2的当前值等于55，则 ...

例 2

```
IF GOutputDnum(go2) = 4294967295 THEN ...
```

如果信号go2的当前值等于4294967295，则 ...

返回值

数据类型：dnum

信号的当前值（正整数）。

读取该组中各信号的值，并解释为无符号二进制数字。随后，将该二进制数字转换成一个整数。

返回的值位于范围内，其取决于该组中信号的数量。

信号数目	允许值
1	0-1
2	0-3
3	0-7
4	0-15
5	0-31
6	0-63
7	0-127
8	0-255
9	0-511
10	0-1023
11	0-2047
12	0-4095
13	0-8191
14	0-16383
15	0-32767
16	0-65535
17	0-131071
18	0-262143
19	0-524287

下一页继续

2.88 GOutputDnum - 读取组输出信号的值

RobotWare - OS

续前页

信号数目	允许值
20	0-1048575
21	0-2097151
22	0-4194303
23	0-8388607
24	0-16777215
25	0-33554431
26	0-67108863
27	0-134217727
28	0-268435455
29	0-536870911
30	0-1073741823
31	0-2147483647
32	0-4294967295

变元

```
GOutputDnum (Signal)
```

`Signal`

数据类型：`signalgo`

待读取信号组的名称。

错误处理

系统会生成下列可恢复错误，并在错误处理器中处理这些错误。系统变量ERRNO将被设置成：

如果信号变量是RAPID中声明的变量，则`ERR_NO_ALIASIO_DEF`。尚未同I/O配置以及指令`AliasIO`中确定的I/O信号相连。

如果与I/O单元无接触，则`ERR_NORUNUNIT`。

如果无法访问I/O信号（仅对ICI现场总线有效），则`ERR_SIG_NOT_VALID`。

语法

```
GOutputDnum '('
  [ Signal ':=' ] < variable (VAR) of signalgo > ')'
```

含数据类型`dnum`的返回值的函数。

相关信息

信息，关于	请参阅
设置一个输出信号组	第591页的*SetGO - 改变一组数字信号输出信号的值*
输入/输出指令	计数参考手册 - *RAPID*概述，*RAPID*摘要 - 输入和输出信号一节
输入/输出功能性概述	计数参考手册 - *RAPID*概述，运动和*I/O*原则一节

下一页继续

信息，关于	请参阅
I/O配置	技术参考手册- 系统参数

2.89 HexToDec - 从十六进制转换为十进制

手册用法

HexToDec用于将基座16中可读字符串中指定的数字转换为基座10中可读字符串中指定的数字。

应当根据字符集【0-9，A-F，a-f】，构建输入字符串。

程序处理数字从0多达9223372036854775807十进制或7FFFFFFFFFFFFFFF十六进制。

基本示例

以下示例介绍了函数HexToDec。

例 1

```
VAR string str;

str := HexToDec("5F5E0FF");
```

变量str被赋予值“99999999”。

返回值

数据类型：string

将字符串转换成代表非参数字符串中给定数字的十进制。

变元

```
HexToDec ( Str )
```

Str

String

数据类型：string

用于转换的字符串。

语法

```
HexToDec '('
  [ Str ':=' ] <expression (IN) of string> ')'
```

返回值的数据类型是 string 的函数。

相关信息

信息，关于	请参阅
字符串功能	技术参考手册 - *RAPID*语言概览，*RAPID*概要 - 字符串函数一节
字符串的定义	第*1489*页的*string - 字符串*
字符串值	技术参考手册 - *RAPID*语言概览，基本特征 - 基本元素一节

2.90 IndInpos - 在位置状态中的独立轴

手册用法

IndInpos用于测试一个独立轴是否已达到选定位置。

基本示例

以下示例介绍了函数IndInpos。

例 1

```
IndAMove Station_A,1\ToAbsNum:=90,20;
WaitUntil IndInpos(Station_A,1) = TRUE;
WaitTime 0.2;
```

等待直至Station_A的轴1位于90度位置。

返回值

数据类型：bool

本表描述了IndInpos的返回值：

返回值	轴状态
TRUE	就位且拥有零速度。
FALSE	未就位和/或未拥有零速度。

变元

```
IndInpos ( MecUnit Axis )
```

MecUnit

Mechanical Unit

数据类型：mecunit

机械单元名称

Axis

数据类型：num

机械单元当前轴的编号（1-6）。

限制

通过指令IndCMove来执行的一个独立轴，始终返回值FALSE，即使在将速度设置为零时。

应当在指令后增加0.2秒的等待时间，以确保已经实现正确的状态。对于性能不佳的外轴而言，该时期应当更长。

错误处理

如果未启用该轴，则将系统变量ERRNO设置为ERR_AXIS_ACT。

如果该轴未处于独立模式，则将系统变量ERRNO设置为ERR_AXIS_IND。

随后，可用错误处理器来处理此类错误。

下一页继续

2.90 IndInpos - 在位置状态中的独立轴

Independent Axis

续前页

语法

```
IndInpos '('
  [ MecUnit ':=' ] < variable (VAR) of mecunit> ','
  [ Axis ':=' ] < expression (IN) of num> ')'
```

含数据类型bool的返回值的函数。

相关信息

信息，关于	请参阅
普通的独立轴	技术参考手册 - *RAPID*语言概览，运动和*I/O*原则 - 程序执行期间的定位一节
其他独立的指令和功能	技术参考手册 - *RAPID*语言概览，*RAPID*概要 - 运动一节
检查独立轴的速度状态	第*1121*页的*IndSpeed* - 独立的速度状态
定义独立接头	技术参考手册 - 系统参数，运动 - 臂一节

2.91 IndSpeed - 独立的速度状态

手册用法

IndSpeed用于测试一个独立轴是否已达到选定速度。

基本示例

以下示例介绍了函数IndSpeed。

例 1

```
IndCMove Station_A, 2, 3.4;
WaitUntil IndSpeed(Station_A,2 \InSpeed) = TRUE;
WaitTime 0.2;
```

等待直至Station_A的轴2已达到速度3.4度/秒。

返回值

数据类型：bool

本表描述了IndSpeed \IndSpeed的返回值：

返回值	轴状态
TRUE	已达到选定的速度。
FALSE	尚未达到选定的速度。

本表描述了IndSpeed \ZeroSpeed:的返回值

返回值	轴状态
TRUE	零速度
FALSE	非零速度

变元

```
IndSpeed ( MecUnit Axis [ \InSpeed ] | [ \ZeroSpeed ] )
```

MecUnit

Mechanical Unit

数据类型：mecunit

机械单元名称

Axis

数据类型：num

机械单元当前轴的编号（1-6）。

[\InSpeed]

数据类型：switch

如果轴已达到选定的速度，则IndSpeed返回值TRUE，否则FALSE。

[\ZeroSpeed]

数据类型：switch

如果轴已达到零速度，则IndSpeed返回值TRUE，否则FALSE。

如果省略参数\InSpeed和\ZeroSpeed，则将显示错误消息。

下一页继续

2.91 IndSpeed - 独立的速度状态

Independent Axis

续前页

限制

在以下情形下，函数IndSpeed\InSpeed将始终返回值FALSE：

- 机械臂处于手动模式，并以减速运行。
- 使用VelSet指令，降低速度。
- 从生产窗口降低速度。

应当在指令后增加0.2秒的等待时间，以确保获得正确的状态。对于性能不佳的外轴而言，该时期应当更长。

错误处理

如果未启用该轴，则将系统变量ERRNO设置为ERR_AXIS_ACT。

如果该轴未处于独立模式，则将系统变量ERRNO设置为ERR_AXIS_IND。

随后，可用错误处理器来处理此类错误。

语法

```
IndSpeed '('
  [ MecUnit ':=' ] < variable (VAR) of mecunit> ','
  [ Axis ':=' ] < expression (IN) of num>
  [ '\' InSpeed ] | [ '\' ZeroSpeed ] ')'
```

含数据类型bool的返回值的函数。

相关信息

信息，关于	请参阅
普通的独立轴	技术参考手册 - *RAPID*语言概览，运动和*I/O*原则 - 程序执行期间的定位一节
其他独立的指令和功能	技术参考手册 - *RAPID*语言概览，*RAPID*概要 - 运动一节
更多示例	第*244*页的*IndCMove* - 独立的连续运动
检查独立轴的位置状态	第*1119*页的*IndInpos* - 在位置状态中的独立轴
定义独立接头	技术参考手册 - 系统参数，运动 - 臂一节

2.92 IOUnitState - 获取I/O单元的当前状态

手册用法

IOUnitState用于查明I/O单元的当前状态。其物理状态和逻辑状态规定I/O单元的状态。

基本示例

以下示例介绍了函数IOUnitState。

例 1

```
IF (IOUnitState("UNIT1" \Phys)=IOUNIT_PHYS_STATE_RUNNING) THEN
  ! Possible to access some signal on the I/O unit
ELSE
  ! Read/Write some signal on the I/O unit result in error
ENDIF
```

完成测试，以查看I/O单元UNIT1是否运行正常。

例 2

```
IF (IOUnitState("UNIT1" \Logic)=IOUNIT_LOG_STATE_DISABLED) THEN
  ! Unit is disabled by user from RAPID or FlexPendant
ELSE
  ! Unit is enabled.
ENDIF
```

完成测试，以查看I/O单元UNIT1是否禁用。

返回值

数据类型：iounit_state

根据使用的可选参数\Logic或\Phys或不使用任何可选参数，返回值将拥有不同的值。

I/O单元逻辑状态描述了用户可命令I/O单元进入的状态。当使用可选参数\Logic时，I/O单元的状态如下表所述。

返回值	符号常量	备注
10	IOUNIT_LOG_STATE_DISABLED	由用户根据RAPID、FlexPendant示教器或系统参数而禁用的I/O单元。
11	IOUNIT_LOG_STATE_ENABLED	由用户根据RAPID、FlexPendant示教器或系统参数而启用的I/O单元。默认在启动后。

当I/O单元逻辑上由用户启用，且现场总线驱动器旨在使I/O单元进入物理状态IOUNIT_PHYS_STATE_RUNNING时，I/O单元会因各种原因（参见下表）而进入其他状态。

当使用可选参数\Phys时，I/O单元的状态如下表所述。

返回值	符号常量	备注
20	IOUNIT_PHYS_STATE_DEACTIVATED	I/O单元未运行，由用户禁用
21	IOUNIT_PHYS_STATE_RUNNING	I/O单元正在运行

下一页继续

2.92 IOUnitState - 获取I/O单元的当前状态

RobotWare - OS

续前页

返回值	符号常量	备注
22	IOUNIT_PHYS_STATE_ERROR	由于一些运行时错误，I/O单元不工作
23	IOUNIT_PHYS_STATE_UNCONNECTED	I/O单元得以配置，但是未与I/O总线相连，或者I/O总线停止。
24	IOUNIT_PHYS_STATE_UNCONFIGURED	I/O单元未配置，但是与I/O总线相连。 1)
25	IOUNIT_PHYS_STATE_STARTUP	I/O单元处于起动模式。 1)
26	IOUNIT_PHYS_STATE_INIT	创建I/O单元。 1)

注意

当未使用任何可选参数\Phys或\Logic时，下表确定了I/O单元的状态。

返回值	符号常量	备注
1	IOUNIT_RUNNING	I/O单元运行正常
2	IOUNIT_RUNERROR	由于一些运行时错误，I/O单元不工作
3	IOUNIT_DISABLE	由用户根据RAPID或FlexPendant示教器 而禁用的I/O单元
4	IOUNIT_OTHERERR	其他配置或启动错误

1) 通过RobotWare - OS的当前版本，不可能在RAPID程序中获得该状态。

变元

```
IOUnitState (UnitName [\Phys] | [\Logic])
```

`UnitName`

数据类型：`string`

待检查I/O单元的名称（与配置的名称相同）。

`[\Phys]`

物理

数据类型：`switch`

如果使用该参数，则读取I/O单元的物理状态。

`[\Logic]`

逻辑

数据类型：`switch`

如果使用该参数，则读取I/O单元的逻辑状态。

语法

```
IOUnitState '('
  [ UnitName ':=' ] < expression (IN) of string >
  [ '\' Phys] | [ '\' Logic] ')'
```

含数据类型`iounit_state`的返回值的函数。

下一页继续

相关信息

信息，关于	请参阅
I/O单元的状态	第*1417*页的*iounit_state - I/O单元的状态*
启用一个I/O单元	第*268*页的*IOEnable - 启用I/O单元*
禁用一个I/O单元	第*265*页的*IODisable - 停用I/O单元*
输入/输出指令	技术参考手册 - *RAPID*语言概览，*RAPID*概要 - 输入和输出信号一节
输入/输出功能性概述	技术参考手册 - *RAPID*语言概览，运动和*I/O*原则 - *I/O*原则一节
I/O配置	技术参考手册 - 系统参数

2.93 IsFile - 检查文件的类型

手册用法

IsFile函数获得有关指定文件或路径的信息，并检查其是否与指定类型相同。如果未指定任何类型，则仅实施存在检查。

路径参数指定文件。不需要有关指定文件的读取、写入或执行许可，但是，指向文件的路径名称中所列的所有路径必须可供搜索。

基本示例

以下示例介绍了函数IsFile。

另请参阅第*1127*页的更多示例

例 1

```
PROC printFT(string filename)
  IF IsFile(filename \Directory) THEN
    TPWrite filename+" is a directory";
    RETURN;
  ENDIF
  IF IsFile(filename \Fifo) THEN
    TPWrite filename+" is a fifo file";
    RETURN;
  ENDIF
  IF IsFile(filename \RegFile) THEN
    TPWrite filename+" is a regular file";
    RETURN;
  ENDIF
  IF IsFile(filename \BlockSpec) THEN
    TPWrite filename+" is a block special file";
    RETURN;
  ENDIF
  IF IsFile(filename \CharSpec) THEN
    TPWrite filename+" is a character special file";
    RETURN;
  ENDIF
ENDPROC
```

该例子打印出filename以及FlexPendant示教器上指定文件的类型。

返回值

数据类型：bool

如果指定的类型与实际类型相匹配，则函数将返回TRUE，否则，FALSE。未指定任何类型时，如果文件存在，则其返回TRUE，否则，FALSE。

变元

```
IsFile (Path [\Directory] [\Fifo] [\RegFile] [\BlockSpec]
     [\CharSpec])
```

Path

数据类型：string

下一页继续

通过完整或相关的路径而指定的文件。

[\Directory]

数据类型：switch

文件为一个路径。

[\Fifo]

数据类型：switch

文件为一个先进先出文件。

[\RegFile]

数据类型：switch

文件为一个常规文件，即标准二进制或ASCII文件。

[\BlockSpec]

数据类型：switch

文件为一个块特殊文件。

[\CharSpec]

数据类型：switch

文件为一个字符特殊文件。

程序执行

该函数返回一个bool，其规定匹配与否。

更多示例

有关于函数IsFile的更多例子阐述如下。

例 1

该例子通常横穿路径结构函数。

```
PROC searchdir(string dirname, string actionproc)
  VAR dir directory;
  VAR string filename;
  IF IsFile(dirname \Directory) THEN
    OpenDir directory, dirname;
    WHILE ReadDir(directory, filename) DO
        ! .. and . is the parent and resp. this directory
        IF filename <> ".." AND filename <> "." THEN
          searchdir dirname+"/"+filename, actionproc;
        ENDIF
    ENDWHILE
    CloseDir directory;
  ELSE
    %actionproc% dirname;
  ENDIF
ERROR
  RAISE;
ENDPROC
```

下一页继续

2.93 IsFile - 检查文件的类型
RobotWare - OS
续前页

```
PROC listfile(string filename)
  TPWrite filename;
ENDPROC

PROC main()
  ! Execute the listfile routine for all files found under the
  ! tree of HOME:
  searchdir "HOME:","listfile";
ENDPROC
```

该程序横穿“HOME:”下的路径结构，并针对各个文件而调用listfile无返回值程序。searchdir为通用部分，其不知晓有关搜索的开始，或者不知晓应当针对各文件调用哪一个程序。其使用IsFile，以检查是否已发现子路径或文件，且其使用后期绑定机制，以调用有关所发现文件的actionproc中指定的无返回值程序。actionproc程序应当为一个无返回值程序，其中一个参数为string型。

错误处理

如果不存在该文件，但是存在指定类型的文件，则将系统变量ERRNO设置为ERR_FILEACC。随后，可用错误处理器对该错误进行处理。

限制

针对串行通道或现场总线使用该函数是不可能的。

如果针对FTP或NFS挂载磁盘而使用，则并非始终更新文件存在或类型信息。为获得正确的信息，在使用IsFile之前，可能需要针对搜索路径（通过指令Open）的明令。

语法

```
Isfile '('
  [ Path ':=' ] < expression (IN) of string>
  [ '\' Directory ]
  | [ '\' Fifo ]
  | [ '\' RegFile ]
  | [ '\' BlockSpec ]
  | [ '\' CharSpec ] ')'
```

含数据类型bool的返回值的函数。

相关信息

信息，关于	请参阅
目录	第*1384*页的*dir - 路径结构*
打开路径	第*419*页的*OpenDir - 打开路径*
关闭路径	第*117*页的*CloseDir - 关闭路径*
读取路径	第*1197*页的*ReadDir - 读取路径中的下个条目*
建立路径	第*322*页的*MakeDir - 创建新路径*
删除路径	第*503*页的*RemoveDir - 删除路径*
重命名文件	第*507*页的*RenameFile - 重命名文件*
删除文件	第*504*页的*RemoveFile - 删除文件*
复制文件	第*126*页的*CopyFile - 复制文件*

下一页继续

信息，关于	请参阅
检查文件大小	第*1077*页的*FileSize - 检索文件的大小*
检查文件系统大小	第*1082*页的*FSSize - 检索文件系统的大小*
文件和串行通道处理	应用手册 - 控制器软件*IRC5*

2.94 IsMechUnitActive - 机械单元是否有效

手册用法

IsMechUnitActive （*Is Mechanical Unit Active*）用于检查机械单元是否启用。

基本示例

以下示例介绍了函数IsMechUnitActive。

例 1

```
IF IsMechUnitActive(SpotWeldGun)
  CloseGun SpotWeldGun;
```

如果机械单元SpotWeldGun有效，则将调用程序CloseGun，并由此关闭焊枪。

返回值

数据类型：bool

函数返回：

- 如果启用机械单元，则TRUE
- 如果停用机械单元，则FALSE

变元

```
IsMechUnitActive ( MechUnit )
```

MechUnit

Mechanical Unit

数据类型：mecunit

机械单元名称

语法

```
IsMechUnitActive '('
  [ MechUnit ':='] < variable (VAR) of mecunit> ')'
```

含数据类型bool的返回值的函数。

相关信息

信息，关于	请参阅
启动机械单元	第*24*页的*ActUnit - 启用机械单元*
停用机械单元	第*140*页的*DeactUnit - 停用机械单元*
机械单元	第*1428*页的*mecunit - 机械单元*

2.95 IsPers - 是永久数据对象

手册用法

IsPers用于测试一个数据对象是否为一个永久变量。

基本示例

以下示例介绍了函数IsPers。

例 1

```
PROC procedure1 (INOUT num parameter1)
  IF IsVar(parameter1) THEN
    ! For this call reference to a variable
    ...
  ELSEIF IsPers(parameter1) THEN
    ! For this call reference to a persistent variable
    ...
  ELSE
    ! Should not happen
    EXIT;
  ENDIF
ENDPROC
```

根据实际参数parameter1是否为一个变量或一个永久变量，无返回值程序procedure1将采取不同的行动。

返回值

数据类型：bool

如果经测试的实际INOUT参数为一个永久变量，则TRUE。如果经测试的实际INOUT参数并非一个永久变量，则FALSE。

变元

```
IsPers (DatObj)
```

DatObj ()

Data Object

数据类型：任意类型

正式INOUT参数的名称。

语法

```
IsPers '('
  [ DatObj ':=' ] < var or pers (INOUT) of any type > ')'
```

含数据类型bool的返回值的函数。

相关信息

信息，关于	请参阅
测试是否为变量	第*1139*页的*IsVar - 是变量*
参数的类型（存取模式）	技术参考手册 - *RAPID*语言概览，基本特征 - 程序一节

2.96 IsStopMoveAct - 停止移动表示是否启用

手册用法

IsStopMoveAct用于获取有关当前或相关运动任务的停止移动标志的状态。

基本示例

以下示例介绍了函数IsStopMoveAct。

例 1

```
stopflag2:= IsStopMoveAct(\FromNonMoveTask);
```

如果在当前或相关运动任务中设置非运动任务的停止移动标志，则stopflag2将为TRUE，否则，其将为FALSE。

例 2

```
IF IsStopMoveAct(\FromMoveTask) THEN
  StartMove;
ENDIF
```

如果在当前运动任务中设置运动任务的停止移动标志，则其将通过StartMove指令重置。

返回值

数据类型：bool

如果选定的停止移动标志得以设置，则返回值将为TRUE，否则，返回值将为FALSE。

变元

```
IsStopMoveAct ( [\FromMoveTask] | [\FromNonMoveTask] )
```

[\FromMoveTask]

数据类型：switch

FromMoveTask用于获取私人运动任务型停止移动标志的状态。

仅可通过以下方面来设置此类停止移动标志：

- 运动任务自身连同指令StopMove
- 在离开任务中的RestoPath后
- 执行期间，用异步错误处理器来处理任意StorePath 之前以及任意RestoPath之后的过程错误或运动错误

[\FromNonMoveTask]

数据类型：switch

FromNonMoveTask用于获取任意非运动任务的停止移动标志的状态。仅可通过相关的任意非运动任务或含指令StopMove的所有运动任务，设置此类停止移动标志。

语法

```
IsStopMoveAct '('
  ['\' FromMoveTask]
  | ['\' FromNonMoveTask] ')'
```

含数据类型bool的返回值的函数。

下一页继续

相关信息

信息，关于	请参阅
停止机械臂的移动	第*690*页的*StopMove - 停止机械臂的移动*
使机器臂重新开始移动	第*663*页的*StartMove - 重启机械臂移动*

2.97 IsStopStateEvent - 测试是否移动程序指针

手册用法

IsStopStateEvent返回有关当前程序任务中程序指针（PP）移动的信息。

基本示例

以下示例介绍了函数IsStopStateEvent。

例 1

```
IF IsStopStateEvent (\PPMoved) = TRUE THEN
  ! PP has been moved during the last program stop
ELSE
  ! PP has not been moved during the last program stop
ENDIF

IF IsStopStateEvent (\PPToMain) THEN
  ! PP has been moved to main routine during the last program stop
ENDIF
```

返回值

数据类型：bool

最后停止状态期间的“如果”状态以及程序指针的移动情形。

如果程序指针已经在最后停止期间移动，则TRUE。

如果程序指针在最后停止期间未曾移动，则FALSE。

如果程序指针已经移动到主程序，则\PPMoved和\PPToMain 均将返回TRUE。

如果程序指针已经移动到一个程序，则\PPMoved和\PPToMain均将返回TRUE。

如果程序指针已经在一系列程序内移动，则\PPMoved将返回TRUE，且\PPToMain将返回FALSE。

在调用服务程序（将执行环境维持在主程序次序中）后，\PPMove将返回FALSE，且\PPToMain将返回FALSE。

变元

```
IsStopStateEvent ([\PPMoved] | [\PPToMain])
```

[\PPMoved]

数据类型：switch

测试是否移动过程序指针。

[\PPToMain]

数据类型：switch

测试是否已将程序指针移动至主程序或一个程序。

限制

向前或向后执行期间，在绝大多数情况下不能使用该函数，因为系统处于各单一步骤之间的停止状态。

下一页继续

语法

```
IsStopStateEvent '('
  ['\' PPMoved] | ['\ä PPToMain] ')'
```

含数据类型bool的返回值的函数。

相关信息

信息，关于	请参阅
制作自己的指令	技术参考手册 - *RAPID*语言概览，离线编程 - 制作你自己的指令一节
Advanced RAPID	产品规格 - 控制器软件*IRC5*

2.98 IsSyncMoveOn - 测试是否处于同步移动模式

手册用法

IsSyncMoveOn用于测试Motion Task型当前程序任务是否处于同步移动模式。

同时可能从一些Non Motion Task来测试相关的Motion Task是否处于同步移动模式。系统参数*Controller/Tasks/Use Mechanical Unit Group*定义相关的Motion Task。

当在StorePath等级下执行Motion Task时，IsSyncMoveOn将测试任务是否处于该等级的同步模式中，其不取决于初始水平的同步模式。

指令IsSyncMoveOn通常用于*MultiMove*系统以及选项*Coordinated Robots*，但是却可以用于任意系统和任意程序任务中。

基本示例

以下示例介绍了函数IsSyncMoveOn。

例 1

位于任务T_ROB1中的程序实例

```
PERS tasks task_list{2} := [ ["T_ROB1"], ["T_ROB2"] ];
VAR syncident sync1;
VAR syncident sync2;
VAR syncident sync3;

PROC main()
  ...
  MoveL p_zone, vmax, z50, tcp1;
  WaitSyncTask sync1, task_list;
  MoveL p_fine, v1000, fine, tcp1;
  syncmove;
  ...
ENDPROC

PROC syncmove()
  SyncMoveOn sync2, task_list;
  MoveL * \ID:=10, v100, z10, tcp1 \WOBJ:= rob2_obj;
  MoveL * \ID:=20, v100, fine, tcp1 \WOBJ:= rob2_obj;
  SyncMoveOff sync3;
  UNDO
    SyncMoveUndo;
ENDPROC
```

位于任务BCK1中的程序实例

```
PROC main()
  ...
  IF IsSyncMoveOn() THEN
    ! Connected Motion Task is in synchronized movement mode
  ELSE
    ! Connected Motion Task is in independent mode
  ENDIF
  ...
```

下一页继续

```
ENDPROC
```

执行IsSyncMoveOn期间，在背景任务BCK1中，我们测试相关运动任务在当时是否处于同步移动模式。

返回值

数据类型：bool

如果当前或相关程序任务此刻处于同步移动模式，则TRUE，否则，FALSE。

程序执行

测试当前或相关程序任务此刻是否处于同步移动模式。如果正在StorePath level下执行 MotionTask，则SyncMoveOn将测试任务是否处于StorePath level而非初始等级上的同步移动中。

语法

```
IsSyncMoveOn '(' ')'
```

含数据类型bool的返回值的函数。

相关信息

信息，关于	请参阅
指定协作的程序任务	第*1500*页的*tasks - RAPID*程序任务
同步点的识别号	第*1496*页的*syncident* - 同步点的识别号
起动协调同步移动	第*709*页的*SyncMoveOn* - 起动协调同步移动
结束协调同步移动	第*704*页的*SyncMoveOff* - 结束协调同步移动
设置独立移动	第*719*页的*SyncMoveUndo* - 设置独立移动
存储路径，并在新的等级上执行	第*695*页的*StorePath* - 发生中断时，存储路径

2.99 IsSysId - 测试系统识别号

手册用法

IsSysId（*System Identity*）可用于测试使用系统序列号的系统识别号。

基本示例

以下示例介绍了函数IsSysId。

例 1

```
IF NOT IsSysId("6400-1234") THEN
  ErrWrite "System identity fault", "Faulty system identity for
       this program";
  EXIT;
ENDIF
```

本程序针对含序列号6400-1234的特殊机器人系统而制定，且无法由另一个机器人系统使用。

返回值

数据类型：bool

TRUE = 机器人系统序列号与试验中指定的序列号相同。

FALSE = 机器人系统序列号不与试验中指定的序列号相同。

变元

```
IsSysId ( SystemId)
```

SystemId

数据类型：string

机器人系统序列号，标记系统识别号。

语法

```
IsSysId '('
  [ SystemId ':=' ] < expression (IN) of string> ')'
```

含数据类型bool的返回值的函数。

相关信息

信息，关于	请参阅
读取系统信息	第*1102*页的*GetSysInfo - 获取系统相关信息*

2.100 IsVar - 是变量

手册用法

IsVar用于测试一个数据对象是否为一个变量。

基本示例

以下示例介绍了函数IsVar。

例 1

```
PROC procedure1 (INOUT num parameter1)
  IF IsVAR(parameter1) THEN
    ! For this call reference to a variable
    ...
  ELSEIF IsPers(parameter1) THEN
    ! For this call reference to a persistent variable
    ...
  ELSE
    ! Should not happen
    EXIT;
  ENDIF
ENDPROC
```

根据实际参数parameter1是否为一个变量或一个永久变量，无返回值程序procedure1将采取不同的行动。

返回值

数据类型：bool

如果经测试的实际INOUT参数为一个变量，则TRUE。如果经测试的实际INOUT参数并非一个变量，则FALSE。

变元

```
IsVar (DatObj)
```

DatObj

Data Object

数据类型：任意类型

正式INOUT参数的名称。

语法

```
IsVar '('
  [ DatObj ':=' ] < var or pers (INOUT) of any type > ')'
```

含数据类型bool的返回值的函数。

相关信息

信息，关于	请参阅
测试永久数据对象是否	第*1131*页的*IsPers - 是永久数据对象*
参数的类型（存取模式）	技术参考手册 - *RAPID*语言概览，基本特征 - 程序一节

2.101 MaxRobSpeed - 最大机械臂速度

手册用法

MaxRobSpeed（*Maximum Robot Speed*）返回有关所用机械臂类型的最大TCP速度。

基本示例

以下示例介绍了函数MaxRobSpeed。

例 1

```
TPWrite "Max. TCP speed in mm/s for my robot="\Num:=MaxRobSpeed();
```

在FlexPendant示教器上写入消息Max. TCP speed in mm/s for my robot = 5000。

返回值

数据类型：num

针对所用机械臂类型和普通实践TCP值，返回最大的TCP速度，以mm/s计。

如果在工具坐标系中使用非常大的TCP值，则应当用大于MaxRobSpeed所返回的TCP速度，创建自己的速度数据。

语法

```
MaxRobSpeed '(' ')'
```

返回值的数据类型是 num 的函数。

相关信息

信息，关于	请参阅
速度的定义	第*1480*页的*speeddata - 速度数据*
最大速率的定义	第*849*页的*VelSet - 改变编程速率*

2.102 MirPos - 一处位置的镜像

手册用法

MirPos（*Mirror Position*）用于反映一处位置的平移和旋转零件。

基本示例

以下示例介绍了函数MirPos。

例 1

```
CONST robtarget p1:= [...];
VAR robtarget p2;
PERS wobjdata mirror:= [...];
...
p2 := MirPos(p1, mirror);
```

p1为一处机器人位置，其储存机械臂的一处位置以及工具的一个方位。通过同世界坐标系相关的mirror所定义的坐标系的xy平面，反映出该位置。该结果为新机器人位置数据，将其储存在p2中。

返回值

数据类型： robtarget

新位置，其为输入位置的镜像位置。

变元

```
MirPos (Point MirPlane [\WObj] [\MirY])
```

Point

数据类型： robtarget

输入机械臂位置。该位置的方位规定了工具坐标系的当前方位。

MirPlane

Mirror Plane

数据类型： wobjdata

用于定义镜面的工件数据。镜面为MirPlane中定义的工件坐标系的xy平面。定义工件坐标系相对于用户坐标系的位置（同时在MirPlane中定义），其反过来相对于世界坐标系来定义。

[\WObj]

Work Object

数据类型： wobjdata

用于定义工件坐标系和用户坐标系的工件数据，由此定义输入位置*Point*。如果遗漏该参数，则相对于世界坐标系来定义该位置。

注意！

如果通过一个有效的工件来创建位置，则必须在参数中参考该工件。

[\MirY]

Mirror Y

数据类型：switch

下一页继续

2.102 MirPos - 一处位置的镜像

RobotWare - OS

续前页

如果默认省去该开关，则将通过x轴和z轴来反映工具坐标系。如果指定开关，则将通过y轴和z轴来反映工具坐标系。

限制

未重新计算输入机器人位置数据的机械臂配置零件。

如果使用一个坐标系，则必须将协调单元置于与机械臂相同的任务中。

语法

```
MirPos '('
  [ Point ':=' ] < expression (IN) of robtarget> ','
  [ MirPlane ':='] <expression (IN) of wobjdata> ','
  ['\' WObj ':=' <expression (IN) of wobjdata> ]
  ['\' MirY ] ')'
```

含数据类型robtarget的返回值的函数。

相关信息

信息，关于	请参阅
数学指令和函数	技术参考手册 - *RAPID*语言概览，*RAPID*概要 - 数学一节
位置数据	第*1467*页的*robtarget - 位置数据*
工件数据	第*1523*页的*wobjdata - 工件数据*

2.103 MOD - 评估一个整数模数

手册用法

MOD是用于评估整数除法模数和余数的条件表达式。

基本示例

以下示例介绍了函数MOD。

例 1

```
reg1 := 14 MOD 4;
```

返回值为2，因为14除以4，得到模数2。

例 2

```
VAR dnum mydnum1 := 11;
VAR dnum mydnum2 := 5;
VAR dnum mydnum3;
...
mydnum3 := mydnum1 MOD mydnum2;
```

返回值为1，因为11除以5，得到模数2。

返回值

数据类型：num、dnum

返回整数除法的模数和余数。

语法

```
<expression of num> MOD <expression of num>
```

含数据类型num的返回值的函数。

```
<expression of dnum> MOD <expression of dnum>
```

含数据类型dnum的返回值的函数。

相关信息

信息，关于	请参阅
num - 数值	第*1435*页的*num - 数值*
dnum - 双数值	第*1385*页的*dnum - 双数值*
DIV	第*1060*页的*DIV - 评估一个整数除法*
表达式	技术参考手册 - *RAPID*语言概览

2.104 ModExist - 检查普通程序模块是否存在

手册用法

ModExist（*Module Exist*）用于检查程序任务中是否存在给定模块。

首先搜索已加载模块，此后，如果未发现已加载模块，则搜索已安装模块。

基本示例

以下示例介绍了函数ModExist。

例 1

```
VAR bool mod_exist;
mod_exist:=ModExist ("MyModule");
```

如果任务内存在模块MyModule，则函数将返回TRUE。否则，函数将返回FALSE。

返回值

数据类型：bool

如果发现模块，则TRUE，否则，FALSE。

变元

```
ModExist (ModuleName)
```

ModuleName

数据类型：string

待搜索模块的名称。

语法

```
ModExist '('
  [ ModuleName ':=' ] < expression (IN) of string > ')'
```

含数据类型bool的返回值的函数。

相关信息

信息，关于	请参阅
发现已加载模块的修改时间	第*1145*页的*ModTimeDnum - 获取加载模块的文件修改时间*

2.105 ModTimeDnum - 获取加载模块的文件修改时间

手册用法

ModTimeDnum（*Modify Time*）将用于检索加载模块的最后一次文件修改时间。该模块将以该模块的名称来指定，必须处于任务存储器中。该时间自1970年1月1日00:00:00（格林威治标准时间）起进行测量，以秒计。该时间以dnum形式返回，或也可选择以stringdig形式返回。

基本示例

下列示例说明了函数ModTimeDnum。

另请参阅第*1145*页的更多示例

例 1

```
MODULE mymod
  VAR dnum mytime;
  PROC printMyTime()
    mytime := ModTimeDnum("mymod");
    TPWrite "My time is "+ValToStr(mytime);
  ENDPROC
ENDMODULE
```

返回值

数据类型：dnum

测得的时间以秒计，始于1970年1月1日格林威治标准时间00:00:00。

变元

```
ModTimeDnum ( Object [\StrDig] )
```

Object

数据类型：string

模块的名称。

[\StrDig]

String Digit

数据类型：stringdig

为获得stringdig表达中的模数加载时间。

程序执行

该函数返回一个数值，其规定了在将文件作为系统中的一个普通程序模块进行加载之前修改文件的最后时间。

更多示例

下文给出了函数ModTimeDnum的更多示例。

例 1

```
IF FileTimeDnum ("HOME:/mymod.mod" \ModifyTime) > ModTimeDnum
     ("mymod") THEN
  UnLoad "HOME:/mymod.mod";
  Load \Dynamic, "HOME:/mymod.mod";
```

下一页继续

2.105 ModTimeDnum - 获取加载模块的文件修改时间
续前页

```
    ENDIF
```

如果源文件更新一些，此程序将重新加载一个模块。其采用ModTimeDnum来检索指定模块的最后一次修改时间，并将之与源处的FileTimeDnum ("HOME:/mymod.mod" \ModifyTime)相比较。如果源更新一些，那么程序将卸载并再次加载此模块。

错误处理

如果程序任务中不存在含指定名称的模块，则将系统变量ERRNO设置为ERR_MOD_NOT_LOADED。随后，可用错误处理器对该错误进行处理。

限制

如果在编码的或进行安装共享的模块上使用该函数，那么，该函数将一直返回0。

语法

```
ModTimeDnum '('
  [ Object ':=' ] < expression (IN) of string>
  [ '\' StrDig ':=' < variable (VAR) of stringdig> ] ')'
```

含数据类型dnum的返回值的函数。

相关信息

信息，关于	请参阅
检索有关文件的时间信息	第*1079*页的*FileTimeDnum - 检索有关文件的时间信息*
只含数字的字符串	第*1491*页的*stringdig - 只含数字的字符串*
将仅含数字的两个字符串进行比较	第*1247*页的*StrDigCmp - 将仅含数字的两个字符串进行比较*

2.106 MotionPlannerNo - 获取相连运动规划器编号

手册用法

MotionPlannerNo返回相连运动规划器编号。如果在运动任务中执行MotionPlannerNo，其将返回规划器编号。否则，如果在非运动任务中执行MotionPlannerNo，其将根据系统参数中的设置，返回相连运动规划器编号。

基本示例

以下示例介绍了函数MotionPlannerNo。

例 1

```
!Motion task T_ROB1
PERS string buffer{6} := ["", "", "", "", "", ""];
VAR num motion_planner;

PROC main()
  ...
  MoveL point, v1000, fine, tcp1;
  motion_planner := MotionPlannerNo();
  buffer{motion_planner} := "READY";
  ...
ENDPROC

!Background task BCK1
PERS string buffer{6};
VAR num motion_planner;
VAR string status;

PROC main()
  ...
  motion_planner := MotionPlannerNo();
  status := buffer{motion_planner};
  ...
ENDPROC

!Motion T_ROB2
PERS string buffer{6};
VAR num motion_planner;

PROC main()
  ...
  MoveL point, v1000, fine, tcp1;
  motion_planner := MotionPlannerNo();
  buffer{motion_planner} := "READY";
  ...
ENDPROC

!Background task BCK2
PERS string buffer{6};
VAR num motion_planner;
```

下一页继续

2.106 MotionPlannerNo - 获取相连运动规划器编号

RobotWare - OS

续前页

```
VAR string status;

PROC main()
  ...
  motion_planner := MotionPlannerNo();
  status := buffer{motion_planner};
  ...
ENDPROC
```

使用函数MotionPlannerNo，以寻找同任务相关的运动规划器编号。可在所有运动任务和背景任务中执行完全相同的代码。随后，各背景任务可检查其相关运动任务的状态。

返回值

数据类型：num

相连运动规划器的编号。针对非运动任务，将返回相关机械单元的运动规划器编号。

返回值范围为1 ... 6。

语法

```
MotionPlannerNo '(' ')'
```

返回值的数据类型是 num 的函数。

相关信息

信息，关于	请参阅
指定协作的程序任务	技术参考手册 - 系统参数，*Controller - Task*一节

2.107 NonMotionMode - 读取非运动执行模式

手册用法

NonMotionMode（*Non-Motion Execution Mode*）用于读取程序任务的当前非运动执行模式。根据菜单*ABB*\控制面板\监测，从FlexPendant示教器选定或删除非运动执行模式。

基本示例

以下示例介绍了函数NonMotionMode。

例 1

```
IF NonMotionMode() =TRUE THEN
  ...
ENDIF
```

仅当机械臂处于非运动执行模式时，方才执行程序段。

返回值

数据类型：bool

当前非运动模式如下表所述。

返回值	符号常量	备注
0	FALSE	未使用非运动执行
1	TRUE	使用非运动执行

变元

```
NonMotionMode ( [ \Main] )
```

[\Main]

数据类型：switch

返回相关运动任务的当前运行模式。如果其为运动任务或相关运动任务，则在一个非运动任务中执行函数NonMotionMode时，用于一个多任务系统中，以获取有关实际任务的当前运行模式。

如果省略该参数，则返回值始终反映有关执行函数NonMotionMode的程序任务的当前运行模式。

注意，执行模式与系统而非任意任务相关。这意味着，一个系统中的所有任务均将得出与NonMotionMode相同的返回值。

语法

```
NonMotionMode '(' ['\' Main] ')'
```

含数据类型bool的返回值的函数。

相关信息

信息，关于	请参阅
读取运行模式	第*1158*页的*OpMode - 读取运行模式*

2.108 NOT - 转化一个逻辑值

手册用法

NOT为用于转化一个逻辑值（真/假）的条件表达式。

基本示例

以下示例介绍了条件表达式NOT。

例 1

```
VAR bool mybool;
mybool := NOT mybool;
```

如果mybool为TRUE，则返回值为FALSE。

如果mybool为FALSE，则返回值为TRUE。

例 2

```
VAR bool a;
VAR bool b;
VAR bool c;
...
c := a AND (NOT b);
```

如果a为TRUE，且b为FALSE，则返回值c为TRUE

返回值

数据类型：bool

返回反值。

语法

```
NOT <logical term>
```

相关信息

信息，关于	请参阅
AND	第*962*页的*AND* - 评估一个逻辑值
OR	第*1159*页的*OR* - 评估一个逻辑值
XOR	第*1341*页的*XOR* - 评估一个逻辑值
表达式	技术参考手册 - *RAPID*语言概览

2.109 NOrient - 规范化方位

手册用法

NOrient（*Normalize Orientation*）用于规范化方位（四元数）。

描述

必须使姿态规范化，即平方和必须等于1：

$$q_1^2+q_2^2+q_3^2+q_4^2=1$$

xx0500002452

如果规范略微非规范化，则可能使其规范化。规范化错误为方位分量平方和的绝对值。如果规范化错误大于0.00001且小于0.1，则认为方位略微非规范化。如果规范化错误大于0.1，则定向不可用。

$$ABS(\sqrt{q_1^2+q_2^2+q_3^2+q_4^2}-1)=normerr$$

xx0500002453

normerr>0.1，不可用

normerr>0.00001且normerr< = 0.1，略微非规范化

normerr< = 0.00001，规范化

基本示例

以下示例介绍了函数NOrient。

例 1

我们拥有略微非规范化的位置（0.707170，0，0，0.707170）

$$ABS(\sqrt{0{,}707170^2+0^2+0^2+0{,}707170^2}-1)=0{,}0000894$$

$$0{,}0000894>0{,}00001\Rightarrow \text{unnormalized}$$

xx0500002451

```
VAR orient unnormorient := [0.707170, 0, 0, 0.707170];
VAR orient normorient;
...
...
normorient := NOrient(unnormorient);
```

方位（0.707170，0，0，0.707170）规范化，成为（0.707107，0，0，0.707107）。

返回值

数据类型：orient

规范化的方位

变元

```
NOrient (Rotation)
```

下一页继续

2.109 NOrient - 规范化方位

RobotWare - OS

续前页

旋转

数据类型：orient

有待规范化的方位

语法

```
NOrient '('
  [Rotation ':='] <expression (IN) of orient> ')'
```

含数据类型orient的返回值的函数。

相关信息

信息，关于	请参阅
数学指令和函数	技术参考手册 - *RAPID*语言概览，*RAPID*概要 - 数学一节

2.110 NumToDnum - 将数值转换为双数值

手册用法

NumToDnum将一个num转换为dnum。

基本示例

以下示例介绍了函数NumToDnum。

例 1

```
VAR num mynum:=55;
VAR dnum mydnum:=0;
mydnum:=NumToDnum(mynum);
```

由函数返回如dnum55的数值55。

返回值

数据类型：dnum

dnum型返回值将拥有与num型输入值相同的值。

变元

```
NumToDnum (Value)
```

Value

数据类型：num

有待转换的数值。

语法

```
NumToDnum '('
  [ Value ':=' ] < expression (IN) of num > ')'
```

含数据类型dnum的返回值的函数。

相关信息

信息，关于	请参阅
Num数据类型	第*1435*页的*num - 数值*
Dnum数据类型	第*1385*页的*dnum - 双数值*

2.111 NumToStr - 将数值转换为字符串

手册用法

NumToStr（*Numeric To String*）用于转换数值为字符串。

基本示例

以下示例介绍了函数NumToStr。

例 1

```
VAR string str;
str := NumToStr(0.38521,3);
```

变量str被赋予值"0.385"。

例 2

```
reg1 := 0.38521;
str := NumToStr(reg1, 2\Exp);
```

变量str被赋予值"3.85E-01"。

返回值

数据类型：string

通过指定的小数位数，如有要求，可通过指数，将数值转换为字符串。如有必要，将数值四舍五入。如果不包括任何小数，则抑制小数点。

变元

```
NumToStr (Val Dec [\Exp])
```

Val

Value

数据类型：num

有待转换的数值。

Dec

Decimals

数据类型：num

小数位数。小数位数不得为负或大于数值的可用精度。

[\Exp]

Exponent

数据类型：switch

为了在返回值中使用指数。

语法

```
NumToStr '('
  [ Val ':=' ] <expression (IN) of num>
  [ Dec ':=' ] <expression (IN) of num>
  ['\' Exp ')'
```

返回值的数据类型是 string 的函数。

下一页继续

相关信息

信息，关于	请参阅
字符串功能	技术参考手册 - *RAPID*语言概览，*RAPID*概要 - 字符串函数一节
字符串的定义	第*1489*页的*string - 字符串*
字符串值	技术参考手册 - *RAPID*语言概览，基本特征 - 基本元素一节
将一个双数值转换为一段字符串	第*1063*页的*DnumToStr - 将数值转换为字符串*

2.112 Offs - 取代一个机械臂位置

手册用法

Offs用于在一个机械臂位置的工件坐标系中添加一个偏移量。

基本示例

以下示例介绍了函数Offs。

另请参阅第*1156*页的更多示例

例 1

```
MoveL Offs(p2, 0, 0, 10), v1000, z50, tool1;
```

将机械臂移动至距位置p2（沿z方向）10 mm的一个点。

例 2

```
p1 := Offs (p1, 5, 10, 15);
```

机械臂位置p1沿x方向移动5 mm，沿y方向移动10 mm，且沿z方向移动15 mm。

返回值

数据类型： robtarget

移动的位置数据。

变元

```
Offs (Point XOffset YOffset ZOffset)
```

Point

数据类型： robtarget

有待移动的位置数据。

XOffset

数据类型：num

工件坐标系中x方向的位移。

YOffset

数据类型：num

工件坐标系中y方向的位移。

ZOffset

数据类型：num

工件坐标系中z方向的位移。

更多示例

有关于函数Offs的更多例子阐述如下。

例 1

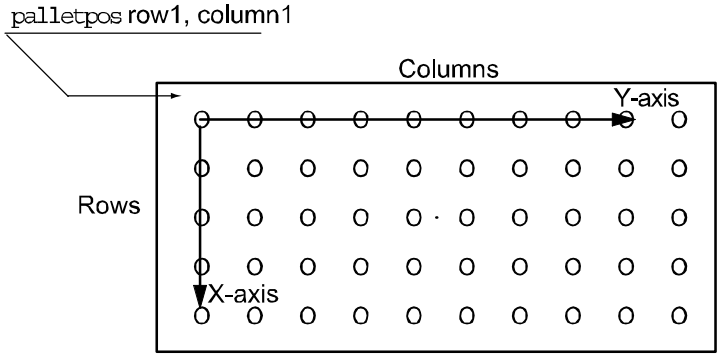
```
PROC pallet (num row, num column, num distance, PERS tooldata tool,
     PERS wobjdata wobj)
VAR robtarget palletpos:=[[0, 0, 0], [1, 0, 0, 0], [0, 0, 0, 0],
     [9E9, 9E9, 9E9, 9E9, 9E9, 9E9]];
```

下一页继续

```
palettpos := Offs (palettpos, (row-1)*distance, (column-1)*distance,
     0);
MoveL palettpos, v100, fine, tool\WObj:=wobj;
ENDPROC
```

制定有关一个托盘的拾料零件的程序。将各托盘定义为一个工件（参见下图）。将待拾取零件（行和列）以及零件之间的距离作为输入参数。在程序外实施行和列指数的增值。

通过定义一个工件，指定可表明托盘位置和方位的图。



xx050002300_

语法

```
Offs '('
  [Point ':='] <expression (IN) of robtarget> ','
  [XOffset ':='] <expression (IN) of num> ','
  [YOffset ':='] <expression (IN) of num> ','
  [ZOffset ':='] <expression (IN) of num> ')'
```

含数据类型robtarget的返回值的函数。

相关信息

信息，关于	请参阅
位置数据	第*1467*页的*robtarget - 位置数据*
数学指令和函数	技术参考手册 - *RAPID*语言概览，*RAPID*概要 - 数学一节
定位器指令	技术参考手册 - *RAPID*语言概览，*RAPID*概要 - 运动一节

2.113 OpMode - 读取运行模式

手册用法

OpMode（*Operating Mode*）用于读取系统的当前运行模式。

基本示例

以下示例介绍了函数OpMode。

例 1

```
TEST OpMode()
CASE OP_AUTO:
  ...
CASE OP_MAN_PROG:
  ...
CASE OP_MAN_TEST:
  ...
DEFAULT:
  ...
ENDTEST
```

根据当前的运行模式，执行不同的程序段。

返回值

数据类型：symnum

当前运行模式如下表所述。

返回值	符号常量	备注
0	OP_UNDEF	未定义的运行模式
1	OP_AUTO	自动的运行模式
2	OP_MAN_PROG	手动运行模式，最快250 mm/s
3	OP_MAN_TEST	手动全速运行模式，100 %

语法

```
OpMode '(' ')'
```

含数据类型symnum的返回值的函数。

相关信息

信息，关于	请参阅
不同的运行模式	操作员手册 - 带 *FlexPendant* 的 *IRC5*
读取运行模式	第*1224*页的*RunMode - 读取运行模式*

2.114 OR - 评估一个逻辑值

手册用法

OR为用于评估一个逻辑值（真/假）的条件表达式。

基本示例

以下示例介绍了函数OR。

例 1

```
VAR num a;
VAR num b;
VAR bool c;
...
c := a>5 OR b=3;
```

如果a大于5，或b等于3，则c的返回值为TRUE。否则，返回值为FALSE。

例 2

```
VAR num mynum;
VAR string mystring;
VAR bool mybool;
VAR bool result;
...
result := mystring="Hello" OR mynum<15 AND mybool;
```

如果mystring为"Hello"，则result的返回值为TRUE。或者如果mynum均小于15，且mybool为TRUE。否则，返回值为FALSE。

首先评估AND声明，随后评估OR声明。由以下行中的插入成分对此进行说明。

```
result := mystring="Hello" OR (mynum<15 AND mybool);
```

返回值

数据类型：bool

如果条件表达式之一或全部正确，则返回值为TRUE，否侧，返回值为FALSE。

语法

```
<expression of bool> OR <expression of bool>
```

含数据类型bool的返回值的函数。

相关信息

信息，关于	请参阅
AND	第962页的*AND - 评估一个逻辑值*
XOR	第1341页的*XOR - 评估一个逻辑值*
NOT	第1150页的*NOT - 转化一个逻辑值*
表达式	技术参考手册 - *RAPID*语言概览

2.115 OrientZYX - 建立欧拉角的定向

手册用法

OrientZYX（*Orient from Euler ZYX angles*）用于建立一个不含欧拉角的定向变量。

基本示例

以下示例介绍了函数OrientZYX。

例 1

```
VAR num anglex;
VAR num angley;
VAR num anglez;
VAR pose object;
...
object.rot := OrientZYX(anglez, angley, anglex)
```

返回值

数据类型：orient

根据欧拉角来确定方位。

按照下列顺序实施旋转：

- 围绕z轴的旋转，
- 围绕新y轴的旋转，
- 围绕新x轴的旋转。

变元

```
OrientZYX (ZAngle YAngle XAngle)
```

ZAngle

数据类型：num

勿扰Z轴的旋转，以度计。

YAngle

数据类型：num

勿扰Y轴的旋转，以度计。

XAngle

数据类型：num

勿扰X轴的旋转，以度计。

按照下列顺序实施旋转：

- 围绕z轴的旋转，
- 围绕新y轴的旋转，
- 围绕新x轴的旋转。

语法

```
OrientZYX '('
  [ZAngle ':='] <expression (IN) of num> ','
  [YAngle ':='] <expression (IN) of num> ','
```

下一页继续

```
[XAngle ':='] <expression (IN) of num> ')'
```

含数据类型orient的返回值的函数。

相关信息

信息，关于	请参阅
数学指令和函数	操作员手册 - 带 *FlexPendant* 的 *IRC5*，*RAPID* 概要 - 数学一节

2.116 ORobT - 从一个位置清除程序位移

手册用法

ORobT（*Object Robot Target*）用于将一个机械臂位置从程序位移坐标系转换至工件坐标系，和/或移除外轴的偏移量。

基本示例

以下示例介绍了函数ORobT。

另请参阅第*1162*页的更多示例

例 1

```
VAR robtarget p10;
VAR robtarget p11;
VAR num wobj_diameter;

p10 := CRobT(\Tool:=tool1 \WObj:=wobj_diameter);
p11 := ORobT(p10);
```

将机械臂和外轴的当前位置储存在p10和p11中。将各值储存在同ProgDisp/ExtOffs坐标系相关的p10中。并将各值储存在同不含任何程序位移和任何外轴偏移量的工件坐标系相关的p11中。

返回值

数据类型： robtarget

转换的位置数据。

变元

```
ORobT (OrgPoint [\InPDisp] | [\InEOffs])
```

OrgPoint

Original Point

数据类型： robtarget

有待转换的原点。

[\InPDisp]

In Program Displacement

数据类型：switch

返回ProgDisp坐标系中的TCP位置，即仅移除外轴偏移量。

[\InEOffs]

In External Offset

数据类型：switch

返回偏移量坐标系中的外轴，即仅移除有关机械臂的程序位移。

更多示例

有关于如何使用函数ORobT的更多例子阐述如下。

例 1

```
p10 := ORobT(p10 \InEOffs );
```

下一页继续

ORobT函数将移除任意有效的程序位移，离开同工件坐标系相关的TCP位置。外轴将维持在偏移量坐标系中。

例 2

```
p10 := ORobT(p10 \InPDisp );
```

ORobT函数将移除外轴的所有偏移量。TCP位置将维持在ProgDisp坐标系中。

语法

```
ORobT '('
  [ OrgPoint ':=' ] < expression (IN) of robtarget>
  ['\' InPDisp] | ['\' InEOffs] ')'
```

含数据类型robtarget的返回值的函数。

相关信息

信息，关于	请参阅
有关机械臂的程序位移的定义	第*448*页的*PDispOn - 启用程序位移* 第*452*页的*PDispSet - 启用使用已知坐标系的程序位移*
有关外轴的偏移量的定义	第*187*页的*EOffsOn - 启用附加轴的偏移量* 第*189*页的*EOffsSet - 启用附加轴（使用已知值）的偏移量*
坐标系	操作员手册 - 带 *FlexPendant* 的 *IRC5*，运动和*I/O*原则 - 坐标系一节

2.117 ParIdPosValid - 用于参数识别的有效机械臂位置

手册用法

ParIdPosValid（*Parameter Identification Position Valid*）检查机械臂位置是否适用于当前的参数识别，例如，工具或有效负载的负载识别。

本指令仅可用于main任务，或者如果在*MultiMove*系统中，则可用于运动任务。

基本示例

以下示例介绍了函数ParIdPosValid。

例 1

```
VAR jointtarget joints;
VAR bool valid_joints{12};

! Check if valid robot type
IF ParIdRobValid(TOOL_LOAD_ID) <> ROB_LOAD_VAL THEN
  EXIT;
ENDIF
! Read the current joint angles
joints := CJointT();
! Check if valid robot position
IF ParIdPosValid (TOOL_LOAD_ID, joints, valid_joints) = TRUE THEN
  ! Valid position for load identification
  ! Continue with LoadId
  ...
ELSE
  ! Not valid position for one or several axes for load
  ! identification
  ! Move the robot to the output data given in variable joints
  ! and do ParIdPosValid once again
  ...
ENDIF
```

在进行工具负载识别之前，检查机械臂位置是否有效。

返回值

数据类型：bool

如果机械臂位置适用于当前的参数识别，则TRUE。

如果机械臂位置不适用于当前的参数识别，则FALSE。

变元

```
ParIdPosValid (ParIdType Pos AxValid [\ConfAngle])
```

ParIdType

数据类型：paridnum

参数识别的类型如下表所述。

值	符号常量	备注
1	TOOL_LOAD_ID	确定工具负载

下一页继续

值	符号常量	备注
2	PAY_LOAD_ID	确定有效负载（参考指令GripLoad）
3	IRBP_K	识别外机械臂IRBP K负载
4	IRBP_L	识别外机械臂IRBP L负载
4	IRBP_C	识别外机械臂IRBP C负载
4	IRBP_C_INDEX	识别外机械臂IRBP C_INDEX负载
4	IRBP_T	识别外机械臂IRBP T负载
5	IRBP_R	识别外机械臂IRBP R负载
6	IRBP_A	识别外机械臂IRBP A负载
6	IRBP_B	识别外机械臂IRBP B负载
6	IRBP_D	识别外机械臂IRBP D负载

Pos

数据类型：jointtarget

变量指定有关所有机械臂和外轴的实际接头角度。根据下表，由ParIdPosValid对变量进行更新。

输入轴接头值	输出轴接头值
有效	未改变
无效	改变为适当值

AxValid

数据类型：bool

含12个元素的数组变量，相当于6个机械臂和6个外轴。根据下表，由ParIdPosValid对变量进行更新。

位置中的输入轴接头值	AxValid中的输出状态
有效	TRUE
无效	FALSE

下一页继续

2.117 ParIdPosValid - 用于参数识别的有效机械臂位置

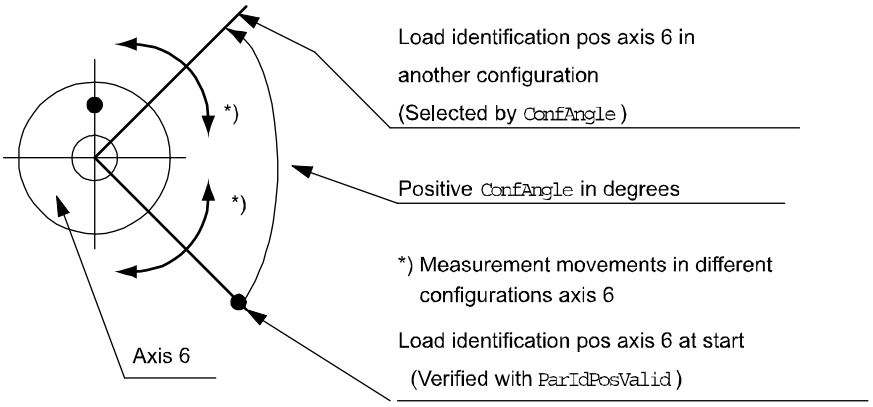
RobotWare - OS

续前页

```
[ \ConfAngle ]
```

数据类型：num

关于特定配置角+/-度数规格的选项参数，用于参数识别。



xx0500002493

如果未指定该参数，则默认+90度。

最小+或-30度。最佳+或-90度。

错误处理

如果出现错误，则将系统变量ERRNO设置为ERR_PID_RAISE_PP。随后，可用错误处理器来该错误。

语法

```
ParIdPosValid '('
  [ ParIdType ':=' ] <expression (IN) of paridnum> ','
  [ Pos ':=' ] <variable (VAR) of jointtarget> ','
  [ AxValid ':=' ] <array variable {*} (VAR) of bool>
  [ '\' ConfAngle ':=' <expression (IN) of num> ] ')'
```

含数据类型bool的返回值的函数。

相关信息

信息，关于	请参阅
参数识别的类型	第*1444*页的*paridnum - 参数识别的类型*
有效的机械臂类型	第*1167*页的*ParIdRobValid - 用于参数识别的有效机械臂类型*
工具或有效负载的负载识别	第*316*页的*LoadId - 工具或有效负载的负载识别*
定位器的负载识别（IRBP）	第*323*页的*ManLoadIdProc - IRBP机械臂的负载识别*

2.118 ParIdRobValid - 用于参数识别的有效机械臂类型

手册用法

ParIdRobValid（*Parameter Identification Robot Valid*）检查机械臂类型是否适用于当前的参数识别，例如，工具或有效负载的负载识别。

本指令仅可用于主任务T_ROB1，或者如果在*MultiMove*系统中，则可用于运动任务中。

基本示例

以下示例介绍了函数ParIdRobValid。

例 1

```
TEST ParIdRobValid (TOOL_LOAD_ID)
  CASE ROB_LOAD_VAL:
    ! Possible to do load identification of tool in actual robot
         type
    ...
  CASE ROB_LM1_LOAD_VAL:
    ! Only possible to do load identification of tool with
    ! IRB 6400FHD if actual load < 200 kg
    ...
  CASE ROB_NOT_LOAD_VAL:
    ! Not possible to do load identification of tool in actual
         robot type
    ...
ENDTEST
```

返回值

数据类型：paridvalidnum

是否可通过当前的机械臂类型，实施指定的参数识别，如下表所述。

值	符号常量	备注
10	ROB_LOAD_VAL	有关实际参数识别的有效机械臂类型
11	ROB_NOT_LOAD_VAL	有关实际参数识别的无效类型。
12	ROB_LM1_LOAD_VAL	如果实际负载<200kg，则有关实际参数识别的有效机械臂类型为IRB 6400FHD。

变元

```
ParIdRobValid(ParIdType [\MechUnit] [\AxisNo])
```

ParIdType

数据类型：paridnum

参数识别的类型如下表所述。

值	符号常量	备注
1	TOOL_LOAD_ID	确定机械臂工具负载
2	PAY_LOAD_ID	确定机械臂有效负载（参考指令GripLoad）
3	IRBP_K	识别外机械臂IRBP K负载

下一页继续

2.118 ParIdRobValid - 用于参数识别的有效机械臂类型

RobotWare - OS

续前页

值	符号常量	备注
4	IRBP_L	识别外机械臂IRBP L负载
4	IRBP_C	识别外机械臂IRBP C负载
4	IRBP_C_INDEX	识别外机械臂IRBP C_INDEX负载
4	IRBP_T	识别外机械臂IRBP T负载
5	IRBP_R	识别外机械臂IRBP R负载
6	IRBP_A	识别外机械臂IRBP A负载
6	IRBP_B	识别外机械臂IRBP B负载
6	IRBP_D	识别外机械臂IRBP D负载

`[ \MechUnit ]`

Mechanical Unit

数据类型：`mecunit`

用于负载识别的机械单元。仅针对外机械臂而指定。如果省略该参数，则使用任务中的TCP机械臂。

`[ \AxisNo ]`

Axis number

数据类型：`num`

机械单元内的轴编号，其保存有待识别的负载。仅针对外机械臂而指定。

当使用参数`\MechUnit`时，必须使用`\AxisNo`。不得在没有`\MechUnit`的情况下使用参数`\AxisNo`。

错误处理

如果出现错误，则将系统变量ERRNO设置为`ERR_PID_RAISE_PP`。随后，可用错误处理器来该错误。

语法

```
ParIdRobValid '('
  [ParIdType ':='] <expression (IN) of paridnum>
  ['\' MechUnit ':=' <variable (VAR) of mecunit>]
  ['\' AxisNo ':=' <expression (IN) of num>] ')'
```

含数据类型`paridvalidnum`的返回值的函数。

相关信息

信息，关于	请参阅
参数识别的类型	第*1444*页的*paridnum - 参数识别的类型*
有待识别的机械单元	第*1428*页的*mecunit - 机械单元*
该函数的结果	第*1446*页的*paridvalidnum - ParIdRobValid的结果*
有效的机械臂位置	第*1164*页的*ParIdPosValid - 用于参数识别的有效机械臂位置*
机械臂工具负载或有效负载的负载识别	第*316*页的*LoadId - 工具或有效负载的负载识别*

下一页继续

信息，关于	请参阅
定位器负载的负载识别	第*323*页的*ManLoadIdProc - IRBP*机械臂的负载识别

2.119 PathLevel - 获取当前的路径等级。

手册用法

PathLevel用于获取当前的路径等级。该函数将显示是否在初始水平上执行任务，或者是否已储存原始路径等级且正在执行新的临时移动。在应用手册 - 控制器软件*IRC5*中读取更多有关于*Path Recovery*的信息。

基本示例

以下示例介绍了函数PathLevel。

另请参阅第*1170*页的更多示例

例 1

```
VAR num level;
level:= PathLevel();
```

如果在一条原始移动路径中执行，则变量level将为1，或者如果在临时的新移动路径中执行，则该变量将为2。

返回值

数据类型：num

存在两种可能的返回值。

返回值	描述
1	在原始移动路径中执行。
2	在StorePath等级中执行，一条临时的新移动路径。

更多示例

有关如何使用函数PathLevel的又一个例子阐述如下。

例 1

```
...
MoveL p100, v100, z10, tool1;
StopMove;
StorePath;
p:= CRobT(\Tool:=tool1);
!New temporary movement
MoveL p1, v100, fine, tool1;
...
level:= PathLevel();
...
MoveL p, v100, fine, tool1;
RestoPath;
StartMove;
...
```

变量level将为2。

限制

必须安装RobotWare附加功能Path Recovery，以便能够在路径等级2下使用函数PathLevel。

下一页继续

语法

```
PathLevel '(' ')'
```

返回值的数据类型是 num 的函数。

相关信息

信息，关于	请参阅
路径恢复	应用手册 - 控制器软件*IRC5*
储存和恢复路径	第*695*页的*StorePath* - 发生中断时，存储路径 第*512*页的*RestoPath* - 中断之后，恢复路径
停止和开始移动	第*663*页的*StartMove* - 重启机械臂移动 第*690*页的*StopMove* - 停止机械臂的移动

2.120 PathRecValidBwd - 是否记录了有效的后退路径

手册用法

PathRecValidBwd用于检查路径记录器是否已启用及所记录的后退路径是否有效。

基本示例

以下示例介绍了函数PathRecValidBwd。

另请参阅第*1173*页的更多示例

例 1

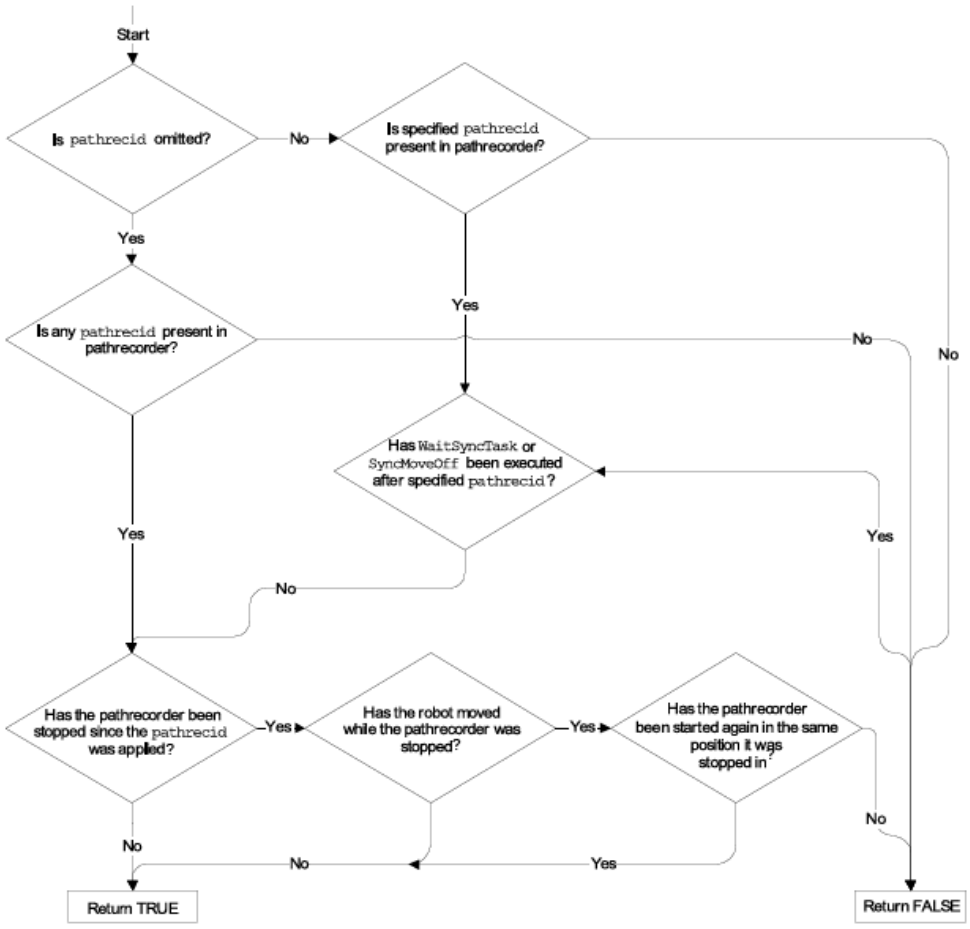
```
VAR bool bwd_path;
VAR pathrecid fixture_id;
bwd_path := PathRecValidBwd (\ID:=fixture_id);
```

如果可能备份至含标识符fixture_id的位置，则将变量bwd_path设置为TRUE。否则，将bwd_path设置为FALSE。

返回值

数据类型：bool

可根据以下流程图，确定函数的返回值：



xx0500002132

下一页继续

变元

```
PathRecValidBwd ([\ID])
```

[\ID]

Identifier

数据类型：pathrecid

指定记录初始位置名称的变量。数据类型pathrecid为非值类型，仅用作命名和记录位置的标识符。

程序执行

在命令路径记录器通过PathRecMoveBwd向后移动之前，有可能通过PathRecValidBwd来检查是否存在有效的记录路径。

更多示例

以下示例介绍了函数PathRecValidBwd。

例 1

```
PathRecStart id1;
MoveL p1, vmax, z50, tool1;
MoveL p2, vmax, z50, tool1;
bwd_path := PathRecValidBwd (\ID := id1);
```

启动路径记录器，并执行两个移动指令。PathRecValidBwd将返回TRUE，且可利用的备份路径将为：

从p2到p1到开始位置。

例 2

```
PathRecStart id1;
MoveL p1, vmax, z50, tool1;
MoveL p2, vmax, z50, tool1;
PathRecStop \Clear;
bwd_path:= PathRecValidBwd (\ID := id1);
```

启动路径记录器，并执行两个移动指令。随后，停止并清除路径记录器。PathRecValidBwd将返回FALSE。

例 3

```
PathRecStart id1;
MoveL p1, vmax, z50, tool1;
PathRecStart id2;
MoveL p2, vmax, z50, tool1;
bwd_path := PathRecValidBwd ();
```

启动路径记录器，并执行一个移动指令。随后，在一个运动指令后，启用一个附加路径标识符。PathRecValidBwd将返回TRUE，且备份路径将为：

从p2到p1。

例 4

```
PathRecStart id1;
MoveL p1, vmax, z50, tool1;
WaitSyncTask sync101, tasklist_r1o1;
MoveL p2, vmax, z50, tool1;
bwd_path1 := PathRecValidBwd ();
```

下一页继续

2.120 PathRecValidBwd - 是否记录了有效的后退路径

Path Recovery

续前页

```
bwd_path2 := PathRecValidBwd (\ID := id1);
```

执行上述程序将指定布尔变量bwd_path1为TRUE，因为存在通往WaitSyncTask声明的有效后退路径。将指定布尔变量bwd_path2为FALSE，因为不可能备份上述WaitSyncTask声明。

语法

```
PathRecValidBwd '('
  ['\' ID ':=' < variable (VAR) of pathrecid >] ')'
```

含数据类型bool的返回值的函数。

相关信息

信息，关于	请参阅
路径记录器标识符	第*1448*页的*pathrecid - 路径记录器标识符*
起动-停止路径记录器	第*440*页的*PathRecStart - 起动路径记录器* 第*442*页的*PathRecStop - 停止路径记录器*
使路径记录器向后	第*431*页的*PathRecMoveBwd - 将路径记录器向后移动*
检查是否存在一条有效的前进路径	第*1175*页的*PathRecValidFwd - 是否记录了有效的前进路径*
使路径记录器向前	第*437*页的*PathRecMoveFwd - 向前移动路径记录器*
一般动作	技术参考手册 - *RAPID*语言概览

2.121 PathRecValidFwd - 是否记录了有效的前进路径

手册用法

PathRecValidFwd用于检查路径记录器是否可用于向前移动。通过路径记录器向前移动的能力意味着必须命令路径记录器尽早向后移动。

基本示例

以下示例介绍了函数PathRecValidFwd。

另请参阅第*1176*页的更多示例

例 1

```
VAR bool fwd_path;
VAR pathrecid fixture_id;

fwd_path:= PathRecValidFwd (\ID:=fixture_id);
```

如果可能向前移动至含标识符fixture_id的位置，则将变量fwd_path设置为TRUE。否则，将fwd_path设置为FALSE。

返回值

数据类型：bool

不含指定\ID的PathRecValidFwd的返回值为：

如果出现以下情况，则TRUE：

- 通过使用PathRecMoveBwd，路径记录器已使机械臂向后移动。
- 机械臂尚未从PathRecMoveBwd所执行的路径移动。

如果出现以下情况，则FALSE：

- 未满足上述条件。

含指定\ID的PathRecValidFwd的返回值为：

如果出现以下情况，则TRUE：

- 通过使用PathRecMoveBwd，路径记录器已使机械臂向后移动。
- 机械臂尚未从PathRecMoveBwd所执行的路径移动。
- 向后移动期间，通过指定的\ID。

如果出现以下情况，则FALSE：

- 未满足上述条件。

变元

```
PathRecValidFwd ([\ID])
```

[\ID]

Identifier

数据类型：pathrecid

指定记录初始位置名称的变量。数据类型pathrecid为非值类型，仅用作命名和记录位置的标识符。

下一页继续

2.121 PathRecValidFwd - 是否记录了有效的前进路径

Path Recovery

续前页

程序执行

在通过使用PathRecMoveBwd，已经命令路径记录器向后移动后，有可能检查是否存在用以使机械臂向前移动的有效记录路径。如果省略标识符\ID，则PathRevValidFwd返回是否可能向前移动至向后移动开始的位置。

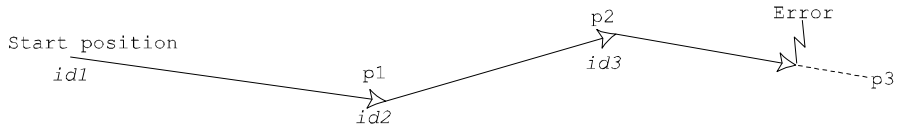
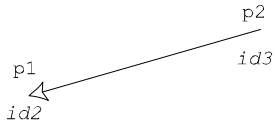
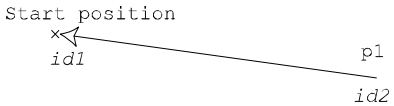
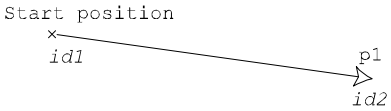
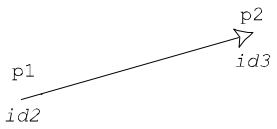
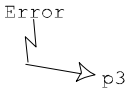
更多示例

以下示例介绍了函数PathRecValidFwd。

例 1

```
VAR pathrecid id1;
VAR pathrecid id2;
VAR pathrecid id3;

PathRecStart id1;
MoveL p1, vmax, z50, tool1;
PathRecStart id2;
MoveL p2, vmax, z50, tool1;
PathRecStart id3;
!See figures 1 and 8 in tbe following table.
MoveL p3, vmax, z50, tool1;
ERROR
  StorePath;
  IF PathRecValidBwd(\ID:=id3) THEN
    !See figure 2 in the following table.
    PathRecMoveBwd \ID:=id3;
    ! Do some other operation
  ENDIF
  IF PathRecValidBwd(\ID:=id2) THEN
    !See figure 3 in the following table.
    PathRecMoveBwd \ID:=id2;
    ! Do some other operation
  ENDIF
  !See figure 4 in the following table.
  PathRecMoveBwd;
  ! Do final service action
  IF PathRecValidFwd(\ID:=id2) THEN
    !See figure 5 in the following table.
    PathRecMoveFwd \ID:=id2;
    ! Do some other operation
  ENDIF
  IF PathRecValidFwd(\ID:=id3) THEN
    !See figure 6 in the following table.
    PathRecMoveFwd \ID:=id3;
    ! Do some other operation
  ENDIF
  !See figure 7 in the following table.
  PathRecMoveFwd;
  RestoPath;
  StartMove;
  RETRY;
```

下一页继续

1	 xx0500002121
2	 xx0500002124
3	 xx0500002126
4	 xx0500002127
5	 xx0500002128
6	 xx0500002129
7	 xx0500002130
8	 xx0500002131

上述例子将启动路径记录器，并在沿执行路径的三处不同位置增加标识符。上述图片参考示例代码，并描述在朝点p3执行期间出现错误时，机械臂将如何移动。分别使用PathRecValidBwd和PathRecValidFwd，因为不可能提前确定程序中可能出现错误的位置。

下一页继续

2.121 PathRecValidFwd - 是否记录了有效的前进路径

Path Recovery

续前页

语法

```
PathRecValidFwd '('
  ['\' ID ':=' < variable (VAR) of pathrecid >] ')'
```

含数据类型bool的返回值的函数。

相关信息

信息，关于	请参阅
路径记录器标识符	第*1448*页的*pathrecid - 路径记录器标识符*
起动-停止路径记录器	第*440*页的*PathRecStart - 起动路径记录器* 第*442*页的*PathRecStop - 停止路径记录器*
检查是否存在有效的后退路径	第*1172*页的*PathRecValidBwd - 是否记录了有效的后退路径*
使路径记录器向后	第*431*页的*PathRecMoveBwd - 将路径记录器向后移动*
使路径记录器向前	第*437*页的*PathRecMoveFwd - 向前移动路径记录器*
一般动作	技术参考手册 - *RAPID*语言概览

2.122 PFRestart - 在电源故障后，检查中断的路径

手册用法

PFRestart *(Power Failure Restart)* 用于检查路径是否已在电源故障时中断。如果是这样的话，可能有必要采取一些特殊行动。函数检查当前等级、基础等级或中断等级的路径。

基本示例

以下示例介绍了函数PFRestart。

例 1

```
IF PFRestart() = TRUE THEN
```

如果存在一条当前等级的中断路径，则进行检查。如果是这样的话，函数将返回TRUE。

返回值

数据类型： bool

如果存在一条指定路径等级的中断路径，则TRUE，否则，FALSE。

变元

```
PFRestart([\Base] | [\Irpt])
```

[\Base]

Base Level

数据类型：switch

如果存在一条基础等级的中断路径，则返回TRUE。

[\Irpt]

Interrupt Level

数据类型：switch

如果存在一条StorePath等级的中断路径，则返回TRUE。

如果未给定参数，则存在当前等级的中断路径时，函数将返回TRUE。

语法

```
PFRestart '('
  ['\' Base] | ['\' Irpt] ')'
```

含数据类型bool的返回值的函数。

相关信息

信息，关于	请参阅
Advanced RAPID	产品规格 - 控制器软件*IRC5*

2.123 PoseInv - 转化姿态数据

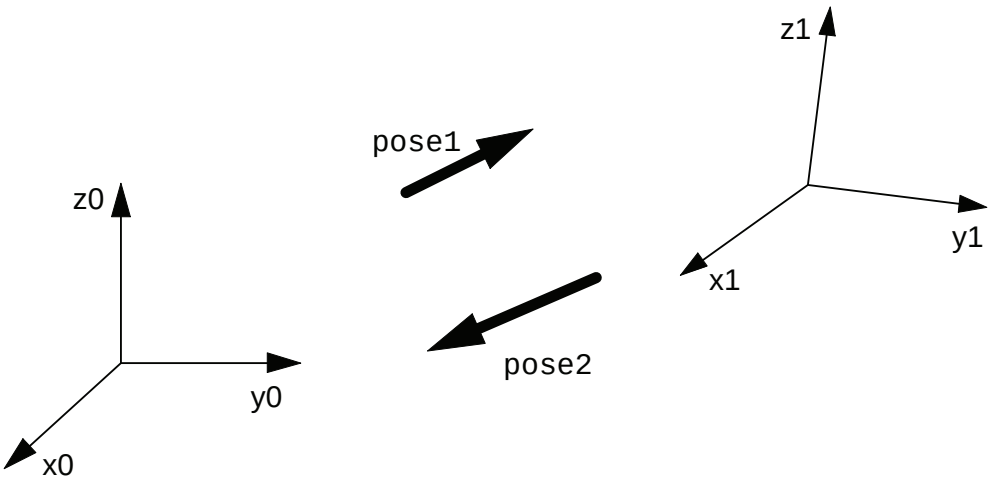
手册用法

PoseInv（*Pose Invert*）计算一个pose的逆向转换。

基本示例

以下示例介绍了函数PoseInv。

例 1



xx0500002443

pose1代表与坐标系0相关的坐标系1。通过逆向转换而获得得出同坐标系1相关的坐标系0的转换，将储存在pose2中。

```
VAR pose pose1;
VAR pose pose2;
...
pose2 := PoseInv(pose1);
```

返回值

数据类型：pose

逆向姿态的值。

变元

```
PoseInv (Pose)
```

Pose

数据类型：pose

用于转换的pose。

语法

```
PoseInv'('
  [Pose ':='] <expression (IN) of pose>
  ')'
```

含数据类型pose的返回值的函数。

下一页继续

相关信息

信息，关于	请参阅
数学指令和函数	技术参考手册 - *RAPID*语言概览

2.124 PoseMult - 倍增姿态数据

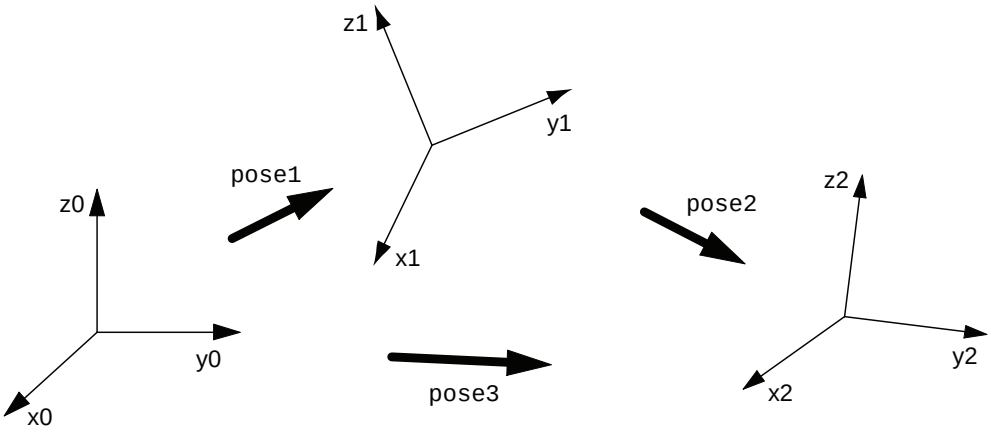
手册用法

PoseMult（*Pose Multiply*）用于计算两次姿态转换的产物。典型用途是计算一个新的姿态，以作为影响最初姿态的位移结果。

基本示例

以下示例介绍了函数PoseMult。

例 1



xx0500002444

pose1代表与坐标系0相关的坐标系1。pose2代表与坐标系1相关的坐标系2。通过两次转换的产物，获得同坐标系0相关的坐标系2，即得出pose3的转换。

```
VAR pose pose1;
VAR pose pose2;
VAR pose pose3;
...
pose3 := PoseMult(pose1, pose2);
```

返回值

数据类型：pose

两个姿态的产物的值。

变元

```
PoseMult (Pose1 Pose2)
```

Pose1

数据类型：pose

第一个姿态。

Pose2

数据类型：pose

第二个姿态。

下一页继续

语法

```
PoseMult '('
  [Pose1 ':='] <expression (IN) of pose> ','
  [Pose2 ':='] <expression (IN) of pose> ')'
```

含数据类型pose的返回值的函数。

相关信息

信息，关于	请参阅
数学指令和函数	技术参考手册 - *RAPID*语言概览

2.125 PoseVect - 将一次转换应用于一个矢量

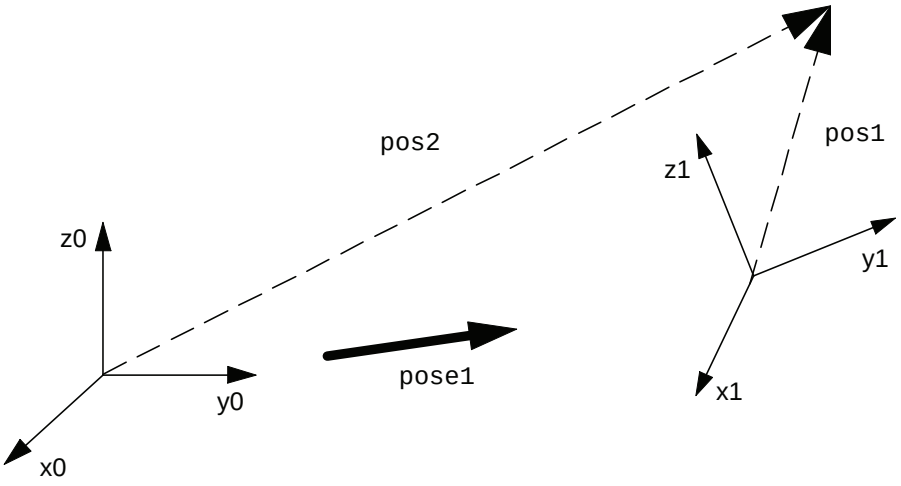
手册用法

PoseVect（姿态矢量）用于计算一个姿态和一个矢量的产物。典型用途是计算一个矢量，以作为一个原始矢量上位移的影响。

基本示例

以下示例介绍了函数PoseVect。

例 1



xx0500002445

pose1代表同坐标系0相关的坐标系1。

pos1为同坐标系1相关的一个矢量。通过产物，获得同坐标系0相关的矢量。

```
VAR pose pose1;
VAR pos pos1;
VAR pos pos2;
...
...
pos2:= PoseVect(pose1, pos1);
```

返回值

数据类型：pos

姿态和原始位置的产物的值。

变元

```
PoseVect (Pose Pos)
```

Pose

数据类型：pose

有待应用的转换。

Pos

数据类型：pos

有待转换的位置。

下一页继续

语法

```
PoseVect '('
  [Pose ':='] <expression (IN) of pose> ','
  [Pos ':='] <expression (IN) of pos> ')'
```

含数据类型pos的返回值的函数。

相关信息

信息，关于	请参阅
数学指令和函数	技术参考手册 - *RAPID*语言概览

2.126 Pow - 计算一个值的幂

手册用法

Pow（*Power*）用于计算任意基座中的指数值。

基本示例

以下示例介绍了函数Pow。

例 1

```
VAR num x;
VAR num y
VAR num reg1;
...
reg1:= Pow(x, y);
```

将reg1指定为值x^y。

返回值

数据类型：num

将Base的值提升到指数的幂，即基础^{指数}。

变元

```
Pow (Base Exponent)
```

Base

数据类型：num

基础参数值。

Exponent

数据类型：num

指数参数值。

限制

如果出现以下情况，则函数x^y的执行将出错：

- x < 0和y为非整数；
- x = 0且y ≤ 0。

语法

```
Pow '('
  [Base ':='] <expression (IN) of num> ','
  [Exponent ':='] <expression (IN) of num> ')'
```

返回值的数据类型是 num 的函数。

相关信息

信息，关于	请参阅
数学指令和函数	技术参考手册 - *RAPID*语言概览

2.127 PowDnum - 计算一个值的幂

手册用法

PowDnum（幂双数值）用于计算任意基座中的指数值。

基本示例

以下示例介绍了函数PowDnum。

例 1

```
VAR dnum x;
VAR num y
VAR dnum value;
...
value:= PowDnum(x, y);
```

将value指定为值x^y。

返回值

数据类型：dnum

将Base的值提升到指数的幂，即基础^{指数}。

变元

```
PowDnum (Base Exponent)
```

Base

数据类型：dnum

基础参数值。

Exponent

数据类型：num

指数参数值。

限制

如果出现以下情况，则函数x^y的执行将出错：

- x < 0和y为非整数；
- x = 0且y ≤ 0。

语法

```
PowDnum '('
  [Base ':='] <expression (IN) of dnum> ','
  [Exponent ':='] <expression (IN) of num> ')'
```

含数据类型dnum的返回值的函数。

相关信息

信息，关于	请参阅
数学指令和函数	技术参考手册 - *RAPID*语言概览

2.128 PPMovedInManMode - 测试是否以手动模式移动程序指针。

手册用法

当控制器处于手动模式，即操作员钥匙位于手动减速度或手动全速时，如果用户已经移动了程序指针，则PPMovedInManMode返回TRUE。当钥匙由自动切换至手动，或者当使用指令ResetPPMoved时，重置程序指针移动状态。

基本示例

以下示例介绍了函数PPMovedInManMode。

例 1

```
IF PPMovedInManMode() THEN
  WarnUserOfPPMovement;
  DoJob;
ELSE
  DoJob;
ENDIF
```

返回值

数据类型：bool

当采用手动模式时，如果已由用户移动程序指针，则TRUE。

程序执行

测试是否已采用手动模式来移动有关当前程序任务的程序指针。

语法

```
PPMovedInManMode '(' ')'
```

含数据类型bool的返回值的函数。

相关信息

信息，关于	请参阅
测试是否已移动程序指针	第*1134*页的*IsStopStateEvent* - 测试是否移动程序指针
重置以手动模式移动的程序指针的状态	第*510*页的*ResetPPMoved* - 重置以手动模式移动的程序指针的状态

2.129 Present - 测试是否使用一个可选参数

手册用法

Present用于测试在调用一个程序时，是否已经使用一个可选参数。

如果在调用程序时未作任何指定，则可能不会使用可选参数。该函数可用于测试是否已指定一个参数，从而防止出现错误。

基本示例

以下示例介绍了函数Present。

另请参阅第*1189*页的更多示例

例 1

```
PROC feeder (\switch on | switch off)
  IF Present (on) Set do1;
  IF Present (off) Reset do1;
ENDPROC
```

当调用程序时，根据使用的参数，设置或重置用于控制送料机的输出do1。

返回值

数据类型：bool

TRUE = 调用程序时，已经确定参数值或开关。

FALSE = 尚未确定参数值或开关。

变元

```
Present (OptPar)
```

OptPar

Optional Parameter

数据类型：任意类型

待测试可选参数的名称。

更多示例

以下示例介绍了函数Present。

例 1

```
PROC glue (\switch on, num glueflow, robtarget topoint, speeddata
     speed, zonedata zone, PERS tooldata tool, \PERS wobjdata wobj)
  IF Present (on) PulseDO glue_on;
    SetAO gluesignal, glueflow;
  IF Present (wobj) THEN
    MoveL topoint, speed, zone, tool \WObj:=wobj;
  ELSE
    MoveL topoint, speed, zone, tool;
  ENDIF
ENDPROC
```

制定粘贴程序。如果在调用程序时指定参数\on，则在信号glue_on上产生脉冲。随后，机械臂设置一个模拟信号输出gluesignal，其控制涂胶枪，并移动至结束位置。由于wobj参数可选，因此，根据是否使用该参数，使用不同的MoveL指令。

下一页继续

2.129 Present - 测试是否使用一个可选参数

RobotWare - OS

续前页

语法

```
Present '('
  [OptPar ':='] <reference (REF) of any type> ')'
```

在这种情况下，REF参数需要可选参数名称。

含数据类型bool的返回值的函数。

相关信息

信息，关于	请参阅
程序参数	技术参考手册 - *RAPID*语言概览

2.130 ProgMemFree - 获得闲置程序内存的容量

手册用法

ProgMemFree（*Program Memory Free*）用于获得闲置程序内存的容量。

基本示例

以下示例介绍了函数ProgMemFree。

例 1

```
FUNC num module_size(string file_path)
  VAR num pgmfree_before;
  VAR num pgmfree_after;

  pgmfree_before:=ProgMemFree();
  Load \Dynamic, file_path;
  pgmfree_after:=ProgMemFree();
  Unload file_path;
  RETURN (pgmfree_before-pgmfree_after);
ENDFUNC
```

ProgMemFree用于一个函数中，其返回一个模块在程序内存中分配的内存量。

返回值

数据类型：num

闲置程序内存的容量，以字节计。

语法

```
ProgMemFree '(' ')'
```

返回值的数据类型是 num 的函数。

相关信息

信息，关于	请参阅
加载普通程序模块	第*312*页的*Load* - 执行期间，加载普通程序模块
卸载普通程序模块	第*843*页的*UnLoad* - 执行期间，卸载普通程序模块

2.131 PrxGetMaxRecordpos - 获得最大传感器位置

手册用法

PrxGetMaxRecordpos用于返回有效记录中的最大位置，以mm计。

最大传感器位置可用于缩放或限制SyncToSensor指令中的max_sync参数。

基本示例

```
maxpos:=PrxGetMaxRecordpos Ssync1;
```

获得有关机械单元Ssync1有效剖面的最大位置。

返回值

数据类型：num

有关记录的传感器移动剖面的最大位置（以mm计）。

变元

```
PrxGetMaxRecordpos MechUnit
```

MechUnit

数据类型：mechunit

同步机械臂运动的移动机械单元工件。

程序执行

必须完成记录，且记录必须有效。

语法

```
PrxGetMaxRecordpos '('
  [ MechUnit ':=' ] < expression (IN) of mechunit> ')'
```

返回值的数据类型是 num 的函数。

相关信息

信息，关于	请参阅
Machine Synchronization	应用手册 - 控制器软件*IRC5*

2.132 RawBytesLen - 获取原始数据字节数据的长度

手册用法

RawBytesLen用于获得原始数据字节变量中有效字节的当前长度。

基本示例

以下示例介绍了函数RawBytesLen。

例 1

```
VAR rawbytes from_raw_data;
VAR rawbytes to_raw_data;
VAR num integer := 8
VAR num float := 13.4;
ClearRawBytes from_raw_data;
PackRawBytes integer, from_raw_data, 1 \IntX := INT;
PackRawBytes float, from_raw_data, (RawBytesLen(from_raw_data)+1)
     \Float4;
CopyRawBytes from_raw_data, 1, to_raw_data, 3;
```

在该例子中，首先清除原始数据字节型变量 from_raw_data，即将所有字节设置为0（与声明的默认值相同）。随后，将整数值置于前2个字节中，并通过函数RawBytesLen的帮助，将float的值置于下4个字节（始于指数3）。

在向from_raw_data填充数据之后，将内容（6个字节）复制到to_raw_data，开始于位置3。

返回值

数据类型：num

范围为0 ... 1024的rawbytes;型变量中有效字节的当前长度。

总的来说，由系统将原始数据字节变量中有效字节的当前长度更新为原始数据字节结构中最后写入的字节。

有关细节，请参见数据类型rawbytes，指令ClearRawBytes、CopyRawBytes、PackDNHeader、PackRawBytes和ReadRawBytes。

变元

```
RawBytesLen (RawData)
```

RawData

数据类型：rawbytes

RawData为数据容器，应当返回其有效字节的当前长度。

程序执行

程序执行期间，返回有效字节的当前长度。

语法

```
RawBytesLen '('
[RawData ':='] < variable (VAR) of rawbytes> ')'
```

返回值的数据类型是 num 的函数。

下一页继续

2.132 RawBytesLen - 获取原始数据字节数据的长度

RobotWare - OS

续前页

相关信息

信息，关于	请参阅
rawbytes数据	第*1455*页的原始数据字节 - 原始数据
清除rawbytes数据的内容	第*110*页的*ClearRawBytes* - 清除原始数据字节数据的内容
复制rawbytes数据的内容	第*128*页的*CopyRawBytes* - 复制原始数据字节数据的内容
将DeviceNet标题装入rawbytes数据	第*421*页的*PackDNHeader* - 将*DeviceNet*标题装入原始数据字节数据中。
将数据装入rawbytes数据	第*424*页的*PackRawBytes* - 将数据装入原始数据字节数据
读取rawbytes数据	第*499*页的*ReadRawBytes* - 读取原始数据字节数据
打开来自rawbytes数据的数据	第*845*页的*UnpackRawBytes* - 打开来自原始数据字节数据的数据
写入rawbytes数据	第*923*页的*WriteRawBytes* - 写入原始数据字节数据
文件和串行通道处理	应用手册 - 控制器软件*IRC5*

2.133 ReadBin - 从一个文件或串行通道读取一个字节

手册用法

ReadBin（*Read Binary*）用于从一个文件或串行通道读取一个字节（8位）。

该函数对基于二进制和字符的文件或串行通道起作用。

基本示例

以下示例介绍了函数ReadBin。

另请参阅第*1196*页的更多示例

例 1

```
VAR num character;
VAR iodev inchannel;
...
Open "com1:", inchannel\Bin;
character := ReadBin(inchannel);
```

从二进制串行通道inchannel读取一个字节。

返回值

数据类型：num

从一个指定文件或串行通道读取一个字节（8位）。将该字节转换为相关的正数值，并作为一个num数据类型返回。如果文件为空（文件末尾），则返回EOF_BIN（数字 -1）。

变元

```
ReadBin (IODevice [\Time])
```

IODevice

数据类型：iodev

有待读取的文件或串行通道的名称（引用）。

[\Time]

数据类型：num

读取操作（超时）的最长时间以秒计。如果未规定该参数，则将最长时间设置为60秒。为了永远等待，使用预定义常量WAIT_MAX。

如果在完成读取操作之前耗尽时间，则将通过错误代码ERR_DEV_MAXTIME来调用错误处理器。如果不存在错误处理器，则将停止执行。

程序停止期间，亦使用超时功能，并由程序开始时的RAPID程序进行通知。

程序执行

程序执行进入等待，直至可以从文件或串行通道读取一个字节（8位）。

上电失败重启时，将关闭系统中所有打开的文件或串行通道，并将重置iodev型变量中的I/O描述符。

下一页继续

2.133 ReadBin - 从一个文件或串行通道读取一个字节

RobotWare - OS

续前页

更多示例

以下示例介绍了函数ReadBin。

例 1

```
VAR num bindata;
VAR iodev file;

Open "HOME:/myfile.bin", file \Read \Bin;
bindata := ReadBin(file);
WHILE bindata <> EOF_BIN DO
  TPWrite ByteToStr(bindata\Char);
  bindata := ReadBin(file);
ENDWHILE
```

由始至终，读取二进制文件myfile.bin的内容，并在FlexPendant示教器上显示所接收到的会转换为字符的二进制数据（各线路上的一个字符）。

限制

本函数仅可用于已通过读取而打开的文件和串行通道（有关基于字符的文件的\Read，有关二进制文件的\Bin或\Append \Bin ）。

错误处理

如果在读取期间出现错误，则将系统变量ERRNO设置为ERR_FILEACC。

如果在完成读取操作之前超时，则将系统变量ERRNO设置为ERR_DEV_MAXTIME。

随后，可通过错误处理器来处理此类错误。

预定义数据

常量EOF_BIN可用于在文件末尾停止读取。

```
CONST num EOF_BIN := -1;
```

语法

```
ReadBin '('
  [IODevice ':='] <variable (VAR) of iodev>
  ['\' Time ':=' <expression (IN) of num>] ')'
```

含num型返回值的函数。

相关信息

信息，关于	请参阅
“打开”等文件或串行通道	技术参考手册 - *RAPID*语言概览
将一个字节转换为一段字符串数据	第*1002*页的*ByteToStr - 将字节转换为字符串数据*
文件和串行通道处理	应用手册 - 控制器软件*IRC5*

2.134 ReadDir - 读取路径中的下个条目

手册用法

ReadDir的作用，是在已用指令OpenDir打开的一个目录中检索下一份文件或子目录的名称。

只要函数返回TRUE，则可以检索更多的文件或子路径。

基本示例

以下示例介绍了函数ReadDir。

另请参阅第*1197*页的更多示例

例 1

```
PROC lsdir(string dirname)
  VAR dir directory;
  VAR string filename;
  OpenDir directory, dirname;
  WHILE ReadDir(directory, filename) DO
    TPWrite filename;
  ENDWHILE
  CloseDir directory;
ENDPROC
```

此例子打印出指定路径下所有文件或子路径的名称。

返回值

数据类型：bool

如果函数已经检索了一个名称，则函数将返回TRUE，否则，FALSE。

变元

```
ReadDir (Dev FileName)
```

Dev

数据类型：dir

通过指令OpenDir，取得有关于路径的变量。

FileName

数据类型：string

检索到的文件或子路径名称。

程序执行

该函数返回一个布尔变量，其规定一个名称的检索成功与否。

更多示例

有关于函数ReadDir的更多例子阐述如下

例 1

该例子通常横穿路径结构函数。

```
PROC searchdir(string dirname, string actionproc)
  VAR dir directory;
```

下一页继续

2.134 ReadDir - 读取路径中的下个条目

RobotWare - OS

续前页

```
  VAR string filename;
  IF IsFile(dirname \Directory) THEN
    OpenDir directory, dirname;
    WHILE ReadDir(directory, filename) DO
        ! .. and . is the parent and resp. this directory
        IF filename <> ".." AND filename <> "." THEN
          searchdir dirname+"/"+filename, actionproc;
        ENDIF
    ENDWHILE
    CloseDir directory;
  ELSE
    %actionproc% dirname;
  ENDIF
ERROR
  RAISE;
ENDPROC

PROC listfile(string filename)
  TPWrite filename;
ENDPROC

PROC main()
  ! Execute the listfile routine for all files found under the
  ! tree in HOME:
  searchdir "HOME:","listfile";
ENDPROC
```

该程序横穿“HOME:”下的路径结构，并针对各个文件而调用listfile无返回值程序。searchdir为通用部分，其不知晓有关搜索的开始，或者不知晓应当针对各文件调用哪一个程序。其使用IsFile，以检查是否已发现子路径或文件，且其使用后期绑定机制，以调用有关所发现文件的actionproc中指定的无返回值程序。actionproc程序应当为一个无返回值程序，其中一个参数为string型。

错误处理

如果未打开路径（参见OpenDir），则将系统变量ERRNO设置为ERR_FILEACC。随后，可用错误处理器对该错误进行处理。

语法

```
ReadDir '('
  [ Dev ':=' ] < variable (VAR) of dir> ','
  [ FileName ':=' ] < var or pers (INOUT) of string> ')'
```

含数据类型bool的返回值的函数。

相关信息

信息，关于	请参阅
目录	第*1384*页的*dir - 路径结构*
建立路径	第*322*页的*MakeDir - 创建新路径*
打开路径	第*419*页的*OpenDir - 打开路径*

下一页继续

信息，关于	请参阅
关闭路径	第*117*页的*CloseDir - 关闭路径*
删除路径	第*503*页的*RemoveDir - 删除路径*
删除文件	第*504*页的*RemoveFile - 删除文件*
重命名文件	第*507*页的*RenameFile - 重命名文件*
文件和串行通道处理	应用手册 - 控制器软件*IRC5*

2.135 ReadMotor - 读取当前电机的角度

手册用法

ReadMotor用于读取机械臂和外轴的不同电机的当前角度。该函数主要用于机械臂的校准无返回值程序。

基本示例

以下示例介绍了函数ReadMotor。

另请参阅第*1200*页的更多示例

例 1

```
VAR num motor_angle2;
motor_angle2 := ReadMotor(2);
```

将机械臂第二个轴的当前电机角度储存在motor_angle2中。

返回值

数据类型：num

机械臂或外轴的指定轴的当前电机角度，以弧度计。

变元

```
ReadMotor [\MecUnit ] Axis
```

MecUnit

Mechanical Unit

数据类型：mecunit

包含待读取轴的机械单元的名称。如果省略该参数，则读取有关相连机械臂的轴。

Axis

数据类型：num

待读取轴的编号（1 - 6）。

程序执行

返回的电机角度代表不含任何校准偏移量的电机的当前位置，以弧度计。该值与机械臂的固定位置无关，仅与旋转变压器内部的零位置相关，即通常最接近校准位置的旋转变压器零位置（旋转变压器零位置与校准位置之间的差异为校准偏移量）。该值代表各轴的完全移动，尽管其可能包含若干转。

限制

其仅可能读取有关由当前程序任务控制的机械单元的当前电机角度。对于非运动任务，读取有关由相关运动任务控制的机械单元的角度是可能的。

更多示例

以下示例介绍了函数ReadMotor。

例 1

```
VAR num motor_angle;
motor_angle := ReadMotor(\MecUnit:=STN1, 1);
```

将STN1第一个轴的当前电机角度储存在motor_angle。

下一页继续

错误处理

系统会生成下列可恢复错误，并在错误处理器中处理这些错误。系统变量ERRNO将被设置成：

ERR_AXIS_PAR	指令中的参数轴错误。

语法

```
ReadMotor '('
  ['\' MecUnit ':=' < variable (VAR) of mecunit> ',']
  [Axis ':=' ] < expression (IN) of num> ')'
```

返回值的数据类型是 num 的函数。

相关信息

信息，关于	请参阅
读取当前接头的角度	第*1026*页的*CJointT* - 读取当前接头的角度

2.136 ReadNum - 从一个文件或串行通道读取一个数字

手册用法

ReadNum（*Read Numeric*）用于从一个基于字符的文件或串行通道读取一个数字。

基本示例

以下示例介绍了函数ReadNum。

另请参阅第*1203*页的更多示例

例 1

```
VAR iodev infile;
...
Open "HOME:/file.doc", infile\Read;
reg1 := ReadNum(infile);
```

向reg1分配一个从文件file.doc读取的数字。

返回值

数据类型：num

从一个指定文件或串行通道读取的数值。如果文件为空（文件末尾），则返回一个大于EOF_NUM (9.998E36)的数字。

变元

```
ReadNum (IODevice [\Delim] [\Time])
```

IODevice

数据类型：iodev

有待读取的文件或串行通道的名称（引用）。

[\Delim]

Delimiters

数据类型：string

当解析文件或串行通道中的一行时，使用包含分隔符的一个字符串。默认（不含\Delim）逐行读取文件，并通过解析，将换行字符（\0A）视为唯一的分隔符。当使用\Delim参数时，将考虑指定字符串参数中的任意字符，以确定该行的重要部分。

当使用参数\Delim时，控制系统始终将回车（\0D）字符和换行（\0A）字符增加到用户所指定的分隔符。

为指定非字母数字字符，使用\xx，其中，xx为字符的ASCII代码十六进制表示（例如：通过\09指定TAB）。

[\Time]

数据类型：num

读取操作（超时）的最长时间以秒计。如果未规定该参数，则将最长时间设置为60秒。为了永远等待，使用预定义常量WAIT_MAX。

如果在完成读取操作之前耗尽时间，则将通过错误代码ERR_DEV_MAXTIME来调用错误处理器。如果不存在错误处理器，则将停止执行。

程序停止期间，亦使用超时功能，并由程序开始时的RAPID程序进行通知。

下一页继续

程序执行

始于当前文件位置，函数读取和删除所有标题分隔符。不含参数\Delim的标题分隔符为一个换行字符。含参数\Delim的标题换行符为\Delim参数加上回车和换行字符中指定的任意字符。随后，读取任意多达且包括下一分隔符字符（将予以删除）的所有字符，但是不超过80个字符。如果重要部分超过80个字符，将在下一次读取中读取其余字符。

随后，将读取的字符串转换成一个数值；例如，将"234.4"转换成数值234.4。

上电失败重启时，将关闭系统中所有打开的文件或串行通道，并将重置iodev型变量中的I/O描述符。

更多示例

以下示例介绍了函数ReadNum。

例 1

```
reg1 := ReadNum(infile\Delim:="\09");
IF reg1 > EOF_NUM THEN
  TPWrite "The file is empty";
  ...
```

从由TAB（"\09"）或SPACE（" "）字符将数字分隔开的行中，读取一个数字。在使用从文件读取的数字之前，实施检查，以确保文件非空。

注意

检查文件是否为空时，使用<或>（小于或大于）。不使用 = （等于）。

限制

函数仅可用于基于字符的文件，已打开此类文件以供读取。

错误处理

如果在读取期间出现访问错误，则将系统变量ERRNO设置为ERR_FILEACC。

如果试图读取非数值数据，则将系统变量ERRNO设置为ERR_RCVDATA。

如果在完成读取操作之前超时，则将系统变量ERRNO设置为ERR_DEV_MAXTIME。

随后，可通过错误处理器来处理此类错误。

预定义数据

常量EOF_NUM可用于在文件末尾停止读取。

```
CONST num EOF_NUM := 9.998E36;
```

语法

```
ReadNum '('
  [IODevice ':='] <variable (VAR) of iodev>
  ['\' Delim ':=' <expression (IN) of string>]
  ['\' Time ':=' <expression (IN) of num>] ')'
```

含num型返回值的函数。

下一页继续

2.136 ReadNum - 从一个文件或串行通道读取一个数字

RobotWare - OS

续前页

相关信息

信息，关于	请参阅
“打开”等文件或串行通道	技术参考手册 - *RAPID*语言概览
文件和串行通道处理	应用手册 - 控制器软件*IRC5*

2.137 ReadStr - 从一个文件或串行通道读取一个字符串

手册用法

ReadStr（*Read String*）用于从一个基于字符的文件或串行通道读取一个字符串。

基本示例

以下示例介绍了函数ReadStr。

另请参阅第*1206*页的更多示例

例 1

```
VAR string text;
VAR iodev infile;
...
Open "HOME:/file.doc", infile\Read;
text := ReadStr(infile);
```

向text分配一个从文件file.doc读取的字符串。

返回值

数据类型：string

从指定文件或串行通道读取的字符串。如果文件为空（文件末尾），则返回字符串"EOF"。

变元

```
ReadStr (IODevice [\Delim] [\RemoveCR] [\DiscardHeaders] [\Time])
```

IODevice

数据类型：iodev

有待读取的文件或串行通道的名称（引用）。

[\Delim]

Delimiters

数据类型：string

当解析文件或串行通道中的一行时，使用包含分隔符的一个字符串。默认逐行读取文件，并通过解析，将换行字符（\0A）视为唯一的分隔符。当使用\Delim参数时，将考虑指定字符串参数加上默认换行字符中的任意字符，以确定该行的重要部分。

为指定非字母数字字符，使用\xx，其中，xx 为字符的ASCII代码十六进制表示（例如：通过\09指定TAB）。

[\RemoveCR]

数据类型：switch

读取PC文件时，用于移除后续回车字符的开关。在PC文件中，通过回车和换行（CRLF）来指定新行。当读取此类文件中的一行时，默认将回车字符读入返回字符串。当使用该参数时，将从文件读取回车字符，但是并不将其囊括到返回字符串中。

[\DiscardHeaders]

数据类型：switch

下一页继续

2.137 ReadStr - 从一个文件或串行通道读取一个字符串
RobotWare - OS
续前页

该参数规定是否略过标题分隔符（通过\Delim 加上默认换行来规定），或不早于将数据转移至返回字符串。如果当前文件位置的第一个字符为一个分隔符，则默认进行读取，但不转移到返回字符串，停止行解析，且返回将作为一个空字符串。如果使用该参数，则将从文件读取该行中包含的所有分隔符，但是予以删除，且不进行任何返回，直至返回字符串将包含始于该行第一个非分隔符字符的数据。

[\Time]

数据类型：num

读取操作（超时）的最长时间以秒计。如果未规定该参数，则将最长时间设置为60秒。为了永远等待，使用预定义常量WAIT_MAX。

如果在完成读取操作之前耗尽时间，则将通过错误代码ERR_DEV_MAXTIME来调用错误处理器。如果不存在错误处理器，则将停止执行。

程序停止期间，亦使用超时功能，并在程序开始时的RAPID程序中进行通知。

程序执行

始于当前文件位置，如果使用\DiscardHeaders参数，则函数读取和删除所有标题分隔符（\Delim参数中指定的）。在所有情况下，其随后读取直至下一分隔符字符的所有字符，但是不超过80个字符。如果重要部分超过80个字符，则将在下一次读取时来读取其余字符。从文件读取导致解析停止的分隔符，但是不转移到返回字符串。如果字符串中的最后一个字符为一个回车字符，且使用\RemoveCR参数，则将从字符串移除该字符。

上电失败重启时，将关闭系统中所有打开的文件或串行通道，并将重置iodev型变量中的I/O描述符。

更多示例

以下示例介绍了函数ReadStr。

例 1

```
text := ReadStr(infile);
IF text = EOF THEN
  TPWrite "The file is empty";
  ...
```

在使用从文件读取的字符串之前，实施检查，以确保该文件非空。

例 2

考虑一个包含以下内容的文件：

```
<LF><SPACE><TAB>Hello<SPACE><SPACE>World<CR><LF>
text := ReadStr(infile);
```

text将成为一个空字符串：文件中的第一个字符为默认<LF>分隔符。

```
text := ReadStr(infile\DiscardHeaders);
```

文本将包含<SPACE><TAB>Hello<SPACE><SPACE>World<CR>：删除文件中的第一个字符，即默认<LF>分隔符。

```
text := ReadStr(infile\RemoveCR\DiscardHeaders);
```

text将包含<SPACE><TAB>Hello<SPACE><SPACE>World:删除文件中的第一个字符，即默认<LF>分隔符；移除最终的回车字符

```
text := ReadStr(infile\Delim:=" \09"\RemoveCR\DiscardHeaders);
```

下一页继续

文本将包含"Hello"：删除文件中匹配默认<LF>分隔符或\Delim所确定的字符集（空格和制表符）。随后，将数据转移至从文件读取的第一个分隔符，但是不得转移到字符串中。相同声明的全新调用将返回"World"。

例 3

考虑一个包含以下内容的文件：

```
<CR><LF>Hello<CR><LF>
```

```
text := ReadStr(infile);
```

text将包含<CR> (\0d)字符：<CR>和<LF> 字符为从文件读取的字符，但是仅将<CR>转移到字符串。相同声明的全新调用将返回"Hello\0d"。

```
text := ReadStr(infile\RemoveCR);
```

text将包含一个空字符串：<CR>和<LF>字符为从文件读取的字符；转移<CR>，但是将其从字符串移除。相同声明的全新调用将返回"Hello"。

```
text := ReadStr(infile\Delim:="\0d");
```

text将包含一个空字符串：<CR>为从文件读取的字符，但是不会转移到返回字符串。相同指令的全新调用将再次返回一个空字符串：<LF>为从文件读取的字符，但是不会转移到返回字符串。

```
text := ReadStr(infile\Delim:="\0d"\DiscardHeaders);
```

text将包含"Hello"。相同指令的全新调用将返回"EOF"（文件末尾）。

限制

函数仅可用于已打开以便通过基于字符的模式来读取的文件或串行通道。

错误处理

如果在读取期间出现错误，则将系统变量ERRNO设置为ERR_FILEACC。

如果在完成读取操作之前超时，则将系统变量ERRNO设置为ERR_DEV_MAXTIME。

随后，可通过错误处理器来处理此类错误。

预定义数据

当尝试读取文件，或者在文件末尾停止读取时，常量EOF可用于检查文件是否为空。

```
CONST string EOF := "EOF";
```

语法

```
ReadStr '('
[IODevice ':='] <variable (VAR) of iodev>
['\' Delim ':='<expression (IN) of string>]
['\' RemoveCR]
['\' DiscardHeaders]
['\' Time ':=' <expression (IN) of num>] ')'
```

含string型返回值的函数。

相关信息

信息，关于	请参阅
“打开”等文件或串行通道	技术参考手册 - *RAPID*语言概览
文件和串行通道处理	应用手册 - 控制器软件*IRC5*

2.138 ReadStrBin - 从一条二进制串行通道或一份文件读取一段字符串

手册用法

ReadStrBin（*Read String Binary*）的作用是从一条二进制串行通道或一份文件中读取一段字符串。

基本示例

以下示例介绍了函数ReadStrBin。

例 1

```
VAR iodev channel;
VAR string text;
...
Open "com1:", channel \Bin;
text := ReadStrBin (channel, 10);
text := ReadStrBin(infile,20);
IF text = EOF THEN
```

向text分配从channel所参考的串行通道读取而来的10字符文本字符串。

在使用从文件读取的字符串之前，实施检查，以确保该文件非空。

返回值

数据类型：string

从指定串行通道或文件读取的文本字符串。如果文件为空（文件末尾），则返回字符串"EOF"。

变元

```
ReadStrBin (IODevice NoOfChars [\Time])
```

IODevice

数据类型：iodev

有待读取的二进制串行通道或文件的名称（参考）。

NoOfChars

Number of Characters

数据类型：num

有待从二进制串行通道或文件读取的字符数。

[\Time]

数据类型：num

读取操作（超时）的最长时间以秒计。如果未规定该参数，则将最长时间设置为60秒。为了永远等待，使用预定义常量WAIT_MAX。

如果在完成读取操作之前耗尽时间，则将通过错误代码ERR_DEV_MAXTIME来调用错误处理器。如果不存在错误处理器，则将停止执行。

程序停止期间，亦使用超时功能，并由程序开始时的RAPID程序进行通知。

程序执行

函数从二进制串行通道或文件读取规定数量的字符。

下一页继续

上电失败重启时，将关闭系统中所有打开的文件或串行通道，并将重置iodev型变量中的I/O描述符。

限制

函数仅可用于为采用二进制模式进行读取而打开的串行通道或文件。

错误处理

如果在读取期间出现错误，则将系统变量ERRNO设置为ERR_FILEACC。

如果在完成读取操作之前超时，则将系统变量ERRNO设置为ERR_DEV_MAXTIME。

随后，可通过错误处理器来处理此类错误。

预定义数据

当尝试读取文件，或者在文件末尾停止读取时，常量EOF可用于检查文件是否为空。

```
CONST string EOF := "EOF";
```

语法

```
ReadStrBin '('
  [IODevice ':='] <variable (VAR) of iodev> ','
  [NoOfChars ':='] <expression (IN) of num>
  ['\' Time ':=' <expression (IN) of num>] ')'
```

含string型返回值的函数。

相关信息

信息，关于	请参阅
“打开”等串行通道或文件	技术参考手册 - *RAPID*语言概览
写入二进制字符串	第*925*页的*WriteStrBin* - 将字符串写入一个二进制串行通道
文件和串行通道处理	应用手册 - 控制器软件*IRC5*

2.139 ReadVar - 从设备读取变量

手册用法

ReadVar用于读取与串行传感器接口连接的装置中的变量。

传感器接口与位于串行通道（使用RTP1传输层协议）上方的传感器进行通信。

此为关于传感器通道配置的实例。

COM_PHY_CHANNEL:

- Name “COM1:”
- Connector “COM1”
- Baudrate 19200

COM_TRP:

- Name “sen1:”
- Type “RTP1”
- PhyChannel “COM1”

基本示例

以下示例介绍了函数ReadVar。

例 1

```
CONST num XCoord := 8;
CONST num YCoord := 9;
CONST num ZCoord := 10;

VAR pos SensorPos;

! Connect to the sensor device "sen1:" (defined in sio.cfg)
SenDevice "sen1:";

! Read a cartesian position from the sensor.

SensorPos.x := ReadVar ("sen1:", XCoord);
SensorPos.y := ReadVar ("sen1:", YCoord);
SensorPos.z := ReadVar ("sen1:", ZCoord);
```

变元

```
ReadVar (device, VarNo, [ \TaskName ])
```

device

数据类型：string

在sio.cfg中配置了I/O设备名称，以供传感器使用。

VarNo

数据类型：num

参数VarNo用于选择有待读取的变量。

[\TaskName]

数据类型：string

下一页继续

参数TaskName使其可能访问其他RAPID任务中的设备。

错误处理

错误常量 (ERRNO值)	描述
SEN_NO_MEAS	测量失败
SEN_NOREADY	传感器不能处理命令
SEN_GENERRO	一般性传感器错误
SEN_BUSY	传感器繁忙
SEN_UNKNOWN	未知的传感器
SEN_EXALARM	外部传感器错误
SEN_CAALARM	内部传感器错误
SEN_TEMP	传感器温度错误
SEN_VALUE	非法通信值
SEN_CAMCHECK	传感器检查失败
SEN_TIMEOUT	通信错误

语法

```
ReadVar
  [ device ':=' ] < expression(IN) of string> ','
  [ VarNo ':=' ] < expression (IN) of num > ','
  [ '\' TaskName ':=' < expression (IN) of string > ] ';'
```

返回值的数据类型是 num 的函数。

相关信息

信息，关于	请参阅
与传感器设备相连	第574页的*SenDevice - 与传感器设备相连*
写入传感器变量	第927页的*WriteVar - 写入变量*
写入传感器数据块	第917页的*WriteBlock - 将数据块写入设备*
读取传感器数据块	第490页的*ReadBlock - 读取设备的数据块*
传感器通信配置	技术参考手册 - *RAPID*语言概览

2.140 RelTool - 实施与工具相关的取代

手册用法

RelTool（*Relative Tool*）用于将通过有效工具坐标系表达的位移和/或旋转增加至机械臂位置。

基本示例

以下示例介绍了函数RelTool。

例 1

```
MoveL RelTool (p1, 0, 0, 100), v100, fine, tool1;
```

沿工具的z方向，将机械臂移动至距p1达100 mm的一处位置。

例 2

```
MoveL RelTool (p1, 0, 0, 0 \Rz:= 25), v100, fine, tool1;
```

将工具围绕其z轴旋转25°。

返回值

数据类型： robtarget

增加一个位移和/或一次旋转的新位置，如存在，则同有效工具相关。

变元

```
RelTool (Point Dx Dy Dz [\Rx] [\Ry] [\Rz])
```

Point

数据类型： robtarget

输入机械臂位置。该位置的方位规定了工具坐标系的当前方位。

Dx

数据类型：num

工具坐标系x方向的位移，以mm计。

Dy

数据类型：num

工具坐标系y方向的位移，以mm计。

Dz

数据类型：num

工具坐标系z方向的位移，以mm计。

[\Rx]

数据类型：num

围绕工具坐标系x轴的旋转，以度计。

[\Ry]

数据类型：num

围绕工具坐标系y轴的旋转，以度计。

下一页继续

[\Rz]

数据类型：num

围绕工具坐标系z轴的旋转，以度计。

如果同时指定两次或三次旋转，则将首先围绕x轴旋转，随后围绕新的y轴旋转，然后围绕新的z轴旋转。

语法

```
RelTool '('
  [ Point ':=' ] < expression (IN) of robtarget> ','
  [Dx ':='] <expression (IN) of num> ','
  [Dy ':='] <expression (IN) of num> ','
  [Dz ':='] <expression (IN) of num>
  ['\' Rx ':=' <expression (IN) of num> ]
  ['\' Ry ':=' <expression (IN) of num> ]
  ['\' Rz ':=' <expression (IN) of num> ] ')'
```

含数据类型robtarget的返回值的函数。

相关信息

信息，关于	请参阅
位置数据	第*1467*页的*robtarget - 位置数据*
数学指令和函数	技术参考手册 - *RAPID*语言概览
定位器指令	技术参考手册 - *RAPID*语言概览

2.141 RemainingRetries - 剩余的重试

手册用法

RemainingRetries用于发现程序中错误处理器尚未完成的RETRY次数。在配置中确定最多重试次数。

基本示例

以下示例介绍了函数RemainingRetries。

例 1

```
...
ERROR
  IF RemainingRetries() > 0 THEN
    RETRY;
  ELSE
    TRYNEXT;
  ENDIF
...
```

无论是否出错，该程序将重试指令，直至完成最多重试次数，随后，尝试下一指令。

返回值

数据类型：num

返回值表明了最多重试次数中尚未进行的尝试次数。

语法

```
RemainingRetries '(' ')'
```

返回值的数据类型是 num 的函数。

相关信息

信息，关于	请参阅
错误处理器	技术参考手册 - *RAPID*语言概览
在错误后恢复执行	第*514*页的*RETRY - 在错误后恢复执行*
配置最多重试次数	技术参考手册 - 系统参数，*System misc*一节
重置统计的重试次数	第*511*页的*ResetRetryCount - 重置重试次数*

2.142 RMQGetSlotName - 获取RMQ客户端的名称

手册用法

RMQGetSlotName（*RAPID Mesasage Queue Get Slot Name*）用于从一个给定的槽识别号，即一个给定的rmqslot，获取一个RMQ或一个SDK客户端的槽名。

基本示例

以下示例介绍了函数RMQGetSlotName。

例 1

```
VAR rmqslot slot;
VAR string client_name;
RMQFindSlot slot, "RMQ_T_ROB1";
...
client_name := RMQGetSlotName(slot);
TPWrite "Name of the client: " + client_name;
```

本例子阐明了如何通过使用客户端的识别号，以获取客户端的名称。

返回值

数据类型：string

返回客户端的名称。其可以是一个RMQ名称，或者是一个使用RMQ功能的机械臂应用开发器客户端的名称。

变元

```
RMQGetSlotName (Slot)
```

Slot

数据类型：rmqslot

用于寻找名称的客户端的槽识别号。

程序执行

指令RMQGetSlotName用于发现含参数Slot中指定识别号的客户端的名称。该客户端可以为另一个RMQ或一个SDK客户端。

错误处理

可能会产生下列可恢复错误。错误可以用ERROR处理器进行处理。将系统变量ERRNO设置为：

ERR_RMQ_INVALID	尚未连接目的槽，或目的槽不再适用。如果未连接，则必须调用RMQFindSlot。如果不可用，则原因在于远程客户端已经同控制器断开。

语法

```
RMQGetSlotName '('
  [ Slot ':=' ] < variable (VAR) of rmqslot > ')'
```

返回值的数据类型是 string 的函数。

下一页继续

2.142 RMQGetSlotName - 获取RMQ客户端的名称

FlexPendant Interface, PC Interface, or Multitasking

续前页

相关信息

信息，关于	请参阅
RAPID Message Queue功能的描述	应用手册 - 控制器软件*IRC5*，*RAPID Message Queue*一节。
发现RAPID Message Queue任务或SDK客户端的识别号	第*520*页的*RMQFindSlot - 从槽名中寻找槽识别号*
将数据发送至RAPID任务或SDK客户端的队列	第*534*页的*RMQSendMessage - 发送RMQ数据消息*
从一个RAPID Message Queue获取第一条消息。	第*522*页的*RMQGetMessage - 获取RMQ消息*
将数据发送至RAPID任务或SDK客户端的队列，并等待客户端的回答	第*537*页的*RMQSendWait - 发送RMQ数据消息，并等待响应*
从`rmqmessage`提取标题数据	第*528*页的*RMQGetMsgHeader - 从RMQ消息获取标题信息*
从`rmqmessage`提取数据	第*525*页的*RMQGetMsgData - 从RMQ消息获取数据部分*
命令和启用特定数据类型的中断	第*274*页的*IRMQMessage - 下达数据类型的RMQ中断指令*
`RMQ Slot`	第*1465*页的*rmqslot - RMQ客户端的识别号*

2.143 RobName - 获取TCP机械臂名称

手册用法

RobName（*Robot Name*）用于获取一些程序任务中TCP机械臂的名称。如果任务未控制任何TCP机械臂，则该函数返回一个空字符串。

基本示例

以下示例介绍了函数RobName。

另请参阅第*1217*页的更多示例

例 1

```
VAR string my_robot;
...
my_robot := RobName();
IF my_robot="" THEN
  TPWrite "This task does not control any TCP robot";
ELSE
  TPWrite "This task controls TCP robot with name "+ my_robot;
ENDIF
```

将本程序任务控制的TCP机械臂的名称写入FlexPendant示教器。如果未控制任何TCP机械臂，则写入：本任务未控制机械臂。

返回值

数据类型：string

由该程序任务控制的TCP机械臂的相关机械单元的名称。如果未控制任何TCP机械臂，则返回空字符串。

更多示例

有关于如何使用指令RobName的更多例子阐述如下。

例 1

```
VAR string my_robot;
...
IF TaskRunRob() THEN
    my_robot := RobName();
    TPWrite "This task controls robot with name "+ my_robot;
ENDIF
```

如果该程序任务控制任何TCP机械臂，则将TCP机械的名称写入FlexPendant示教器。

语法

```
    RobName '(' ')'
```

返回值的数据类型是 string 的函数。

下一页继续

2.143 RobName - 获取TCP机械臂名称

RobotWare - OS

续前页

相关信息

信息，关于	请参阅
检查任务是否使一些TCP机械臂运行	第*1269*页的*TaskRunRob* - 检查任务是否控制一些机械臂
检查任务是否使一些机械单元运行	第*1268*页的*TaskRunMec* - 检查任务是否控制所有机械单元
获得系统中机械单元的名称	第*1092*页的*GetNextMechUnit* - 获取有关机械单元的名称和数据
字符串功能	技术参考手册 - *RAPID*指令、函数和数据类型
字符串的定义	第*1489*页的*string* - 字符串

2.144 RobOS - 检查是否在RC或VC上执行

手册用法

RobOS（*Robot Operating System*）可用于检查是否在机器人控制器RC或虚拟控制器VC上执行。

基本示例

以下示例介绍了函数RobOS。

例 1

```
IF RobOS() THEN
  ! Execution statements in RC
ELSE
  ! Execution statements in VC
ENDIF
```

返回值

数据类型：bool

如果在机器人控制器RC上执行，则TRUE，否则，FALSE。

语法

```
RobOS '(' ')'
```

含数据类型bool的返回值的函数。

2.145 Round - 按四舍五入计算数值

手册用法

Round用于使数值舍入至规定位数的小数或整数值。

基本示例

以下示例介绍了函数Round。

例 1

```
VAR num val;
val := Round(0.3852138\Dec:=3);
```

变量val被赋予值0.385。

例 2

```
val := Round(0.3852138\Dec:=1);
```

变量val被赋予值0.4。

例 3

```
val := Round(0.3852138);
```

变量val被赋予值0。

例 4

```
val := Round(0.3852138\Dec:=6);
```

变量val被赋予值0.385214。

返回值

数据类型：num

将数值舍入至规定位数的小数。

变元

```
Round ( Val [\Dec])
```

Val

Value

数据类型：num

有待舍入的数值。

[\Dec]

Decimals

数据类型：num

小数位数。

如果指定的小数位数为0，或者如果省略参数，则将值舍入为一个整数。

小数位数不得为负或大于数值的可用精度。

可以使用的最多小数位数为6。

语法

```
Round '('
  [ Val ':=' ] <expression (IN) of num>
```

下一页继续

```
[ \Dec ':=' <expression (IN) of num> ] ')'
```

返回值的数据类型是 num 的函数。

相关信息

信息，关于	请参阅
数学指令和函数	技术参考手册 - *RAPID*语言概览
截断一个值	第*1286*页的*Trunc* - 截断一个数值

2.146 RoundDnum - 按四舍五入计算数值

手册用法

RoundDnum用于使数值舍入至规定位数的小数或整数值。

基本示例

以下示例介绍了函数RoundDnum。

例 1

```
VAR dnum val;
val := RoundDnum(0.3852138754655357\Dec:=3);
```

变量val被赋予值0.385。

例 2

```
val := RoundDnum(0.3852138754655357\Dec:=1);
```

变量val被赋予值0.4。

例 3

```
val := RoundDnum(0.3852138754655357);
```

变量val被赋予值0。

例 4

```
val := RoundDnum(0.3852138754655357\Dec:=15);
```

变量val被赋予值0.385213875465536。

例 5

```
val := RoundDnum(1000.3852138754655357\Dec:=15);
```

变量val被赋予值1000.38521387547。

返回值

数据类型：dnum

将数值舍入至规定位数的小数。

变元

```
RoundDnum ( Val [\Dec])
```

Val

Value

数据类型：dnum

有待舍入的数值。

[\Dec]

Decimals

数据类型：num

小数位数。

如果指定的小数位数为0，或者如果省略参数，则将值舍入为一个整数。

小数位数不得为负或大于数值的可用精度。

可以使用的最多小数位数为15。

下一页继续

语法

```
RoundDnum '('
  [ Val ':=' ] <expression (IN) of dnum>
  [ \Dec ':=' <expression (IN) of num> ] ')'
```

含数据类型dnum的返回值的函数。

相关信息

信息，关于	请参阅
数学指令和函数	技术参考手册 - *RAPID*语言概览
将一个值舍入	第*1220*页的*Round* - 按四舍五入计算数值
截断一个值	第*1286*页的*Trunc* - 截断一个数值
截断一个值	第*1288*页的*TruncDnum* - 截断一个数值

2.147 RunMode - 读取运行模式

手册用法

RunMode（*Running Mode*）用于读取程序任务的当前运行模式。

基本示例

以下示例介绍了函数RunMode。

例 1

```
IF RunMode() = RUN_CONT_CYCLE THEN
...
ENDIF
```

仅针对持续或循环运行而执行的程序段。

返回值

数据类型：symnum

当前运行模式如下表所述。

返回值	符号常量	备注
0	RUN_UNDEF	未定义的运行模式
1	RUN_CONT_CYCLE	持续或循环运行模式
2	RUN_INSTR_FWD	指示向前运行模式
3	RUN_INSTR_BWD	指示向后运行模式
4	RUN_SIM	模拟运行模式。尚未发布。
5	RUN_STEP_MOVE	向前运动模式下的移动指令以及持续运行模式下的逻辑指令

变元

```
RunMode ( [ \Main] )
```

[\Main]

数据类型：switch

如果为一个运动任务，则返回任务的当前模式。如果用于一个非运动任务中，则其将返回非运动任务的相关运动任务的当前模式。

如果省略该参数，则返回值始终反映有关执行函数RunMode的程序任务的当前运行模式。

语法

```
RunMode '('
  ['\' Main] ')'
```

含数据类型symnum的返回值的函数。

相关信息

信息，关于	请参阅
读取运行模式	第*1158*页的*OpMode - 读取运行模式*

2.148 SafetyControllerGetChecksum - 获取用户配置文件的检查和。

手册用法

SafetyControllerGetChecksum将用于获取用户配置文件的安全控制柜检查和。

基本示例

下列示例说明了函数SafetyControllerGetChecksum。

例 1

```
VAR string mystring;
...
mystring:=SafetyControllerGetChecksum();
```

获取用户配置文件的检查和，并将之保存在变量mystring中。

返回值

数据类型：string

用户配置的检查和。

语法

```
SafetyControllerGetChecksum '(' ')'
```

返回值的数据类型是 string 的函数。

相关信息

信息，关于	请参阅
SafetyControllerGetUserChecksum	第*1227*页的*SafetyControllerGetUserChecksum - 获取受保护参数的检查和*
SafetyControllerGetSWVersion	第*1226*页的*SafetyControllerGetSWVersion - 获取安全控制柜固件版本*
SafetyControllerSyncRequest	第*541*页的*SafetyControllerSyncRequest - 硬件同步程序的初始化*
SafeMove安全配置	应用手册 - *Functional safety and SafeMove*

2.149 SafetyControllerGetSWVersion - 获取安全控制柜固件版本

手册用法

SafetyControllerGetSWVersion将用于获取安全控制柜固件版本

基本示例

下列示例说明了函数SafetyControllerGetSWVersion。

例 1

```
VAR string mystring;
...
mystring:=SafetyControllerGetSWVersion();
```

获取安全控制柜固件版本，并将之保存在变量mystring中。

返回值

数据类型：string

安全控制柜固件版本。如果在虚拟控制柜上使用该函数，那么将返回带“VC”的字符串。

语法

```
SafetyControllerGetSWVersion '(' ')'
```

返回值的数据类型是 string 的函数。

相关信息

信息，关于	请参阅
SafetyControllerGetUserChecksum	第*1227*页的*SafetyControllerGetUserChecksum - 获取受保护参数的检查和*
SafetyControllerGetSWVersion	第*1226*页的*SafetyControllerGetSWVersion - 获取安全控制柜固件版本*
SafetyControllerSyncRequest	第*541*页的*SafetyControllerSyncRequest - 硬件同步程序的初始化*
SafeMove安全配置	应用手册 - *Functional safety and SafeMove*

2.150 SafetyControllerGetUserChecksum - 获取受保护参数的检查和

手册用法

SafetyControllerGetUserChecksum将用于获取用户配置文件中带受保护参数的区域的安全控制柜检查和。

基本示例

下列示例说明了函数SafetyControllerGetUserChecksum。

例 1

```
VAR string mystring;
...
mystring:=SafetyControllerGetUserChecksum();
```

获取用户配置文件中带受保护参数的区域的检查和，并将之保存在变量mystring中。

返回值

数据类型：string

带受保护参数的区域的检查和。

语法

```
SafetyControllerGetUserChecksum '(' ')'
```

返回值的数据类型是 string 的函数。

相关信息

信息，关于	请参阅
SafetyControllerGetChecksum	第*1225*页的*SafetyControllerGetChecksum - 获取用户配置文件的检查和。*
SafetyControllerGetSWVersion	第*1226*页的*SafetyControllerGetSWVersion - 获取安全控制柜固件版本*
SafetyControllerSyncRequest	第*541*页的*SafetyControllerSyncRequest - 硬件同步程序的初始化*
SafeMove安全配置	应用手册 - *Functional safety and SafeMove*

2.151 Sin - 计算正弦值

手册用法

Sin（*Sine*）用于计算一个角值的正弦值。

基本示例

以下示例介绍了函数Sin。

例 1

```
VAR num angle;
VAR num value;
...
...
value := Sin(angle);
```

value将获得angle的正弦值。

返回值

数据类型：num

正弦值，范围 = 【-1，1】。

变元

```
Sin (Angle)
```

Angle

数据类型：num

角度值，以度表示。

语法

```
Sin '('
  [Angle ':='] <expression (IN) of num> ')'
```

返回值的数据类型是 num 的函数。

相关信息

信息，关于	请参阅
数学指令和函数	技术参考手册 - *RAPID*语言概览

2.152 SinDnum - 计算正弦 (sine) 值

手册用法

SinDnum (*Sine dnum*) 用于从数据类型 dnum 的角度值上计算正弦 (sine) 值。

基本示例

以下示例介绍了 SinDnum 函数。

例 1

```
VAR dnum angle;
VAR dnum value;
...
...
value := SinDnum(angle);
```

value将获得angle的正弦值。

返回值

数据类型：dnum

正弦值，范围 = 【-1，1】。

变元

```
SinDnum (Angle)
```

Angle

数据类型：dnum

角度值，以度表示。

语法

```
SinDnum '('
  [Angle ':='] <expression (IN) of dnum> ')'
```

含数据类型dnum的返回值的函数。

相关信息

信息，关于	请参阅
数学指令和函数	技术参考手册 - *RAPID*语言概览

2.153 SocketGetStatus - 获得当前套接字的状态

手册用法

SocketGetStatus返回一个套接字的当前状态。

基本示例

以下示例介绍了函数SocketGetStatus。

另请参阅第*1230*页的更多示例

例 1

```
VAR socketdev socket1;
VAR socketstatus state;
...
SocketCreate socket1;
state := SocketGetStatus( socket1 );
```

套接字状态SOCKET_CREATED将储存在变量state中。

返回值

数据类型：socketstatus

套接字的当前状态。

仅socketstatus型预定义符号常量可用于检查状态。

变元

```
SocketGetStatus( Socket )
```

Socket

数据类型：socketdev

状态相关的套接字变量。

程序执行

函数返回以下预定义套接字状态之一：

SOCKET_CREATED、SOCKET_CONNECTED、SOCKET_BOUND、SOCKET_LISTENING或SOCKET_CLOSED。

更多示例

以下示例介绍了函数SocketGetStatus。

例 1

```
VAR socketstatus status;
VAR socketdev my_socket;
...
SocketCreate my_socket;
SocketConnect my_socket, "192.168.0.1", 1025;
! A lot of RAPID code
status := SocketGetStatus( my_socket );
!Check which instruction that was executed last, not the state of
!the socket
```

下一页继续

```
IF status = SOCKET_CREATED THEN
  TPWrite "Instruction SocketCreate has been executed";
ELSEIF status = SOCKET_CLOSED THEN
  TPWrite "Instruction SocketClose has been executed";
ELSEIF status = SOCKET_BOUND THEN
  TPWrite "Instruction SocketBind has been executed";
ELSEIF status = SOCKET_LISTENING THEN
  TPWrite "Instruction SocketListen or SocketAccept has been
       executed";
ELSEIF status = SOCKET_CONNECTED THEN
  TPWrite "Instruction SocketConnect, SocketReceive or SocketSend
       has been executed";
ELSE
  TPWrite "Unknown socket status";
ENDIF
```

创建客户端套接字，并与一台远程计算机相连。在一个SocketSend指令中使用套接字之前，检查套接字的状态，以便其始终相连。

限制

仅可通过执行RAPID套接字指令来改变一个套接字的状态。例如，如果套接字相连，且连接随后断开，则将由SocketGetStatus函数记录。取而代之的是，当在一个SocketSend或SocketReceive指令中使用套接字时，将返回一个错误。

语法

```
SocketGetStatus '('
  [ Socket ':=' ] < variable (VAR) of socketdev > ')'
```

含数据类型socketstatus的返回值的函数。

相关信息

信息，关于	请参阅
一般的套接字通信	应用手册 - 控制器软件*IRC5*
创建新套接字	第*623*页的*SocketCreate - 创建新套接字*
连接远程计算机（仅限客户端）	第*620*页的*SocketConnect - 连接远程计算机*
向远程计算机发送数据	第*635*页的*SocketSend - 向远程计算机发送数据*
接收来自远程计算机的数据	第*627*页的*SocketReceive - 接收来自远程计算机的数据*
关闭套接字	第*618*页的*SocketClose - 关闭套接字*
绑定套接字（仅限服务器）	第*616*页的*SocketBind - 将套接字与我的IP地址和端口绑定*
监听连接（仅限服务器）	第*625*页的*SocketListen - 监听输入连接*
接受连接（仅服务器）	第*613*页的*SocketAccept - 接受输入连接*

2.154 SocketPeek - 测试套接字上数据的存在

手册用法

SocketPeek用于测试一个套接字上数据的存在。其返回可在规定套接字上收到的字节数。

基本示例

以下示例介绍了函数SocketPeek。

例 1

```
VAR socketdev socket1;
VAR socketdev client_socket;
VAR num peek_value;
...
SocketCreate socket1;
SocketBind socket1, "192.168.0.1", 1025;
SocketListen socket1;
SocketAccept socket1, client_socket;
..
peek_value := SocketPeek( client_socket );
IF peek_value >= 64 THEN
  SocketReceive client_socket \Str := str_data \ReadNoOfBytes:=64;
  ..
ELSE
  ! Not enough data to receive. Do something else.
ENDIF
```

首先，创建一个服务器套接字，并与控制器网络地址192.168.0.1上的端口1025相连。随后，SocketPeek用于检查能否在套接字上收到64个字节的数据。

返回值

数据类型：num

指定套接字上可获得的字节数。

变元

```
SocketPeek( Socket )
```

Socket

数据类型：socketdev

应当予以窥视的套接字变量。

错误处理

系统会生成下列可恢复错误，并在错误处理器中处理这些错误。系统变量ERRNO将被设置成：

ERR_SOCK_CLOSED	关闭套接字。连接损坏。

限制

在上电失败重启后，关闭所有套接字。可通过错误恢复来处理该问题。

下一页继续

将一次调用中可收到的最大数据规模限制为1024字节。因此，SocketPeek所能返回的最大值为1024。

语法

```
SocketPeek '('
  [ Socket ':=' ] < variable (VAR) of socketdev > ')'
```

返回值的数据类型是 num 的函数。

相关信息

信息，关于	请参阅
一般的套接字通信	应用手册 - 控制器软件*IRC5*
创建新套接字	第*623*页的*SocketCreate* - 创建新套接字
连接远程计算机（仅限客户端）	第*620*页的*SocketConnect* - 连接远程计算机
向远程计算机发送数据	第*635*页的*SocketSend* - 向远程计算机发送数据
向远程计算机发送数据	第*639*页的*SocketSendTo* - 向远程计算机发送数据
关闭套接字	第*618*页的*SocketClose* - 关闭套接字
绑定套接字（仅限服务器）	第*616*页的*SocketBind* - 将套接字与我的*IP*地址和端口绑定
监听连接（仅限服务器）	第*625*页的*SocketListen* - 监听输入连接
接受连接（仅服务器）	第*613*页的*SocketAccept* - 接受输入连接
获得当前套接字的状态	第*1230*页的*SocketGetStatus* - 获得当前套接字的状态
客户端套接字应用实例	第*635*页的*SocketSend* - 向远程计算机发送数据
接收来自远程计算机的数据	第*627*页的*SocketReceive* - 接收来自远程计算机的数据
接收来自远程计算机的数据	第*631*页的*SocketReceiveFrom* - 接收来自远程计算机的数据

2.155 Sqrt - 计算平方根值

手册用法

Sqrt（*Square root*）用于计算平方根值。

基本示例

以下示例介绍了函数Sqrt。

例 1

```
VAR num x_value;
VAR num y_value;
...
...
y_value := Sqrt( x_value);
```

y-value将获得x_value的平方根，即√（x_value）。

返回值

数据类型：num

平方根值（√）。

变元

```
Sqrt (Value)
```

Value

数据类型：num

平方根的参数值，即√value。

Value需要≥0。

限制

如果x < 0，则函数Sqrt(x)的执行将出错。

语法

```
Sqrt '('
  [Value ':='] <expression (IN) of num> ')'
```

返回值的数据类型是 num 的函数。

相关信息

信息，关于	请参阅
计算一个双数值的平方根值	第*1235*页的*SqrtDnum - 计算平方根值*
数学指令和函数	技术参考手册 - *RAPID*语言概览

2.156 SqrtDnum - 计算平方根值

手册用法

SqrtDmum（平方根双数值）用于计算平方根值。

基本示例

以下示例介绍了函数SqrtDnum。

例 1

```
VAR dnum x_value;
VAR dnum y_value;
...
...
y_value := SqrtDnum(x_value);
```

y_value将获得x_value的平方根，即√（x_value）。

返回值

数据类型：dnum

平方根值（√）。

变元

```
SqrtDnum (Value)
```

Value

数据类型：dnum

平方根的参数值，即√value。

Value需要≥0。

限制

如果x < 0，则函数Sqrt(x)的执行将出错。

语法

```
SqrtDnum '('
  [ Value ':=' ] < expression (IN) of dnum > ')'
```

含数据类型dnum的返回值的函数。

相关信息

信息，关于	请参阅
计算一个数值的平方根值	第*1234*页的*Sqrt - 计算平方根值*
数学指令和函数	技术参考手册 - *RAPID*语言概览

2.157 STCalcForce - 计算一个伺服工具的焊枪头压力

手册用法

STCalcForce用于计算一个伺服工具的焊枪头压力。该函数的作用是，例如，寻找一个伺服工具的最大容许焊枪头压力。

基本示例

以下示例介绍了函数STCalcForce。

例 1

```
VAR num tip_force;
tip_force := STCalcForce(gun1, 7);
```

当所需电机扭矩为7 Nm时，计算焊枪头压力。

返回值

数据类型：num

计算出的焊枪头压力【N】。

变元

```
STCalcForce ToolName MotorTorque
```

ToolName

数据类型：string

机械单元名称

MotorTorque

数据类型：num

所需电机扭矩【Nm】。

错误处理

如果指定伺服工具名称并非配置伺服工具，则将系统变量ERRNO设置为ERR_NO_SGUN。

可以用Rapid错误处理器来处理错误。

语法

```
STCalcForce '('
  [ ToolName ':=' ] < expression (IN) of string > ','
  [ MotorTorque ':=' ] < expression (IN) of num > ';'
```

返回值的数据类型是 num 的函数。

相关信息

信息，关于	请参阅
打开伺服工具	第*688*页的*STOpen - 打开伺服工具*
关闭伺服工具	第*671*页的*STClose - 关闭伺服工具*
计算电机扭矩	第*1237*页的*STCalcTorque - 计算一个伺服工具的电机扭矩*

2.158 STCalcTorque - 计算一个伺服工具的电机扭矩

手册用法

STCalcTorque用于计算一个伺服工具的电机扭矩。该函数的作用是，例如，用于实施力校准。

基本示例

以下示例介绍了函数STCalcTorque。

例 1

```
VAR num curr_motortorque;
curr_motortorque := STCalcTorque( gun1, 1000);
```

当所需焊枪头压力为1000 N时，计算电机扭矩。

返回值

数据类型：num

计算出的电机扭矩【Nm】。

变元

```
STCalcTorque ToolName TipForce
```

ToolName

数据类型：string

机械单元名称

TipForce

数据类型：num

所需焊枪头压力[N]。

错误处理

如果指定伺服工具名称并非配置伺服工具，则将系统变量ERRNO设置为ERR_NO_SGUN。

可以用Rapid错误处理器来处理错误。

语法

```
STCalcTorque '('
  [ ToolName ':=' ] < expression (IN) of string > ','
  [ TipForce ':=' ] < expression (IN) of num > ';'
```

返回值的数据类型是 num 的函数。

相关信息

信息，关于	请参阅
打开伺服工具	第688页的*STOpen - 打开伺服工具*
关闭伺服工具	第671页的*STClose - 关闭伺服工具*
计算焊枪头压力	第1236页的*STCalcForce - 计算一个伺服工具的焊枪头压力*

2.159 STIsCalib - 测试是否计算出一个伺服工具

手册用法

STIsCalib用于测试是否计算出一个伺服工具，即检查焊枪头是否完成校准或同步。

基本示例

以下示例介绍了函数STIsCalib。

例 1

```
IF STIsCalib(gun1\sguninit) THEN
  ...
ELSE
  !Start the gun calibration
  STCalib gun1\TipChg;
ENDIF
```

例 2

```
IF STIsCalib(gun1\sgunsynch) THEN
  ...
ELSE
  !Start the gun calibration to synchronize the gun position with
       the revolution counter
  STCalib gun1\ToolChg;
ENDIF
```

返回值

数据类型：bool

如果经测试的工具完成校准，即工具头之间的距离完成校准，或者如果经测试的工具完成同步，即工具头的位置与工具的转数计数器同步，则TRUE。

如果经测试的工具未校准或同步，则FALSE。

变元

```
STIsCalib ToolName [\sguninit] | [\sgunsynch]
```

ToolName

数据类型：string

机械单元名称

[\sguninit]

数据类型：switch

该参数用于检查焊枪位置是否得以初始化和校准。

[\sgunsynch]

数据类型：switch

该参数用于检查焊枪位置是否与转数计数器同步。

语法

```
STIsCalib '('
  [ ToolName ':=' ] < expression (IN) of string >
```

下一页继续

```
[ '\' sguninit ] | [ '\'sgunsynch ] ')'
```

含数据类型bool的返回值的函数。

相关信息

信息，关于	请参阅
校准一个伺服工具	第*667*页的*STCalib* - 校准伺服工具

2.160 STIsClosed - 测试是否已关闭一个伺服工具

手册用法

STIsClosed用于测试是否已关闭一个伺服工具。

基本示例

以下示例介绍了函数STIsClosed。

例 1

```
IF STIsClosed(gun1) THEN
  !Start the weld process
  Set weld_start;
ELSE
  ...
ENDIF
```

检查是否已关闭焊枪。

例 2

```
STClose "sgun", 1000, 3 \Conc;
WHILE NOT(STIsClosed("sgun"\RetThickness:=thickness)) DO
  WaitTime 0.1;

ENDWHILE
IF thickness > max_thickness THEN...
```

开始关闭名为sgun的焊枪。立即继续下一个指令，程序由此等待焊枪关闭。当指令STIsClosed已返回TRUE时，读取所实现的厚度值。

例 3

无效组合实例：

```
STClose "sgun", 1000, 3 \RetThickness:=thickness \Conc;
WHILE NOT(STIsClosed("sgun"\RetThickness:=thickness_2)) DO;
...
```

关闭焊枪。由于使用了\Conc开关，参数厚度将不会保存任何有效值。等待，直至焊枪关闭。当焊枪已关闭，且STIsClosed返回TRUE时，参数厚度_2将保存一个有效值，因为\Conc开关用于STClose。

```
STClose "sgun", 1000, 3 \RetThickness:=thickness;
WHILE NOT(STIsClosed("sgun"\RetThickness:=thickness_2)) DO;
...
```

关闭焊枪。由于未使用\Conc开关，因此，当已经关闭焊枪时，参数厚度将保存一个有效值。由于\Conc开关未用于STClose指令，因此，参数厚度_2将不会保存任何有效值。

返回值

数据类型：bool

如果经测试的工具已关闭，即已实现所需焊枪头压力，则TRUE。

如果经测试的工具未关闭，则FALSE。

下一页继续

变元

```
STIsClosed ToolName [\RetThickness]
```

`ToolName`

数据类型：`string`

机械单元名称

`[\RetThickness]`

数据类型：`num`

实现的厚度【mm】。

注意！仅当`\Conc`**已用于先前的**`STClose`**指令时方才有效。**

语法

```
STIsClosed '('
  [ ToolName ':=' ] < expression (IN) of string > ')'
  ['\' RetThickness ':=' < variable or persistent (INOUT) of num
       > ] ')'
```

含数据类型`bool`**的返回值的函数。**

相关信息

信息，关于	请参阅
打开伺服工具	第*688*页的*STOpen - 打开伺服工具*
关闭伺服工具	第*671*页的*STClose - 关闭伺服工具*
测试是否已打开一个伺服工具	第*1243*页的*STIsOpen - 测试是否已打开一个伺服工具*

2.161 STIsIndGun - 测试一个伺服工具是否处于独立模式

手册用法

STIsIndGun用于测试一个伺服工具是否处于独立模式。

基本示例

以下示例介绍了函数STIsIndGun。

例 1

```
IF STIsIndGun(gun1) THEN
  ! Start the gun calibration
  STCalib gun1\TipChg;
ELSE
  ...
ENDIF
```

返回值

数据类型：bool

如果经测试的工具处于独立模式，即可不依赖于机械臂移动而移动焊枪，则TRUE。

如果经测试的工具不处于同步模式，则FALSE。

变元

```
STIsIndGun ToolName
```

ToolName

数据类型：string

机械单元名称

语法

```
STIsIndGun '('
  [ ToolName ':=' ] < expression (IN) of string > ')'
```

含数据类型bool的返回值的函数。

相关信息

信息，关于	请参阅
校准一个伺服工具	第*667*页的*STCalib - 校准伺服工具*
以独立模式设置焊枪	第*676*页的*STIndGun - 以独立模式设置焊枪*
根据独立模式重置焊枪	第*678*页的*STIndGunReset - 以独立模式重置焊枪*

2.162 STIsOpen - 测试是否已打开一个伺服工具

手册用法

STIsOpen用于测试是否已开启一个伺服工具。

基本示例

以下示例介绍了函数STIsOpen。

例 1

```
IF STIsOpen(gun1) THEN
  !Start the motion
  MoveL ...
ELSE
  ...
ENDIF
```

检查是否已开启焊枪。

例 2

```
STCalib "sgun" \TipWear \Conc;
WHILE NOT(STIsOpen("sgun") \RetTipWear:=tipwear \RetPosAdj:=posadj)
     DO;
WaitTime 0.1;

ENDWHILE
```

如果焊枪头磨损> 20...

如果posadj > 25...

实施焊枪头磨损校准。等待，直至焊枪sgun开启。读取焊枪头磨损和定位调整值。

例 3

无效组合实例：

```
STCalib "sgun" \TipWear \RetTipWear:=tipwear_1 \Conc;
WHILE NOT(STIsOpen("sgun") \RetTipWear:=tipwear_2) DO;
  WaitTime 0.1;

ENDWHILE
```

开始焊枪头磨损校准。由于使用\Conc开关，因此，参数tipwear_1将不会保存任何有效值。当校准就绪，且STIsOpen返回TRUE时，参数tipwear_2将保存一个有效值。

```
STCalib "sgun" \TipWear \RetTipWear:=tipwear_1;

WHILE NOT(STIsOpen("sgun") \RetTipWear:=tipwear_2) DO;
  WaitTime 0.1;

ENDWHILE
```

实施焊枪头磨损校准。由于未使用\Conc开关，因此，参数tipwear_1将保存一个有效值。由于 \Conc开关未用于STCalib，因此，当STIsOpen返回TRUE时，参数tipwear_2将不会保存任何有效值。

下一页继续

2.162 STIsOpen - 测试是否已打开一个伺服工具

Servo Tool Control

续前页

返回值

数据类型：bool

如果经测试的工具开启，即工具臂处于编程开启位置，则TRUE。

如果经测试的工具未开启，则FALSE。

变元

```
STIsOpen ToolName [\RetTipWear] [\RetPosAdj]
```

ToolName

数据类型：string

机械单元名称

[\RetTipWear]

数据类型：num

造成的焊枪头磨损[mm]。

注意！仅当\Conc已用于先前的STCalib指令，且STIsOpen返回TRUE时，方才有效。

[\RetPosAdj]

数据类型：num

自最后一次校准起的位置调整[mm]。

注意！仅当\Conc已用于先前的STCalib指令，且STIsOpen返回TRUE时，方才有效。

语法

```
STIsOpen '('
  [ ToolName ':=' ] < expression (IN) of string > ')'
  [ '\' RetTipWear ':=' < variable or persistent(INOUT) of num >
       ] ';'
  [ '\' RetPosAdj ':=' < variable or persistent(INOUT) of num > ]
```

含数据类型bool的返回值的函数。

相关信息

信息，关于	请参阅
打开伺服工具	第*688*页的*STOpen - 打开伺服工具*
关闭伺服工具	第*671*页的*STClose - 关闭伺服工具*
测试是否已关闭一个伺服工具	第*1240*页的*STIsClosed - 测试是否已关闭一个伺服工具*

2.163 StrDigCalc - 有关数据类型数字字符串的算术运算

手册用法

StrDigCalc用于采用与有关正整数值的数字算术运算相同的方式，对两个正数字字符串实施算术运算（+、-、*、/、%）。

该函数可通过准确的表达式，处理高于8 388 608的正整数。

基本示例

以下示例介绍了函数StrDigCalc。

另请参阅第*1246*页的更多示例

例 1

```
res := StrDigCalc(str1, OpAdd, str2);
```

向res分配有关数字字符串str1和str2所代表值的加法运算的结果。

返回值

数据类型：stringdig

stringdig用于代表纯数字字符串中的较大正整数。

因为数据类型num无法通过准确的表达式来处理高于8 388 608的正整数，因而引入该数据类型。

变元

```
StrDigCalc (StrDig1 Operation StrDig2)
```

StrDig1

String Digit 1

数据类型：stringdig

代表一个正整数值的字符串。

Operation

Arithmetic operator

数据类型：opcalc

定义在两个数字字符串上实施的算术运算。在所能使用的数据类型opcalc的算术运算之后；OpAdd，OpSub，OpMult，OpDiv 和OpMod。

StrDig2

String Digit 2

数据类型：stringdig

代表一个正整数值的字符串。

程序执行

该函数将：

- 仅检查StrDig1和StrDig2中的数字0...9
- 将两个数字字符串转换为long integers
- 对两个long integers实施算术运算
- 将结构由long integer转换为stringdig

下一页继续

2.163 StrDigCalc - 有关数据类型数字字符串的算术运算

RobotWare - OS

续前页

更多示例

以下示例介绍了函数StrDigCalc。

例 1

```
res := StrDigCalc(str1, OpSub, str2);
```

向res分配有关数字字符串str1和str2所代表值的减法运算的结果。

例 2

```
res := StrDigCalc(str1, OpMult, str2);
```

向res分配有关数字字符串str1和str2所代表值的乘法运算的结果。

例 3

```
res := StrDigCalc(str1, OpDiv, str2);
```

向res分配有关数字字符串str1和str2所代表值的除法运算的结果。

例 4

```
res := StrDigCalc(str1, OpMod, str2);
```

向res分配有关数字字符串str1和str2所代表值的模数运算的结果。

错误处理

可以用Rapid错误处理器来处理以下错误。

错误代码	描述
ERR_INT_NOTVAL	除数字或模数零以外的输入值
ERR_INT_MAXVAL	高于4294967295的输入值
ERR_CALC_OVERFLOW	范围0...4294967295外的结果
ERR_CALC_NEG	减法，即StrDig2 > StrDig1
ERR_CALC_DIVZERO	除零

限制

StrDigCalc仅接受包含数字（字符0...9）的字符串。stringdig中的所有其他字符将引起错误。

该函数仅可处理高达4 294 967 295的正整数。

语法

```
StrDigCalc '('
  [ StrDig1 ':=' ] < expression (IN) of stringdig > ','
  [ Operation ':=' ] < expression (IN) of opcalc > ','
  [ StrDig2 ':=' ] < expression (IN) of stringdig > ')'‘
```

含数据类型stringdig的返回值的函数。

相关信息

信息，关于	请参阅
只含数字的字符串。	第*1491*页的*stringdig - 只含数字的字符串*
算术运算符。	第*1437*页的*opcalc - Arithmetic Operator*

2.164 StrDigCmp - 将仅含数字的两个字符串进行比较

手册用法

通过正整数的数值比较方法，StrDigCmp用于比较两个正数字字符串。

该函数可通过准确的表达式，处理高于8 388 608的正整数。

基本示例

以下示例介绍了函数StrDigCmp。

例 1

```
VAR stringdig digits1 := "1234";
VAR stringdig digits2 := "1256";
VAR bool is_equal;
is_equal := StrDigCmp(digits1, EQ, digits2);
```

因为数值1234与1256不相等，因此，将变量is_equal设置为FALSE。

返回值

数据类型：bool

如果满足给定条件，则TRUE，否则，FALSE。

变元

```
StrDigCmp (StrDig1 Relation StrDig2)
```

StrDig1

String Digit 1

数据类型：stringdig

仅含有待进行数值比较的数字的第一个字符串。

Relation

数据类型：opnum

定义如何比较两个数字字符串。在所能使用的数据类型opnum的预定义常量后，LT, LTEQ, EQ, NOTEQ, GTEQ或GT。

StrDig2

String Digit 2

数据类型：stringdig

仅含有待进行数值比较的数字的第二个字符串。

程序执行

该函数将：

- 检查是否仅数字0...9用于StrDig1和StrDig2
- 将两个数字字符串转换为long integers
- 从数字上比较两个long integers

下一页继续

2.164 StrDigCmp - 将仅含数字的两个字符串进行比较

RobotWare - OS

续前页

错误处理

可以用Rapid错误处理器来处理以下错误。

错误代码	描述
ERR_INT_NOTVAL	不仅包含数字的输入值
ERR_INT_MAXVAL	高于4294967295的值

限制

StrDigCmp仅接受包含数字（字符0...9）的字符串。stringdig中的所有其他字符将引起错误。

该函数仅可处理高达4 294 967 295的正整数。

语法

```
StrDigCmp '('
  [ StrDig1 ':=' ] < expression (IN) of stringdig > ','
  [ Relation ':=' ] < expression (IN) of opnum > ','
  [ StrDig2 ':=' ] < expression (IN) of stringdig > ')'
```

含数据类型bool的返回值的函数。

相关信息

信息，关于	请参阅
只含数字的字符串	第*1491*页的*stringdig - 只含数字的字符串*
比较运算符	第*1438*页的*opnum - 比较运算符*
文件时间信息	第*1079*页的*FileTimeDnum - 检索有关文件的时间信息*
已加载模块的文件修改时间	第*1145*页的*ModTimeDnum - 获取加载模块的文件修改时间*

2.165 StrFind - 搜索一个字符串中的一个字符

手册用法

StrFind（*String Find*）用于在一个字符串中搜索始于一个指定位置、属于一组指定字符的一个字符。

基本示例

以下示例介绍了函数StrFind。

例 1

```
VAR num found;
found := StrFind("Robotics",1,"aeiou");
```

变量found被赋予值“2”。

```
found := StrFind("Robotics",1,"aeiou"\NotInSet);
```

变量found被赋予值“1”。

```
found := StrFind("IRB 6400",1,STR_DIGIT);
```

变量found被赋予值“5”。

```
found := StrFind("IRB 6400",1,STR_WHITE);
```

变量found被赋予值4。

返回值

数据类型：num

位于属于指定集合的规定位置或其后位置的第一个字符的字符位置。如果未发现此类字符，返回字符串长度+1。

变元

```
StrFind (Str ChPos Set [\NotInSet])
```

Str

String

数据类型：string

用于搜索的字符串。

ChPos

Character Position

数据类型：num

开始字符位置。如果位于字符串以外，则产生运行时错误。

Set

数据类型：string

有待测试的字符串集合。亦请参见第*1250*页的预定义数据。

[\NotInSet]

数据类型：switch

搜索未在Set中呈现的字符集合中的一个字符。

下一页继续

2.165 StrFind - 搜索一个字符串中的一个字符

RobotWare - OS

续前页

语法

```
StrFind '('
  [ Str ':=' ] <expression (IN) of string> ','
  [ ChPos ':=' ] <expression (IN) of num> ','
  [ Set ':=' ] <expression (IN) of string>
  ['\' NotInSet ] ')'
```

返回值的数据类型是 num 的函数。

预定义数据

可在系统中利用大量的预定义字符串常量，并且可与字符串函数一同使用。

名称	字符集
STR_DIGIT	<digit> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
STR_UPPER	<upper case letter> ::= A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | À | Á | Â | Ã | Ä | Å | Æ | Ç | È | É | Ê | Ë | Ì | Í | Î | Ï | 1)| Ñ | Ò | Ó | Ô | Õ | Ö | Ø | Ù | Ú | Û | Ü | 2) | 3)
STR_LOWER	<lower case letter> ::= a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p | q | r | s | t | u | v | w | x | y | z | à | á | â | ã | ä | å | æ | ç | è | é | ê | ë | ì | í | î | ï | 1) | ñ | ò | ó | ô | õ | ö | ø | ù | ú | û | ü | 2) | 3) | ß | ÿ-
STR_WHITE	<blank character> ::=

相关信息

信息，关于	请参阅
字符串功能	技术参考手册 - *RAPID*语言概览
字符串的定义	第*1489*页的*string - 字符串*
字符串值	技术参考手册 - *RAPID*语言概览

2.166 StrLen - 获取字符串长度

手册用法

StrLen（*String Length*）用于发现一个字符串的当前长度。

基本示例

以下示例介绍了函数StrLen。

例 1

```
VAR num len;
len := StrLen("Robotics");
```

变量len被赋予值8。

返回值

数据类型：num

字符串中的字符数量（> = 0）。

变元

```
StrLen (Str)
```

Str

String

数据类型：string

字符数量有待统计的字符串。

语法

```
StrLen '('
  [ Str ':=' ] <expression (IN) of string> ')'
```

返回值的数据类型是 num 的函数。

相关信息

信息，关于	请参阅
字符串功能	技术参考手册 - *RAPID*指令、函数和数据类型
字符串的定义	第*1489*页的*string - 字符串*
字符串值	技术参考手册 - *RAPID*指令、函数和数据类型

2.167 StrMap - 在地图上绘制一个字符串

手册用法

StrMap（*String Mapping*）用于复制一个字符串，根据指定的映射，转换该字符串中所有的字符。

基本示例

以下示例介绍了函数StrMap。

例 1

```
VAR string str;
str := StrMap("Robotics","aeiou","AEIOU");
```

变量str被赋予值RObOtIcs。

例 2

```
str := StrMap("Robotics",STR_LOWER, STR_UPPER);
```

变量str被赋予值ROBOTICS。

返回值

数据类型：string

通过转换指定字符串中的字符而创建的字符串，如“始于”和“截至”字符串所规定。在“始于”字符串中发现的指定字符串的各个字符由“截至”字符串中相应位置的字符所取代。将未定义任何映射的字符串不作改变地复制到生成的字符串。

变元

```
StrMap ( Str FromMap ToMap)
```

Str

String

数据类型：string

用于转换的字符串。

FromMap

数据类型：string

映射的变址部分。亦请参见第*1253*页的预定义数据。

ToMap

数据类型：string

映射的值部分。亦请参见第*1253*页的预定义数据。

语法

```
StrMap '('
  [ Str ':=' ] <expression (IN) of string> ','
  [ FromMap ':=' ] <expression (IN) of string> ','
  [ ToMap ':=' ] <expression (IN) of string> ')'
```

返回值的数据类型是 string 的函数。

下一页继续

预定义数据

可在系统中利用大量的预定义字符串常量，并且可与字符串函数一同使用。

名称	字符集
STR_DIGIT	<digit> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
STR_UPPER	<upper case letter> ::= A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | À | Á | Â | Ã | Ä | Å | Æ | Ç | È | É | Ê | Ë | Ì | Í | Î | Ï | 1)| Ñ | Ò | Ó | Ô | Õ | Ö | Ø | Ù | Ú | Û | Ü | 2) | 3)
STR_LOWER	<lower case letter> ::= a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p | q | r | s | t | u | v | w | x | y | z | à | á | â | ã | ä | å | æ | ç | è | é | ê | ë | ì | í | î | ï | 1) | ñ | ò | ó | ô | õ | ö | ø | ù | ú | û | ü | 2) | 3) | ß | ÿ-
STR_WHITE	<blank character> ::=

相关信息

信息，关于	请参阅
字符串功能	技术参考手册 - *RAPID*语言概览
字符串的定义	第*1489*页的*string - 字符串*
字符串值	技术参考手册 - *RAPID*语言概览

2.168 StrMatch - 在字符串中搜索模板

手册用法

StrMatch（*String Match*）用于在始于一个指定位置的一个字符串中搜索一个特定模板。

基本示例

以下示例介绍了函数StrMatch。

例 1

```
VAR num found;

found := StrMatch("Robotics",1,"bo");
```

变量found被赋予值3。

返回值

数据类型：num

位于等同于指定模板字符串的规定位置或其后位置的第一个子串的字符位置。如果未发现此类子串，则返回字符串长度+1。

变元

```
StrMatch (Str ChPos Pattern)
```

Str

String

数据类型：string

用于搜索的字符串。

ChPos

Character Position

数据类型：num

开始字符位置。如果位于字符串以外，则产生运行时错误。

Pattern

数据类型：string

有待搜索的模板字符串。

语法

```
StrMatch '('
  [ Str ':=' ] <expression (IN) of string> ','
  [ ChPos ':=' ] <expression (IN) of num> ','
  [ Pattern ':=' ] <expression (IN) of string> ')'
```

返回值的数据类型是 num 的函数。

相关信息

信息，关于	请参阅
字符串功能	技术参考手册 - *RAPID*语言概览

下一页继续

信息，关于	请参阅
字符串的定义	第*1489*页的*string - 字符串*
字符串值	技术参考手册 - *RAPID*语言概览

2.169 StrMemb - 检查字符是否属于一组

手册用法

StrMemb（字符串组成）用于检查一个字符串中的指定字符是否属于一个指定的字符组。

基本示例

以下示例介绍了函数StrMemb。

例 1

```
VAR bool memb;
memb := StrMemb("Robotics",2,"aeiou");
```

因为o是集合"aeiou"的组成部分，因此，假定变量memb的值TRUE。

```
memb := StrMemb("Robotics",3,"aeiou");
```

因为b不是集合"aeiou"的组成部分，因此，假定变量memb的值FALSE。

```
memb := StrMemb("S-721 68 VÄSTERÅS",3,STR_DIGIT);
```

因为7是集合STR_DIGIT的组成部分，因此，假定变量memb的值TRUE。

返回值

数据类型：bool

如果位于指定字符串指定位置的字符属于指定的字符集合，则TRUE。

变元

```
StrMemb (Str ChPos Set)
```

Str

String

数据类型：string

用于登记的字符串。

ChPos

Character Position

数据类型：num

有待检查的字符位置。如果位于字符串以外，则产生运行时错误。

Set

数据类型：string

有待测试的字符串集合。

语法

```
StrMemb '('
  [ Str ':=' ] <expression (IN) of string> ','
  [ ChPos ':=' ] <expression (IN) of num> ','
  [ Set ':=' ] <expression (IN) of string> ')'
```

含数据类型bool的返回值的函数。

下一页继续

预定义数据

可在系统中利用大量的预定义字符串常量，并且可与字符串函数一同使用。

名称	字符集
STR_DIGIT	<digit> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
STR_UPPER	<upper case letter> ::= A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | À | Á | Â | Ã | Î | Ï | 1)| Ñ | Ò | Ó | Ô | Õ | Ö | Ø | Ù | Ú | Û | Ü | 2) | 3)
STR_LOWER	<lower case letter> ::= a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p | q | r | s | t | u | v | w | x | y | z | à | á | â | ã | ä | å | æ | ç | è | é | ê | ë | ì | í | î | ï | 1) | ñ | ò | ó | ô | õ | ö | ø | ù | ú | û | ü | 2) | 3) | ß | ÿ-
STR_WHITE	<blank character> ::=

相关信息

信息，关于	请参阅
字符串功能	技术参考手册 - *RAPID*语言概览
字符串的定义	第*1489*页的*string* - 字符串
字符串值	技术参考手册 - *RAPID*语言概览

2.170 StrOrder - 检查是否已下达字符串指令

手册用法

StrOrder（*String Order*）将两个字符串进行比较（按字符），并返回一个布尔值，以此表明两个字符串是否符合指定字符排序序列。

基本示例

以下示例介绍了函数StrOrder。

例 1

```
VAR bool le;

le := StrOrder("FIRST","SECOND",STR_UPPER);
```

因为在字符排序序列STR_UPPER中，“F”位于“S”之前，因此，假定变量le的值为TRUE。

例 2

```
VAR bool le;

le := StrOrder("FIRST","FIRSTB",STR_UPPER);
```

因为"FIRSTB"在字符排序序列中具有一个附加字符（与"B"相比，无字符），因此，假定变量le的值为TRUE。

例 3

```
VAR bool le;

le := StrOrder("FIRSTB","FIRST",STR_UPPER);
```

因为"FIRSTB"在字符排序序列中具有一个附加字符（与无字符相比，"B"），因此，假定变量le的值为FALSE。

返回值

数据类型：bool

当按照说明下达字符指令时，如果第一个字符串位于第二个字符串之前（Str1 <= Str2），则TRUE。

假定不包括在规定排序中的字符均位于当前字符之后。

变元

```
StrOrder ( Str1 Str2 Order)
```

Str1

String 1

数据类型：string

第一个字符串值。

Str2

String 2

数据类型：string

第二个字符串值。

下一页继续

`Order`

数据类型：`string`

用于确定排序的字符顺序。亦请参见第*1259*页的预定义数据。

语法

```
StrOrder '('
  [ Str1 ':=' ] <expression (IN) of string> ','
  [ Str2 ':=' ] <expression (IN) of string> ','
  [ Order ':=' ] <expression (IN) of string> ')'
```

含数据类型`bool`的返回值的函数。

预定义数据

可在系统中利用大量的预定义字符串常量，并且可与字符串函数一同使用。

名称	字符集
`STR_DIGIT`	<digit> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
`STR_UPPER`	<upper case letter> ::= A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | À | Á | Â | Ã | Ä | Å | Æ | Ç | È | É | Ê | Ë | Ì | Í | Î | Ï | 1)| Ñ | Ò | Ó | Ô | Õ | Ö | Ø | Ù | Ú | Û | Ü | 2) | 3)
`STR_LOWER`	<lower case letter> ::= a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p | q | r | s | t | u | v | w | x | y | z | à | á | â | ã | ä | å | æ | ç | è | é | ê | ë | ì | í | î | ï | 1) | ñ | ò | ó | ô | õ | ö | ø | ù | ú | û | ü | 2) | 3) | ß | ÿ-
`STR_WHITE`	<blank character> ::=

相关信息

信息，关于	请参阅
字符串功能	技术参考手册 - *RAPID*语言概览
字符串的定义	第*1489*页的*string* - 字符串
字符串值	技术参考手册 - *RAPID*语言概览

2.171 StrPart - 寻找一部分字符串

手册用法

StrPart（*String Part*)用于寻找一部分字符串，以作为一个新的字符串。

基本示例

以下示例介绍了函数StrPart。

例 1

```
VAR string part;
part := StrPart("Robotics",1,5);
```

变量part被赋予值"Robot"。

返回值

数据类型：string

指定字符串的子串，其拥有规定的长度，并始于指定字符位置。

变元

```
StrPart (Str ChPos Len)
```

Str

String

数据类型：string

字符串，有待发现其组成部分。

ChPos

Character Position

开始字符位置。如果位于字符串以外，则产生运行时错误。

Len

长度

数据类型：num

字符串组成部分的长度。如果长度为负或大于字符串的长度，或者如果子串（部分地）位于字符串之外，则会产生运行时错误。

语法

```
StrPart '('
[ Str ':=' ] <expression (IN) of string> ','
[ ChPos ':=' ] <expression (IN) of num> ','
[ Len ':=' ] <expression (IN) of num> ')'
```

返回值的数据类型是 string 的函数。

相关信息

信息，关于	请参阅
字符串功能	技术参考手册 - *RAPID*语言概览
字符串的定义	第*1489*页的*string - 字符串*

下一页继续

信息，关于	请参阅
字符串值	技术参考手册 - *RAPID*语言概览

2.172 StrToByte - 将一段字符串转换为一个字节数据

手册用法

StrToByte （*String To Byte*）用于通过规定的字节数据格式，将一个字符串转换为一个字节数据。

基本示例

以下示例介绍了函数StrToByte。

例 1

```
VAR string con_data_buffer{5} := ["10", "AE", "176", "00001010",
     "A"];
VAR byte data_buffer{5};
data_buffer{1} := StrToByte(con_data_buffer{1});
```

在StrToByte ...函数后，数组分量data_buffer{1}的容量将为10个小数。

```
data_buffer{2} := StrToByte(con_data_buffer{2}\Hex);
```

在StrToByte ...函数后，数组分量data_buffer{2}的容量将为174个小数。

```
data_buffer{3} := StrToByte(con_data_buffer{3}\Okt);
```

在StrToByte ... 函数后，数组分量 data_buffer{3}的容量将为126个小数。

```
data_buffer{4} := StrToByte(con_data_buffer{4}\Bin);
```

在StrToByte ... 函数后，数组分量data_buffer{4}的容量将为10个小数。

```
data_buffer{5} := StrToByte(con_data_buffer{5}\Char);
```

在StrToByte ...函数后，数组分量data_buffer{5}的容量将为65个小数。

返回值

数据类型：byte

换算运算的结果，以小数表示。

变元

```
StrToByte (ConStr [\Hex] | [\Okt] | [\Bin] | [\Char])
```

ConStr

Convert String

数据类型：string

有待转换的字符串数据。

如果省略可选开关参数，则有待转换的字符串具有decimal（Dec）格式。

[\Hex]

Hexadecimal

数据类型：switch

有待转换的字符串具有hexadecimal格式。

[\Okt]

Octal

数据类型：switch

有待转换的字符串具有 octal格式。

下一页继续

[\Bin]

Binary

数据类型：switch

有待转换的字符串具有binary格式。

[\Char]

Character

数据类型：switch

有待转换的字符串具有ASCII字符格式。

限制

根据有待转换的字符串的格式，以下字符串数据有效：

格式	字符串长度	范围
Dec ..... : ‘0’ - ‘9’	3	‘0’-‘255’
Hex ..... : ‘0’ - ‘9’, ‘a’ -‘f’, ‘A’ - ‘F’	2	“0” - “FF”
Okt ..... : ‘0’ - ‘7’	3	“0” - “377”
Bin ..... : ‘0’ - ‘1’	8	“0” - “11111111”
Char .... : 任意ASCII字符	1	一个ASCII字符

RAPID字符代码（例如，有关BEL控制字符的“\07”）可用作ConStr中的参数。

语法

```
StrToByte '('
  [ConStr ':=' ] <expression (IN) of string>
  ['\' Hex ] | ['\' Okt] | ['\' Bin] | ['\' Char]
  ')'
```

含数据类型byte的返回值的函数。

相关信息

信息，关于	请参阅
将一个字节转换为一段字符串数据	第*1002*页的*ByteToStr - 将字节转换为字符串数据*
其他位（字节）函数	技术参考手册 - *RAPID*语言概览
其他字符串函数	技术参考手册 - *RAPID*语言概览

2.173 StrToVal - 将一段字符串转换为一个值

手册用法

StrToVal（*String To Value*）用于将一段字符串转换为任意数据类型的一个值。

基本示例

以下示例介绍了函数StrToVal。

另请参阅第*1264*页的更多示例

例 1

```
VAR bool ok;
VAR num nval;
ok := StrToVal("3.85",nval);
```

假定变量ok的值为TRUE，并假定nval的值为3.85。

返回值

数据类型：bool

如果所需转换成功，则TRUE，否则，FALSE。

变元

```
StrToVal ( Str Val )
```

Str

String

数据类型：string

一个包含文字数据的字符串值，其格式符合参数Val中使用的数据类型。有关RAPID文字总量的有效格式。

Val

Value

数据类型：ANYTYPE

用于储存转换结果的任意数据类型的变量或永久变量的名称。

原子结构、记录、记录分量、数组或数组元素均可使用的各类值数据。因为格式不符合参数Str中使用的数据，因此，如果所需转换失败，则数据不会发生改变。

更多示例

有关于函数StrToVal的更多例子阐述如下。

例 1

```
VAR string str15 := "[600, 500, 225.3]";
VAR bool ok;
VAR pos pos15;

ok := StrToVal(str15,pos15);
```

假定变量ok的值为TRUE，并假定变量pos15的值为字符串str15中的规定值。

下一页继续

语法

```
StrToVal '('
  [ Str ':=' ] <expression (IN) of string> ','
  [ Val ':=' ] <var or pers (INOUT) of ANYTYPE>
  ')'
```

含数据类型bool的返回值的函数。

相关信息

信息，关于	请参阅
字符串功能	技术参考手册 - *RAPID*语言概览
字符串的定义	第*1489*页的*string - 字符串*
字符串值	技术参考手册 - *RAPID*语言概览

2.174 Tan - 计算正切值

手册用法

Tan（*Tangent*）用于计算一个角值的正切值。

基本示例

以下示例介绍了函数Tan。

例 1

```
VAR num angle;
VAR num value;
...
...
value := Tan(angle);
```

value将获得angle的正切值。

返回值

数据类型：num

正切值。

变元

```
Tan (Angle)
```

Angle

数据类型：num

角度值，以度表示。

语法

```
Tan '('
  [Angle ':='] <expression (IN) of num>
  ')'
```

返回值的数据类型是 num 的函数。

相关信息

信息，关于	请参阅
数学指令和函数	技术参考手册 - *RAPID*语言概览
包含范围【-180，180】中返回值的反正切	第*972*页的*ATan2* - 计算反正切*2*的值

2.175 TanDnum - 计算正切值

手册用法

TanDnum（*Tangent*）用于根据有关数据类型dnum的角度值，计算正切值。

基本示例

以下示例介绍了函数TanDnum。

例 1

```
VAR dnum angle;
VAR dnum value;
...
...
value := TanDnum(angle);
```

value将获得angle的正切值。

返回值

数据类型：dnum

正切值。

变元

```
TanDnum (Angle)
```

Angle

数据类型：dnum

角度值，以度表示。

语法

```
TanDnum '('
  [Angle ':='] <expression (IN) of dnum>
  ')'
```

含数据类型dnum的返回值的函数。

相关信息

信息，关于	请参阅
数学指令和函数	技术参考手册 - *RAPID*语言概览
包含范围【-180，180】中返回值的反正切	第*973*页的*ATan2Dnum - 计算反正切2*的值

2.176 TaskRunMec - 检查任务是否控制所有机械单元

手册用法

TaskRunMec用于检查程序任务是否控制所有机械单元（包含TCP的机械臂或不含TCP的机械臂）。

基本示例

以下示例介绍了函数TaskRunMec。

例 1

```
VAR bool flag;
...
flag := TaskRunMec( );
```

如果当前任务控制所有机械单元，则flag将为TRUE，否则，FALSE。

返回值

数据类型：bool

如果当前任务控制所有机械单元，则返回值将为TRUE，否则，FALSE。

程序执行

检查当前程序任务是否控制所有机械单元。

语法

```
TaskRunMec '(' ')'
```

含数据类型bool的返回值的函数。

相关信息

信息，关于	请参阅
检查任务是否控制一些机械臂	第*1269*页的*TaskRunRob - 检查任务是否控制一些机械臂*
启用和停用机械单元	第*24*页的*ActUnit - 启用机械单元* 第*140*页的*DeactUnit - 停用机械单元*
机械单元的配置	技术参考手册 - 系统参数

2.177 TaskRunRob - 检查任务是否控制一些机械臂

手册用法

TaskRunRob用于检查程序任务是否控制一些机械臂（包含TCP的机械单元）。

基本示例

以下示例介绍了函数TaskRunRob。

例 1

```
VAR bool flag;
...
flag := TaskRunRob( );
```

如果当前任务控制一些机械臂，则将flag设置为TRUE，否则，FALSE。

返回值

数据类型：bool

如果当前任务控制一些机械臂，则返回值将为TRUE，否则，FALSE。

程序执行

检查当前程序任务是否控制一些机械臂。

语法

```
TaskRunRob '(' ')'
```

含数据类型bool的返回值的函数。

相关信息

信息，关于	请参阅
检查任务是否控制所有机械单元	第*1268*页的*TaskRunMec* - 检查任务是否控制所有机械单元
启用和停用机械单元	第*24*页的*ActUnit* - 启用机械单元 第*140*页的*DeactUnit* - 停用机械单元
机械单元的配置	技术参考手册 - 系统参数

2.178 TasksInSync - 返回同步任务量

手册用法

TasksInSync用于检索同步任务量。

基本示例

以下示例介绍了函数TaskInSync。

例 1

```
VAR tasks tasksInSyncList{6};
...

PROC main ()
  VAR num noOfSynchTasks;
  ...
  noOfSynchTasks:= TasksInSync (tasksInSyncList);
  TPWrite "No of synchronized tasks = "\Num:=noOfSynchTasks;
ENDPROC
```

向变量noOfSynchTasks分配同步任务量，且tasksInSyncList将包含同步任务的名称。在本例子中，任务列表为一个变量，但是其亦可以是一个永久数据对象。

返回值

数据类型：num

同步任务量。

变元

```
TaskInSync (TaskList)
```

TaskList

数据类型：tasks

任务列表（数组）中的输入输出参数将表明已同步程序任务的名称（string）。任务列表可以为VAR或PERS型。

程序执行

函数返回系统中的同步任务量。在输入输出参数任务列表中表明同步任务的名称。如果不存在任何同步任务，则列表将仅包含空字符串。

限制

当前，仅支持一个同步组，因此，TasksInSync返回该组中同步的任务量。

语法

```
TasksInSync
  [ TaskList ':=' ] < var or pers array {*} (INOUT) of tasks> ','
```

返回值的数据类型是 num 的函数。

下一页继续

相关信息

信息，关于	请参阅
指定协作的程序任务	第*1500*页的*tasks - RAPID*程序任务
起动协调同步移动	第*709*页的*SyncMoveOn* - 起动协调同步移动

2.179 TestAndSet - 未设置时，测试变量并予以设置

手册用法

TestAndSet可以与bool型普通数据对象一同使用，其作为一个二进制信号，用以检索特定RAPID代码区域或系统资源的专有权。函数可以在相同程序任务内的不同程序任务与不同执行等级（TRAP或事件程序）之间使用。

同时需要访问保护的资源实例：

- 并行执行时，一些RAPID程序的使用会产生函数问题。
- FlexPendant示教器的使用-运算符日志

基本示例

以下示例介绍了函数TestAndSet。

另请参阅第*1273*页的更多示例

例 1

MAIN程序任务：

```
PERS bool tproutine_inuse := FALSE;
...
WaitUntil TestAndSet(tproutine_inuse);
TPWrite "First line from MAIN";
TPWrite "Second line from MAIN";
TPWrite "Third line from MAIN";
tproutine_inuse := FALSE;
```

BACK1程序任务：

```
PERS bool tproutine_inuse := FALSE;
...
WaitUntil TestAndSet(tproutine_inuse);
TPWrite "First line from BACK1";
TPWrite" Second line from BACK1";
TPWrite "Third line from BACK1";
tproutine_inuse := FALSE;
```

为避免运算符日志中的各行相混，一行来自MAIN，一行来自BACK1，TestAndSet函数的使用可保证各任务的全部三行均不会分离。

如果程序任务MAIN首先采用信号TestAndSet(tproutine_inuse)，则程序任务BACK1必须等待，直至程序任务MAIN已留下信号。

返回值

数据类型：bool

如果已由我（TestAndSet函数的执行者）取得信号，则TRUE，否则，FALSE。

变元

```
TestAndSet Object
```

Object

数据类型：bool

下一页继续

将用作信号的用户定义数据对象。数据对象可为一个变量VAR或一个永久变量PERS。如果在不同的程序任务之间使用TestAndSet，则对象必须为一个永久变量PERS或一个安装变量VAR（内部任务对象）。

程序执行

本函数将在一个不可分割的步骤中，检查用户确定的变量，且如果未予以设置，则进行设置，并返回TRUE，否则，其将返回FALSE。

```
IF Object = FALSE THEN
  Object := TRUE;
  RETURN TRUE;
ELSE
  RETURN FALSE;
ENDIF
```

更多示例

以下示例介绍了函数TestAndSet。

例 1

```
LOCAL VAR bool doit_inuse := FALSE;
...
PROC doit(...)
  WaitUntil TestAndSet (doit_inuse);
  ...
  doit_inuse := FALSE;
ENDPROC
```

如果一个模块内置并共享，则有可能使用一个局部模块变量，以保护不同程序任务同时进行访问。

注意

在这种情况下，通过已安装的内置模块，且将永久变量用作信号对象时：如果在程序doit中停止程序执行，且程序指针移动至main，则将不会重置变量doit_inuse。为避免这种情况，重置变量doit_inuse为START事件程序中的FALSE。

语法

```
TestAndSet '('
  [ Object ':=' ] < variable or persistent (INOUT) of bool> ')'
```

含数据类型bool的返回值的函数。

相关信息

信息，关于	请参阅
等待，直至变量复原-随后进行设置（等待中断控制的类型）	第893页的*WaitTestAndSet - 等待，直至变量FALSE，随后设置*

2.180 TestDI - 测试有没有设置数字信号输入

手册用法

TestDI用于测试是否已设置数字信号输入。

基本示例

以下示例介绍了函数TestDI。

例 1

```
IF TestDI (di2) THEN . . .
```

如果信号di2的当前值等于1，则...

```
IF NOT TestDI (di2) THEN . . .
```

如果信号di2的当前值等于0，则...

```
WaitUntil TestDI(di1) AND TestDI(di2);
```

仅在已完成di1输入和di2输入后，方才继续程序执行。

返回值

数据类型：bool

TRUE = 信号的当前值等于1。

FALSE = 信号的当前值等于0。

变元

```
TestDI (Signal)
```

Signal

数据类型：signaldi

待测试信号的名称。

错误处理

系统会生成下列可恢复错误，并在错误处理器中处理这些错误。系统变量ERRNO将被设置成：

如果信号变量是RAPID中声明的变量，则ERR_NO_ALIASIO_DEF。尚未同I/O配置以及指令AliasIO中确定的I/O信号相连。

如果与I/O单元无接触，则ERR_NORUNUNIT。

如果无法访问I/O信号（仅对ICI现场总线有效），则ERR_SIG_NOT_VALID。

语法

```
TestDI '('
  [ Signal ':=' ] < variable (VAR) of signaldi > ')'
```

含数据类型bool的返回值的函数。

相关信息

信息，关于	请参阅
读取数字信号输入信号值	第*1476*页的*signalxx - 数字信号和模拟信号*
读取数字信号输出信号值	第*1067*页的*DOutput - 读取数字信号输出信号值*

下一页继续

信息，关于	请参阅
输入/输出指令	技术参考手册 - *RAPID*语言概览

2.181 TestSignRead - 读取测试信号值

手册用法

TestSignRead用于读取实际测试信号值。

根据指令TestSignDefine中的通道规格，该函数返回最后样本的瞬时值或平均值。

基本示例

以下示例介绍了函数TestSignRead。

另请参阅第*1276*页的更多示例

例 1

```
CONST num speed_channel:=1;
VAR num speed_value;
...
TestSignDefine speed_channel, speed, orbit, 1, 0;
...
! During some movements with orbit’s axis 1
speed_value := TestSignRead(speed_channel);
...
TestSignReset;
```

向speed_value 分配最新8个样本的平均值。在定义为通道1的通道speed_channel上，每0.5 ms的测试信号speed均会产生此类样本。通道speed_channel测量机械单元orbit上的axis 1速度。

返回值

数据类型：num

根据指令TestSignDefine中的定义，有关指定通道电机侧SI单元中的数值。

变元

```
TestSignRead (Channel)
```

Channel

数据类型：num

有待读取的测试信号的通道编号1-12。必须在定义指令TestSignDefine中使用相同的编号。

程序执行

根据指令TestSignDefine中的通道规格，返回最后样本的瞬时值或平均值。

有关预定义测试信号以及外机械臂轴的有效SI单元，请参见数据类型testsignal。

更多示例

以下示例介绍了函数TestSignRead。

例 1

```
CONST num torque_channel:=2;
VAR num torque_value;
VAR intnum timer_int;
CONST jointtarget psync := [...];
```

下一页继续

```
...
PROC main()
  CONNECT timer_int WITH TorqueTrap;
  ITimer \Single, 0.05, timer_int;
  TestSignDefine torque_channel, torque_ref, IRBP_K, 2, 0.001;
  ...
  MoveAbsJ psync \NoEOffs, v5, fine, tool0;
  ...
  IDelete timer_int;
  TestSignReset;

TRAP TorqueTrap
  IF (TestSignRead(torque_channel) > 6) THEN
    TPWrite "Torque pos = " + ValToStr(CJointT());
    Stop;
  ELSE
    IDelete timer_int;
    CONNECT timer_int WITH TorqueTrap;
    ITimer \Single, 0.05, timer_int;
  ENDIF
ENDTRAP
```

缓慢移动至位置psync期间，当机械臂IRBP_K轴2的扭矩参考量首次大于电机侧的6 Nm时，接头位置得以在FlexPendant示教器上显示。

语法

```
TestSignRead '('
  [ Channel ':='] <expression (IN) of num> ')'
```

含num型返回值的函数。

相关信息

信息，关于	请参阅
测试信号	第*1501*页的*testsignal - 测试信号*
定义测试信号	第*727*页的*TestSignDefine - 定义测试信号*
重置测试信号	第*729*页的*TestSignReset - 重置所有测试信号定义*

2.182 TextGet - 从系统文本表格中获取文本

手册用法

TextGet的作用是是从系统文件表格中获取一段文本字符串。

基本示例

以下示例介绍了函数TextGet。

例 1

```
VAR string text1;
...
text1 := TextGet(14, 5);
```

向变量text1分配存储于文本资源14和索引5中的文本。

返回值

数据类型：string

指定来自系统文本表格的文本。

变元

```
TextGet ( Table Index )
```

Table

数据类型：num

文本表格编号（正整数）。

Index

数据类型：num

文本表格内的索引编号（正整数）。

错误处理

如果表格或索引无效，且无法从系统文本表格取得任何文本字符串，则将系统变量ERRNO设置为ERR_TXTNOEXIST。通过错误处理器继续执行。

语法

```
TextGet '('
  [ Table ':=' ] < expression (IN) of num > ','
  [ Index ':=' ] < expression (IN) of num> ')'
```

返回值的数据类型是 string 的函数。

相关信息

信息，关于	请参阅
获取文本表格编号	第*1282*页的*TextTabGet - 获取文本表格编号*
安装文本表格	第*730*页的*TextTabInstall - 安装文本表格*
格式化文本文件	技术参考手册 - *RAPID*语言内核
字符串功能	技术参考手册 - *RAPID*语言概览
字符串的定义	第*1489*页的*string - 字符串*

下一页继续

信息，关于	请参阅
字符串值	技术参考手册 - *RAPID*语言概览
Advanced RAPID	产品规格 - 控制器软件*IRC5*

2.183 TextTabFreeToUse - 测试文本表格是否已解除

手册用法

TextTabFreeToUse应当用于测试是否能够自由使用（尚未在系统中安装）文本表格名称（文本资源字符串），即是否可能在系统中安装文本表格。

基本示例

以下示例介绍了函数TextTabFreeToUse。

例 1

```
! System Module with Event Routine to be executed at event
! POWER ON, RESET or START

PROC install_text()
  IF TextTabFreeToUse("text_table_name") THEN
    TextTabInstall "HOME:/text_file.eng";
  ENDIF
ENDPROC
```

首次执行事件程序install_text时，函数TextTabFreeToUse返回TRUE，并在系统中安装文本文件text_file.eng。此后，可通过函数TextTabGet和TextGet，从RAPID系统获取已安装的文本字符串。

下一次执行事件程序install_text时，函数TextTabFreeToUse返回FALSE，且不重复安装。

返回值

数据类型：bool

该函数返回：

- 如果文本表格尚未在系统中安装，则TRUE。
- 如果文本表格已在系统中安装，则FALSE。

变元

```
TextTabFreeToUse ( TableName )
```

TableName

数据类型：string

文本表格名称（一段最多80个字符的字符串）。参考*RAPID*参考手册 - *RAPID*核心，文本文件一节中的<text_resource>。字符串text_resource为文本表格名称。

限制

关于在系统中安装文本表格（文本资源）的限制：

- 不可能在系统中多次安装同一文本表格
- 不可能从系统中卸载（解除）单一文本表格。从系统卸载文本表格的唯一方式是通过使用重启模式重置系统，以此来重启控制器。随后，将卸载所有文本表格（包括系统和用户定义的文本表格）。

下一页继续

语法

```
TextTabFreeToUse '('
  [ TableName ':=' ] < expression (IN) of string > ')'
```

含数据类型bool的返回值的函数。

相关信息

信息，关于	请参阅
安装文本表格	第730页的*TextTabInstall - 安装文本表格*
文本文件的格式	技术参考手册 - *RAPID*语言内核
获取文本表格编号	第1282页的*TextTabGet - 获取文本表格编号*
从系统文本表格中获取文本	第1278页的*TextGet - 从系统文本表格中获取文本*
字符串功能	技术参考手册 - *RAPID*语言概览
字符串的定义	第1489页的*string - 字符串*
Advanced RAPID	产品规格 - 控制器软件*IRC5*

2.184 TextTabGet - 获取文本表格编号

手册用法

TextTabGet的作用是获取由用户在运行时期间定义的文本表格的文本表格编号。

基本示例

以下示例介绍了函数TextTabGet。

针对用户定义的文本，上述例子均使用一个名为deburr_part1的新文本表格。该新文本表格拥有文件名deburr.eng。

```
# deburr.eng - USERS deburr_part1 english text description file
#
# DESCRIPTION:
# Users text file for RAPID development
#
deburr_part1::
0:
RAPID S4: Users text table deburring part1
1:
Part 1 is not in pos
2:
Identity of worked part: XYZ
3:
Part error in line 1
#
# End of file
```

例 1

```
VAR num text_res_no;
...
text_res_no := TextTabGet("deburr_part1");
```

向变量text_res_no分配有关已确定文本表格deburr_part1的文本表格编号。

例 2

```
ErrWrite TextGet(text_res_no, 1), TextGet(text_res_no, 2);
```

将消息储存在机械臂日志中。亦使该消息在FlexPendant示教器显示器上显示。将从文本表格deburr_part1取得该消息：

Part 1 is not in pos

Identity of worked part: XYZ

返回值

数据类型：num

已确定文本表格的文本表格编号。

变元

```
TextTabGet ( TableName )
```

TableName

数据类型：string

下一页继续

文本表格名称。

语法

```
TextTabGet '('
  [ TableName '=' ] < expression (IN) of string > ';)'
```

返回值的数据类型是 num 的函数。

相关信息

信息，关于	请参阅
从系统文本表格中获取文本	第*1278*页的*TextGet - 从系统文本表格中获取文本*
安装文本表格	第*730*页的*TextTabInstall - 安装文本表格*
格式化文本文件	技术参考手册 - *RAPID*语言内核
字符串功能	技术参考手册 - *RAPID*语言概览
字符串的定义	第*1489*页的*string - 字符串*
字符串值	技术参考手册 - *RAPID*语言概览
Advanced RAPID	产品规格 - 控制器软件*IRC5*

2.185 TriggDataValid - 检查触发数据变量中的内容是否有效

手册用法

TriggDataValid函数用于检查triggdata变量是否有效。有效的triggdata变量为先前已在程序的指令TriggIO、TriggEquip、TriggInt、TriggSpeed、TriggCheckIO或TriggRampAO之一中使用过的变量，从而规定触发条件和触发活动。

基本示例

以下示例介绍了函数TriggDataValid。

例 1

```
VAR triggdata trigg_array{25};
...
PROC MyTriggProcL(robtarget myrobt, \VAR triggdata T1 \VAR triggdata
     T2 \VAR triggdata T3)
  VAR num triggcnt:=2;
  ! Reset entire trigg_array array before using it
  FOR i FROM 1 TO 25 DO
    TriggDataReset trigg_array{i};
  ENDFOR
  TriggEquip trigg_array{1}, 10 \Start, 0 \DOp:=do1, SetValue:=1;
  TriggEquip trigg_array{2}, 40 \Start, 0 \DOp:=do2, SetValue:=1;
  ! Check if optional argument is present,
  ! and if any trigger condition has been setup in T1
  IF Present(T1) AND TriggDataValid(T1) THEN
    ! Copy actual trigger condition to trigg_array
    TriggDataCopy trigg_array{triggcnt}, T1;
    Incr triggcnt;
  ENDIF
  IF Present(T2) AND TriggDataValid(T2) THEN
    Incr triggcnt;
    TriggDataCopy trigg_array{triggcnt}, T2;
  ENDIF
  IF Present(T3) AND TriggDataValid(T3) THEN
    Incr triggcnt;
    TriggDataCopy trigg_array{triggcnt}, T3;
  ENDIF
  TriggL p1, v500, trigg_array, z30, tool2;
  ...
```

上述无返回值程序MyTriggProcL使用TriggDataValid指令，以检查是否在可选参数T1、T2和T3中使用了任何有效数据。

返回值

数据类型：bool

下一页继续

该函数返回：

- 如果变量有效，即指令TriggIO、TriggEquip、TriggInt、TriggSpeed、TriggCheckIO或TriggRampAO之一已用于指定触发条件和触发活动，则TRUE。
- 当设置任何触发条件和触发活动时，如果尚未使用变量，则FALSE。

变元

```
TriggDataValid TriggData
```

TriggData

数据类型：triggdata

用于检查是否有效的triggdata变量。

语法

```
TriggDataValid
  [TriggData ':=' ] < variable (VAR) of triggdata > ';'
```

含数据类型bool的返回值的函数。

相关信息

信息，关于	请参阅
触发时的线性移动	第*785*页的*TriggL* - 关于事件的机械臂线性运动
触发时的接头移动	第*778*页的*TriggJ* - 关于事件的轴式机械臂运动
触发时的圆周移动	第*747*页的*TriggC* - 关于事件的机械臂圆周移动
触发的定义	第*773*页的*TriggIO* - 定义停止点附近的固定位置或时间*I/O*事件 第*763*页的*TriggEquip* - 定义路径上的固定位置和时间*I/O*事件 第*769*页的*TriggInt* - 定义与位置相关的中断 第*754*页的*TriggCheckIO* - 定义位于固定位置的*IO*检查 第*804*页的*TriggRampAO* - 定义路径上的固定位置斜坡*AO*事件 第*810*页的*TriggSpeed* - 定义与固定位置-时间尺度事件成比例的*TCP*速度模拟信号输出
处理triggdata	第*1510*页的触发数据 - 定位事件，触发 第*761*页的*TriggDataReset* - 重置触发数据变量中的内容 第*759*页的*TriggDataCopy* - 复制触发数据变量中的内容

2.186 Trunc - 截断一个数值

手册用法

Trunc（*Truncate*）用于截断一个数值至规定位数的小数或整数值。

基本示例

以下示例介绍了函数Trunc。

例 1

```
VAR num val;
val := Trunc(0.3852138\Dec:=3);
```

变量val被赋予值0.385。

例 2

```
reg1 := 0.3852138
val := Trunc(reg1\Dec:=1);
```

变量val被赋予值0.3。

例 3

```
val := Trunc(0.3852138);
```

变量val被赋予值0。

例 4

```
val := Trunc(0.3852138\Dec:=6);
```

变量val被赋予值0.385213。

返回值

数据类型：num

将数值截断至规定位数的小数。

变元

```
Trunc ( Val [\Dec] )
```

Val

Value

数据类型：num

有待截断的数值。

[\Dec]

Decimals

数据类型：num

小数位数。

如果指定的小数位数为0，或者如果省略参数，则将值截断为一个整数。

小数位数不得为负或大于数值的可用精度。

可以使用的最多小数位数为6。

下一页继续

语法

```
Trunc '('
  [ Val ':=' ] <expression (IN) of num>
  [ \Dec ':=' <expression (IN) of num> ] ')'
```

返回值的数据类型是 num 的函数。

相关信息

信息，关于	请参阅
数学指令和函数	技术参考手册 - *RAPID*语言概览
将一个值舍入	第*1220*页的*Round* - 按四舍五入计算数值

2.187 TruncDnum - 截断一个数值

手册用法

TruncDnum（*Truncate dnum*）用于截断一个数值至规定位数的小数或整数值。

基本示例

以下示例介绍了函数TruncDnum。

例 1

```
VAR dnum val;
val := TruncDnum(0.3852138754655357\Dec:=3);
```

变量val被赋予值0.385。

例 2

```
val := TruncDnum(0.3852138754655357\Dec:=1);
```

变量val被赋予值0.3。

例 3

```
val := TruncDnum(0.3852138754655357);
```

变量val被赋予值0。

例 4

```
val := TruncDnum(0.3852138754655357\Dec:=15);
```

变量val被赋予值0.385213875465535。

例 5

```
val := TruncDnum(1000.3852138754655357\Dec:=15);
```

变量val被赋予值1000.38521387547。

返回值

数据类型：dnum

将数值截断至规定位数的小数。

变元

```
TruncDnum ( Val [\Dec] )
```

Val

Value

数据类型：dnum

有待截断的数值。

[\Dec]

Decimals

数据类型：num

小数位数。

如果指定的小数位数为0，或者如果省略参数，则将值截断为一个整数。

小数位数不得为负或大于数值的可用精度。

可以使用的最多小数位数为15。

下一页继续

语法

```
TruncDnum '('
  [ Val ':=' ] <expression (IN) of dnum>
  [ \Dec ':=' <expression (IN) of num> ] ')'
```

含数据类型dnum的返回值的函数。

相关信息

信息，关于	请参阅
数学指令和函数	技术参考手册 - *RAPID*语言概览
截断一个值	第*1286*页的*Trunc - 截断一个数值*
将一个值舍入	第*1220*页的*Round - 按四舍五入计算数值*
将一个值舍入	第*1222*页的*RoundDnum - 按四舍五入计算数值*

2.188 Type - 获取有关一个变量的数据类型名称

手册用法

Type用于获取有关参数Data中指定变量的数据类型名称。

基本示例

以下示例介绍了函数Type。

例 1

```
VAR string rettype;
VAR intnum intnumtype;
...
PROC main()
  rettype := Type(intnumtype);
  TPWrite "Data type name: " + rettype;
```

打印出的内容将为：**"Data type name: intnum"**

例 2

```
VAR string rettype;
VAR intnum intnumtype;
...
PROC main()
  rettype := Type(intnumtype \BaseName);
  TPWrite "Data type name: " + rettype;
```

打印出的内容将为：**"Data type name: num"**

例 3

```
VAR string rettype;
VAR num numtype;
...
PROC main()
  rettype := Type(numtype);
  TPWrite "Data type name: " + rettype;
```

打印出的内容将为：**"Data type name: num"**

返回值

数据类型：string

一个包含参数Data中指定变量的数据类型名称的字符串。

变元

```
Type (Data [\BaseName])
```

Data

Data object name

数据类型：anytype

有待获取数据类型名称的变量的名称。

[\BaseName]

Base data type Name

下一页继续

数据类型：switch

当指定的Data为一个ALIAS声明数据类型时，如果使用，则函数返回潜在的数据类型名称。

语法

```
Type '('
  [ Data ':=' ] < reference (REF) of anytype >
  [ '\' BaseName ] ')'
```

返回值的数据类型是 string 的函数。

相关信息

信息，关于	请参阅
Alias类型的定义	技术参考手册 - *RAPID*语言内核 第*1344*页的*ALIAS - 分配一种别名数据类型*

2.189 UIAlphaEntry - 用户α条目

手册用法

UIAlphaEntry（*User Interaction Alpha Entry*）用于让操作员通过可用的用户设备，例如FlexPendant示教器，输入一段字符串。将一条消息写给操作员，其以一段文本字符串作为应答。随后，将字符串转换回程序。

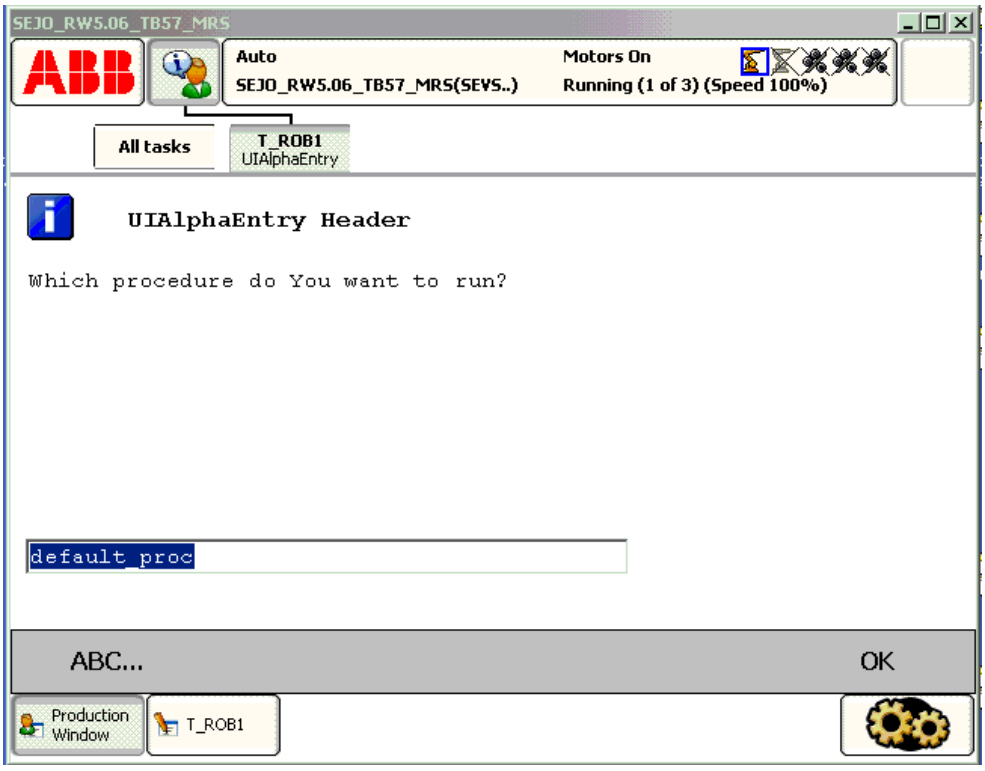
基本示例

以下示例介绍了函数UIAlphaEntry。

请参阅 第*1295*页的更多示例。

例 1

```
VAR string answer;
...
answer := UIAlphaEntry(
  \Header:= "UIAlphaEntry Header",
  \Message:= "Which procedure do You want to run?"
  \Icon:=iconInfo
  \InitString:= "default_proc");
%answer%;
```



xx0500002437

在上文中，阿尔法消息框以及图标、标题、消息和初始化字符串得以写入FlexPendant示教器显示器。用户通过起支持作用的阿尔法平板电脑，编辑初始化字符串或写入一段新的字符串。程序执行进入等待，直至按下OK，随后，通过变量answer，返回写入的字符串。随后，程序通过后期绑定，调用指定无返回值程序。

下一页继续

返回值

数据类型：string

该函数返回输入字符串。

如果函数通过\BreakFlag:中断

- 如果指定参数\InitString，则返回该字符串
- 如果未指定参数\InitString，则返回空字符串""。

如果函数通过ERROR处理器而中断，则将不会返回任何返回值。

变元

```
UIAlphaEntry ([\Header] [\Message]|[\MsgArray]
     [\Wrap][\Icon][\InitString] [\MaxTime] [\DIBreak] [\DIPassive]
     [\DOBreak] [\DOPassive] [\BreakFlag])
```

[\Header]

数据类型：string

将在消息框顶部写入的标题文本。最多40个字符。

[\Message]

数据类型：string

有待在显示器上写入的一个文本行。最多55个字符。

[\MsgArray]

Message Array

数据类型：string

Several text lines from an array to be written on the display.

仅可在同一时间使用参数\Message 或\MsgArray之一。

最大布置间距为9行，每行包含55个字符。

[\Wrap]

数据类型：switch

如果选择，则参数\MsgArray中的所有指定字符串均将连接到一个字符串，各单独字符串之间仅存在单间距，并以尽可能少的行展开。

默认将在显示器上单独的行中显示参数\MsgArray中的各字符串。

[\Icon]

数据类型：icondata

定义有待显示的图标。Only one of the predefined icons of type icondata can be used.参见第*1295*页的预定义数据。

默认没有图标。

[\InitString]

数据类型：string

默认在文本输入框中显示的一段初始字符串。

[\MaxTime]

数据类型：num

下一页继续

2.189 UIAlphaEntry - 用户α条目
RobotWare-OS
续前页

程序执行等待的最长时间，以秒计。如果未在该时间内按下OK按钮，则用错误处理器继续执行程序，除非使用BreakFlag （参见下文）。常量ERR_TP_MAXTIME 可用于测试是否已经过最长时间。

[\DIBreak]

Digital Input Break

数据类型：signaldi

可能中断运算符对话框的数字信号输入信号。如果在将信号设置为1（或已经为1）之前未按下OK按钮，则用错误处理器继续执行程序，除非使用BreakFlag（参见下文）。可使用常量ERR_TP_DIBREAK来测试是否已出现该情况。

[\DIPassive]

Digital Input Passive

数据类型：switch

当使用DIBreak可选参数时，该开关会覆盖默认行为。将信号设置为1（或已经为1）时，取代反应；将信号DIBreak设置为0（或已经为0）时，应当用错误处理器继续本指令（如果未使用BreakFlag）。常量ERR_TP_DIBREAK可用于测试是否已出现上述情况。

[\DOBreak]

Digital Output Break

数据类型：signaldo

可能中断运算符对话框的数字信号输出信号。如果在将信号设置为1（或已经为1）之前未按下OK按钮，则用错误处理器继续执行程序，除非使用 BreakFlag（参见下文）。可使用常量ERR_TP_DOBREAK 来测试是否已出现该情况。

[\DOPassive]

Digital Output Passive

数据类型：switch

当使用DOBreak可选参数时，该开关会覆盖默认行为。将信号设置为1（或已经为1）时，取代反应；将信号 DOBreak设置为0（或已经为0）时，应当用错误处理器继续本指令（如果未使用BreakFlag）。常量ERR_TP_DOBREAK可用于测试是否已出现上述情况。

[\BreakFlag]

数据类型：errnum

使用\MaxTime、\DIBreak或\DOBreak时将保存错误代码的变量（使用前，由系统将其设置为0）。常量ERR_TP_MAXTIME, ERR_TP_DIBREAK和ERR_TP_DOBREAK可用于选择原因。如果省略该可选变量，则将执行错误处理器。

程序执行

根据编程参数，显示阿尔法消息框以及阿尔法平板电脑、图标、标题、消息行和初始化字符串。程序执行进入等待，直至用户编辑或创建一段新的字符串，并按下OK，或消息框被超时或信号行动中断。将输入字符串和中断原因转移回程序。

从基础等级的消息框转而关注软中断等级的新消息框。

下一页继续

预定义数据

```
!Icons:
  CONST icondata iconNone := 0;
  CONST icondata iconInfo := 1;
  CONST icondata iconWarning := 2;
  CONST icondata iconError := 3;
```

更多示例

以下示例介绍了函数UIAlphaEntry。

例 1

```
VAR errnum err_var;
VAR string answer;
VAR string logfile;
...
answer := UIAlphaEntry (\Header:= "Log file name:"
  \Message:= "Enter the name of the log file to create?"
  \Icon:=iconInfo
  \InitString:= "signal.log"
  \MaxTime:=60
  \DIBreak:=di5\BreakFlag:=err_var);
TEST err_var
  CASE ERR_TP_MAXTIME:
  CASE ERR_TP_DIBREAK:
    ! No operator answer
    logfile:="signal.log";
  CASE 0:
    ! Operator answer
    logfile := answer;
  DEFAULT:
    ! No such case defined
ENDTEST
```

显示消息框，且操作员可输入一段字符串，并按下OK。消息框亦可因数字信号输入信号超时或中断而中断。在本程序中，发现原因，并采取适当的措施是可能的。

错误处理

如果未使用参数\BreakFlag，则可通过错误处理器来处理此类情形：

如果在运算符输入之前出现超时（参数\MaxTime），则将系统变量ERRNO设置为ERR_TP_MAXTIME，并用错误处理器继续执行。

如果在运算符输入之前设置数字信号输入（参数\DIBreak），则将系统变量ERRNO设置为ERR_TP_DIBREAK，并用错误处理器继续执行。

如果在运算符输入之前设置数字信号输出（参数\DOBreak），则将系统变量ERRNO设置为ERR_TP_DOBREAK，并用错误处理器继续执行。

仅可通过错误处理器来处理该情形：

如果信号变量是在RAPID中声明的变量，且尚未与通过指令AliasIO而在I/O配置中确定的I/O信号相连，则将系统变量ERRNO设置为ERR_NO_ALIASIO_DEF，并用错误处理器继续执行。

下一页继续

2.189 UIAlphaEntry - 用户α条目
RobotWare-OS
续前页

如果没有客户端（例如，FlexPendant示教器）来关注指令，则将系统变量ERRNO设置为ERR_TP_NO_CLIENT，并用错误处理器继续执行。

限制

当频繁地执行UIAlphaEntry时，例如，在一个回路中，避免使用过小的超时参数\MaxTime。其可能导致系统性能出现不可预测的行为，诸如FlexPendant示教器响应放慢。

语法

```
UIAlphaEntry '('
  ['\' Header ':=' <expression (IN) of string>]
  ['\' Message ':=' <expression (IN) of string>]
  | ['\' MsgArray ':='<array {*} (IN) of string>]
  ['\' Wrap]
  ['\' Icon ':=' <expression (IN) of icondata>]
  ['\' InitString ':='<expression (IN) of string>]
  ['\' MaxTime ':=' <expression (IN) of num>]
  ['\' DIBreak ':=' <variable (VAR) of signaldi>]
  ['\' DIPassive]
  ['\' DOBreak ':='<variable (VAR) of signaldo>]
  ['\' DOPassive]
  ['\' BreakFlag ':=' <var or pers (INOUT) of errnum>] ')'
```

含数据类型string的返回值的函数。

相关信息

信息，关于	请参阅
图标显示数据	第*1411*页的*icondata - 图标显示数据*
基础型用户交互消息框	第*832*页的*UIMsgBox - 用户消息对话框，基本类型*
先进型用户交互消息框	第*1317*页的*UIMessageBox - 先进型用户消息框*
用户交互数字条目	第*1324*页的*UINumEntry - 用户数字条目*
用户交互数字调节	第*1330*页的*UINumTune - 用户数字调节*
用户交互列表视图	第*1310*页的*UIListView - 用户列表视图*
同FlexPendant示教器等相连的系统	第*1297*页的*UIClientExist - 现有客户端*
通过后期绑定进行过程调用	技术参考手册 - *RAPID*语言概览
清除运算符窗口	第*732*页的*TPErase - 擦除在FlexPendant示教器上印刷的文本*

2.190 UIClientExist - 现有客户端

手册用法

UIClientExist（*User Interaction Client Exist*）用于检查一些用户设备（诸如FlexPendant示教器）是否与控制器相连。

基本示例

以下示例介绍了函数UIClientExist。

例 1

```
IF UIClientExist() THEN
  ! Possible to get answer from the operator
  ! The TPReadFK and UIMsgBox ... can be used
ELSE
  ! Not possible to communicate with any operator
ENDIF
```

如果可能从系统运算符获得一些答案，则测试完成。

返回值

数据类型：bool

如果一个FlexPendant示教器与系统相连，则返回TRUE，否则，返回FALSE。

限制

在多达16秒的时间里，UIClientExist返回TRUE。此后，移除FlexPendant示教器。随后，UIClientExist返回FALSE（即从FlexPendant示教器探测到网络连接中断时）。再次连接FlexPendant示教器时，存在相同的限制。

语法

```
UIClientExist '(' ')'
```

含bool型返回值的函数。

相关信息

信息，关于	请参阅
基础型用户交互消息框	第*832*页的*UIMsgBox - 用户消息对话框，基本类型*
先进型用户交互消息框	第*1317*页的*UIMessageBox - 先进型用户消息框*
用户交互数字条目	第*1324*页的*UINumEntry - 用户数字条目*
用户交互数字调节	第*1330*页的*UINumTune - 用户数字调节*
用户交互α条目	第*1292*页的*UIAlphaEntry - 用户α条目*
用户交互列表视图	第*1310*页的*UIListView - 用户列表视图*
清除运算符窗口	第*732*页的*TPErase - 擦除在FlexPendant示教器上印刷的文本*

2.191 UIDnumEntry - 用户数字条目

手册用法

UIDnumEntry（*User Interaction Number Entry*）用于让操作员通过可用的用户设备，例如FlexPendant示教器，输入一个数值。将一条消息写给操作员，其以一个数值作为应答。随后，检查、批准该数值，并将其转换回程序。

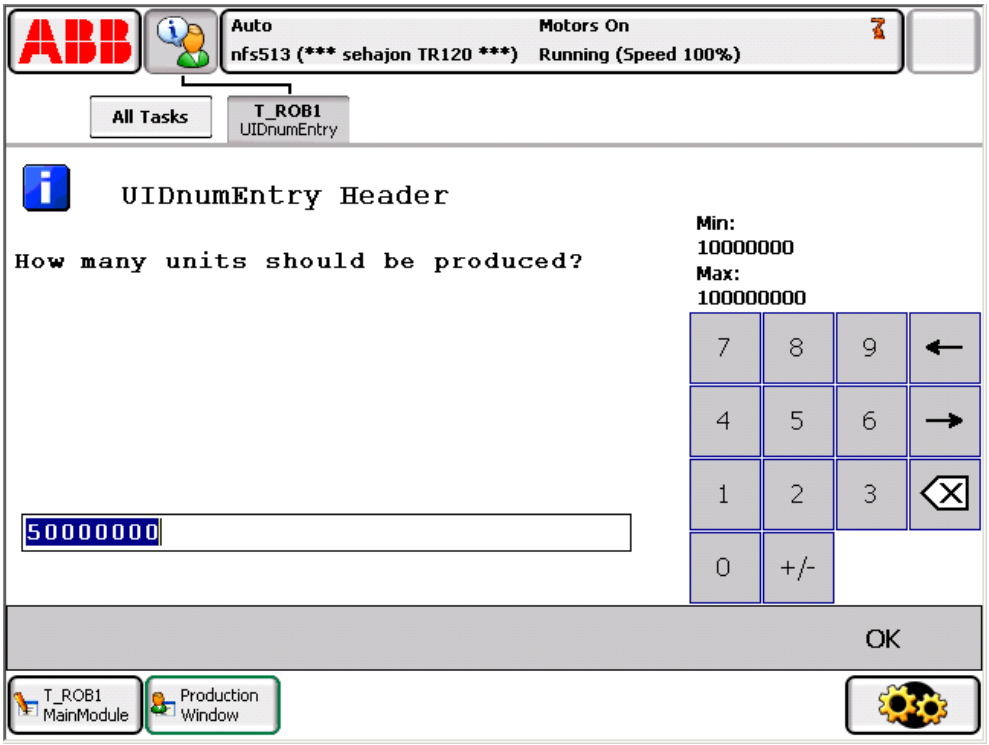
基本示例

以下示例介绍了函数UIDnumEntry。

另请参阅第*1301*页的更多示例

例 1

```
VAR dnum answer;
...
answer := UIDnumEntry(
  \Header:="UIDnumEntry Header"
  \Message:="How many units should be produced?"
  \Icon:=iconInfo
  \InitValue:=50000000
  \MinValue:=10000000
  \MaxValue:=100000000
  \AsInteger);
```



xx0900001064

在上文中，数字消息框以及图标、标题、消息、初始值、最大值和最小值得以写入FlexPendant示教器显示器。消息框检查该运算符是否选择值范围内的一个整数。程序执行进入等待，直至按下OK，随后，返回选定的数值。

下一页继续

返回值

数据类型：dnum

该函数返回输入数值。

如果函数通过\BreakFlag中断：

- 如果指定参数\InitValue，则返回该值
- 如果未指定参数\InitValue，则返回值0。

如果函数通过ERROR处理器而中断，则不存在任何返回值。

变元

```
UIDnumEntry ( [\Header] [\Message] | [\MsgArray]
     [\Wrap][\Icon][\InitValue] [\MinValue] [\MaxValue]
     [\AsInteger][\MaxTime] [\DIBreak] [\DIPassive] [\DOBreak]
     [\DOPassive] \BreakFlag])
```

[\Header]

数据类型：string

将在消息框顶部写入的标题文本。最多40个字符。

[\Message]

数据类型：string

有待在显示器上写入的一个文本行。最多40个字符。

[\MsgArray]

Message Array

数据类型：string

Several text lines from an array to be written on the display.

仅可在同一时间使用参数\Message或\MsgArray之一。

最大布置间距为9行，每行包含40个字符。

[\Wrap]

数据类型：switch

如果选择，则参数\MsgArray中的所有指定字符串均将连接到一个字符串，各单独字符串之间仅存在单间距，并以尽可能少的行展开。

默认将在显示器上单独的行中显示参数\MsgArray中的各字符串。

[\Icon]

数据类型：icondata

定义有待显示的图标。仅可使用一种icondata型预定义图标。参见第*1301*页的预定义数据。

默认没有图标。

[\InitValue]

数据类型：dnum

在输入框中显示的初始值。

[\MinValue]

数据类型：dnum

下一页继续

2.191 UIDnumEntry - 用户数字条目
RobotWare - OS
续前页

有关返回值的最小值。

[\MaxValue]

数据类型：dnum

有关返回值的最大值。

[\AsInteger]

数据类型：switch

消除数字键盘的小数点，以确保返回值为一个整数。

[\MaxTime]

数据类型：num

程序执行等待的最长时间，以秒计。如果未在该时间内按下OK按钮，则用错误处理器继续执行程序，除非使用BreakFlag（参见下文）。常量ERR_TP_MAXTIME可用于测试是否已经过最长时间。

[\DIBreak]

Digital Input Break

数据类型：signaldi

可能中断运算符对话框的数字信号输入信号。如果在将信号设置为1（或已经为1）之前未按下OK按钮，则用错误处理器继续执行程序，除非使用BreakFlag（参见下文）。可使用常量ERR_TP_DIBREAK来测试是否已出现该情况。

[\DIPassive]

Digital Input Passive

数据类型：switch

当使用DIBreak可选参数时，该开关会覆盖默认行为。将信号设置为1（或已经为1）时，取代反应；将信号DIBreak设置为0（或已经为0）时，应当用错误处理器继续本指令（如果未使用BreakFlag）。常量ERR_TP_DIBREAK可用于测试是否已出现上述情况。

[\DOBreak]

Digital Output Break

数据类型：signaldo

可能中断运算符对话框的数字信号输出信号。如果在将信号设置为1（或已经为1）之前未按下OK按钮，则用错误处理器继续执行程序，除非使用BreakFlag（参见下文）。可使用常量ERR_TP_DOBREAK来测试是否已出现该情况。

[\DOPassive]

Digital Output Passive

数据类型：switch

当使用DOBreak可选参数时，该开关会覆盖默认行为。将信号设置为1（或已经为1）时，取代反应；将信号 DOBreak设置为0（或已经为0）时，应当用错误处理器继续本指令（如果未使用BreakFlag）。常量ERR_TP_DOBREAK可用于测试是否已出现上述情况。

[\BreakFlag]

数据类型：errnum

下一页继续

使用\MaxTime、\DIBreak或\DOBreak时将保存错误代码的变量（使用前，由系统将其设置为0）。常量ERR_TP_MAXTIME、ERR_TP_DIBREAK和ERR_TP_DOBREAK可用于选择原因。如果省略该可选变量，则将执行错误处理器。

程序执行

根据编程参数，显示数字消息框以及数字平板电脑、图标、标题、消息行、初始值、最大值和最小值。程序执行进入等待，直至用户已输入一个经批准的数值，并按下OK，或消息框由超时或信号行动中断。将输入数值和中断原因转移回程序。

从基础等级的消息框转而关注软中断等级的新消息框。

预定义数据

```
!Icons:
  CONST icondata iconNone := 0;
  CONST icondata iconInfo := 1;
  CONST icondata iconWarning := 2;
  CONST icondata iconError := 3;
```

更多示例

以下示例介绍了函数UIDnumEntry。

例 1

```
VAR errnum err_var;
VAR dnum answer;
VAR dnum distance;
...
answer := UIDnumEntry (\Header:= "BWD move on path"
  \Message:="Enter the path overlap?" \Icon:=iconInfo
  \InitValue:=5 \MinValue:=0 \MaxValue:=10
  \MaxTime:=60 \DIBreak:=di5 \BreakFlag:=err_var);
TEST err_var
  CASE ERR_TP_MAXTIME:
  CASE ERR_TP_DIBREAK:
    ! No operator answer distance := 5;
  CASE 0
    ! Operator answer
    distance := answer;
  DEFAULT:
    ! No such case defined
ENDTEST
```

显示消息框，且操作员可输入一个字符，并按下OK。消息框亦可因数字信号输入信号超时或中断而中断。在本程序中，发现原因，并采取适当的措施是可能的。

错误处理

如果未使用参数\BreakFlag，则可通过错误处理器来处理此类情形：

- 如果在运算符输入之前出现超时（参数\MaxTime），则将系统变量ERRNO设置为ERR_TP_MAXTIME，并用错误处理器继续执行。
- 如果在运算符输入之前设置数字信号输入（参数\DIBreak），则将系统变量ERRNO设置为ERR_TP_DIBREAK，并用错误处理器继续执行。

下一页继续

2.191 UIDnumEntry - 用户数字条目
RobotWare - OS
续前页

- 如果在运算符输入之前设置数字信号输出（参数\DOBreak），则将系统变量ERRNO设置为ERR_TP_DOBREAK，并用错误处理器继续执行。

仅可通过错误处理器来处理该情形：

- 如果信号变量是在RAPID中声明的变量，且尚未与通过指令AliasIO而在I/O配置中确定的I/O信号相连，则将系统变量ERRNO设置为ERR_NO_ALIASIO_DEF，并用错误处理器继续执行。
- 如果没有客户端（例如，FlexPendant示教器）来关注指令，则将系统变量ERRNO设置为ERR_TP_NO_CLIENT，并用错误处理器继续执行。
- 如果未在最小和最大值（参数\MinValue和\MaxValue）范围内指定初始值（参数\InitValue），则将系统变量ERRNO设置为ERR_UI_INITVALUE，并通过错误处理器继续执行。
- 如果最小值（参数\MinValue）大于最大值（参数\MaxValue），则将系统变量ERRNO设置为ERR_UI_MAXMIN，并通过错误处理器继续执行。
- 如果初始值（参数 \InitValue）并非参数\AsInteger中规定的一个整数，则将系统变量ERRNO设置为ERR_UI_NOTINT，并通过错误处理器继续执行。

限制

当频繁地执行UIDnumEntry时，例如，在一个回路中，避免使用过小的超时参数\MaxTime。其可能导致系统性能出现不可预测的行为，诸如FlexPendant示教器响应放慢。

语法

```
UIDnumEntry '('
  ['\' Header ':=' <expression (IN) of string>]
  [Message ':=' <expression (IN) of string> ]
  | ['\' MsgArray ':=' <array {*} (IN) of string>]
  ['\' Wrap]
  ['\' Icon ':=' <expression (IN) of icondata>]
  ['\' InitValue ':=' <expression (IN) of dnum>]
  ['\' MinValue ':=' <expression (IN) of dnum>]
  ['\' MaxValue ':=' <expression (IN) of dnum>]
  ['\' AsInteger]
  ['\' MaxTime ':=' <expression (IN) of num>]
  ['\' DIBreak ':=' <variable (VAR) of signaldi>]
  ['\' DIPassive]
  ['\' DOBreak ':=' <variable (VAR) of signaldo>]
  ['\' DOPassive]
  ['\' BreakFlag ':=' <var or pers (INOUT) of errnum>] ')'
```

含数据类型dnum的返回值的函数。

相关信息

信息，关于	请参阅
图标显示数据	第*1411*页的*icondata - 图标显示数据*
基础型用户交互消息框	第*832*页的*UIMsgBox - 用户消息对话框，基本类型*
先进型用户交互消息框	第*1317*页的*UIMessageBox - 先进型用户消息框*

下一页继续

信息，关于	请参阅
用户交互数字条目	第*1324*页的*UINumEntry - 用户数字条目*
用户交互数字调节	第*1304*页的*UIDnumTune - 用户数字调节*
用户交互数字调节	第*1330*页的*UINumTune - 用户数字调节*
用户交互α条目	第*1292*页的*UIAlphaEntry - 用户α条目*
用户交互列表视图	第*1310*页的*UIListView - 用户列表视图*
同FlexPendant示教器等相连的系统	第*1297*页的*UIClientExist - 现有客户端*
清除运算符窗口	第*732*页的*TPErase - 擦除在FlexPendant示教器上印刷的文本*

2.192 UIDnumTune - 用户数字调节

手册用法

UIDnumTune（*User Interaction Number Tune*）用于让操作员通过可用的用户设备，例如FlexPendant示教器，调节一个数值。将一条消息写给操作员，其调节一个数值。随后，检查、批准经调节的数值，并将其转换回程序。

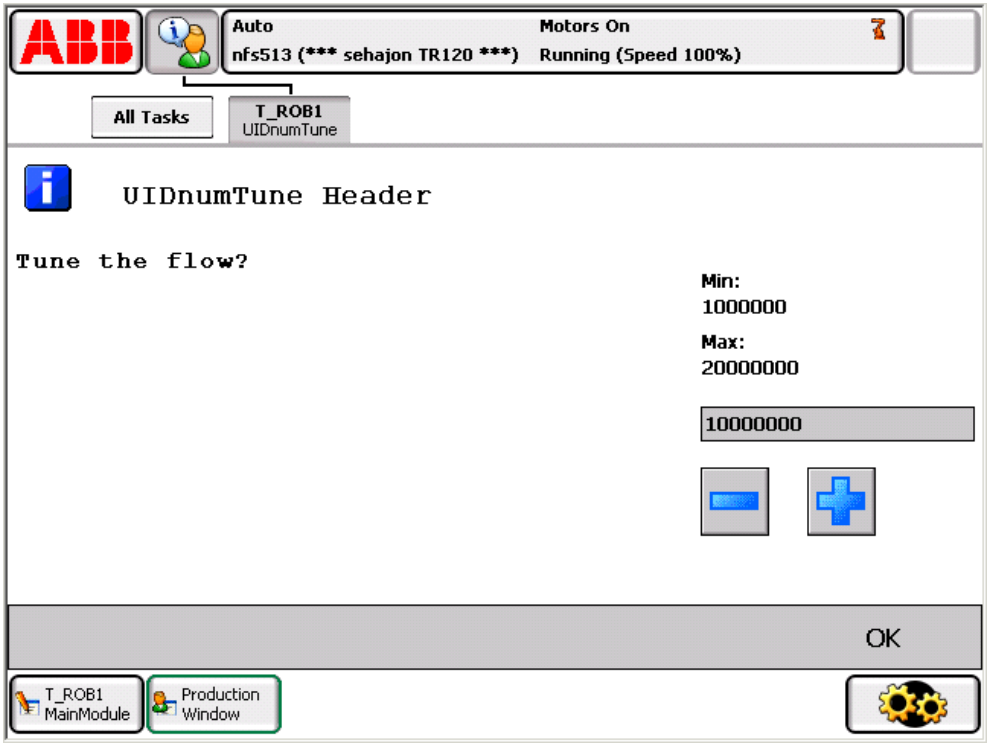
基本示例

以下示例介绍了函数UIDnumTune。

另请参阅第*1307*页的更多示例

例 1

```
VAR dnum flow;
...
flow := UIDnumTune(
  \Header:="UIDnumTune Header"
  \Message:="Tune the flow?"
  \Icon:=iconInfo,
  10000000,
  1000000
  \MinValue:=1000000
  \MaxValue:=20000000);
```



xx0900001063

在上文中，数字调节消息框以及图标、标题、消息、初始值、增量值、最大值和最小值得以写入FlexPendant示教器显示器。消息框检查操作员是否调节flow值以及由初始值10000000步进1000000，且是否处于值范围1000000-20000000内。程序执行进入等待，直至按下OK，随后，返回选定的数值，并将其储存在变量flow中。

下一页继续

返回值

数据类型：dnum

该函数返回经调整的数值。

如果函数通过\BreakFlag而中断，则返回指定的InitValue。

如果函数通过ERROR处理器而中断，则不会返回任何返回值。

变元

```
UIDnumTune ( [\Header] [\Message] | [\MsgArray] [\Wrap]
     [\Icon]InitValue Increment [\MinValue] [\MaxValue]
     [\MaxTime][\DIBreak] [\DIPassive] [\DOBreak] [\DOPassive]
     [\BreakFlag] )
```

[\Header]

数据类型：string

将在消息框顶部写入的标题文本。最多40个字符。

[\Message]

数据类型：string

有待在显示器上写入的一个文本行。最多40个字符。

[\MsgArray]

Message Array

数据类型：string

Several text lines from an array to be written on the display.

仅可在同一时间使用参数\Message或\MsgArray之一。

最大布置间距为11行，每行包含40个字符。

[\Wrap]

数据类型：switch

如果选择，则参数\MsgArray中的所有指定字符串均将连接到一个字符串，各单独字符串之间仅存在单间距，并以尽可能少的行展开。

默认将在显示器上单独的行中显示参数\MsgArray中的各字符串。

[\Icon]

数据类型：icondata

定义有待显示的图标。Only one of the predefined icons of type icondata can be used.参见第*1307*页的预定义数据。

默认没有图标。

InitValue

Initial Value

数据类型：dnum

在输入框中显示的初始值。

Increment

数据类型：dnum

该参数规定了按下加或减按钮时，值所应改变的程度。

下一页继续

2.192 UIDnumTune - 用户数字调节
RobotWare - OS
续前页

[\MinValue]

数据类型：dnum

有关返回值的最小值。

[\MaxValue]

数据类型：dnum

有关返回值的最大值。

[\MaxTime]

数据类型：num

程序执行等待的最长时间，以秒计。如果未在该时间内按下OK按钮，则用错误处理器继续执行程序，除非使用BreakFlag（参见下文）。常量ERR_TP_MAXTIME可用于测试是否已经过最长时间。

[\DIBreak]

Digital Input Break

数据类型：signaldi

可能中断运算符对话框的数字信号输入信号。如果在将信号设置为1（或已经为1）之前未按下OK按钮，则用错误处理器继续执行程序，除非使用BreakFlag（参见下文）。可使用常量ERR_TP_DIBREAK来测试是否已出现该情况。

[\DIPassive]

Digital Input Passive

数据类型：switch

当使用DIBreak可选参数时，该开关会覆盖默认行为。将信号设置为1（或已经为1）时，取代反应；将信号DIBreak设置为0（或已经为0）时，应当用错误处理器继续本指令（如果未使用BreakFlag）。常量ERR_TP_DIBREAK可用于测试是否已出现上述情况。

[\DOBreak]

Digital Output Break

数据类型：signaldo

可能中断运算符对话框的数字信号输出信号。如果在将信号设置为1（或已经为1）之前未按下OK按钮，则用错误处理器继续执行程序，除非使用BreakFlag（参见下文）。可使用常量ERR_TP_DOBREAK来测试是否已出现该情况。

[\DOPassive]

Digital Output Passive

数据类型：switch

当使用DOBreak可选参数时，该开关会覆盖默认行为。将信号设置为1（或已经为1）时，取代反应；将信号 DOBreak设置为0（或已经为0）时，应当用错误处理器继续本指令（如果未使用BreakFlag）。常量ERR_TP_DOBREAK可用于测试是否已出现上述情况。

[\BreakFlag]

数据类型：errnum

下一页继续

使用\MaxTime、\DIBreak或\DOBreak时将保存错误代码的变量（使用前，由系统将其设置为0）。常量ERR_TP_MAXTIME、ERR_TP_DIBREAK和ERR_TP_DOBREAK可用于选择原因。如果省略该可选变量，则将执行错误处理器。

程序执行

根据编程参数，显示数字调节消息框以及调节+/-按钮、图标、标题、消息行、初始值、增量值、最大值和最小值。程序执行进入等待，直至用户已调节数值，并按下OK，或消息框由超时或信号行动中断。将输入数值和中断原因转移回程序。

从基础等级的消息框转而关注TRAP等级的新消息框。

预定义数据

```
!Icons:
CONST icondata iconNone := 0;
CONST icondata iconInfo := 1;
CONST icondata iconWarning := 2;
CONST icondata iconError := 3;
```

更多示例

以下示例介绍了函数UIDnumTune。

例 1

```
VAR errnum err_var;
VAR dnum tune_answer;
VAR dnum distance;
...
tune_answer := UIDnumTune (\Header:=" BWD move on path"
  \Message:="Enter the path overlap?" \Icon:=iconInfo,
  5, 1 \MinValue:=0 \MaxValue:=10
  \MaxTime:=60 \DIBreak:=di5 \BreakFlag:=err_var);
TEST err_var
  CASE ERR_TP_MAXTIME:
  CASE ERR_TP_DIBREAK:
    ! No operator answer
    distance := 5;
  CASE 0:
    ! Operator answer
    distance := tune_answer;
  DEFAULT:
    ! No such case defined
ENDTEST
```

显示调节消息框，且操作员可调节数值，并按下OK。消息框亦可因数字信号输入信号超时或中断而中断。在本程序中，发现原因，并采取适当的措施是可能的。

错误处理

如果未使用参数\BreakFlag，则可通过错误处理器来处理此类情形：

- 如果在运算符输入之前出现超时（参数\MaxTime），则将系统变量ERRNO设置为ERR_TP_MAXTIME，并用错误处理器继续执行。
- 如果在运算符输入之前设置数字信号输入（参数\DIBreak），则将系统变量ERRNO设置为ERR_TP_DIBREAK，并用错误处理器继续执行。

下一页继续

2.192 UIDnumTune - 用户数字调节
RobotWare - OS
续前页

- 如果在运算符输入之前设置数字信号输出（参数\DOBreak，则将系统变量ERRNO设置为ERR_TP_DOBREAK，并用错误处理器继续执行。

仅可通过错误处理器来处理该情形：

- 如果信号变量是在RAPID中声明的变量，且尚未与通过指令AliasIO而在I/O配置中确定的I/O信号相连，则将系统变量ERRNO设置为ERR_NO_ALIASIO_DEF，并用错误处理器继续执行。
- 如果没有客户端（例如，FlexPendant示教器）来关注指令，则将系统变量ERRNO设置为ERR_TP_NO_CLIENT，并用错误处理器继续执行。
- 如果未在最小和最大值（参数\MinValue和\MaxValue）范围内指定初始值（参数\InitValue），则将系统变量ERRNO设置为ERR_UI_INITVALUE，并通过错误处理器继续执行。
- 如果最小值（参数\MinValue）大于最大值（参数\MaxValue），则将系统变量ERRNO设置为ERR_UI_MAXMIN，并通过错误处理器继续执行。

限制

当频繁地执行UIDnumTune时，例如，在一个回路中，避免使用过小的超时参数\MaxTime。其可能导致系统性能出现不可预测的行为，诸如FlexPendant示教器响应放慢。

语法

```
UIDnumTune '('
  ['\' Header ':=' <expression (IN) of string>]
  ['\' Message ':=' <expression (IN) of string> ]
  | ['\' MsgArray ':=' <array {*} (IN) of string>]
  ['\' Wrap]
  ['\' Icon ':=' <expression (IN) of icondata>] ',']
  [InitValue ':=' ] <expression (IN) of dnum> ','
  [Increment ':=' ] <expression (IN) of dnum>
  ['\' MinValue ':=' <expression (IN) of dnum>]
  ['\' MaxValue ':=' <expression(IN) of dnum>]
  ['\' MaxTime ':=' <expression (IN) of num>]
  ['\' DIBreak ':=' <variable (VAR) of signaldi>]
  ['\' DIPassive]
  ['\' DOBreak ':=' <variable (VAR) of signaldo>]
  ['\' DOPassive]
  ['\' BreakFlag ':=' <var or pers (INOUT) of errnum>] ')'
```

含数据类型dnum的返回值的函数。

相关信息

信息，关于	请参阅
图标显示数据	第*1411*页的*icondata - 图标显示数据*
基础型用户交互消息框	第*832*页的*UIMsgBox - 用户消息对话框，基本类型*
先进型用户交互消息框	第*1317*页的*UIMessageBox - 先进型用户消息框*
用户交互数字条目	第*1298*页的*UIDnumEntry - 用户数字条目*
用户交互数字条目	第*1324*页的*UINumEntry - 用户数字条目*

下一页继续

信息，关于	请参阅
用户交互数字调节	第*1330*页的*UINumTune - 用户数字调节*
用户交互α条目	第*1292*页的*UIAlphaEntry - 用户α条目*
用户交互列表视图	第*1310*页的*UIListView - 用户列表视图*
同FlexPendant示教器等相连的系统	第*1297*页的*UIClientExist - 现有客户端*
清除运算符窗口	第*732*页的*TPErase - 擦除在FlexPendant示教器上印刷的文本*

2.193 UIListView - 用户列表视图

手册用法

UIListView（*User Interaction List View*）用于确定有关可用用户设备（例如FlexPendant示教器）的文本和可选图标的菜单列表。本菜单具有两种不同的模式，一种包含确认按钮，一种立即对用户选择作出反应。

基本示例

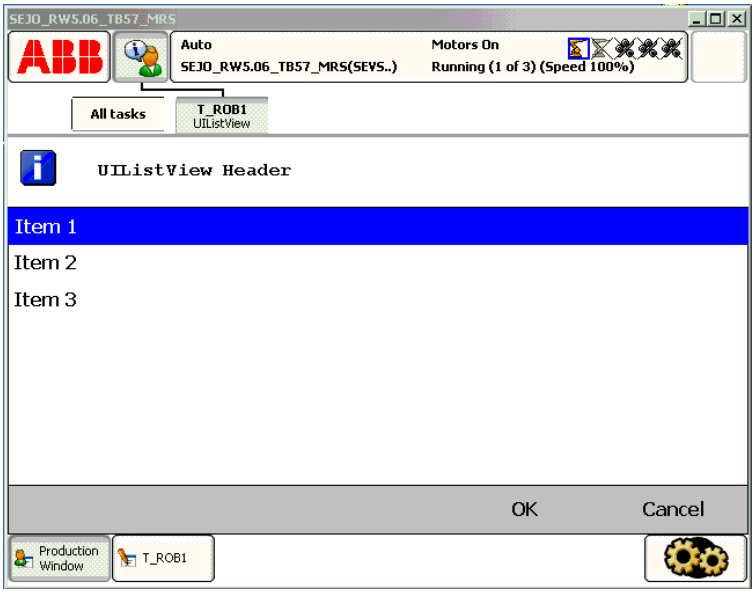
以下示例介绍了函数UIListView。

另请参阅第*1314*页的更多示例

例 1

```
CONST listitem list{3} := [ ["","Item 1"], ["","Item 2"],
     ["","Item3"] ];
VAR num list_item;
VAR btnres button_answer;
...
list_item := UIListView (
  \Result:=button_answer
  \Header:="UIListView Header",
  list
  \Buttons:=btnOKCancel
  \Icon:=iconInfo
  \DefaultIndex:=1);
IF button_answer = resOK THEN
  IF list_item = 1 THEN
    ! Do item1
  ELSEIF list_item = 2 THEN
    ! Do item 2
  ELSE
    ! Do item3
  ENDIF
ELSE
  ! User has select Cancel
ENDIF
```

下一页继续



xx0500002416

在上文中，菜单列表以及图标、标题、菜单Item 1 ... Item 3和按钮得以写入FlexPendant示教器显示器。程序执行进入等待，直至按下确定（OK）或取消（Cancel）。列表和按钮中的选择均得以转移至程序。

返回值

数据类型：num

在符合参数ListItems中指定数组中索引的列表菜单中，该函数返回用户选择。

如果函数通过 \BreakFlag而中断：

- 如果指定参数\DefaultIndex，则返回该索引
- 如果未指定参数\DefaultIndex，则返回0

如果函数通过ERROR处理器而中断，则不会返回任何返回值。

变元

```
UIListView ( [\Result] [\Header] ListItems [\Buttons] |
     [\BtnArray][\Icon] [\DefaultIndex ] [\MaxTime] [\DIBreak]
     [\DIPassive][\DOBreak] [\DOPassive] [\BreakFlag])
```

[\Result]

数据类型：btnres

从列表菜单栏选择的按钮数值。

如果使用参数\Buttons，则返回btnres型预定义符号常量。如果使用参数\BtnArray，则返回相关的数组索引。

如果为以下条件之一，则设置为resUnkwn的参数\Result等于0：

- 未使用任何参数\Buttons或\BtnArray
- 使用参数\Buttons:=btnNone
- 如果函数通过\BreakFlag或ERROR处理器而中断

请参阅 第*1314*页的预定义数据。

下一页继续

2.193 UIListView - 用户列表视图
RobotWare - OS
续前页

[\Header]

数据类型：string

将在列表菜单栏顶部写入的标题文本。最多40个字符。

ListItems

数据类型：listitem

包含一个或多个待显示列表菜单项的一个数组由以下部分组成：

string型部件image：

应当使用的图标图像的名称。为推出自己的图像，必须将图像置于有效系统的HOME:路径中，或直接置于有效系统中。

建议将文件置于HOME:路径中，以便在完成备份和储存时将其保存。

需要重启，随后，FlexPendant示教器加载图像。

对系统的要求是使用RobotWare附加功能*FlexPendant Interface*。

有待显示的图像可拥有28像素的宽度和高度。如果图像过大，则调整其大小，以便仅显示28 * 28像素。

无法指明关于图像尺寸或能够加载至FlexPendant示教器的图像数量的确切值。其取决于加载至FlexPendant示教器的其他文件的尺寸。如果使用的图像尚未加载至FlexPendant示教器，则将继续程序执行。

如果无图标显示，则使用空字符串“”或stEmpty。

string型部件text：

- 菜单行有待显示的文本。
- 各列表菜单项最多75个字符。

[\Buttons]

数据类型：buttondata

定义有待显示的按钮。仅可使用一种按钮数据型预定义按钮组合。参见第*1314*页的预定义数据。

[\BtnArray]

Button Array

数据类型：string

将自己有关按钮的定义储存在一个字符串数组中。当选择相关的字符串后，该函数返回数组索引。

在同一时间，仅可使用参数 \Buttons或\BtnArray之一。如果未使用任何参数\Buttons或\BtnArray或参数\Buttons:=btnNone，则菜单列表立即对用户选择作出反应。

最多5个按钮，每个按钮42个字符。

[\Icon]

数据类型：icondata

定义有待显示的图标。仅可使用一种icondata型预定义图标。

默认没有图标。参见第*1314*页的预定义数据。

[\DefaultIndex]

数据类型：num

下一页继续

在符合参数ListItems中指定数组中索引的列表菜单中的默认用户选择。

[\MaxTime]

数据类型：num

程序执行等待的最长时间，以秒计。如果在该时间内未按下任何按钮或实施任何选择，则用错误处理器继续执行程序，除非使用BreakFlag（参见下文）。常量ERR_TP_MAXTIME可用于测试是否已经过最长时间。

[\DIBreak]

Digital Input Break

数据类型：signaldi

可能中断运算符对话框的数字信号输入信号。如果在将信号设置为1（或已经为1）之前未按下任何按钮或实施任何选择，则用错误处理器继续执行程序，除非使用BreakFlag（参见下文）。可使用常量ERR_TP_DIBREAK来测试是否已出现该情况。

[\DIPassive]

Digital Input Passive

数据类型：switch

当使用DIBreak可选参数时，该开关会覆盖默认行为。将信号设置为1（或已经为1）时，取代反应；将信号DIBreak设置为0（或已经为0）时，应当用错误处理器继续本指令（如果未使用BreakFlag）。常量ERR_TP_DIBREAK可用于测试是否已出现上述情况。

[\DOBreak]()

Digital Output Break

数据类型：signaldo

可能中断运算符对话框的数字信号输出信号。如果在将信号设置为1（或已经为1）之前未按下任何按钮或实施任何选择，则用错误处理器继续执行程序，除非使用BreakFlag（参见下文）。可使用常量ERR_TP_DOBREAK来测试是否已出现该情况。

[\DOPassive]

Digital Output Passive

数据类型：switch

当使用DOBreak可选参数时，该开关会覆盖默认行为。将信号设置为1（或已经为1）时，取代反应；将信号 DOBreak设置为0（或已经为0）时，应当用错误处理器继续本指令（如果未使用BreakFlag）。常量ERR_TP_DOBREAK可用于测试是否已出现上述情况。

[\BreakFlag]

数据类型：errnum

使用\MaxTime、\DIBreak或\DOBreak时将保存错误代码的变量。常量ERR_TP_MAXTIME、ERR_TP_DIBREAK和ERR_TP_DOBREAK可用于选择原因。如果省略该可选变量，则将执行错误处理器。

下一页继续

2.193 UIListView - 用户列表视图

RobotWare - OS

续前页

程序执行

根据编程参数，显示菜单列表以及图标、标题、列表项和默认项。程序执行进入等待，直至操作员已作出选择，或菜单列表被超时或信号行动打断。将选定的列表项和中断原因转移回程序。

从基础等级菜单列表转而关注软中断等级新菜单列表。

预定义数据

```
!Icons:
CONST icondata iconNone := 0;
CONST icondata iconInfo := 1;
CONST icondata iconWarning := 2;
CONST icondata iconError := 3;
!Buttons:
CONST buttondata btnNone := -1;
CONST buttondata btnOK := 0;
CONST buttondata btnAbrtRtryIgn := 1;
CONST buttondata btnOKCancel := 2;
CONST buttondata btnRetryCancel := 3;
CONST buttondata btnYesNo := 4;
CONST buttondata btnYesNoCancel := 5;
!Results:
CONST btnres resUnkwn := 0;
CONST btnres resOK := 1;
CONST btnres resAbort := 2;
CONST btnres resRetry := 3;
CONST btnres resIgnore := 4;
CONST btnres resCancel := 5;
CONST btnres resYes := 6;
CONST btnres resNo := 7;
```

更多示例

以下示例介绍了函数UIListView。

例 1

```
CONST listitem list{2} := [ ["","Calibrate tool1"], ["","Calibrate
     tool2"] ];
VAR num list_item;
VAR errnum err_var;
...
list_item := UIListView
  ( \Header:="Select tool ?",
  list \Icon:=iconInfo
  \MaxTime:=60
  \DIBreak:=di5
  \BreakFlag:=err_var);
TEST err_var
  CASE ERR_TP_MAXTIME:
  CASE ERR_TP_DIBREAK:
    ! No operator answer
```

下一页继续

```
  CASE 0:
    ! Operator answer
    IF list_item =1 THEN
      ! Calibrate tool1
    ELSEIF list_item=2 THEN
      ! Calibrate tool2
    ENDIF
  DEFAULT:
    ! Not such case defined
ENDTEST
```

显示消息框，且操作员可在列表中选择一项。消息框亦可因数字信号输入信号超时或中断而中断。在本程序中，发现原因，并采取适当的措施是可能的。

错误处理

如果未使用参数\BreakFlag，则可通过错误处理器来处理此类情形：

- 如果在运算符输入之前出现超时（参数 \MaxTime），则将系统变量ERRNO设置为ERR_TP_MAXTIME，并用错误处理器继续执行。
- 如果在运算符输入之前设置数字信号输入（参数\DIBreak），则将系统变量ERRNO设置为ERR_TP_DIBREAK，并用错误处理器继续执行。
- 如果在运算符输入之前设置数字信号输出（参数\DOBreak，则将系统变量ERRNO设置为ERR_TP_DOBREAK，并用错误处理器继续执行。

仅可通过错误处理器来处理该情形：

- 如果信号变量是在RAPID中声明的变量，且尚未与通过指令AliasIO而在I/O配置中确定的I/O信号相连，则将系统变量ERRNO设置为ERR_NO_ALIASIO_DEF，并用错误处理器继续执行。
- 如果没有客户端（例如，FlexPendant示教器）来关注指令，则将系统变量ERRNO设置为ERR_TP_NO_CLIENT，并用错误处理器继续执行。

限制

当频繁地执行UIListView时，例如，在一个回路中，避免使用过小的超时参数\MaxTime。其可能导致系统性能出现不可预测的行为，诸如FlexPendant示教器响应放慢。

语法

```
UIListView '('
  [['\' Result ':=' <var or pers (INOUT) of btnres>]
  ['\' Header ':=' <expression (IN) of string>] ',']
  [ListItems '='] <array {*} (IN) of listitem>
  ['\' Buttons ':=' <expression (IN) of buttondata>]
  | ['\' BtnArray ':=' <array {*} (IN) of string>]
  ['\' Icon ':=' <expression (IN) of icondata>]
  ['\' DefaultIndex ':=' <expression (IN) of num>]
  ['\' MaxTime ':=' <expression (IN) of num>]
  ['\' DIBreak ':=' <variable (VAR) of signaldi>]
  ['\' DIPassive]
  ['\' DOBreak ':=' <variable (VAR) of signaldo>]
  ['\' DOPassive]
```

下一页继续

2.193 UIListView - 用户列表视图
RobotWare - OS
续前页

```
['\' BreakFlag ':=' <var or pers (INOUT) of errnum>] ')'
```

含数据类型num的返回值的函数。

相关信息

信息，关于	请参阅
图标显示数据	第*1411*页的*icondata - 图标显示数据*
按钮数据	第*1349*页的*buttondata - 按钮数据*
按钮结果数据	第*1346*页的*btnres - 按钮结果数据*
列表项数据结构	第*1420*页的*listitem - 列表项数据结构*
基础型用户交互消息框	第*832*页的*UIMsgBox - 用户消息对话框，基本类型*
先进型用户交互消息框	第*1317*页的*UIMessageBox - 先进型用户消息框*
用户交互数字条目	第*1324*页的*UINumEntry - 用户数字条目*
用户交互数字调节	第*1330*页的*UINumTune - 用户数字调节*
用户交互α条目	第*1292*页的*UIAlphaEntry - 用户α条目*
同FlexPendant示教器等相连的系统	第*1297*页的*UIClientExist - 现有客户端*
清除运算符窗口	第*732*页的*TPErase - 擦除在FlexPendant示教器上印刷的文本*

2.194 UIMessageBox - 先进型用户消息框

手册用法

UIMessageBox（*User Interaction Message Box*）用于同有关可用用户设备（例如，FlexPendant示教器）的机器人系统的用户进行沟通。写入消息，以供操作员使用，操作员通过选择按钮来进行回答。随后，将用户选择转移回程序。

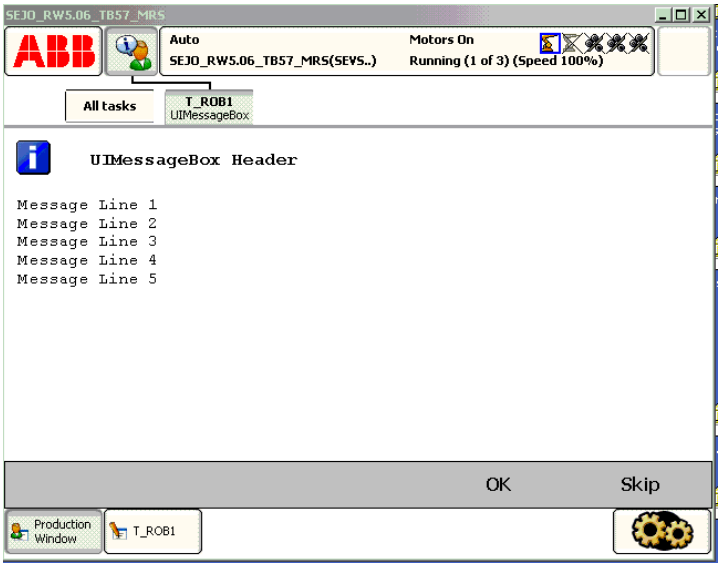
基本示例

以下示例介绍了函数UIMessageBox。

另请参阅第*1321*页的更多示例

例 1

```
VAR btnres answer;
CONST string my_message{5}:= ["Message Line 1","Message Line 2",
     "Message Line 3","Message Line 4","Message Line 5"];
CONST string my_buttons{2}:=["OK","Skip"];
...
answer:= UIMessageBox (
  \Header:="UIMessageBox Header"
  \MsgArray:=my_message
  \BtnArray:=my_buttons
  \Icon:=iconInfo);
IF answer = 1 THEN
  ! Operator selection OK
ELSEIF answer = 2 THEN
  ! Operator selection Skip
ELSE
  ! No such case defined
ENDIF
```



xx0500002409

下一页继续

2.194 UIMessageBox - 先进型用户消息框

RobotWare - OS

续前页

在上文中，消息框以及图标、标题、消息和用户定义按钮得以写入FlexPendant示教器显示器。程序执行进入等待，直至按下确定（OK）或跳过（Skip）。换句话说，根据按下的按钮（相应的数组索引），向answer 分配1（OK）或2（Skip）。

> **注意**
>
> 在单独的行1到5上显示Message Line 1到Message Line 5（未使用开关\Wrap）。

返回值

数据类型：btnres

从消息框选择的按钮数值。

如果使用参数\Buttons，则返回btnres型预定义符号常量。

如果使用参数\BtnArray，则返回相应的数组索引。

如果函数通过\BreakFlag中断，或者如果\Buttons:=btnNone：

- 如果指定参数\DefaultBtn，则返回该索引。
- 如果未指定参数\DefaultBtn，则返回等于0的resUnkwn。

如果函数通过ERROR处理器而中断，则不存在任何返回值。

变元

```
UIMessageBox ( [\Header] [\Message] | [\MsgArray] [\Wrap][\Buttons]
     | [\BtnArray] [\DefaultBtn] [\Icon][\Image] [\MaxTime]
     [\DIBreak] [\DIPassive] [\DOBreak] [\DOPassive] [\BreakFlag])
```

[\Header]

数据类型：string

将在消息框顶部写入的标题文本。最多40个字符。

[\Message]

数据类型：string

有待在显示器上写入的一个文本行。最多55个字符。

[\MsgArray]

Message Array

数据类型：string

Several text lines from an array to be written on the display.

仅可在同一时间使用参数\Message或\MsgArray之一。

最大布置间距为11行，每行包含55个字符。

[\Wrap]

数据类型：switch

如果选择，则参数\MsgArray中的所有指定字符串均将连接到一个字符串，各单独字符串之间仅存在单间距，并以尽可能少的行展开。

默认将在显示器上单独的行中显示参数\MsgArray中的各字符串。

下一页继续

[\Buttons]

数据类型：buttondata

Defines the push buttons to be displayed. Only one of the predefined buttons combination of type buttondata can be used.，参见第*1320*页的预定义数据。

系统默认显示OK按钮。

[\BtnArray]

Button Array

数据类型：string

将自己有关按钮的定义储存在一个字符串数组中。当选择相关的字符串后，该函数返回数组索引。

仅可在同一时间使用参数 \Buttons或\BtnArray之一。

最多5个按钮，每个按钮42个字符。

[\DefaultBtn]

Default Button

数据类型：btnres

如果消息框被\MaxTime、 \DIBreak或\DOBreak中断，则允许指定应当返回的值。指定btnres型预定义符号常量或任何用户定义值是可能的。参见第*1320*页的预定义数据。

[\Icon]

数据类型：icondata

定义有待显示的图标。仅可使用一种icondata型预定义图标。参见第*1320*页的预定义数据。

默认没有图标。

[\Image]

数据类型：string

应当使用的图像的名称。为推出自己的图像，必须将图像置于有效系统的HOME:路径中，或直接置于有效系统中。

建议将文件置于HOME:路径中，以便在完成备份和储存时将其保存。

需要重启，随后，FlexPendant示教器加载图像。

对系统的要求是使用RobotWare附加功能*FlexPendant Interface*。

将显示的图像可拥有185个像素点的宽度和300个像素点的高度。如果图像较大，则将仅显示该图像左上方185*300像素点的图像。

无法指明关于图像尺寸或能够加载至FlexPendant示教器的图像数量的确切值。其取决于加载至FlexPendant示教器的其他文件的尺寸。如果使用的图像尚未加载至FlexPendant示教器，则将继续程序执行。

[\MaxTime]

数据类型：num

程序执行等待的最长时间，以秒计。如果在该时间内未选择任何按钮，则用错误处理器继续执行程序，除非使用BreakFlag（参见下文）。常量ERR_TP_MAXTIME可用于测试是否已经过最长时间。

下一页继续

2.194 UIMessageBox - 先进型用户消息框
RobotWare - OS
续前页

[\DIBreak]

Digital Input Break

数据类型：signaldi

可能中断运算符对话框的数字信号输入信号。如果将信号设置为1（或已经为1）时未选择任何按钮，则用错误处理器继续执行程序，除非使用BreakFlag （参见下文）。可使用常量ERR_TP_DIBREAK来测试是否已出现该情况。

[\DIPassive]

Digital Input Passive

数据类型：switch

当使用DIBreak可选参数时，该开关会覆盖默认行为。将信号设置为1（或已经为1）时，取代反应；将信号DIBreak设置为0（或已经为0）时，应当用错误处理器继续本指令（如果未使用BreakFlag）。常量ERR_TP_DIBREAK可用于测试是否已出现上述情况。

[\DOBreak]

Digital Output Break

数据类型：signaldo

可能中断运算符对话框的数字信号输出信号。如果将信号设置为1（或已经为1）时未选择任何按钮，则用错误处理器继续执行程序，除非使用BreakFlag（参见下文）。可使用常量ERR_TP_DOBREAK来测试是否已出现该情况。

[\DOPassive]

Digital Output Passive

数据类型：switch

当使用DOBreak可选参数时，该开关会覆盖默认行为。将信号设置为1（或已经为1）时，取代反应；将信号 DOBreak设置为0（或已经为0）时，应当用错误处理器继续本指令（如果未使用BreakFlag）。常量ERR_TP_DOBREAK可用于测试是否已出现上述情况。

[\BreakFlag]

数据类型：errnum

使用\MaxTime,、\DIBreak或\DOBreak时将保存错误代码的变量（使用前，由系统将其设置为0）。常量ERR_TP_MAXTIME、ERR_TP_DIBREAK和ERR_TP_DOBREAK可用于选择原因。如果省略该可选变量，则将执行错误处理器。

程序执行

根据编程参数，显示消息框以及图标、标题、消息行、图像和按钮。程序执行进入等待，直至用户选择一个按钮，或消息框被超时或信号行动打断。将用户选择和中断原因转移回程序。

从基础等级的消息框转而关注软中断等级的新消息框。

预定义数据

```
!Icons:
  CONST icondata iconNone := 0;
  CONST icondata iconInfo := 1;
  CONST icondata iconWarning := 2;
```

下一页继续

```
  CONST icondata iconError := 3;
!Buttons:
  CONST buttondata btnNone := -1;
  CONST buttondata btnOK := 0;
  CONST buttondata btnAbrtRtryIgn := 1;
  CONST buttondata btnOKCancel := 2;
  CONST buttondata btnRetryCancel := 3;
  CONST buttondata btnYesNo := 4;
  CONST buttondata btnYesNoCancel := 5;
!Results:
  CONST btnres resUnkwn := 0;
  CONST btnres resOK := 1;
  CONST btnres resAbort := 2;
  CONST btnres resRetry := 3;
  CONST btnres resIgnore := 4;
  CONST btnres resCancel := 5;
  CONST btnres resYes := 6;
  CONST btnres resNo := 7;
```

更多示例

以下示例介绍了函数UIMessageBox。

例 1

```
VAR errnum err_var;
VAR btnres answer;
...
answer := UIMessageBox (\Header:= "Cycle step 3"
  \Message:="Continue with the calibration ?" \Buttons:=btnOKCancel
  \DefaultBtn:=resCancel \Icon:=iconInfo \MaxTime:=60 \DIBreak:=di5
  \BreakFlag:=err_var);
IF answer = resOK THEN
  ! OK from the operator
ELSE
  ! Cancel from the operator or operation break
  TEST err_var
    CASE ERR_TP_MAXTIME:
      ! Time out
    CASE ERR_TP_DIBREAK:
      ! Input signal break
    DEFAULT:
      ! Not such case defined
  ENDTEST
ENDIF
```

显示消息框，且操作员可回应确定（OK）或取消（Cancel）。消息框亦可因数字信号输入信号超时或中断而中断。在本程序中，发现原因是可能的。

错误处理

如果未使用参数\BreakFlag，则可通过错误处理器来处理此类情形：

- 如果在运算符输入之前出现超时（参数\MaxTime），则将系统变量ERRNO设置为ERR_TP_MAXTIME，并用错误处理器继续执行。

下一页继续

2.194 UIMessageBox - 先进型用户消息框
RobotWare - OS
续前页

- 如果在运算符输入之前设置数字信号输入（参数\DIBreak），则将系统变量ERRNO设置为ERR_TP_DIBREAK，并用错误处理器继续执行。
- 如果在运算符输入之前设置数字信号输出（参数\DOBreak，则将系统变量ERRNO设置为ERR_TP_DOBREAK，并用错误处理器继续执行。

仅可通过错误处理器来处理该情形：

- 如果信号变量是在RAPID中声明的变量，且尚未与通过指令AliasIO而在I/O配置中确定的I/O信号相连，则将系统变量ERRNO设置为ERR_NO_ALIASIO_DEF，并用错误处理器继续执行。
- 如果没有客户端（例如，FlexPendant示教器）来关注指令，则将系统变量ERRNO设置为ERR_TP_NO_CLIENT，并用错误处理器继续执行。

限制

当频繁地执行UIMessageBox时，例如，在一个回路中，避免使用过小的超时参数\MaxTime。其可能导致系统性能出现不可预测的行为，诸如FlexPendant示教器响应放慢。

语法

```
UIMessageBox '('
  ['\' Header ':=' <expression (IN) of string>] ',']
  ['\' Message ':=' <expression (IN) of string>]
  | ['\' MsgArray ':=' <array {*} (IN) of string>]
  ['\' Wrap]
  ['\' Buttons ':=' <expression (IN) of buttondata>]
  | ['\' BtnArray ':=' <array {*} (IN) of string>]
  ['\' DefaultBtn ':=' <expression (IN) of btnres>]
  ['\' Icon ':=' <expression (IN) of icondata>]
  ['\' Image ':=' <expression (IN) of string>]
  ['\' MaxTime ':=' <expression (IN) of num>]
  ['\' DIBreak ':=' <variable (VAR) of signaldi>]
  ['\' DIPassive]
  ['\' DOBreak ':=' <variable (VAR) of signaldo>]
  ['\' DOPassive]
  ['\' BreakFlag ':=' <var or pers (INOUT) of errnum>] ')'
```

含数据类型btnres的返回值的函数。

相关信息

信息，关于	请参阅
图标显示数据	第*1411*页的*icondata - 图标显示数据*
按钮数据	第*1349*页的*buttondata - 按钮数据*
按钮结果数据	第*1346*页的*btnres - 按钮结果数据*
基础型用户交互消息框	第*832*页的*UIMsgBox - 用户消息对话框，基本类型*
用户交互数字条目	第*1324*页的*UINumEntry - 用户数字条目*
用户交互数字调节	第*1330*页的*UINumTune - 用户数字调节*
用户交互α条目	第*1292*页的*UIAlphaEntry - 用户α条目*
用户交互列表视图	第*1310*页的*UIListView - 用户列表视图*

下一页继续

信息，关于	请参阅
同FlexPendant示教器等相连的系统	第*1297*页的*UIClientExist - 现有客户端*
FlexPendant示教器界面	产品规格 - 控制器软件*IRC5*
清除运算符窗口	第*732*页的*TPErase - 擦除在FlexPendant示教器上印刷的文本*

2.195 UINumEntry - 用户数字条目

手册用法

UINumEntry（*User Interaction Number Entry*）用于让操作员通过可用的用户设备，例如FlexPendant示教器，输入一个数值。将一条消息写给操作员，其以一个数值作为应答。随后，检查、批准该数值，并将其转换回程序。

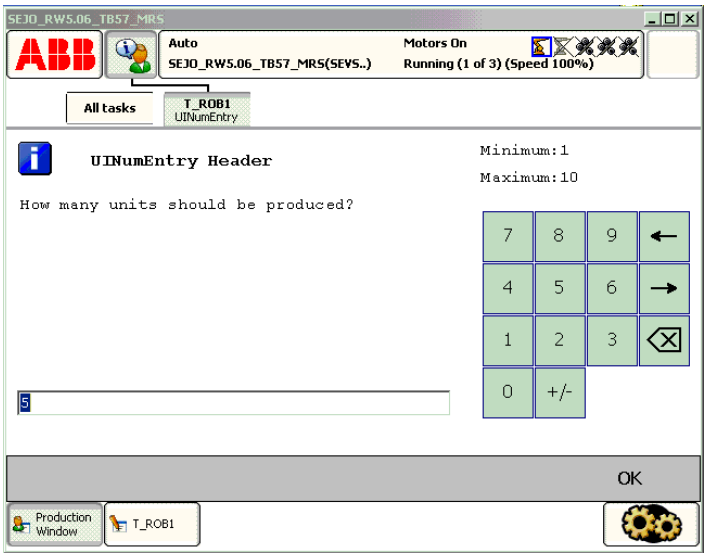
基本示例

以下示例介绍了函数UINumEntry。

另请参阅第*1327*页的更多示例

例 1

```
VAR num answer;
...
answer := UINumEntry(
  \Header:="UINumEntry Header"
  \Message:="How many units should be produced?"
  \Icon:=iconInfo
  \InitValue:=5
  \MinValue:=1
  \MaxValue:=10
  \AsInteger);
FOR i FROM 1 TO answer DO
  produce_part;
ENDFOR
```



xx0500002412

在上文中，数字消息框以及图标、标题、消息、初始值、最大值和最小值得以写入FlexPendant示教器显示器。消息框检查该运算符是否选择值范围内的一个整数。程序执行进入等待，直至按下OK，然后，返回选定的数值。随后，程序produce_part得以重复通过FlexPendant示教器的输入时间量。

下一页继续

返回值

数据类型：num

该函数返回输入数值。

如果函数通过\BreakFlag中断：

- 如果指定参数\InitValue，则返回该值
- 如果未指定参数\InitValue ，则返回值0。

如果函数通过ERROR处理器而中断，则不存在任何返回值。

变元

```
UINumEntry ( [\Header] [\Message] | [\MsgArray]
     [\Wrap][\Icon][\InitValue] [\MinValue] [\MaxValue]
     [\AsInteger][\MaxTime] [\DIBreak] [\DIPassive] [\DOBreak]
     [\DOPassive] \BreakFlag])
```

[\Header]

数据类型：string

将在消息框顶部写入的标题文本。最多40个字符。

[\Message]

数据类型：string

有待在显示器上写入的一个文本行。最多40个字符。

[\MsgArray]

Message Array

数据类型：string

Several text lines from an array to be written on the display.

仅可在同一时间使用参数\Message或\MsgArray之一。

最大布置间距为9行，每行包含40个字符。

[\Wrap]

数据类型：switch

如果选择，则参数\MsgArray中的所有指定字符串均将连接到一个字符串，各单独字符串之间仅存在单间距，并以尽可能少的行展开。

默认将在显示器上单独的行中显示参数\MsgArray中的各字符串。

[\Icon]

数据类型：icondata

定义有待显示的图标。仅可使用一种icondata型预定义图标。参见第*1327*页的预定义数据。

默认没有图标。

[\InitValue]

数据类型：num

在输入框中显示的初始值。

[\MinValue]

数据类型：num

下一页继续

有关返回值的最小值。

[\MaxValue]

数据类型：num

有关返回值的最大值。

[\AsInteger]

数据类型：switch

消除数字键盘的小数点，以确保返回值为一个整数。

[\MaxTime]

数据类型：num

程序执行等待的最长时间，以秒计。如果未在该时间内按下OK按钮，则用错误处理器继续执行程序，除非使用BreakFlag（参见下文）。常量ERR_TP_MAXTIME可用于测试是否已经过最长时间。

[\DIBreak]

Digital Input Break

数据类型：signaldi

可能中断运算符对话框的数字信号输入信号。如果在将信号设置为1（或已经为1）之前未按下OK按钮，则用错误处理器继续执行程序，除非使用BreakFlag（参见下文）。可使用常量ERR_TP_DIBREAK来测试是否已出现该情况。

[\DIPassive]

Digital Input Passive

数据类型：switch

当使用DIBreak可选参数时，该开关会覆盖默认行为。将信号设置为1（或已经为1）时，取代反应；将信号DIBreak设置为0（或已经为0）时，应当用错误处理器继续本指令（如果未使用BreakFlag）。常量ERR_TP_DIBREAK可用于测试是否已出现上述情况。

[\DOBreak]

Digital Output Break

数据类型：signaldo

可能中断运算符对话框的数字信号输出信号。如果在将信号设置为1（或已经为1）之前未按下OK按钮，则用错误处理器继续执行程序，除非使用BreakFlag（参见下文）。可使用常量ERR_TP_DOBREAK来测试是否已出现该情况。

[\DOPassive]

Digital Output Passive

数据类型：switch

当使用DOBreak可选参数时，该开关会覆盖默认行为。将信号设置为1（或已经为1）时，取代反应；将信号 DOBreak设置为0（或已经为0）时，应当用错误处理器继续本指令（如果未使用BreakFlag）。常量ERR_TP_DOBREAK可用于测试是否已出现上述情况。

[\BreakFlag]

数据类型：errnum

下一页继续

使用\MaxTime、\DIBreak或\DOBreak时将保存错误代码的变量（使用前，由系统将其设置为0）。常量ERR_TP_MAXTIME、ERR_TP_DIBREAK和ERR_TP_DOBREAK可用于选择原因。如果省略该可选变量，则将执行错误处理器。

程序执行

根据编程参数，显示数字消息框以及数字平板电脑、图标、标题、消息行、初始值、最大值和最小值。程序执行进入等待，直至用户已输入一个经批准的数值，并按下OK，或消息框由超时或信号行动中断。将输入数值和中断原因转移回程序。

从基础等级的消息框转而关注软中断等级的新消息框。

预定义数据

```
!Icons:
  CONST icondata iconNone := 0;
  CONST icondata iconInfo := 1;
  CONST icondata iconWarning := 2;
  CONST icondata iconError := 3;
```

更多示例

以下示例介绍了函数UINumEntry。

例 1

```
VAR errnum err_var;
VAR num answer;
VAR num distance;
...
answer := UINumEntry (\Header:= "BWD move on path"
  \Message:="Enter the path overlap ?" \Icon:=iconInfo
  \InitValue:=5 \MinValue:=0 \MaxValue:=10
  \MaxTime:=60 \DIBreak:=di5 \BreakFlag:=err_var);
TEST err_var
  CASE ERR_TP_MAXTIME:
  CASE ERR_TP_DIBREAK:
    ! No operator answer distance := 5;
  CASE 0
    ! Operator answer
    distance := answer;
  DEFAULT:
    ! Not such case defined
ENDTEST
```

显示消息框，且操作员可输入一个字符，并按下OK。消息框亦可因数字信号输入信号超时或中断而中断。在本程序中，发现原因，并采取适当的措施是可能的。

错误处理

如果未使用参数\BreakFlag，则可通过错误处理器来处理此类情形：

- 如果在运算符输入之前出现超时（参数\MaxTime），则将系统变量ERRNO设置为ERR_TP_MAXTIME，并用错误处理器继续执行。
- 如果在运算符输入之前设置数字信号输入（参数\DIBreak），则将系统变量ERRNO设置为ERR_TP_DIBREAK，并用错误处理器继续执行。

下一页继续

2.195 UINumEntry - 用户数字条目
RobotWare - OS
续前页

- 如果在运算符输入之前设置数字信号输出（参数\DOBreak），则将系统变量ERRNO设置为ERR_TP_DOBREAK，并用错误处理器继续执行。

仅可通过错误处理器来处理该情形：

- 如果信号变量是在RAPID中声明的变量，且尚未与通过指令AliasIO而在I/O配置中确定的I/O信号相连，则将系统变量ERRNO设置为ERR_NO_ALIASIO_DEF，并用错误处理器继续执行。
- 如果没有客户端（例如，FlexPendant示教器）来关注指令，则将系统变量ERRNO设置为ERR_TP_NO_CLIENT，并用错误处理器继续执行。
- 如果未在最小和最大值（参数\MinValue和\MaxValue）范围内指定初始值（参数\InitValue），则将系统变量ERRNO设置为ERR_UI_INITVALUE，并通过错误处理器继续执行。
- 如果最小值（参数\MinValue）大于最大值（参数\MaxValue），则将系统变量ERRNO设置为ERR_UI_MAXMIN，并通过错误处理器继续执行。
- 如果初始值（参数 \InitValue）并非参数\AsInteger中规定的一个整数，则将系统变量ERRNO设置为ERR_UI_NOTINT，并通过错误处理器继续执行。

限制

当频繁地执行UINumEntry时，例如，在一个回路中，避免使用过小的超时参数\MaxTime。其可能导致系统性能出现不可预测的行为，诸如FlexPendant示教器响应放慢。

语法

```
UINumEntry '('
  ['\' Header ':=' <expression (IN) of string>]
  [Message ':=' <expression (IN) of string> ]
  | ['\' MsgArray ':=' <array {*} (IN) of string>]
  ['\' Wrap]
  ['\' Icon ':=' <expression (IN) of icondata>]
  ['\' InitValue ':=' <expression (IN) of dnum>]
  ['\' MinValue ':=' <expression (IN) of dnum>]
  ['\' MaxValue ':=' <expression (IN) of dnum>]
  ['\' AsInteger]
  ['\' MaxTime ':=' <expression (IN) of num>]
  ['\' DIBreak ':=' <variable (VAR) of signaldi>]
  ['\' DIPassive]
  ['\' DOBreak ':=' <variable (VAR) of signaldo>]
  ['\' DOPassive]
  ['\' BreakFlag ':=' <var or pers (INOUT) of errnum>] ')'
```

含数据类型num的返回值的函数。

相关信息

信息，关于	请参阅
图标显示数据	第*1411*页的*icondata - 图标显示数据*
基础型用户交互消息框	第*832*页的*UIMsgBox - 用户消息对话框，基本类型*
先进型用户交互消息框	第*1317*页的*UIMessageBox - 先进型用户消息框*

下一页继续

信息，关于	请参阅
用户交互数字调节	第*1330*页的*UINumTune - 用户数字调节*
用户交互α条目	第*1292*页的*UIAlphaEntry - 用户α条目*
用户交互列表视图	第*1310*页的*UIListView - 用户列表视图*
同FlexPendant示教器等相连的系统	第*1297*页的*UIClientExist - 现有客户端*
清除运算符窗口	第*732*页的*TPErase - 擦除在FlexPendant示教器上印刷的文本*

2.196 UINumTune - 用户数字调节

手册用法

UINumTune（*User Interaction Number Tune*）用于让操作员通过可用的用户设备，例如FlexPendant示教器，调节一个数值。将一条消息写给操作员，其调节一个数值。随后，检查、批准经调节的数值，并将其转换回程序。

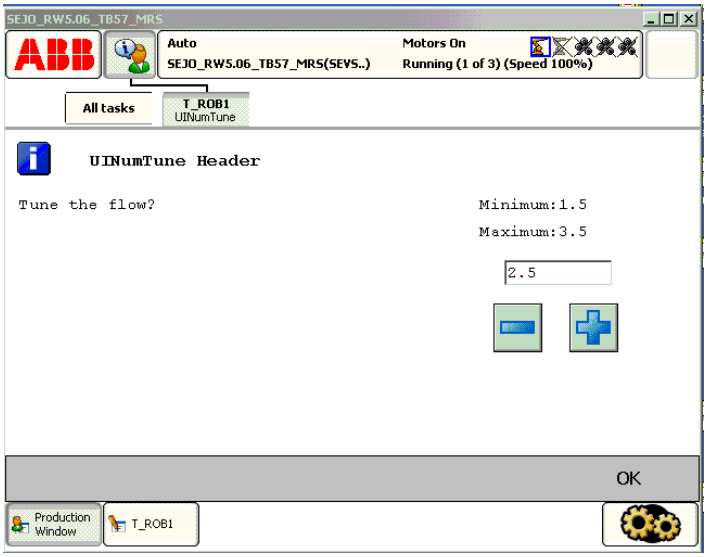
基本示例

以下示例介绍了函数UINumTune。

另请参阅第*1333*页的更多示例

例 1

```
VAR num flow;
...
flow := UINumTune(
  \Header:="UINumTune Header"
  \Message:="Tune the flow?"
  \Icon:=iconInfo,
  2.5,
  0.1
  \MinValue:=1.5
  \MaxValue:=3.5);
```



xx0500002414

在上文中，数字调节消息框以及图标、标题、消息、初始值、增量值、最大值和最小值得以写入FlexPendant示教器显示器。消息框检查操作员是否调节流值以及由初始值2.5步进0.1，且是否处于值范围1.5 .. 3.5内。程序执行进入等待，直至按下OK，随后，返回选定的数值，并将其储存在变量流中。

返回值

数据类型：num

该函数返回经调整的数值。

下一页继续

如果函数通过\BreakFlag而中断，则返回指定的\InitValue。

如果函数通过ERROR处理器而中断，则不会返回任何返回值。

变元

```
UINumTune ( [\Header] [\Message] | [\MsgArray] [\Wrap] [\Icon]
     InitValue Increment [\MinValue] [\MaxValue] [\MaxTime]
     [\DIBreak] [\DIPassive] [\DOBreak] [\DOPassive] [\BreakFlag]
     )
```

[\Header]

数据类型：string

将在消息框顶部写入的标题文本。最多40个字符。

[\Message]

数据类型：string

有待在显示器上写入的一个文本行。最多40个字符。

[\MsgArray]

Message Array

数据类型：string

一个数组的若干文本行，有待写入显示器。

仅可在同一时间使用参数\Message或\MsgArray之一。

最大布置间距为11行，每行包含40个字符。

[\Wrap]

数据类型：switch

如果选择，则参数\MsgArray中的所有指定字符串均将连接到一个字符串，各单独字符串之间仅存在单间距，并以尽可能少的行展开。

默认将在显示器上单独的行中显示参数\MsgArray中的各字符串。

[\Icon]

数据类型：icondata

定义有待显示的图标。仅可使用一种icondata型预定义图标。参见第*1333*页的预定义数据。

默认没有图标。

InitValue

数据类型：num

在输入框中显示的初始值。

Increment

数据类型：num

该参数规定了按下加或减按钮时，值所应改变的程度。

[\MinValue]

数据类型：num

有关返回值的最小值。

下一页继续

2.196 UINumTune - 用户数字调节
RobotWare - OS
续前页

[\MaxValue]

数据类型：num

有关返回值的最大值。

[\MaxTime]

数据类型：num

程序执行等待的最长时间，以秒计。如果未在该时间内按下OK按钮，则用错误处理器继续执行程序，除非使用BreakFlag（参见下文）。常量ERR_TP_MAXTIME可用于测试是否已经过最长时间。

[\DIBreak]

Digital Input Break

数据类型：signaldi

可能中断运算符对话框的数字信号输入信号。如果在将信号设置为1（或已经为1）之前未按下OK按钮，则用错误处理器继续执行程序，除非使用BreakFlag（参见下文）。可使用常量ERR_TP_DIBREAK来测试是否已出现该情况。

[\DIPassive]

Digital Input Passive

数据类型：switch

当使用DIBreak可选参数时，该开关会覆盖默认行为。将信号设置为1（或已经为1）时，取代反应；将信号DIBreak设置为0（或已经为0）时，应当用错误处理器继续本指令（如果未使用BreakFlag）。常量ERR_TP_DIBREAK可用于测试是否已出现上述情况。

[\DOBreak]

Digital Output Break

数据类型：signaldo

可能中断运算符对话框的数字信号输出信号。如果在将信号设置为1（或已经为1）之前未按下OK按钮，则用错误处理器继续执行程序，除非使用BreakFlag（参见下文）。可使用常量ERR_TP_DOBREAK来测试是否已出现该情况。

[\DOPassive]

Digital Output Passive

数据类型：switch

当使用DOBreak可选参数时，该开关会覆盖默认行为。将信号设置为1（或已经为1）时，取代反应；将信号 DOBreak设置为0（或已经为0）时，应当用错误处理器继续本指令（如果未使用BreakFlag）。常量ERR_TP_DOBREAK可用于测试是否已出现上述情况。

[\BreakFlag]

数据类型：errnum

使用\MaxTime、\DIBreak或\DOBreak时将保存错误代码的变量（使用前，由系统将其设置为0）。常量ERR_TP_MAXTIME、ERR_TP_DIBREAK和ERR_TP_DOBREAK可用于选择原因。如果省略该可选变量，则将执行错误处理器。

下一页继续

程序执行

根据编程参数，显示数字调节消息框以及调节+/-按钮、图标、标题、消息行、初始值、增量值、最大值和最小值。程序执行进入等待，直至用户已调节数值，并按下OK，或消息框由超时或信号行动中断。将输入数值和中断原因转移回程序。

从基础等级的消息框转而关注软中断等级的新消息框。

预定义数据

```
!Icons:
  CONST icondata iconNone := 0;
  CONST icondata iconInfo := 1;
  CONST icondata iconWarning := 2;
  CONST icondata iconError := 3;
```

更多示例

以下示例介绍了函数UINumTune。

例 1

```
VAR errnum err_var;
VAR num tune_answer;
VAR num distance;
...
tune_answer := UINumTune (\Header:=" BWD move on path"
     \Message:="Enter the path overlap ?" \Icon:=iconInfo, 5, 1
     \MinValue:=0 \MaxValue:=10 \MaxTime:=60 \DIBreak:=di5
     \BreakFlag:=err_var);
TEST err_var
  CASE ERR_TP_MAXTIME:
  CASE ERR_TP_DIBREAK:
    ! No operator answer
    distance := 5;
  CASE 0:
    ! Operator answer
    distance := tune_answer;
  DEFAULT:
    ! No such case defined
ENDTEST
```

显示调节消息框，且操作员可调节数值，并按下OK。消息框亦可因数字信号输入信号超时或中断而中断。在本程序中，发现原因，并采取适当的措施是可能的。

错误处理

如果未使用参数\BreakFlag，则可通过错误处理器来处理此类情形：

- 如果在运算符输入之前出现超时（参数\MaxTime），则将系统变量ERRNO设置为ERR_TP_MAXTIME，并用错误处理器继续执行。
- 如果在运算符输入之前设置数字信号输入（参数\DIBreak），则将系统变量ERRNO设置为ERR_TP_DIBREAK，并用错误处理器继续执行。
- 如果在运算符输入之前设置数字信号输出（参数\DOBreak），则将系统变量ERRNO设置为ERR_TP_DOBREAK，并用错误处理器继续执行。

下一页继续

2.196 UINumTune - 用户数字调节
RobotWare - OS
续前页

仅可通过错误处理器来处理该情形：

- 如果信号变量是在RAPID中声明的变量，且尚未与通过指令AliasIO而在I/O配置中确定的I/O信号相连，则将系统变量ERRNO设置为ERR_NO_ALIASIO_DEF，并用错误处理器继续执行。
- 如果没有客户端（例如，FlexPendant示教器）来关注指令，则将系统变量ERRNO设置为ERR_TP_NO_CLIENT，并用错误处理器继续执行。
- 如果未在最小和最大值（参数\MinValue和\MaxValue）范围内指定初始值（参数\InitValue），则将系统变量ERRNO设置为ERR_UI_INITVALUE，并通过错误处理器继续执行。
- 如果最小值（参数\MinValue）大于最大值（参数\MaxValue），则将系统变量ERRNO设置为ERR_UI_MAXMIN，并通过错误处理器继续执行。

限制

当频繁地执行UINumTune时，例如，在一个回路中，避免使用过小的超时参数\MaxTime。其可能导致系统性能出现不可预测的行为，诸如FlexPendant示教器响应放慢。

语法

```
UINumTune '('
  ['\' Header ':=' <expression (IN) of string>]
  [Message ':=' <expression (IN) of string> ]
  | ['\' MsgArray ':='<array {*} (IN) of string>]
  ['\' Wrap]
  ['\' Icon ':=' <expression (IN) of icondata>]
  [InitValue ':=' <expression (IN) of num>]
  [Increment ':=' <expression (IN) of num>]
  ['\' MinValue ':=' <expression (IN) of num>]
  ['\' MaxValue ':=' <expression (IN) of num>]
  ['\' MaxTime ':=' <expression (IN) of num>]
  ['\' DIBreak ':=' <variable (VAR) of signaldi>]
  ['\' DIPassive]
  ['\' DOBreak ':=' <variable (VAR) of signaldo>]
  ['\' DOPassive]
  ['\' BreakFlag ':=' <var or pers (INOUT) of errnum>] ')'
```

含数据类型num的返回值的函数。

相关信息

信息，关于	请参阅
图标显示数据	第*1411*页的*icondata - 图标显示数据*
基础型用户交互消息框	第*832*页的*UIMsgBox - 用户消息对话框，基本类型*
先进型用户交互消息框	第*1317*页的*UIMessageBox - 先进型用户消息框*
用户交互数字条目	第*1324*页的*UINumEntry - 用户数字条目*
用户交互α条目	第*1292*页的*UIAlphaEntry - 用户α条目*
用户交互列表视图	第*1310*页的*UIListView - 用户列表视图*
同FlexPendant示教器等相连的系统	第*1297*页的*UIClientExist - 现有客户端*

下一页继续

信息，关于	请参阅
清除运算符窗口	第*732*页的*TPErase - 擦除在FlexPendant示教器上印刷的文本*

2.197 ValidIO - 有待访问的有效I/O信号

手册用法

ValidIO用户检查是否可以访问指定的I/O信号，且当前不产生任何错误。

基本示例

以下示例介绍了函数ValidIO。

例 1

```
IF ValidIO(mydosignal) SetDO mydosignal, 1;
```

如果数字信号输出信号的I/O单元正常运行，则将数字信号输出信号mydosignal设置为1。

返回值

数据类型：bool

如果I/O信号有效，且有关信号的I/O单元正常运行，则返回TRUE.

如果I/O单元并非正常运行，或者如果尚未执行任何AliasIO指令，以便将RAPID程序中声明的信号变量连接到I/O配置中确定的信号，则返回FALSE。

变元

```
ValidIO (Signal)
```

Signal

数据类型：signalxx

I/O信号名称。必须为数据类型signaldo, signaldi, signalgo, signalgi, signalao或signalai。

程序执行

执行行为：

- 检查是否为有效的I/O信号
- 检查有关信号的I/O单元是否正常运行。

未产生错误消息。

语法

```
ValidIO '('
  [Signal ':='] <variable (VAR) of anytype> ')'
```

含数据类型bool的返回值的函数。

相关信息

信息，关于	请参阅
输入/输出指令	技术参考手册 - *RAPID*语言概览
输入/输出功能性概述	技术参考手册 - *RAPID*语言概览
I/O配置	技术参考手册 - 系统参数
确定I/O信号以及别名	第*28*页的*AliasIO - 确定I/O信号以及别名*
读取系统参数的属性	第*492*页的*ReadCfgData - 读取系统参数的属性*

2.198 ValToStr - 将一个值转换为一段字符串

手册用法

ValToStr（*Value To String*）用于将一个任意数据类型的值转换为一段字符串。

基本示例

以下示例介绍了函数ValToStr。

例 1

```
VAR string str;
VAR pos p := [100,200,300];
str := ValToStr(p);
```

变量str被赋予值"[100,200,300]"。

例 2

```
str := ValToStr(TRUE);
```

变量str被赋予值TRUE。

例 3

```
str := ValToStr(1.234567890123456789);
```

变量str被赋予值"1.23456789012346"。

例 4

```
VAR num numtype:=1.234567890123456789;

str := ValToStr(numtype);
```

变量str被赋予值"1.23457"。

例 5

```
VAR dnum dnumtype:=1.234567890123456789;

str := ValToStr(dnumtype);
```

变量str被赋予值"1.23456789012346"。

返回值

数据类型：string

将值转换为具有标准RAPID格式的一段字符串。这意味着，原则上存在6个有效数字。将字面值解释成一个dnum（参见例3），但是，dnum变量（参见例5）拥有15个有效数字。

如果生成的字符串过长，则会产生一个运行时错误。

变元

```
ValToStr ( Val )
```

Val

Value

数据类型：anytype

一个任意数据类型的值。可以使用有关结构原子、记录、记录成分、数组或数组元素的各类值数据。

下一页继续

2.198 ValToStr - 将一个值转换为一段字符串

RobotWare - OS

续前页

语法

```
ValToStr '('
  [ Val ':=' ] <expression (IN) of anytype> ')'
```

返回值的数据类型是 `string` 的函数。

相关信息

信息，关于	请参阅
字符串功能	技术参考手册 - *RAPID*语言概览
字符串的定义	第*1489*页的*string - 字符串*
字符串值	技术参考手册 - *RAPID*语言概览

2.199 VectMagn - 位置矢量的大小

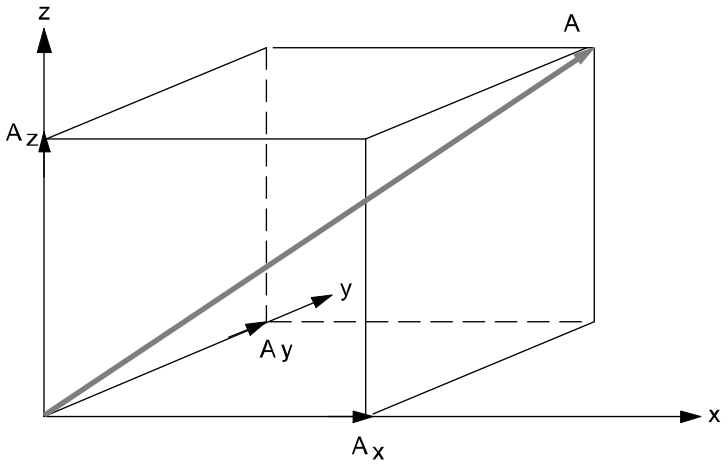
手册用法

VectMagn（*Vector Magnitude*）用于计算一个pos 矢量的大小。

基本示例

以下示例介绍了函数VectMagn。

例 1



xx0500002446

可将矢量**A**作为其三个正交方向的组件总和而写入。

$A=A_x x+A_y y+A_y y$

A的大小为：

$$|A| = \sqrt{A_x^2 + A_y^2 + A_z^2}$$

通过数据类型pos来描述矢量，并通过数据类型num来描述大小：

```
VAR num magnitude;
VAR pos vector;
...
vector := [1,1,1];
magnitude := VectMagn(vector);
```

返回值

数据类型：num

矢量的大小（数据类型pos）。

变元

```
VectMagn (Vector)
```

Vector

数据类型：pos

通过数据类型pos来描述矢量。

下一页继续

2.199 VectMagn - 位置矢量的大小

RobotWare - OS

续前页

语法

```
VectMagn '('
  [Vector ':='] <expression (IN) of pos> ')'
```

返回值的数据类型是 num 的函数。

相关信息

信息，关于	请参阅
数学指令和函数	技术参考手册 - *RAPID*语言概览

2.200 XOR - 评估一个逻辑值

手册用法

XOR（*Exclusive Or*）为用于评估一个逻辑值（真/假）的条件表达式。

基本示例

以下示例介绍了函数XOR。

例 1

```
VAR bool a;
VAR bool b;
VAR bool c;
c := a XOR b;
```

如果仅a或b之一为TRUE，则返回值c为TRUE。否则，返回值为FALSE。

例 2

```
VAR num a;
VAR num b;
VAR bool c;
...
c := a>5 XOR b=3;
```

如果仅各种条件之一为TRUE，则c的返回值为TRUE。或者a大于5，或者b等于3。否则，返回值为FALSE。

返回值

数据类型：bool

如果仅条件表达式之一正确，则返回值为TRUE，否侧，返回值为FALSE。

语法

```
<expression of bool> XOR <expression of bool>
```

含数据类型bool的返回值的函数。

相关信息

信息，关于	请参阅
AND	第*962*页的*AND - 评估一个逻辑值*
OR	第*1159*页的*OR - 评估一个逻辑值*
NOT	第*1150*页的*NOT - 转化一个逻辑值*
表达式	技术参考手册 - *RAPID*语言概览

此页刻意留白

3 数据类型

3.1 aiotrigg - 模拟I/O触发条件

手册用法

aiotrigg（*Analog I/O Trigger*）用于确定产生有关模拟信号输入或输出信号的中断所需要的条件。

描述

aiotrigg型数据定义了使用高低阈值来确定模拟信号逻辑值是否满足条件的方式，从而产生一个中断。

基本示例

以下示例介绍了数据类型aiotrigg：

例 1

```
VAR intnum sig1int;
PROC main()
  CONNECT sig1int WITH iroutine1;
  ISignalAI \Single, ai1, AIO_BETWEEN, 1.5, 0.5, 0, sig1int;
```

下达0.5和1.5之间的模拟信号输入信号ai1逻辑值首次出现的中断指令。随后，调用iroutine1 软中断程序。

预定义数据

当指定有关指令 ISignalAI和ISignalAO的一个条件时，预定义数据类型aiotrigg的以下符号常量，并且可以使用。

值	符号常量	备注
1	AIO_ABOVE_HIGH	如果高于指定高值，则信号将产生中断
2	AIO_BELOW_HIGH	如果低于指定高值，则信号将产生中断
3	AIO_ABOVE_LOW	如果高于指定低值，则信号将产生中断
4	AIO_BELOW_LOW	如果低于指定低值，则信号将产生中断
5	AIO_BETWEEN	如果介于指定低值与高值之间，则信号将产生中断
6	AIO_OUTSIDE	如果低于指定低值或高于指定高值，则信号将产生中断
7	AIO_ALWAYS	信号将始终产生中断

特征

aiotrigg为有关num的一种别名数据类型，并因此继承其特征。

相关信息

信息，关于	请参阅
模拟信号输入信号的中断	第*278*页的*ISignalAI - 模拟信号输入信号的中断*
信号输出信号的中断	第*287*页的*ISignalAO - 模拟信号输出信号的中断*
一般数据类型、别名数据类型	技术参考手册 - *RAPID*语言概览，基本特征 - 数据类型一节

3.2 ALIAS - 分配一种别名数据类型

手册用法

ALIAS用于确定一种等同于另一种数据类型的数据类型。别名类型提供了一种用于分类对象的方式。系统可使用别名分类来查找和呈现与类型相关的对象。通过一种别名定义来介绍一种别名类型。

内置别名类型为errnum和intnum，两种别名均与num有关。

errnum类型

errnum类型为有关num的一种别名，并且用于代表错误编号。

intnum类型

intnum类型为有关num的一种别名，并且用于代表中断编号。

基本示例

以下示例介绍了ALIAS定义。

例 1

```
ALIAS num level;
CONST level low := 2.5;
CONST level high := 4.0;
```

定义了一种别名类型level（有关num的别名）。

限制

为了被RAPID识别，必须在所有其他数据声明之前，于程序或系统程序模块顶部声明所有别名定义。允许在别名之前予以声明的唯一数据类型为RECORD。

无法根据另一种别名类型进行定义的一种别名类型。

语法

```
ALIAS <type name> <identifier> ';'
```

别名定义。

相关信息

信息，关于	请参阅
errnum - 错误编号	第*1395*页的*errnum - 错误编号*
intnum - 中断识别号	第*1414*页的*intnum - 中断识别号*
词汇元素	技术参考手册 - *RAPID*语言内核

3.3 bool - 逻辑值

手册用法

bool用于逻辑值（真/假）。

描述

bool型数据值可以为TRUE或FALSE。

基本示例

以下示例介绍了数据类型bool ：

例 1

```
flag1 := TRUE;
```

向标志分配值TRUE。

例 2

```
VAR bool highvalue;
VAR num reg1;
...
highvalue := reg1 > 100;
```

如果reg1大于100，则向highvalue分配值TRUE；否则，分配FALSE。

例 3

```
IF highvalue Set do1;
```

如果highvalue为TRUE，则设置do1信号。

例 4

```
highvalue := reg1 > 100;
mediumvalue := reg1 > 20 AND NOT highvalue;
```

如果reg1介于20与100之间，则向mediumvalue分配值TRUE。

相关信息

信息，关于	请参阅
逻辑表达式	技术参考手册 - *RAPID*语言概览，基本特征 - 表达式一节
使用逻辑值进行运算	技术参考手册 - *RAPID*语言概览，基本特征 - 表达式一节

3.4 btnres - 按钮结果数据

手册用法

btnres（*button result*）用于表示用户在用户设备（例如FlexPendant示教器）上按钮显示器的选择。

描述

当检查来自指令UIMsgBox的结果值以及来自指令UIMessageBox和UIListView的返回值时，旨在使用btnres常量。

基本示例

以下示例介绍了数据类型btnres：

例 1

```
VAR btnres answer;

UIMsgBox "More ?" \Buttons:=btnYesNo \Result:= answer;
IF answer= resYes THEN
  ...
ELSEIF answer =ResNo THEN
  ...
ENDIF
```

标准按钮枚举btnYesNo将在用户界面上给出一个是（Yes）和一个否（No）按钮。将用户选择储存在变量answer中。

预定义数据

在系统中预定义数据类型btnres的以下常量

值	常量	按钮应答
0	resUnkwn	未知结果
1	resOK	确定
2	resAbort	中止
3	resRetry	重试
4	resIgnore	忽略
5	resCancel	取消
6	resYes	是
7	resNo	否

对应答函数UIMessageBox和UIListView的用户定义按钮起作用是可能的。

特征

btnres为有关num的一种别名数据类型，并因此继承其特征。

相关信息

信息，关于	请参阅
用户交互消息框	第832页的*UIMsgBox - 用户消息对话框，基本类型*

下一页继续

信息，关于	请参阅
用户交互消息框	第*1317*页的*UIMessageBox - 先进型用户消息框*
用户交互列表视图	第*1310*页的*UIListView - 用户列表视图*
别名数据类型按钮数据	第*1349*页的*buttondata - 按钮数据*

3.5 busstate - I/O总线的状态

手册用法

busstate用于反映I/O总线当前所在状态。

描述

当检查指令IOBusState的返回值时，打算使用busstate常量。

基本示例

以下示例介绍了数据类型busstate：

例 1

```
VAR busstate bstate;

IOBusState "IBS", bstate \Phys;
TEST bstate
CASE IOBUS_PHYS_STATE_RUNNING:
  ! Possible to access some signal on the IBS bus
DEFAULT:
  ! Actions for not up and running IBS bus
ENDTEST
```

预定义数据

通过指令IOBusState，可观察数据类型busstate的预定义符号常量。

特征

busstate为有关num的一种别名数据类型，并因此继承其特征。

相关信息

信息，关于	请参阅
获取I/O总线的当前状态	第*262*页的*IOBusState - 获取I/O总线的当前状态*
输入/输出指令	技术参考手册 - *RAPID*语言概览，*RAPID*概要 - 输入和输出信号一节
输入/输出功能性概述	技术参考手册 - *RAPID*语言概览，运动和*I/O*原则 - *I/O*原则一节
I/O配置	技术参考手册 - 系统参数

3.6 buttondata - 按钮数据

手册用法

buttondata用于表示有关用户设备（例如FlexPendant示教器）上显示器的标准按钮组合。

描述

buttondata常量用于表示指令UIMsgBox以及函数UIMessageBox和UIListView中的响应按钮。

基本示例

以下示例介绍了数据类型buttondata：

例 1

```
VAR btnres answer;
UIMsgBox "More ?" \Buttons:=btnYesNo \Result:= answer;
IF answer= resYes THEN
...
ELSE
...
ENDIF
```

标准按钮枚举btnYesNo将给出一个是（Yes）和一个否（No）按钮。

预定义数据

在系统中预定义数据类型buttondata的以下常量。

值	常量	显示的按钮
-1	btnNone	无按钮
0	btnOK	确定
1	btnAbrtRtryIgn	中止、重试和忽略
2	btnOKCancel	确定和取消
3	btnRetryCancel	重试和取消
4	btnYesNo	是和否
5	btnYesNoCancel	是、否和取消

通过函数UIMessageBox和UIListView来显示用户定义的按钮是可能的。

特征

buttondata为有关num的一种别名数据类型，并因此继承其特征。

相关信息

信息，关于	请参阅
用户交互消息框	第*832*页的*UIMsgBox - 用户消息对话框，基本类型*
用户交互消息框	第*1317*页的*UIMessageBox - 先进型用户消息框*
用户交互列表视图	第*1310*页的*UIListView - 用户列表视图*

下一页继续

3.6 buttondata - 按钮数据
RobotWare - OS
续前页

信息，关于	请参阅
别名数据类型按钮结果	第*1346*页的*btnres - 按钮结果数据*
一般数据类型、别名数据类型	技术参考手册 - *RAPID*语言概览，基本特征 - 数据类型一节

3.7 字节-整数值0 - 255

手册用法

byte用于符合字节范围的整数值（0 - 255）。

该数据类型连同处理位操作并转换特征的指令和函数一同使用。

描述

byte型数据代表一个整数字节值。

基本示例

以下示例介绍了数据类型byte：

例 1

```
VAR byte data1 := 130;
```

含小数值130的变量data1的定义。

例 2

```
CONST num parity_bit := 8;
VAR byte data1 := 130;
BitClear data1, parity_bit;
```

将变量data1中的位号8（parity_bit）设置为0，例如，变量data1的容量将从130改变为2（整数表示）。

错误处理

如果一个byte型参数拥有一个范围0到255以外的值，则程序执行会返回一个错误。

特征

byte为有关num的一种别名数据类型，并因此继承其特征。

相关信息

信息，关于	请参阅
别名数据类型	技术参考手册 - *RAPID*语言概览，基本特征 - 数据类型一节
位功能	技术参考手册 - *RAPID*语言概览，*RAPID*概要 - 位功能一节
Advanced RAPID	产品规格 - 控制器软件*IRC5*

3.8 cameradev - 摄像头设备

手册用法

cameradev *(*摄像头设备*)*用于定义可以从 RAPID 程序控制和访问的不同摄像头设备。数据类型 cameradev 用于与摄像头的指令和函数通信。摄像头设备的名称在系统参数中定义，因此切勿在程序中定义。

描述

cameradev 类数据仅包含对摄像头设备的引用。

限制

cameradev 类数据部的在程序中定义。但是如果定义了，则会任何引用此 cameradev 的指令或函数执行时立即显示一个错误消息。另外，此数据类型也可以在声明例行程序时用作参数。

预定义数据

系统参数中定义的所有摄像头在每个程序任务中预定义。

基本示例

以下示例介绍了 cameradev 数据类型。

例 1

```
CamLoadJob mycamera, "myjob.job";
```

特征

cameradev 是一个非值数据类型。这就表示此类数据不允许面向数值的操作。

3.9 cameratarget - 摄像头数据

手册用法

cameratarget 用于从摄像头图像交换图像数据到 RAPID 程序。

描述

cameratarget 类的数据是一个用户定义的集合数据，可以用于设置为从摄像头图像交换数据到 RAPID 程序。

此数据有一系列组件，可根据当前图像应用的具体需要设置。cframe 组件旨在传输对象的位置信息，数值以及字符串用于保存检查数据。

组件

此数据类型有以下组件：

name

数据类型：string

cameratarget 的名称标识符。

cframe

current frame

数据类型：pose

用于存储位置数据，通常用于通过修改工件引导机器人。

val1

value 1

数据类型：num

用于存储测量结果等数值输出。

...

val5

value 5

数据类型：num

用于存储测量结果等数值输出。

string1

数据类型：string

用于存储检查或识别输出等数值图像输出。

string2

数据类型：string

用于存储检查或识别输出等数值图像输出。

type

数据类型：num

摄像头对象的数字标识符。与 name 组件作用类似。

下一页继续

3.9 cameratarget - 摄像头数据

Integrated Vision

续前页

`cameraname`

数据类型：`string`

摄像头名称。

`sceneid`

scene identification

数据类型：`num`

用于产生 `cameratarget` **的图像的唯一标识符。**

基本示例

以下示例介绍了 `cameratarget` **数据类型。**

例 1

```
VAR cameratarget target1;
...
wobjmycamera.oframe := target1.cframe;
MoveL pickpart, v100, fine, mygripper \WObj:= wobjmycamera;
```

`cframe` **坐标转换分配到工件的对象框架。**`robtarget` `pickpart` **此前已经在工件框架内微调为正确的捡取位置。**

结构

```
< dataobject of cameratarget >
  < name of string >
  < cframe of pose >
    < trans of pos >
    < rot of orient >
  < val1 of num >
  < val2 of num >
  < val3 of num >
  < val4 of num >
  < val5 of num >
  < string1 of string >
  < string2 of string >
  < type of num >
  < cameraname of string >
  < sceneid of num >
```

3.10 capdata - CAP数据

手册用法

capdata包含定义CAP进程的行为所需的所有数据。

组件

start_fly

飞焊开始

数据类型：bool

定义是否执行飞焊开始。

值	后果
TRUE	飞焊开始被执行
FALSE	飞焊开始未被执行

飞焊开始意味着在进程启动前机器人运动已经开始。进程随后在运行中被启动（参见第*1407*页的*flypointdata - 飞焊开始/结束数据*）。

end_fly

飞焊结束

数据类型：bool

定义是否执行飞焊结束。

值	后果
TRUE	飞焊结束被执行
FALSE	飞焊结束未被执行

飞焊结束意味着在机器人到达终点前，CAP进程可以结束，从而能利用一个区域点，让机器人离开运行所在的进程路径（参见第*1407*页的*flypointdata - 飞焊开始/结束数据*）。

first_instr

首个指令

数据类型：bool

定义CapL/CapC指令是否是CapL/CapC指令序列中的首个指令：

值	后果
TRUE	这是CapL/CapC指令序列中的首个指令
FALSE	这不是CapL/CapC指令序列中的首个指令

last_instr

最后一个指令

数据类型：bool

定义CapL/CapC指令是否是CapL/CapC指令序列中的最后一个指令：

值	后果
TRUE	这是CapL/CapC指令序列中的最后一个指令

下一页继续

3.10 capdata - CAP数据
Continuous Application Platform (CAP)
续前页

值	后果
FALSE	这不是CapL/CapC指令序列中的最后一个指令

restart_dist

重启距离，单位：毫米

数据类型：num

定义在CAP进程处于活跃状态时，在已出现停止后，在重启期间，机器人必须沿路径返回的距离。

在MultiMove系统中，所有同步机器人必须采用同样的重启距离。

speed_data

*CAP*的速度数据

数据类型：capspeeddata

定义有关速度的所有CAP数据（参见第*1361*页的*capspeeddata - CAP*的速度数据）。

start_fly_point

数据类型：flypointdata

当start_fly为TRUE时，仅应考虑这类数据。

定义CAP进程的飞焊开始信息（参见第*1407*页的*flypointdata - 飞焊开始/结束数据*）。

end_fly_point

数据类型：flypointdata

当start_fly为TRUE时，仅应考虑这类数据。

定义CAP进程的飞焊结束信息（参见第*1407*页的*flypointdata - 飞焊开始/结束数据*）。

sup_timeouts

数据类型：supervtimeouts

定义所有摇手监控阶段采用的超时（参见第*1492*页的*supervtimeouts - 摇手监控超时*以及应用手册 - *Continuous Application Platform*中的监控章节）。

proc_times

数据类型：processtimes

定义状态监控阶段PRE、POST1和POST2采用的超时（参见第*1452*页的*processtimes - 进程时间*以及应用手册 - *Continuous Application Platform*中的监控和进程阶段）。

block_at_restart

数据类型：restartblkdata

定义重启期间CAP进程的行为（参见第*1457*页的*restartblkdata - 重启用块数据*）。

结构

```
< data object of capdata >
  < start_fly of bool >
  < end_fly of bool >
  < first_instr of bool >
  < last_instr of bool >
  < restart_dist of num >
  < speed_data of capspeeddata >
    < fly_start of num >
```

下一页继续

```
    < start of num >
    < startspeed_time of num >
    < startmove_delay of num >
    < main of num >
    < fly_end of num >
  < start_fly_point of flypointdata >
    < from_start of bool >
    < time_before of num >
    < distance of num >
  < end_fly_point of flypointdata >
    < from_start of bool >
    < time_before of num >
    < distance of num >
  < sup_timeouts of supervtimeouts >
    < pre_cond of num >
    < start_cond of num >
    < end_main_cond of num >
    < end_post1_cond of num >
    < end_post2_cond of num >
  < proc_times of processtimes >
    < pre of num >
    < post1 of num >
    < post2 of num >
  < block_at_restart of restartblkdata >
    < weave_start of bool >
    < motion_delay of bool >
    < pre_phase of bool >
    < startspeed_phase of bool >
    < post1_phase of bool >
    < post2_phase of bool >
```

相关信息

	参见：
Continuous Application Platform	应用手册 - *Continuous Application Platform*
capspeeddata数据类型	第*1361*页的*capspeeddata - CAP*的速度数据
flypointdata数据类型	第*1407*页的*flypointdata* - 飞焊开始/结束数据
supervtimeouts数据类型	第*1492*页的*supervtimeouts* - 摇手监控超时
processtimes数据类型	第*1452*页的*processtimes* - 进程时间
block_at_restart数据类型	第*1457*页的*restartblkdata* - 重启用块数据
CapL指令	第*79*页的*CapL - CAP*线性运动指令
CapC指令	第*67*页的*CapC - CAP*圆周运动指令

3.11 caplatrackdata - CAP Look-Ahead-Tracker跟踪数据

手册用法

caplatrackdata包含有数据，可供用户用于影响CapL/CapC指令对Look-Ahead-Tracker（比如，激光跟踪器）产生的路径校正数据的纳入方式。caplatrackdata是captrackdata的一部分。

组件

joint_no

数据类型：num

定义跟踪期间相关传感器设备应采用的关节类型（用一个数字表示）。

filter

数据类型：num

定义低通传感器用于路径校正的时间常量。分量可设置为处于1至10之间的值，其中，1对传感器检测到的路径错误给出最快的响应（无滤波）。

calibframe_no

数据类型：num

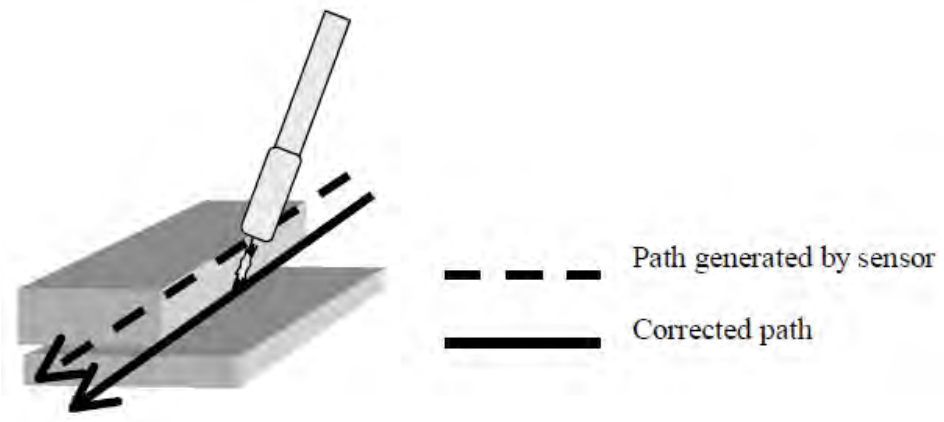
定义应采用CapLATrSetup中定义的三个坐标系中的哪个校准坐标系。

值	校准坐标系	描述
1	calibframe	在CapLATrSetup中是强制性的
2	calibframe2	在CapLATrSetup中是可选的
3	calibframe3	在CapLATrSetup中是可选的)

seamoffs_y、seamoffs_z

数据类型：num

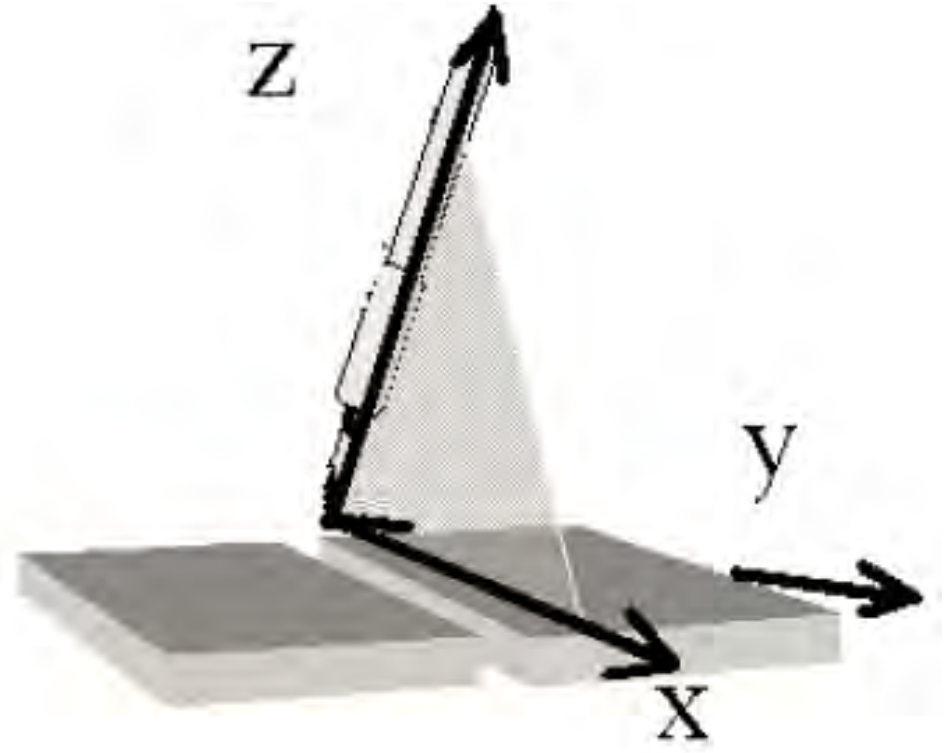
利用焊缝偏移分量来将恒定偏移量添加到传感器产生路径（单位：毫米）。如果举例而言，传感器认为搭接接头的上边缘为正确的焊缝位置，如下图所示，那么，可利用焊缝偏移量来校正路径。



xx1200000199

下一页继续

校正被定义在路径坐标系中，路径坐标系位于右手侧。



xx1200000200

- x方向平行于路径切线
- z方向是工具z矢量。
- x方向垂直于x方向和z方向所在的平面。

seamadapt_y、seamadapt_z

数据类型：num

自适应焊缝调节设置类似于焊缝偏移调节参数。自适应焊缝偏移参数不是固定值。它由自适应调节参数与测量的焊缝坡口宽度的乘积获得。

利用这些分量来相对于焊缝，调整工具偏移量，从而优化不同坡口尺寸的工艺。

track_mode

数据类型：num

凭借track_mode分量，可以选择性地影响激光跟踪器的跟踪行为。

值	跟踪模式
0	法向跟踪。同时考虑到y校正和z校正。
1	如同激光跟踪器发送的y校正为零一样进行跟踪。考虑到z校正。 [i]
2	如同激光跟踪器发送的z校正为零一样进行跟踪。考虑到y校正。 [i]
3	如同激光跟踪器发送的y校正和z校正为零一样进行跟踪。 [i]
4	y校正总体关闭，即，在发送到机器人之前，y分量的校正被设为零。考虑到z校正。 [ii]

下一页继续

3.11 caplatrackdata - CAP Look-Ahead-Tracker跟踪数据

Continuous Application Platform (CAP)

续前页

值	跟踪模式
5	z校正总体关闭，即，在发送到机器人之前，z分量的校正被设为零。考虑到y校正。 [ii]
6	y校正和z校正总体关闭，即，在发送到机器人之前，y分量和z分量的校正被设为零。 [ii]
7	TCP在Y方向渐渐地向示教路径逼近，Z方向保持跟踪模式。
8	TCP在Z方向渐渐地向示教路径逼近，Y方向保持跟踪模式。
9	TCP在Y方向和Z方向渐渐地向示教路径逼近。
10	TCP在Y方向渐渐地远离示教路径，Z方向保持跟踪模式。
11	TCP在Z方向渐渐地远离示教路径，Y方向保持跟踪模式。
12	TCP在Y方向和Z方向渐渐地远离示教路径。

i 对于track_mode 1、2或3，来自先前CapL/CapC指令的累积校正将为y和/或z进行保存，并传递到下一个CapL/CapC指令。在CAP进程的整个寿命中均如此。新CAP进程不会受到影响。

ii 对于track_mode取值为4、5或6时，虽然在发送给机器人本体的Y方向和/或Z方向的偏移为0，但之前的传感器累计的读数还是存在的，这就意味着，在CapL/CapC指令开始和结束的位置，TCP移动会发生一次突变抖动。

基本示例

```
PERS captrackdata captrack := ["laser1:",50,[1,10,1,0,0,0,0,0]]
CapL p1, v200, cd, wsd, cwd, z20, tWeldGun \Track:=captrack;
```

语法

```
< data object of caplatrackdata >
  < joint_no of num >
  < filter of num >
  < calibframe_no of num >
  < seamoffs_y of num >
  < seamoffs_z of num >
  < seamadapt_y of num >
  < seamadapt_z of num >
  < track_mode of num >
```

相关信息

	参见：
captrackdata数据类型	第1363页的*captrackdata - CAP*跟踪数据
Continuous Application Platform	应用手册 - *Continuous Application Platform*

3.12 capspeeddata - CAP的速度数据

手册用法

capspeeddata用于定义有关CAP进程速度的所有数据 - 它是capdata的一部分，定义了CAP进程所需的所有速度数据和进程时间：

- 速度以及在CAP进程开始时该速度应采用多久，
- 相对于CAP进程的开始，机器人运动的延时，
- CAP进程的速度，

速度受到机器人性能限制。根据机器人类型和运动路径，这有所不同。

组件

fly_start

数据类型：num

未使用。

start

数据类型：num

CAP进程开始时采用的速度（单位：mm/s）。

startspeed_time

数据类型：num

以start速度运行的时间（单位：秒）。

startmove_delay

数据类型：num

相对于CAP进程的开始，机器人运动延迟的时间（单位：秒）。

main

数据类型：num

主CAP进程速度（mm/s）。

fly_end

数据类型：num

未使用。

结构

```
< data object of capspeeddata >
  < fly_start of num >
  < start of num >
  < startspeed_time of num >
  < startmove_delay of num >
  < main of num >
  < fly_end of num >
```

下一页继续

3.12 capspeeddata - CAP的速度数据

Continuous Application Platform (CAP)

续前页

相关信息

	参见：
`capdata`数据类型	第*1355*页的*capdata - CAP*数据
Continuous Application Platform	应用手册 - *Continuous Application Platform*

3.13 captrackdata - CAP跟踪数据

手册用法

captrackdata利用Look-Ahead-或At-Point-Tracker，向CapL/CapC指令提供了路径校正所必需的所有数据。这些数据将利用可选参数\Track，传递到CapL/C指令。

device分量决定了将采用哪类跟踪器。在CapL/CapC指令序列中，无法改变captrackdata。利用第一个CapL/C指令设置device分量 - 如果在同一CapL/CapC指令序列的其余CapL/C指令中该分量不同，那么不会有任何影响。

为了能改变CapL/CapC指令中将使用的captrackdata，必须通过在capdata中将last_inst分量设为TRUE来首先结束此序列。

如果在第一个CapL/C指令以及同一CapL/CapC指令序列的所有后续指令中不存在\Track，那么将不施加任何校正。

组件

device

传感器装置

数据类型：string

定义传感器将连接哪个装置来用于CapL/CapC指令以便生成路径校正。

max_corr

允许的最大路径校正

数据类型：num

定义允许的最大路径校正

对于Look-Ahead跟踪器：

- 如果路径校正引起的TCP偏移量超出max_corr并且在CapLATrSetup中指定了\WarnMaxCorr，那么机器人会继续遵循自己的路径，但施加的路径校正不会超出max_corr。
- 如果未指定\WarnMaxCorr，那么，将报告跟踪错误，程序将停止执行。

对于At-Point跟踪器：

- 如果因路径校正，TCP偏移量超出max_corr，那么将报告跟踪错误，程序将停止执行。

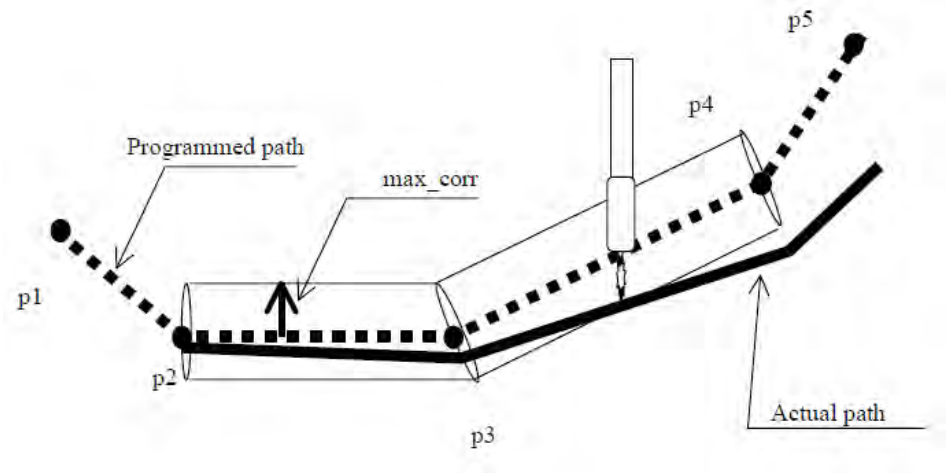
下一页继续

3.13 captrackdata - CAP跟踪数据

Continuous Application Platform (CAP)

续前页

图中展示了在报告max_corr跟踪错误的情况下，相对于编程路径，处于位置上的工具，单位：毫米。



xx1200000198

la_trackdata

*Look-Ahead-Tracker*跟踪数据

数据类型：caplatrackdata

定义Look-Ahead-Trackers（比如，激光跟踪器）特定的跟踪数据。

基本示例

SIO.cfg:

```
COM_TRP:
-Name "SCOUT:" -Type "RTP1"
-Name "digi-ip:" -Type "SOCKDEV" -PhyChannel "LAN1" -RemoteAdress
     "192.168.125.5"
```

RAPID程序：

```
PERS captrackdata captrack1 := ["digi-ip:",50,[1,10,1,0,0,0,0,0]];
CONST string laser := "digi-ip:";
PERS pose pose1 := [[137.867,-326.31,18.5],
     [0.640984,0.766438,0.0348674,0.0223137]];
PROC main()
  VAR pos sensorPos;
  CapLATrSetup laser, pose1, pos \SensorFreq:=10 \CorrFilter:=5
       \MaxBlind:=100 \MaxIncCorr:=2;
  WriteVar laser, 6, 1;
  ! sensor ON
  CapL p1, v200, cd, wsd_event, cwd, z20, tWeldGun
       \Track:=captrack1;
  CapC p2, p3, v200, cd2, wsd, cwd, z20, tWeldGun \Track:=captrack1;
  CapL p4, v200, cd3, wsd, cwd, fine, tWeldGun \Track:=captrack1;
  WriteVar laser, 6, 0;
  ! sensor OFF
ENDPROC
```

下一页继续

语法

```
< data object of captrackdata >
  < device of string >
  < max_corr of num>
  < la_trackdata of caplatrackdata >
```

相关信息

	参见：
caplatrackdata	第*1358*页的*caplatrackdata - CAP Look-Ahead-Tracker*跟踪数据
CapAPTrSetup	第*65*页的*CapAPTrSetup - 设置At-Point-Tracker*
CapLATrSetup	第*86*页的*CapLATrSetup - 设置Look-Ahead-Tracker*
Continuous Application Platform	应用手册 - *Continuous Application Platform*

3.14 capweavedata - CAP摆动数据

手册用法

capweavedata将用于定义处于主阶段时CAP进程的摆动（参见应用手册 - *Continuous Application Platform*）。

摆动描述

摆动在进程的基本路径上重叠，即，进程速度（在capspeeddata中定义）保持为定义速度，但TCP速度增加，直至达到机器人实际限值。

可用的摆动类型：

- 几何摆动：最准确形状
- 腕摆动：仅对摆动采用机器人轴6
- 快速摆动轴1-3：仅对摆动采用机器人轴1、2和3
- 快速摆动轴4-6：仅对摆动采用机器人轴4、5和6

可用的摆动形状：

- “之”字形摆动
- V字形摆动
- 三角形摆动
- 圆弧形摆动

所有capweavedata分量适用于主阶段。

组件

路径坐标系的定义方式：

- X：路径/运动方向
- Z：工具z方向
- Y：为了构建右手侧坐标系，同时垂直于X和Z

active

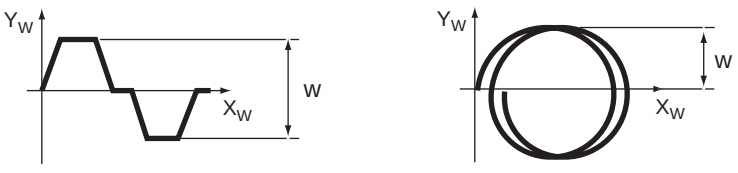
数据类型：bool

值	描述
TRUE	在CAP进程主阶段中进行摆动
FALSE	在CAP进程主阶段中不进行摆动

width

数据类型：num

对于圆弧形摆动，宽度为圆弧的半径（下图中的W）。对于其他摆动形状，宽度为摆动运动的总幅度。



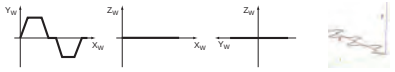
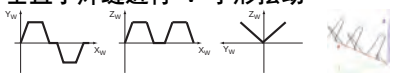
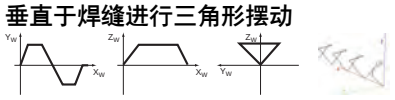
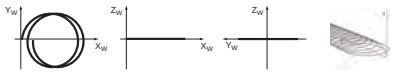
xx1200000721

下一页继续

shape

数据类型：num

主阶段中摆动运动的形状。

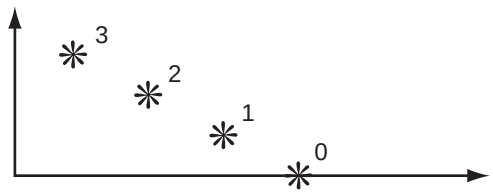
值	形状的几何结构	结果
0	不摆动	
1	“之”字形摆动	相对于焊缝进行水平摆动  xx1200000714
2	V字形摆动	垂直于焊缝进行“V”字形摆动  xx1200000715
3	三角形摆动	垂直于焊缝进行三角形摆动  xx1200000716
4	圆弧形摆动 （仅可为几何摆动提供，摆动类型0）	垂直于焊缝进行圆弧形摆动  xx1200000717

type

数据类型：num

在主阶段中定义为摆动采用哪些轴

指定值	摆动类型
0	几何摆动。摆动期间使用所有轴。
1	腕摆动。
2	快速摆动。使用轴1、2和3。
3	快速摆动。使用轴4、5和6。



xx1200000718

下一页继续

3.14 capweavedata - CAP摆动数据

Continuous Application Platform (CAP)

续前页

注意

相较TrueMove/QuickMove第一代，采用TrueMove/QuickMove第二代的所有机器人产生了RW弧和CAP中可用的不同摆动类型的下列变更行为：

- 几何摆动：无变化。
- 腕摆动：主要采用腕轴（4、5和6），但也可向主轴添加小校正，以便能够将摆动运动保持在想要处于的平面。
- 快速摆动：在TrueMove/QuickMove第二代中，几何摆动和腕摆动均实现性能大幅提升。因此，快速摆动（两种摆动）无需再作为特殊摆动类型。

  快速摆动轴1、2和3与几何摆动的一样。

  快速摆动轴4、5和6与腕摆动的一样。

  针对步退兼容性，仍可提供这些摆动类型。

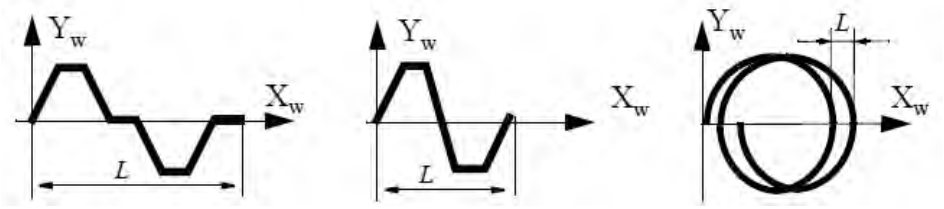
如果在*MOTION_PLANNER* 数据中的MOC.cfg中有开关`dyn_ipol_type 1`，那么，系统采用TrueMove/QuickMove第二代。

length

数据类型：`num`

在几何摆动（类型= 0）和腕摆动（类型= 1）的主阶段中定义摆动圆弧长度。不为其他摆动类型采用长度参数。

对于圆弧形摆动，如果`cycle_time`参数设为0，那么，`length`分量定义了两个相继圆弧之间的距离（L）。如果`cycle_time`具备一个值，那么长度可被替代。长度值为正时，TCP向左旋转，长度值为负时，向右旋转。



xx1200000187

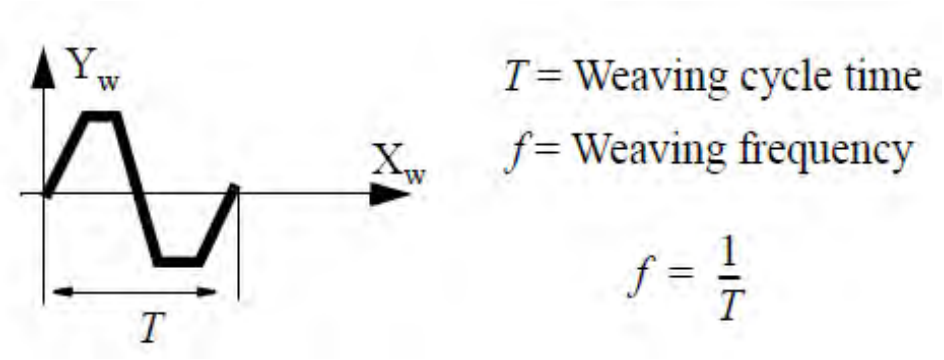
cycle_time

数据类型：`num`

在两类快速摆动（类型 = 2或3）以及圆弧形摆动的主阶段，定义摆动频率（单位：Hz）。不为另两种摆动类型采用`cycle_time`参数。

下一页继续

对于圆弧形摆动，cycle_time参数定义了每秒的圈数。在cycle_time值为正的情况下，TCP向左旋转，在cycle_time值为负的情况下，TCP向右旋转。



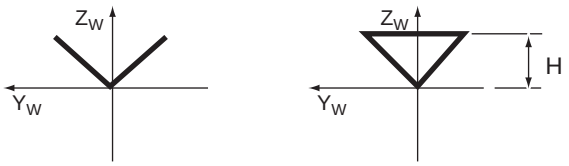
xx1200000188

height

数据类型：num

定义V形摆动和三角形摆动期间摆动运动的高度（单位：毫米）。

不为圆弧形摆动提供。

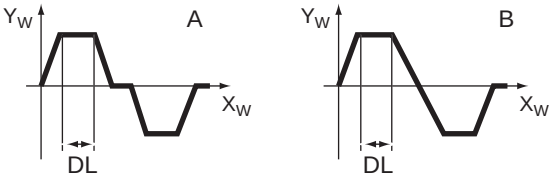


xx1200000722

dwell_left

数据类型：num

用于强制TCP在摆动的左旋转点仅沿焊缝方向移动的停延长度（DL）。不为圆弧形摆动提供。



xx1200000723

A	“之”字形摆动和V形摆动
B	三角形摆动

下一页继续

3.14 capweavedata - CAP摆动数据

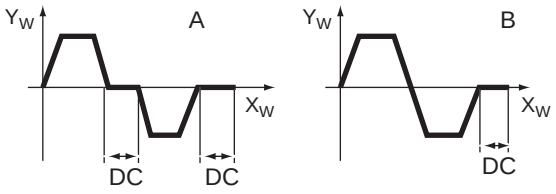
Continuous Application Platform (CAP)

续前页

dwell_center

数据类型：num

用于强制TCP在摆动中心点仅沿焊缝方向移动的停延长度（DC）。不为圆弧形摆动提供。



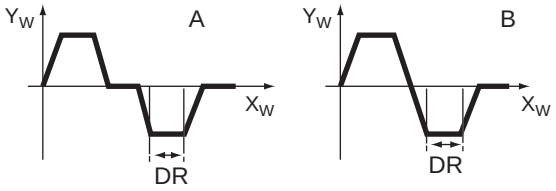
xx1200000724

A	“之”字形摆动和V形摆动
B	三角形摆动

dwell_right

数据类型：num

用于强制TCP在摆动的右旋转点仅沿焊缝方向移动的停延长度（DR）。不为圆弧形摆动提供。



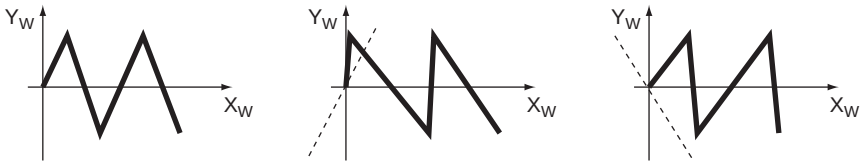
xx1200000725

A	“之”字形摆动和V形摆动
B	三角形摆动

dir

数据类型：num

与焊缝呈水平的摆动方向角度。角度为零度，会让摆动垂直于焊缝。

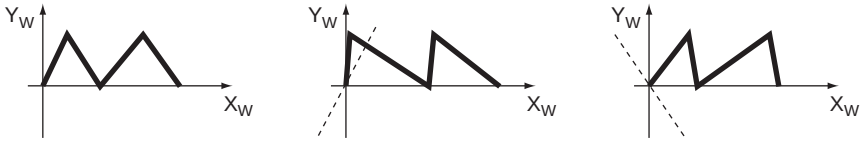


xx1200000726

tilt

数据类型：num

垂直于焊缝的摆动倾斜角度。角度为零度，会让摆动垂直于焊缝。



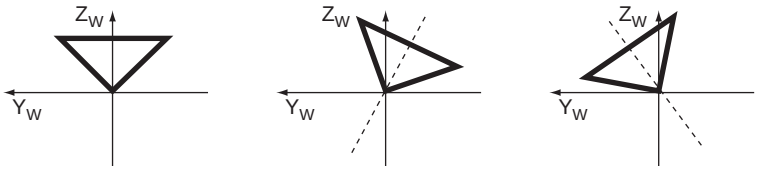
xx1200000727

下一页继续

rot

数据类型：num

与焊缝呈纵横态的摆动姿态角度。角度为零度，会造成对称摆动。



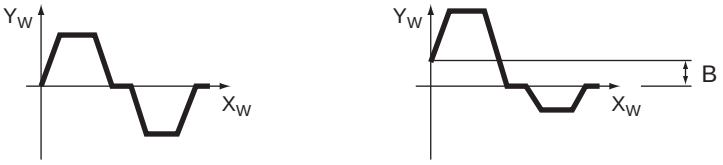
xx1200000728

bias

数据类型：num

与摆动运动呈水平的偏斜。仅可为“之”字形摆动指定偏斜量。偏斜量不可大于摆动宽度的一半。不为圆弧形摆动提供。

下图展示了带偏斜和不带偏斜（B）的“之”字形摆动。



xx1200000729

ptrn_sync_on

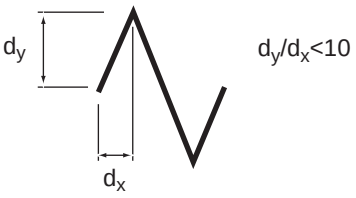
数据类型：bool

值	描述
TRUE	在摆动运动的右旋转点和左旋转点处，发送同步脉冲。
FALSE	在摆动运动的右旋转点和左旋转点处，不发送同步脉冲。

限制

最大摆动频率为2Hz。

摆动运动的倾斜率不得超出1:10（84度）。请参见下图。



xx1200000730

在区域点，weavedata中的weave_type不可能改变，只有在精确点才能改变。这是样条&decbuf插补器的行为。相较*TrueMove*或*QuickMove*第一代，采用*TrueMove*或*QuickMove*第二代的所有机器人，产生了RW弧可提供的不同摆动类型的下列变更行为：

- 几何摆动 - 无变化。
- 腕摆动 - 主要采用腕轴（4、5和6），但也可向主轴添加小校正，以便能够将摆动运动保持在想要处于的平面。

下一页继续

3.14 capweavedata - CAP摆动数据

Continuous Application Platform (CAP)

续前页

- 快速摆动 - 在*TrueMove*或*QuickMove*第二代中，几何摆动和腕摆动均实现性能大幅提升。因此，快速摆动（两种摆动）无需再作为特殊摆动类型。

  快速摆动轴1、2和3与几何摆动的一样。

  快速摆动轴4、5和6与腕摆动的一样。

  针对步退兼容性，仍可提供这些摆动类型。

如果在MOTION_PLANNER数据（系统参数）中的MOC.cfg中有开关`dyn_ipol_type 1`，那么，系统采用*TrueMove*或*QuickMove*第二代。

语法

```
< data object of capweavedata >
  < active of bool>
  < width of num >
  < shape of num >
  < type of num >
  < length of num >
  < cycle_time of num >
  < height of num >
  < dwell_left of num >
  < dwell_center of num >
  < dwell_right of num >
  < dir of num >
  < tilt of num >
  < rot of num >
  < bias of num >
  < ptrn_sync_on of bool >
```

相关信息

	参见：
`capdata`数据类型	第*1355*页的*capdata - CAP*数据
Continuous Application Platform	应用手册 - *Continuous Application Platform*

3.15 cfgdomain - 配置域

手册用法

cfgdomain (配置域)用于指定一个配置域。

描述

cfgdomain型数据旨在用于确定应当通过指令SaveCfgData来保存的配置域。

基本示例

以下示例介绍了数据类型cfgdomain：

例 1

```
SaveCfgData "SYSPAR" \File:="MYEIO.cfg", EIO_DOMAIN;
```

将I/O域配置保存至路径SYSPAR中的文件MYEIO.cfg。

预定义数据

以下预定义常量可用于指定一个配置域。

名称	描述
EIO_DOMAIN	I/O系统配置
MOC_DOMAIN	运动配置
SIO_DOMAIN	通信域
PROC_DOMAIN	过程域
SYS_DOMAIN	控制器域
MMC_DOMAIN	人机连接
ALL_DOMAINS	上文列出的所有域

特征

cfgdomain为有关string的一种别名数据类型，并因此继承其特征。

相关信息

信息，关于	请参阅
将系统参数保存至文件	第*545*页的*SaveCfgData - 将系统参数保存至文件*
系统参数	技术参考手册 - 系统参数

3.16 时钟 - 时间测量

手册用法

Clock用于时间测量。一个功能类似秒表的时钟，用于定时。

描述

clock型数据储存时间测量值，以秒计，且分辨率为0.001秒。

基本示例

以下示例介绍了数据类型clock：

例 1

```
VAR clock myclock;
ClkReset myclock;
```

声明和重置时钟myclock,。在使用ClkReset、ClkStart、ClkStop和ClkRead之前，必须在程序中声明一个数据类型clock的变量。

限制

可储存在时钟变量中的最长时间大约为49天（4,294,967秒）。指令ClkStart、ClkStop和ClkRead报告极不可能出现事件中的时钟溢出。

必须将时钟声明为一个VAR变量类型，而非一个persistent变量类型。

特征

clock为非值数据类型，且无法用于以值为导向的运算。

相关信息

信息，关于	请参阅
时间和日期指令的概要	技术参考手册 - *RAPID*语言概览，*RAPID*概要 - 系统&时间一节
非值数据类型特征	技术参考手册 - *RAPID*语言概览，基本特征 - 数据类型一节

3.17 confdata - 机械臂配置数据

手册用法

confdata用于定义机械臂的轴配置。

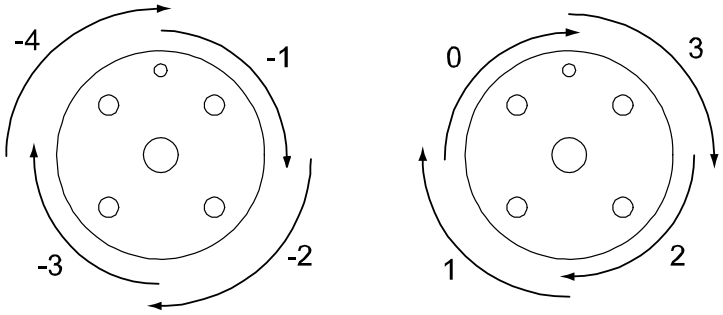
描述

通过使用直角坐标系，定义和储存机械臂的所有位置。当计算相关的轴位置时，通常可能存在两种或更多的解决方案。这意味着机械臂能够达到相同的位置，即工具处于相同的位置且具有相同的方向，机械臂轴具有多种不同的位置或配置。

一些机械臂类型使用迭代数值法来确定机械臂轴位置。在此类情况下，配置参数可用于确定有关迭代无返回值程序所用接头的良好起始值。

为明白地表示可能配置之一，通过使用四个轴的值来指定机械臂配置。针对一个旋转轴，该值确定机械臂轴的当前象限。将象限编号0、1、2等等（其亦可为负）。象限编号与轴的当前接头角相关。对于各个轴，象限0为四个旋转中的第一个，0到90°，处于始于零位置的正方向；象限1为下一个旋转，90到180°，以此类推。象限-1为旋转0°到（-90°），以此类推。

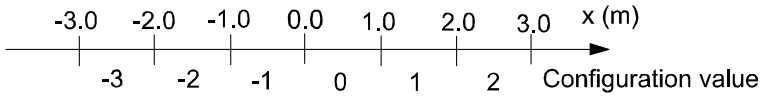
本图显示有关轴6的配置象限。



xx0500002398

对于线性轴，该值规定有关机械臂轴的间隔米数。对于各个轴，值0意味着一个介于0到1米之间的位置，值1意味着一个介于1到2米之间的位置。对于负值，-1意味着一个介于-1到0米之间的位置，以此类推。

本图显示了有关一个线性轴的配置值。



xx0500002399

配置监督

对于一些机械臂模型而言，如果ConfL\On得以设置，则配置数据（confdata）亦用于对有关线性移动的编程点实施监督。

在开始进行指定运动之前，进行验证，以查看是否可能实现编程配置。如果不可能，则停止程序。完成运动时（在区域或精点中），同时验证机械臂已经达到编程配置。

下一页继续

3.17 confdata - 机械臂配置数据
RobotWare - OS
续前页

不同机械臂的配置监督有所不同。有关细节，请参见以下章节。

6轴机械臂

配置监督将检查轴1、4和6是否不会移动180度以上，且安排的移动无需改变cfx（cfx仅用于串行线机械臂）。

4轴机械臂

配置监督将检查轴1和6是否不会移动180度以上。

平行臂机械臂

配置监督将检查轴4是否不会移动180度以上。

SCARA机器人

配置监控将核实轴1和4不会移动超出180度。也会核实轴2的标志。

7轴机械臂

配置监督将检查轴1、4和6是否不会移动180度以上，且安排的移动无需改变cfx。

涂漆机械臂

未实施任何配置监督。

机械臂配置数据

含串行线的6轴机械臂

在机械臂工作范围内存在三个奇异点。欲知有关奇异点的更多信息，请参见技术参考手册 - *RAPID*语言概览。

- cf1为轴1的象限编号。
- cf4为轴4的象限编号。
- cf6为轴6的象限编号。

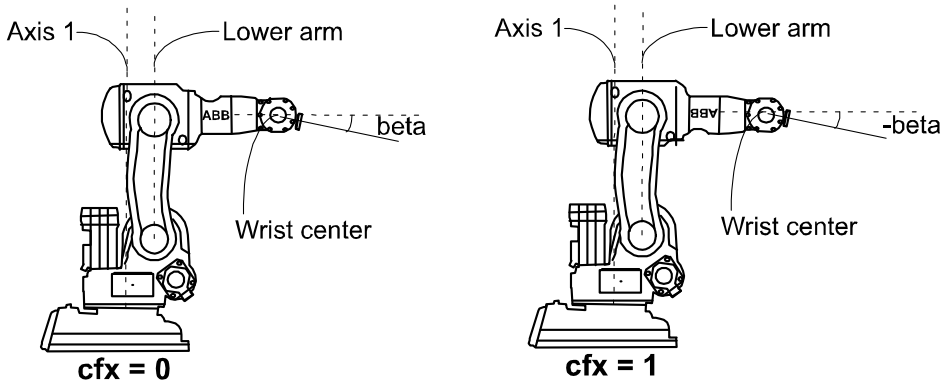
cfx用于从编号0到7的八种可能的机械臂配置中选择一种。下表描述了有关如何相对于三个奇异点来放置机械臂的各种配置。

cfx	相对于轴1的腕中心。	相对于下臂的腕中心。	轴5角
0	在前面	在前面	正
1	在前面	在前面	负
2	在前面	在后面	正
3	在前面	在后面	负
4	在后面	在前面	正
5	在后面	在前面	负
6	在后面	在后面	正
7	在后面	在后面	负

下图描述了有关相同工具位置和方位的八种不同的配置。

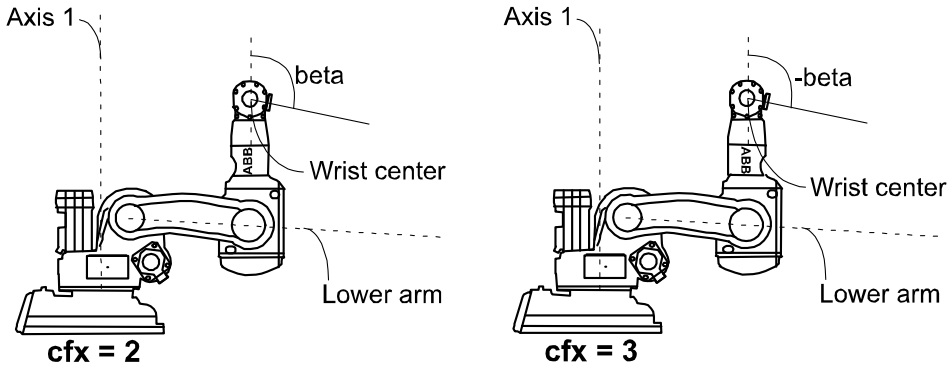
下一页继续

下图显示了一个机械臂配置0和1的例子。注意轴5角的不同标志。



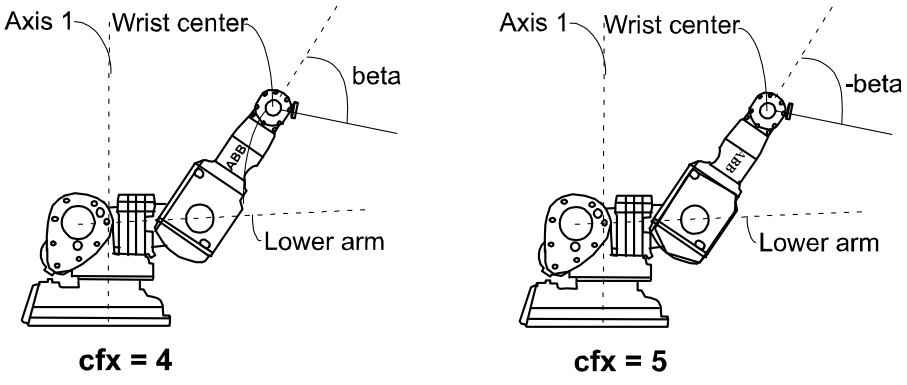
xx0500002400

下图显示了一个机械臂配置2和3的例子。注意轴5角的不同标志。



xx0500002401

下图显示了一个机械臂配置4和5的例子。注意轴5角的不同标志。

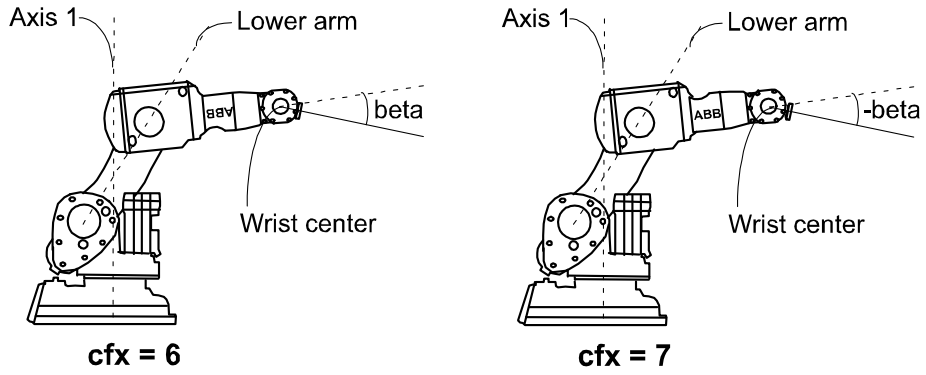


xx0500002402

下一页继续

3.17 confdata - 机械臂配置数据
RobotWare - OS
续前页

下图显示了一个机械臂配置6和7的例子。注意轴5角的不同标志。



xx0500002403

含平行杆的6轴机械臂

仅使用三种配置参数cf1、cf4和cf6。

4轴机械臂

仅使用配置参数cf6。

平行臂机械臂

仅使用配置参数cf4。

SCARA机器人

仅采用三个配置参数cf1、cf4和cfx。

采用cfx值来展示轴2角度的标志。如果轴2角度为负，那么，cfx为1，否则，cfx为0。

7轴机械臂

使用全部四种配置参数。分别针对接头1、4和6使用cf1、cf4、cf6。cfx用于从16种可能的机械臂配置中选择一种。

将cfx值表示为从‘0000’到‘1111’的十进制形式的比特字符串。下表描述了有关如何放置机械臂的各种形式。

cfx	描述
第四位（1000）	如果轴5的角度等于零，或拥有一个正值，则第四位为0。 否则，第四位为1。
第三位（0100）	如果轴3的角度大于或等于-90度，则第三位为0。 否则，第三位为1。
第二位（0010）	如果轴2的角度等于零，或拥有一个正值，则第二位为0。 否则，第二位为1。
第一位（0001）	第一位为具有兼容性的一位。当编程线性移动时，兼容性位应当与先前目标相同。

注意

注意，并未显示前导字符零，参见以下例子。

下一页继续

有关cfx的配置实例：

- cfx = '0000' = 0
  轴5 = 15度，轴3 = -55度，轴2 = 0度，兼容性位 = 0
- cfx = '0110' = 110
  轴5 = 15度，轴3 = -100度，轴2 = -1度，兼容性位 = 0
- cfx = '1000' = 1000
  轴5 = -15度，轴3 = 100度，轴2 = 1度，兼容性位 = 0

涂漆机械臂

使用全部四种配置参数。分别针对接头1、4和6使用cf1、cf4、cf6，并针对接头5使用cfx。

IRB 5500

使用全部四种配置参数。分别针对接头1、4和6使用cf1、cf4、cf6。cfx参数包含接头5象限号以及有关轴2和轴3的四种可能的配置。
有关更多信息，请参见*Product Manual - IRB 5500*。

IRB 5350

机械臂拥有两个旋转轴（臂1和臂2）和一个线性轴（臂3）。

- cf1用于旋转轴1。
- cfx用于旋转轴2。
- 未使用cf4和cf6。

组件

cf1

数据类型：num
旋转轴：
轴1的当前象限，表示为一个正整数或负整数。
线性轴：
轴1的当前间隔米数，表示为一个正整数或负整数。

cf4

数据类型：num
旋转轴：
轴4的当前象限，表示为一个正整数或负整数。
线性轴：
轴4的当前间隔米数，表示为一个正整数或负整数。

cf6

数据类型：num
旋转轴：
轴6的当前象限，表示为一个正整数或负整数。
线性轴：
轴6的当前间隔米数，表示为一个正整数或负整数。

下一页继续

3.17 confdata - 机械臂配置数据

RobotWare - OS

续前页

`cfx`

数据类型：num

旋转轴：

对于串行线机械臂，当前机械臂配置表示为从0到7范围中的一个整数。

对于SCARA机器人，当前机器人配置表示为0至1范围内的一个整数，请参见第*1378*页的*SCARA*机器人。

对于7轴机械臂，当前机械臂配置表示为从0到1111范围中的一个整数，参见第*1378*页的*7*轴机械臂。

对于涂漆机械臂，轴5的当前象限表示为一个正整数或负整数。有关IRB 5500，请参见第*1379*页的*IRB 5500*。

对于其他机械臂，使用轴2的当前象限，表示为一个正整数或负整数。

线性轴：

轴2的当前间隔米数，表示为一个正整数或负整数。

基本示例

以下示例介绍了数据类型confdata：

例 1

```
VAR confdata conf15 := [1, -1, 0, 0]
```

有关一种涂漆机械臂类型的机械臂配置conf15定义如下：

- 机械臂轴1的轴配置为象限1，即90-180º。
- 机械臂轴4的轴配置为象限-1，即0-(-90º)。
- 机械臂轴6的轴配置为象限0，即0 - 90º。
- 机械臂轴5的轴配置为象限0，即0 - 90º。

结构

```
< dataobject of confdata >
  < cf1 of num >
  < cf4 of num >
  < cf6 of num >
  < cfx of num >
```

相关信息

信息，关于	请参阅
坐标系 处理配置数据 奇点	技术参考手册 - *RAPID*语言概览
位置数据	第*1467*页的*robtarget* - 位置数据

3.18 corrdescr - 修正发电机描述符

手册用法

corrdescr（*Correction generator descriptor*）为校正发生器所用。校正发生器会增加路径坐标系中的几何偏移量。

描述

corrdescr型数据包含对校正发生器的引用。

通过指令CorrCon，完成同校正发生器的连接，通过指令CorrWrite，描述符（校正发生器的引用）可用于传递路径坐标系中的几何偏移量。

通过指令CorrDiscon，断开一台校正发生器，可移除早先提供的偏移量。通过指令CorrClear，可移除所有相连的校正发生器。

函数CorrRead返回迄今为止所有已交付偏移量的总和（包括所有相连的校正发生器）。

基本示例

以下示例介绍了数据类型corrdescr：

例 1

```
VAR corrdescr id;
VAR pos offset;
...
CorrCon id;
offset := [1, 2 ,3];
CorrWrite id, offset;
```

通过指令CorrCon，一台校正发生器得以连接，并由描述符id引用。随后，通过使用指令CorrWrite，将偏移量传送至校正发生器（参考id）。

特征

corrdescr是一种非数值的数据类型。

相关信息

信息，关于	请参阅
与修正发电机相连	第*131*页的*CorrCon - 与修正发电机相连*
与修正发电机断开	第*136*页的*CorrDiscon - 与修正发电机断开*
写入修正发电机	第*137*页的*CorrWrite - 写入修正发电机*
读取当前的总偏移量	第*1030*页的*CorrRead - 读取当前的总偏移量*
移除所有修正发电机	第*130*页的*CorrClear - 移除所有修正发电机*
非值数据类型的特征	技术参考手册 - *RAPID*语言概览，基本特征 - 数据类型一节

3.19 datapos - 数据类型的封闭块

手册用法

datapos是用函数GetNextSym检索出的某数据对象（内部系统数据）的封闭块。

描述

datapos型数据包含有关系统中所确定的特定对象的信息。其用于指令GetDataVal和SetDataVal。

基本示例

以下示例介绍了数据类型datapos：

例 1

```
VAR datapos block;
VAR string name;
VAR bool truevar:=TRUE;
...
SetDataSearch "bool" \Object:="my.*" \InMod:="mymod"\LocalSym;
WHILE GetNextSym(name,block) DO
  SetDataVal name\Block:=block,truevar;
ENDWHILE
```

该会话将模块mymod中以my开始的所有局部bool数据对象设置为TRUE。

特征

datapos是一种非数值的数据类型。

相关信息

信息，关于	请参阅
定义在搜索会话中设置的符号	第*582*页的*SetDataSearch - 定义在搜索序列中设置的符号*
获取下一个匹配的符号	第*1095*页的*GetNextSym - 获取下一个匹配的符号*
获取数据对象的值	第*213*页的*GetDataVal - 获得数据对象的值*
设置数据对象的值	第*586*页的*SetDataVal - 设置数据对象的值*
设置许多对象的值	第*578*页的*SetAllDataVal - 在定义设置下，设置所有数据对象的值*
Advanced RAPID	产品规格 - 控制器软件*IRC5*

3.20 dionum - 数字值（0 - 1）

手册用法

dionum*(digital input output numeric)*用于数字值（0或1）。

该数据类型连同处理数字信号输入或输出信号的指令和函数一同使用。

描述

dionum型数据代表数字值0或1。

基本示例

以下示例介绍了数据类型dionum：

例 1

```
CONST dionum close := 1;
SetDO grip1, close;
```

包含一个等于1的值的常量close 的定义。随后，将信号grip1设置为close，即1。

预定义数据

在系统中预定义常量high、low和edge：

```
CONST dionum low:=0;
CONST dionum high:=1;
CONST dionum edge:=2;
```

设计有关I/O指令的常量low和high 。

Edge可以同中断指令ISignalDI和ISignalDO一同使用。

特征

dionum为有关num的一种别名数据类型，并因此继承其特征。

相关信息

信息，关于	请参阅
输入/输出指令概要	技术参考手册 - *RAPID*语言概览，*RAPID*概要 - 输入和输出信号一节
I/O配置	技术参考手册 - 系统参数
别名数据类型	技术参考手册 - *RAPID*语言概览，基本特征 - 数据类型一节

3.21 dir - 路径结构

手册用法

dir（*directory*）用于穿过路径结构。

描述

dir型数据包含对磁盘或网络上路径的引用。通过指令OpenDir，其可以与物理路径相连，随后，用于读取。

基本示例

以下示例介绍了数据类型dir：

例 1

```
PROC lsdir(string dirname)
  VAR dir directory;
  VAR string filename;
  OpenDir directory, dirname;
  WHILE ReadDir(directory, filename) DO
    TPWrite filename;
  ENDWHILE
  CloseDir directory;
ENDPROC
```

此例子打印出指定路径下所有文件或子路径的名称。

特征

dir为非值数据类型，且无法用于以值为导向的运算。

相关信息

信息，关于	请参阅
打开路径	第*419*页的*OpenDir - 打开路径*
建立路径	第*322*页的*MakeDir - 创建新路径*
读取路径	第*1197*页的*ReadDir - 读取路径中的下个条目*
关闭路径	第*117*页的*CloseDir - 关闭路径*
删除路径	第*503*页的*RemoveDir - 删除路径*
删除文件	第*504*页的*RemoveFile - 删除文件*
重命名文件	第*507*页的*RenameFile - 重命名文件*
检查文件类型	第*1126*页的*IsFile - 检查文件的类型*
文件和串行通道处理	应用手册 - 控制器软件*IRC5*

3.22 dnum - 双数值

手册用法

dnum用于数值，例如计数器。其可以处理大于数据num的整数值，但是，其特征和函数与num相同。

描述

dnum数据类型的值可以为：

- 一个整数，例如-5
- 一个小数，例如3.45

其亦可呈指数地写入，例如2E3（= 2*10^3 = 2000），2.5E-2（= 0.025）。

始终将-4503599627370496与+4503599627370496之间的整数作为准确的整数储存。

基本示例

以下示例介绍了数据类型dnum：

例 1

```
VAR dnum reg1;
...
reg1:=1000000;
```

将reg1指定为值1000000。

例 2

```
VAR dnum hex;
Var dnum bin;
VAR dnum oct;
! Hexadecimal representation of decimal value 4294967295
hex := 0xFFFFFFFF;
! Binary representation of decimal value 255
bin := 0b11111111;
! Octal representation of decimal value 255
oct := 0o377;
```

例 3

```
VAR dnum a:=0;
VAR dnum b:=0;
a := 10 DIV 3;
b := 10 MOD 3;
```

整数除法，向a分配一个整数（= 3），并向b分配余数（= 1）。

限制

将分配给一个dnum变量的介于-4503599627370496到4503599627370496之间的字面值作为准确的整数来储存。

如果将已作为一个num解释的字面值作为一个dnum来分配/使用，其自动转换为一个dnum。

下一页继续

3.22 dnum - 双数值
RobotWare - OS
续前页

相关信息

信息，关于	请参阅
使用数据类型num的数值	第*1435*页的*num - 数值*
数字表达式	技术参考手册 - *RAPID*概述，基础*RAPID*编程一节
使用数值进行运算	技术参考手册 - *RAPID*概述，基础*RAPID*编程一节

3.23 egmframetype - 定义EGM所需的框架类型

手册用法

egmframetype的作用是为EGM中的校正和传感器测量定义所需的框架类型。

描述

egmframetype旨在供指令EGMActJ和EGMActPose使用。

基本示例

```
CONST egm_minmax egm_minmax_lin1:=[-0.1,0.1];
CONST egm_minmax egm_minmax_rot1:=[-0.1,0.2];

EGMActPose egmID1\Tool:=tFroniusCMT\WObj:=wobj0, posecor,
     EGM_FRAME_WOBJ, posesens, EGM_FRAME_TOOL \x:=egm_minmax_lin
     \y:=egm_minmax_lin \z:=egm_minmax_lin \rx:=egm_minmax_rot
     \ry:=egm_minmax_rot \rz:=egm_minmax_rot \LpFilter:=20;
```

预定义数值

值	描述
EGM_FRAME_BASE	以相对于基本框架的方式来定义该框架（姿态模式）。
EGM_FRAME_TOOL	以相对于所用工具的方式来定义该框架（姿态模式）。
EGM_FRAME_WOBJ	以相对于所用工件的方式来定义该框架（姿态模式）。
EGM_FRAME_WORLD	以相对于全局框架的方式来定义该框架（姿态模式）。
EGM_FRAME_JOINT	这些数值为关节值（关节模式）。

特征

egmframetype是针对num的一种别名数据类型。

相关信息

信息，关于	请参阅
Externally Guided Motion	应用手册 - 控制器软件*IRC5*

3.24 egmident - 识别一项特定的EGM进程

手册用法

egmident会识别一项特定的EGM进程。

描述

用指令EGMGetId保留某一egmident，然后用其识别指令EGMSetupXX、EGMActX、EGMRunX和EGMReset，并将这些指令与同一项EGM操作关联起来。

egmident的标识就是其名称，即是说如果用相同的egmident来第二次或第三次调用EGMGetId，那么系统既不会保留一项新进程，也不会更改其内容。只有EGMReset才会释放一项egmident。

基本示例

```
VAR egmident egmID1;
VAR egmstate egmSt1;
TASK PERS wobjdata wobj_EGM1:=[FALSE, TRUE, "", [[500,700,900],
     [1,0,0,0]], [[0,0,0], [1,0,0,0]]];
CONST pose posecor:=[[1200,400,900], [0,0,1,0]];
CONST pose posesens:=[[12.3313,-0.108707,416.142],
     [0.903899,-0.00320735,0.427666,0.00765917]];
CONST egm_minmax egm_minmax_lin1:=[-0.1,0.1];
CONST egm_minmax egm_minmax_rot1:=[-0.1,0.2];
CONST egm_minmax egm_minmax_joint1:=[-0.1,0.1];

PROC testAI()
  EGMReset egmID1;
  EGMGetId egmID1;
  mvHome;
  mvHome_EGMLinear;

  egmSt1:=EGMGetState(egmID1);
  TPWrite "EGM state 1: " \Num:=egmSt1;

  IF egmSt1<=EGM_STATE_CONNECTED THEN
    EGMSetupAI ROB_1, egmID1, "default" \Pose \aiR1x:=ai_MoveX
         \aiR2y:=ai_MoveY \aiR3z:=ai_MoveZ \aiR5ry:=ai_RotY
         \aiR6rz:=ai_RotZ;
  ENDIF

  EGMActPose egmID1 \Tool:=tFroniusCMT \WObj:=wobj0, posecor,
       EGM_FRAME_WOBJ, posesens, EGM_FRAME_TOOL \x:=egm_minmax_lin1
       \y:=egm_minmax_lin1 \z:=egm_minmax_lin1 \rx:=egm_minmax_rot1
       \ry:=egm_minmax_rot1 \rz:=egm_minmax_rot1 \LpFilter:=20;
  EGMRunPose egmID1, EGM_STOP_HOLD \x \y \z \rx \ry \rz
       \RampInTime:=0.05;

  egmSt1:=EGMGetState(egmID1);
  IF egmSt1=EGM_STATE_CONNECTED THEN
    TPWrite "Reset lin 1";
    EGMReset egmID1;
```

下一页继续

```
    ENDIF
ENDPROC
```

限制

每项RAPID任务最多有4个并行实例可用。

特征

`egmident`是一种非数值的数据类型。请调用`EGMGetId`来设置该类型。

相关信息

信息，关于	请参阅
Externally Guided Motion	应用手册 - 控制器软件*IRC5*

3.25 egm_minmax - EGM的收敛标准

手册用法

egm_minmax的作用是定义结束EGM的收敛标准。

描述

egm_minmax旨在供指令EGMActJ和EGMActPose使用。

组件

Min

数据类型：num

最小偏差

定义相关位置偏差的最小值。默认值为-0.5度。

Max

数据类型：num

最大偏差

定义相关位置偏差的最大值。默认值为0.5度。

基本示例

```
CONST egm_minmax egm_minmax_lin1:=[-0.1,0.1];
CONST egm_minmax egm_minmax_rot1:=[-0.1,0.2];

EGMActPose egmID1\Tool:=tFroniusCMT\WObj:=wobj0, posecor,
     EGM_FRAME_WOBJ, posesens, EGM_FRAME_TOOL \x:=egm_minmax_lin1
     \y:=egm_minmax_lin1 \z:=egm_minmax_lin1 \rx:=egm_minmax_rot1
     \ry:=egm_minmax_rot1 \rz:=egm_minmax_rot1 \LpFilter:=20;
```

特征

Egm_minmax具有下列单位：

- 毫米，用于直线移动下的x、y和z。
- 度，用于直线移动和关节移动下的rx、ry和rz。

结构

```
< dataobject of egm_minmax >
  < min of num >
  < max of num >
```

相关信息

信息，关于	请参阅
Externally Guided Motion	应用手册 - 控制器软件*IRC5*

3.26 egmstate - 定义EGM所需的状态

手册用法

egmstate的作用是为EGM中的校正和传感器测量定义所需的状态。

描述

egmstate是函数EGMGetState的返回值。

基本示例

```
VAR egmstate egmSt1;
VAR egmident egmID1;

EGMReset egmID1;
EGMGetId egmID1;

egmSt1:=EGMGetState(egmID1);
TPWrite "EGM state: "\Num:=egmSt1;
```

预定义数值

值	描述
EGM_STATE_DISCONNECTED	未定义这一具体进程的EGM状态。 未激活任何设定。
EGM_STATE_CONNECTED	未激活指定的EGM进程。 已进行过设置，但并未激活任何EGM移动。
EGM_STATE_RUNNING	正在执行指定的EGM进程。 EGM移动处于激活状态，即是说移动了相关机器人。

特征

egmstate是针对num的一种别名数据类型。

相关信息

信息，关于	请参阅
Externally Guided Motion	应用手册 - 控制器软件*IRC5*

3.27 egmstopmode - 定义EGM所需的停止模式

手册用法

egmstopmode的作用是为EGM中的校正和传感器测量定义所需的停止模式。

描述

egmstopmode旨在供指令EGMRunJoint、EGMRunPose和EGMStop使用。

基本示例

来自RAPID运动任务：

```
VAR egmstate egmSt1;
VAR egmident egmID1;

EGMReset egmID1;
EGMGetId egmID1;
CONST egm_minmax egm_minmax_lin1:=[-0.1,0.1];
CONST egm_minmax egm_minmax_rot1:=[-0.1,0.2];

EGMActPose egmID1 \Tool:=tFroniusCMT \WObj:=wobj0, posecor,
     EGM_FRAME_WOBJ, posesens, EGM_FRAME_TOOL \x:=egm_minmax_lin
     \y:=egm_minmax_lin \z:=egm_minmax_lin \rx:=egm_minmax_rot
     \ry:=egm_minmax_rot \rz:=egm_minmax_rot \LpFilter:=20;
EGMRunPose egmID1, EGM_STOP_HOLD \x \y \z \rx \ry \rz
     \RampInTime:=0.05;
```

来自一项RAPID软中断或后台任务：

```
EGMStop egmID1, EGM_STOP_RAMP_DOWN\RampOutTime:=5.0;
```

预定义数值

值	描述
EGM_STOP_HOLD	保持EGM结束位置。
EGM_STOP_RAMP_DOWN	从EGM的结束位置返回启动位置。

特征

egmstopmode是针对num的一种别名数据类型。

相关信息

信息，关于	请参阅
Externally Guided Motion	应用手册 - 控制器软件*IRC5*

3.28 errdomain - 错误域

手册用法

errdomain（错误域）用于指定一个错误域。

描述

errdomain型数据代表错误、警告或状态变更已作记录的域。

基本示例

以下示例介绍了数据类型errdomain：

例 1

```
VAR errdomain err_domain;
VAR num err_number;
VAR errtype err_type;
VAR trapdata err_data;
...
TRAP trap_err
  GetTrapData err_data;
  ReadErrData err_data, err_domain, err_number, err_type;
ENDTRAP
```

当错误被困于软中断程序trap_err时，将错误域、错误编号和错误类型保存在适当的变量中。

预定义数据

以下预定义常量可用于指定一个错误域。

名称	错误域	值
COMMON_ERR	所有错误和状态变更域	0
OP_STATE	操作状态变更	1
SYSTEM_ERR	系统错误	2
HARDWARE_ERR	硬件错误	3
PROGRAM_ERR	程序错误	4
MOTION_ERR	运动错误	5
OPERATOR_ERR	运算符错误-废弃，不再使用	6
IO_COM_ERR	I/O和通信错误	7
USER_DEF_ERR	用户定义的错误（由RAPID引起）	8
SAFETY_ERR	安全相关事件	9
PROCESS_ERR	过程错误	11
CFG_ERR	配置错误	12

特征

errdomain 为有关数字的一种别名数据类型，并因此继承其特征。

下一页继续

3.28 errdomain - 错误域
RobotWare - OS
续前页

相关信息

信息，关于	请参阅
下达错误时中断的指令	第*234*页的*IError - 调整关于错误的中断*
错误编号	操作员手册 - *IRC5* 故障排除
别名数据类型	技术参考手册 - *RAPID*语言概览，基本特征 - 数据类型一节
Advanced RAPID	产品规格 - 控制器软件*IRC5*

3.29 errnum - 错误编号

手册用法

errnum用于描述程序执行期间出现的所有可恢复（非重大）错误，例如除以零。

描述

如果机械臂探测到程序执行期间的一个错误，则可用程序的错误处理器进行处理。此类错误的例子包括值过高以及除以零。因此，根据错误的性质，向errnum型系统变量ERRNO分配不同的值。通过读取该变量，本错误处理器也许能够更正错误，随后，可通过正确的方式继续程序执行。

通过使用RAISE 指令，亦可在程序内产生一个错误。通过指定一个作为RAISE的参数的错误编号（处于范围1-90内，或者通过指令BookErrNo预定），可用错误处理器来探测此类错误。

基本示例

以下示例介绍了数据类型errnum：

例 1

```
reg1 := reg2 / reg3;
...
ERROR
  IF ERRNO = ERR_DIVZERO THEN
    reg3 := 1;
    RETRY;
  ENDIF
```

如果reg3 = 0，则在进行除法运算时，机械臂探测到一个错误。但是，通过向reg3分配值1，可探测和更正该错误。此后，可再次进行除法运算，并继续程序执行。

例 2

```
CONST errnum machine_error := 1;
...
IF di1=0 RAISE machine_error;
...
ERROR
  IF ERRNO=machine_error RAISE;
```

在一台机器中出现一个错误（通过输入信号di1来探测）。迅速地转到程序中的错误处理器，其反过来调用有关调用程序的错误处理器，并可能由此更正错误。常量machine_error用于使错误处理器知晓已出现错误的正确类型。

预定义数据

系统变量ERRNO可用于读取最新出现的错误。许多预定义常量可用于确定已出现错误的类型。

名称	错误原因
ERR_ACC_TOO_LOW	指令PathAccLim或WorldAccLim中指定的过低加速度/减速度
ERR_ADDR_INUSE	地址和端口已在使用中，且无法再次使用。在SocketBind中使用一个不同的端口号或地址。

下一页继续

3.29 errnum - 错误编号

RobotWare - OS

续前页

名称	错误原因
ERR_ALIASIO_DEF	未在IO配置中确定FromSignal，或未在RAPID程序中声明ToSignal，或已在IO配置中确定。指令AliasIO
ERR_ALIASIO_TYPE	有关参数FromSignal和ToSignal的信号类型不相同（signalx）。指令AliasIO.
ERR_ALRDYCNT	已经将中断变量与 TRAP程序相连
ERR_ALRDY_MOVING	当执行一个StartMove 或StartMoveRetry指令时，机械臂已正在运动。
ERR_AO_LIM	限制外的模拟信号值
ERR_ARGDUPCND	提出有关相同参数的多个条件参数
ERR_ARGNAME	当执行ArgName时，参数为一个表达式，而非直接提出，或者为switch型
ERR_ARGNOTPER	参数并非为一个永久数据对象引用
ERR_ARGNOTVAR	参数并非为一个变量引用
ERR_ARGVALERR	参数值错误
ERR_AXIS_ACT	轴无效
ERR_AXIS_IND	轴不独立
ERR_AXIS_MOVING	轴正在移动
ERR_AXIS_PAR	指令中的参数轴错误
ERR_BUSSTATE	已完成IOEnable，且在启用I/O单元之前，I/O总线处于错误状态或进入错误状态。
ERR_BWDLIMIT	限制StepBwdPath
ERR_CALC_NEG	StrDig反向计算错误
ERR_CALC_OVERFLOW	StrDig计算溢出
ERR_CALC_DIVZERO	StrDig除以零
ERR_CALLPROC	运行时（后期绑定）出现过程调用错误（非无返回值程序）
ERR_CAM_BUSY	摄像头正忙于处理其他请求，无法执行当前命令。
ERR_CAM_COM_TIMEOUT	与摄像机的通信超时。摄像机无应答。
ERR_CAM_GET_MISMATCH	用 CamGetParameter 指令从摄像头获取的参数的数据类型错误。
ERR_CAM_MAXTIME	当执行一个CamLoadJob或一个CamGetResult指令时，出现超时。
ERR_CAM_NO_MORE_DATA	无法获得更多的视觉效果。
ERR_CAM_NO_PROGMODE	摄像头未处于编程模式
ERR_CAM_NO_RUNMODE	摄像头未处于运行模式
ERR_CAM_SET_MISMATCH	使用命令 CamSetParameter 写入摄像头的参数数据类型错误，或者其值超出范围。
ERR_CFG_INTERNAL	不允许读取内部参数-ReadCfgData
ERR_CFG_ILL_DOMAIN	指令SaveCfgData中使用的cfgdomain无效，或未在使用当中。
ERR_CFG_ILLTYPE	类型不匹配-ReadCfgData、WriteCfgData
ERR_CFG_LIMIT	数据限制-WriteCfgData

下一页继续

名称	错误原因
ERR_CFG_NOTFND	未发现-ReadCfgData、WriteCfgData
ERR_CFG_OUTOFBOUNDS	如果列表编号在输入端为-1，或大于可用实例的数量-ReadCfgData、WriteCfgData
ERR_CFG_WRITEFILE	路径不存在，或使用的FilePath和File是一个路径，或关于使用指令SaveCfgData时保存文件的一些其他问题。
ERR_CNTNOTVAR	CONNECT目标并非为一个变量引用
ERR_CNV_NOT_ACT	未启用传送带
ERR_CNV_CONNECT	已经启用了WaitWobj指令
ERR_CNV_DROPPED	已经删除指令WaitWobj正在等待的对象。
ERR_COLL_STOP	停止因运动碰撞而引起的移动。
ERR_COMM_EXT	有关外部系统的通信错误。
ERR_CONC_MAX	已经超过运用参数\Conc的连续运动指令的数量
ERR_COMM_INIT_FAILED	通信界面无法初始化。
ERR_DATA_RECV	从远程系统收到的数据不正确。
ERR_DEV_MAXTIME	当执行一个ReadBin、ReadNum、ReadStr、ReadStrBinReadAnyBin或一个ReadRawBytes指令时，出现超时。
ERR_DIPLAG_LIM	同当前TriggL/TriggC/TriggJ相关的指令TriggSpeed中的DipLag 过大
ERR_DIVZERO	除零
ERR_EXECPHR	通过使用一个放置支架，尝试执行一个指令
ERR_FILEACC	不正确地访问一个文件
ERR_FILEEXIST	已经存在一个文件
ERR_FILEOPEN	无法打开一个文件
ERR_FILNOTFND	未找到文件
ERR_FNCNORET	无返回值
ERR_FRAME	无法计算新坐标系
ERR_GO_LIM	限制外的数字组信号值
ERR_ILLDIM	不正确的数组维度
ERR_ILLQUAT	试图使用不合法的方位（四元数）值
ERR_ILLRAISE	超出范围的RAISE中的错误编号
ERR_INDCNV_ORDER	执行指令之前，需要执行IndCnvInit。
ERR_INOISSAFE	如果尝试暂时停用安全中断，则采用ISleep。
ERR_INOMAX	无法获得更多的中断编号
ERR_INT_NOTVAL	无效的整数、小数值
ERR_INT_MAXVAL	无效的整数、过大或过小的值
ERR_INVDIM	维度不相等
ERR_IODISABLE	执行IODisable时出现超时
ERR_IOENABLE	执行IOEnable时出现超时

下一页继续

3.29 errnum - 错误编号
RobotWare - OS
续前页

名称	错误原因
ERR_IOERROR	源于指令Save的I/O错误
ERR_LINKREF	程序任务中的引用错误
ERR_LOADED	已经加载普通程序模块
ERR_LOADID_FATAL	仅在LoadId中作内部使用
ERR_LOADID_RETRY	仅在LoadId中作内部使用
ERR_LOADNO_INUSE	加载会话用于StartLoad
ERR_LOADNO_NOUSE	加载会话不用于CancelLoad
ERR_MAXINTVAL	整数值过大
ERR_MODULE	指令Save和EraseModule中的不正确模块名称
ERR_MOD_NOT_LOADED	此模块不存在，符号并非一个模块，或者名称对于符号来说太长。来自函数ModTimeDnum的错误
ERR_NAME_INVALID	如果I/O单元名称不存在
ERR_NO_ALIASIO_DEF	该信号变量是在RAPID中声明的一个变量，与用指令AliasIO在I / O配置中定义的I / O信号无关。
ERR_NORUNUNIT	如果与I/O单元无接触
ERR_NOTARR	数据并非一个数组
ERR_NOTEQDIM	调用程序时使用的数组维度不符合其参数
ERR_NOTINTVAL	并非一个整数值
ERR_NOTPRES	使用一个参数，尽管未在程序调用时使用相关的参数
ERR_NOTSAVED	模块自加载到系统中起已有所改变
ERR_NOT_MOVETASK	指定任务为非运动任务
ERR_NUM_LIMIT	该值高于3.40282347E+38，或低于-3.40282347E+38
ERR_OUTOFBND	数组索引位于允许限制以外
ERR_OVERFLOW	时钟溢出
ERR_OUTSIDE_REACH	位置（机器人位置）位于有关函数CalcJoinT的机械臂工作区域以外。
ERR_PATH	错过指令Save中的目标路径
ERR_PATHDIST	有关StartMove或StartMoveRetry指令的恢复距离过长
ERR_PATH_STOP	停止因某些过程错误而引起的移动
ERR_PERSSUPSEARCH	持续变量在搜索过程的开始就已经是 TRUE。
ERR_PID_MOVESTOP	仅在LoadId中作内部使用
ERR_PID_RAISE_PP	源于ParIdRobValid、ParIdPosValid或LoadId的错误。
ERR_PRGMEMFULL	程序内存已满
ERR_PROCSIGNAL_OFF	过程信号关闭
ERR_PROGSTOP	当执行StartMove或StartMoveRetry指令时，机械臂处于程序停止状态
ERR_RANYBIN_CHK	通过指令ReadAnyBin，检查数据传送时探测到的错误总和
ERR_RANYBIN_EOF	在通过指令ReadAnyBin读取所有字节之前，探测文件结尾

下一页继续

名称	错误原因
ERR_RCVDATA	通过ReadNum，试图读取非数字数据
ERR_REFUNKDAT	整个未知数据对象的引用
ERR_REFUNKFUN	未知的函数的引用
ERR_REFUNKPRC	链接时或运行时（后期绑定）未知无返回值程序的引用
ERR_REFUNKTRP	未知软中断的引用
ERR_RMQ_DIM	错误的维度，给定数据的维度与消息中数据的维度不相等。
ERR_RMQ_FULL	目的消息序列已满。
ERR_RMQ_INVALID	目的槽丢失或无效
ERR_RMQ_INVMSG	无效的消息，可能从其他客户端发送，然后至RAPID任务
ERR_RMQ_MSGSIZE	消息过大。减小消息大小。
ERR_RMQ_NAME	给定槽名无效或未能发现。
ERR_RMQ_NOMSG	序列中无消息，可能是上电失败的结果。
ERR_RMQ_TIMEOUT	等待RMQSendWait中的应答时，出现超时。
ERR_RMQ_VALUE	值的语法与数据类型不匹配。
ERR_ROBLIMIT	位置可以到达，但是至少一个轴位于关节限制外或超出限制至少一个耦合关节（函数 CalcJointT）
ERR_SC_WRITE	当发送至外部计算机时出现错误
ERR_SIGSUPSEARCH	搜索过程开始时，信号已经拥有一个正值
ERR_SIG_NOT_VALID	无法访问I/O信号。原因可能在于I/O单元未运行，或者配置中出现错误（仅对ICI现场总线有效）。
ERR_SOCK_CLOSED	套接字关闭或未创建
ERR_SOCK_TIMEOUT	不会在超时时间内建立连接
ERR_SPEED_REFRESH_LIM	超出SpeedRefresh中限制的覆盖
ERR_SPEEDLIM_VALUE	指令SpeedLimAxis和SpeedLimCheckPoint中使用的速度过低。
ERR_STARTMOVE	当执行StartMove或StartMoveRetry指令时，机械臂处于停止状态
ERR_STRTOOLNG	字符串过长
ERR_SYM_ACCESS	符号读/写权限错误
ERR_SYMBOL_TYPE	在参数Value中使用的数据对象和变量具有不同的类型。如果使用ALIAS数据类型，则亦将出现此错误，尽管上述类型可能具有相同的基础数据类型。指令GetDataVal、SetDataVal和SetAllDataVal。
ERR_SYNCMOVEOFF	源于SyncMoveOff的超时
ERR_SYNCMOVEON	源于SyncMoveOn的超时
ERR_SYNTAX	已加载模块中的语法错误
ERR_TASKNAME	未在系统中发现的任务名称
ERR_TP_DIBREAK	来自FlexPendant的读取指令被数字信号输入中断
ERR_TP_DOBREAK	来自FlexPendant的读取指令被数字信号输出中断
ERR_TP_MAXTIME	当执行来自FlexPendant的读取指令时，出现超时

下一页继续

3.29 errnum - 错误编号
RobotWare - OS
续前页

名称	错误原因
ERR_TP_NO_CLIENT	当使用来自FlexPendant的读取指令时，没有相互作用的客户端
ERR_TRUSTLEVEL	不允许禁用I/O单元
ERR_TXTNOEXIST	函数TextGet中的错误表格或索引
ERR_UDPUC_COMM	UdpUc装置通信超时
ERR_UI_INITVALUE	函数UINumEntry中的初始值错误
ERR_UI_MAXMIN	最小值大于函数UINumEntry中的最大值
ERR_UI_NOTINT	使用UINumEntry时，规定应当使用一个整数，然而值并非一个整数
ERR_UISHOW_FATAL	除指令UIShow中的ERR_UISHOW_FATAL以外的其他错误
ERR_UISHOW_FULL	当使用指令UIShow时，未针对另一种应用而在FlexPendant上留下任何空间
ERR_UNIT_PAR	TestSignDefine中的参数Mech_unit错误
ERR_UNKINO	未知中断编号
ERR_UNKPROC	对指令WaitLoad中加载会话的不正确引用
ERR_UNLOAD	指令UnLoad或WaitLoad中的卸载错误
ERR_WAITSYNCTASK	源于WaitSyncTask的超时
ERR_WAIT_MAXTIME	当执行WaitDI或WaitUntil指令时，出现超时
ERR_WHLSEARCH	没有搜索停止
ERR_WOBJ_MOVING	含工件的机械单元正在移动CalcJointT

特征

errnum为有关num的一种别名数据类型，并因此继承其特征。

相关信息

信息，关于	请参阅
错误恢复	技术参考手册 - *RAPID*语言概览
一般数据类型、别名数据类型	技术参考手册 - *RAPID*语言概览

3.30 errstr - 错误字符串

手册用法

errstr 用于将文本写入错误消息。

基本示例

以下示例介绍了数据类型errstr：

例 1

```
VAR errstr arg:= "This is an example";

ErrLog 5100, \W, ERRSTR_TASK, ERRSTR_CONTEXT, arg, ERRSTR_EMPTY,
     ERRSTR_UNUSED;
```

预定义数据

名称	描述
ERRSTR_EMPTY	参数为空
ERRSTR_UNUSED	参数未使用
ERRSTR_TASK	当前任务的名称
ERRSTR_CONTEXT	上下文

特征

errstr为有关string 的一种别名数据类型，并因此继承其特征。

相关信息

信息，关于	请参阅
一般数据类型、别名数据类型	技术参考手册 - *RAPID*语言概览，基本特征 - 数据类型一节

3.31 errtype - 错误类型

手册用法

errtype（*error type*）用于指定一种错误类型。

描述

errtype 型数据代表一条错误消息的类型（状态变更、警告、错误）。

基本示例

以下示例介绍了数据类型errtype：

例 1

```
VAR errdomain err_domain;
VAR num err_number;
VAR errtype err_type;
VAR trapdata err_data;
...
TRAP trap_err
  GetTrapData err_data;
  ReadErrData err_data, err_domain, err_number, err_type;
ENDTRAP
```

当错误被困于软中断程序trap_err时，将错误域、错误编号和错误类型保存在适当的变量中。

预定义数据

以下预定义常量可用于指定一种错误类型。

名称	错误类型	值
TYPE_ALL	任意类型的错误（状态变更、警告、错误）	0
TYPE_STATE	状态变更（操作消息）	1
TYPE_WARN	警告（例如RAPID可恢复错误）	2
TYPE_ERR	Error	3

特征

errtype为有关num的一种别名数据类型，并因此继承其特征。

相关信息

信息，关于	请参阅
下达错误时中断的指令	第*234*页的*IError - 调整关于错误的中断*
错误编号	操作员手册 - *IRC5* 故障排除
别名数据类型	技术参考手册 - *RAPID*语言概览，基本特征 - 数据类型一节
Advanced RAPID	产品规格 - 控制器软件*IRC5*

3.32 event_type - 事件程序类型

手册用法

`event_type` 用于代表有关一个符号常量的实际事件程序类型。

描述

通过函数`EventType`，检查是否由于某些特定系统事件而执行实际RAPID代码是可能的。

基本示例

以下示例介绍了数据类型`event_type`：

例 1

```
VAR event_type my_type;
...
my_type := EventType( );
```

将执行的事件程序类型储存在变量`my_type`中。

预定义数据

预定义以下`event_type`型常量：

RAPID常量	值	所执行事件类型
EVENT_NONE	0	未执行任何事件
EVENT_POWERON	1	POWER_ON事件
EVENT_START	2	START事件
EVENT_STOP	3	STOP事件
EVENT_QSTOP	4	QSTOP事件
EVENT_RESTART	5	RESTART事件
EVENT_RESET	6	RESET 事件
EVENT_STEP	7	STEP 事件

特征

`event_type` 为有关`num`的一种别名数据类型，并因此继承其特征。

相关信息

信息，关于	请参阅
一般的事件程序	技术参考手册 - 系统参数，*Controller - Event Routine*一节
获得事件类型	第*1072*页的*EventType* - 获取任意事件程序内的当前事件类型
一般数据类型、别名数据类型	技术参考手册 - *RAPID*语言概览，基本特征 - 数据类型一节

3.33 exec_level - 执行等级

手册用法

exec_level用于规定程序执行等级。

描述

通过函数ExecLevel，可能获得当前所执行RAPID事件号的实际执行等级。

预定义数据

预定义以下exec_level型常量：

RAPID常量	值	执行等级
LEVEL_NORMAL	0	在基础等级上执行
LEVEL_TRAP	1	在软中断程序中执行
LEVEL_SERVICE	2	在服务程序中执行[i]

[i] LEVEL_SERVICE意指来自系统输入信号的事件程序、服务程序（包括调用程序）和中断程序。

特征

exec_level为有关num的一种别名数据类型，并因此继承其特征。

相关信息

信息，关于	请参阅
获取当前执行等级	第*1075*页的*ExecLevel - 获取执行等级*

3.34 extjoint - 外接头的位置

手册用法

extjoint用于确定附加轴、定位器或工件机械臂的轴位置。

描述

除六个内部轴外，机械臂可控制多达六个附加轴，即总共十二个轴。从逻辑上表示六个附加轴：a、b、c、d、e和f。各逻辑轴均可物理轴相连，且在这种情况下，用系统参数来定义连接。

extjoint型数据用于保存各逻辑轴（a - f）的位置值。

对于与物理轴相连的各逻辑轴，其位置定义如下：

- 对于旋转轴，其位置定义为从校准位置起旋转的度数。
- 对于线性轴，其位置定义为与校准位置的距离（以mm计）。

如果逻辑轴未与物理轴相连，则将值9E9作为位置值，以此表明该轴未予以连接。执行期间，检查各轴的位置数据以及是否连接相关轴。如果储存的位置值不符合实际的轴连接，则以下内容适用：

- 如果未在位置数据（值为9E9）中定义位置，则当连接且未激活该轴时，将忽略该值。但是如果激活该轴，则将会产生错误。
- 如果在位置数据中定义该位置，则将忽略该值，即便未连接该轴。

如果未激活轴，则不会进行任何运动，且不会产生有关轴（含正确位置数据）的错误。

如果使用附加轴偏移量（指令EOffsOn或EOffsSet），则在ExtOffs坐标系中指定出位置。

如果一个附加轴以独立模式运行，且应由机械臂及其附加轴进行新的运动，则采用独立模式的附加轴的位置数据不得为9E9。数据必须为系统未使用的任意值。

组件

eax_a

external axis a

数据类型：num

外部逻辑轴“a”的位置以度或毫米来表示（取决于轴的类型）。

...

eax_f

external axis f

数据类型：num

外部逻辑轴“f”的位置以度或毫米来表示（取决于轴的类型）。

基本示例

以下示例介绍了数据类型extjoint：

例 1

```
VAR extjoint axpos10 := [ 11, 12.3, 9E9, 9E9, 9E9, 9E9] ;
```

外部定位器的位置axpos10,定义如下：

- 将外部逻辑轴“a”的位置设置为11，以度或毫米来表示（取决于轴的类型）。

下一页继续

3.34 extjoint - 外接头的位置

RobotWare - OS

续前页

- 将外部逻辑轴“b”的位置设置为12.3，以度或毫米来表示（取决于轴的类型）。
- 未定义轴c到轴f。

结构

```
< dataobject of extjoint >
  < eax_a of num >
  < eax_b of num >
  < eax_c of num >
  < eax_d of num >
  < eax_e of num >
  < eax_f of num >
```

相关信息

信息，关于	请参阅
位置数据	第*1467*页的*robtarget - 位置数据* 第*1418*页的*jointtarget - 接头位置数据*
ExtOffs坐标系	第*187*页的*EOffsOn - 启用附加轴的偏移量*

3.35 flypointdata - 飞焊开始/结束数据

手册用法

flypointdata将用于定义CAP进程的飞焊开始或飞焊结束的所有数据 - 属于飞焊开始和飞焊结束的capdata的一部分。

定义

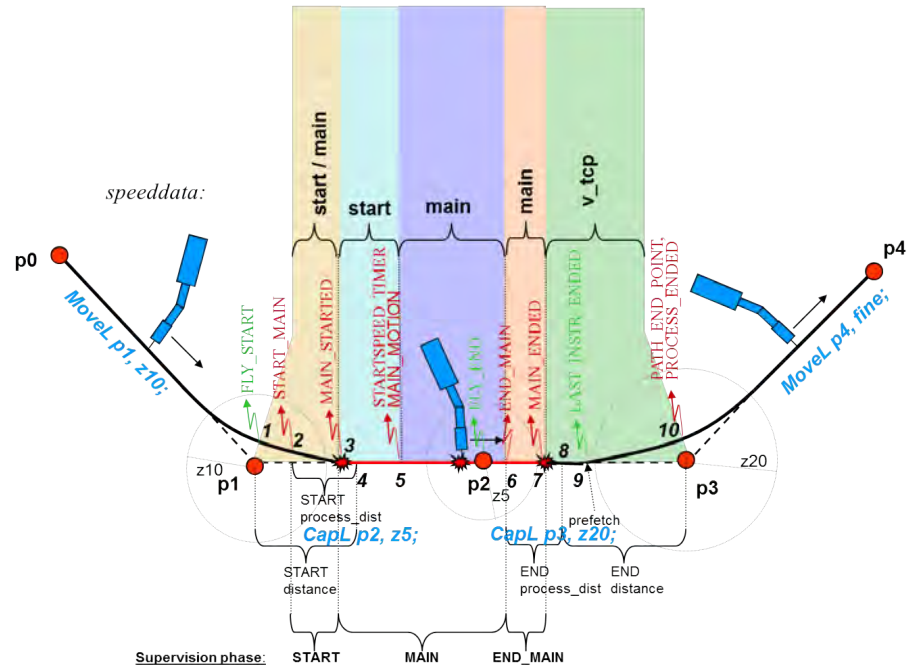
flypointdata定义飞焊开始：

- 仅为CAP提供此功能。
- 飞焊开始在第一个指令 = 真和区域点的共同作用下触发。
- 飞焊结束在*last_instr* = 真和区域点的共同作用下触发。
- 摆动启动将被忽略。
- 如果起点是一个精确点，那么不执行飞焊开始。
- 如果终点是一个精确点，那么不执行飞焊结束。
- 运动延时将被忽略。
- 出错后重启将如常工作：发生错误时，如果应用工艺处于活跃状态，那么没有特定的飞焊开始特征，刮刀启动可供使用。
- 如果摆动被激活，那么将从区域入口开始让摆动运动逐渐逼近，从而在区域中进行过渡，直至当TCP离开区域时，完成整个摆动运动。
- 在开始阶段（TCP移动）、主阶段和END_MAIN阶段（TCP移动），监控处于活跃状态。
- 路径返回将限于返回到位置4（请参见下图）。
- 用户必须调整应用工艺的离开角度、距离以及途径：比如，在应建立弧的位置（图中点4处）进行的弧焊，选择时必须能确保点火。
- 位置4和位置6之间的距离不得=0。
- START process_dist必须等于或短于START distance。
- 如果程序执行停止并且应用工艺处于活跃状态（位置3和位置6之间），那么，CAP行为如常，即，路径返回（仅在已经经过位置4的情况下）、摆动启动、运动延时和运动启动超时可供采用。
- 如果在位置1和位置3之间或者位置7和位置10之间，程序执行停止，那么，CapX指令将如TrigX指令那样发挥作用。
- 含飞焊开始的第一个CAP段建议至少为START distance那么长。
- 如果第一段短于START distance但长于START process_dist，那么位置2和位置4将被移向位置1。
- 如果第一段短于或等于START process_dist，那么，位置1和位置2将重合，位置4将处于该段终点处。
- 含飞焊结束的最后一个CAP段建议至少为END distance + END process_dist那么长。
- 如果最后一段短于END distance + END process_dist但长于END process_dist，那么位置7和位置9将被移向位置10。
- 如果最后一段短于或等于END process_dist，那么，位置8和位置10将重合，位置6将处于该段起点处。

下一页继续

3.35 flypointdata - 飞焊开始/结束数据
Continuous Application Platform (CAP)
续前页

- 仅在应用工艺重启时，采用capdata中指定的开始阶段超时。
- 如果来自运动的预取请求已经到达最后一个CAP指令后（位置9之后）出现进程错误，即，PGM从CAP指令解除，可在下一指令继续，那么将发送错误日志消息，进程将停止，但机器人继续移动。



xx1200000180

组件

from_start

数据类型：bool

未使用。

process_dist

数据类型：num

进程的启动距离（对于飞焊开始）或结束距离（对于飞焊结束）（单位：毫米）。

distance

数据类型：num

将CAP进程监控开始/结束设置为离起点/终点的距离（单位：毫米）。

结构

```
< databases of flypointdata >
  < from_start of bool >
  < process_dist of num >
  < distance of num >
```

相关信息

	参见：
capdata数据类型	第1355页的capdata - CAP数据

下一页继续

	参见：
Continuous Application Platform	应用手册 - *Continuous Application Platform*

3.36 handler_type - 执行处理器的类型

手册用法

handler_type用于指定RAPID程序中执行处理器的类型。

描述

使用函数ExecHandler，有可能检查出是否在RAPID程序中的某些执行处理器中执行实际RAPID事件号。

基本示例

以下示例介绍了数据类型handler_type：

例 1

```
VAR handler_type my_type;
...
my_type := ExecHandler( );
```

将执行事件号的执行处理器的类型储存在变量my_type中。

预定义数据

预定义以下handler_type型常量：

RAPID常量	值	执行处理器的类型
HANDLER_NONE	0	未在任意处理器中执行
HANDLER_BWD	1	在BACKWARD处理器中执行
HANDLER_ERR	2	在ERROR处理器中执行
HANDLER_UNDO	3	在UNDO处理器中执行

特征

handler_type为有关num的一种别名数据类型，并因此继承其特征。

相关信息

信息，关于	请参阅
获取执行处理器的类型	第*1074*页的*ExecHandler - 获取执行处理器的类型*

3.37 icondata - 图标显示数据

手册用法

icondata用于代表用户设备（例如FlexPendant示教器）上的标准图标。

描述

可将icondata枚举常量传递至指令UIMsgBox和函数UIMessageBox、UINumEntry、UINumTune、UIAlphaEntry和UIListView中的Icon参数。

基本示例

以下示例介绍了数据类型icondata：

例 1

```
VAR btnres answer;

UIMsgBox "More ?" \Buttons:=btnYesNo \Icon:=iconInfo \Result:=
     answer;
IF answer= resYes THEN
...
ELSEIF answer =ResNo THEN
...
ENDIF
```

标准按钮枚举常量iconInfo将在用户界面上的消息框顶部给出信息图标。

预定义数据

在系统中预定义数据类型icondata的以下常量：

值	常量	图标
0	iconNone	无图标
1	iconInfo	信息图标
2	iconWarning	警告图标
3	iconError	错误图标

特征

icondata为有关num的一种别名数据类型，并因此继承其特征。

相关信息

信息，关于	请参阅
用户交互消息框	第832页的*UIMsgBox - 用户消息对话框，基本类型*
用户交互消息框	第1317页的*UIMessageBox - 先进型用户消息框*
用户交互数字条目	第1324页的*UINumEntry - 用户数字条目*
用户交互数字调节	第1330页的*UINumTune - 用户数字调节*
用户交互α条目	第1292页的*UIAlphaEntry - 用户α条目*
用户交互列表视图	第1310页的*UIListView - 用户列表视图*

下一页继续

3.37 icondata - 图标显示数据

RobotWare - OS

续前页

信息，关于	请参阅
一般数据类型、别名数据类型	技术参考手册 - *RAPID*语言概览，基本特征 - 数据类型一节

3.38 identno - 移动指令的识别号

手册用法

identno（*Identity Number*）用于控制两个或多个协调同步移动的相互同步。

数据类型identno仅可用于含选项*Coordinated Robots*的*MultiMove*系统中，且仅可用于定义为运动任务的程序任务中。

描述

若为协调同步移动，则必须通过数据类型identno的参数\ID来编程MultiMove系统中的移动指令，且在所有其他情况下均不允许\ID。

在所有合作程序任务中，指定\ID编号必须相同。id编号确保各运动不会在运行时混淆。

在协调同步模式中，所有程序任务中必须存在相同数量的执行移动指令。数据类型identno的可选参数\ID将用于检查相关移动指令在开始移动之前并行运行。\ID编号必须与并行运行的移动指令相同。

用户无须声明任何identno型变量，但是可在指令中直接使用编号（参见基本例子）。

基本示例

以下示例介绍了数据类型identno：

例 1

```
PERS tasks task_list{2} := [["T_ROB1"],["T_ROB2"]];
VAR syncident sync1;
VAR syncident sync2;

PROC proc1()
  ...
  SyncMoveOn sync1, task_list;
  MoveL *\ID:=10,v100,z50,mytool;
  MoveL *\ID:=20,v100,fine,mytool;
  SyncMoveOff sync2;
  ...
ENDPROC
```

特征

identno为有关num的一种别名数据类型，并因此继承其属性。

相关信息

信息，关于	请参阅
别名数据类型	技术参考手册 - *RAPID*语言概览，基本特征 - 数据类型一节
起动协调同步移动	第*709*页的*SyncMoveOn - 起动协调同步移动*
结束协调同步移动	第*704*页的*SyncMoveOff - 结束协调同步移动*

3.39 intnum - 中断识别号

手册用法

intnum（*interrupt numeric*）用于识别一次中断。

描述

当intnum型变量同软中断程序相连时，向其给出识别中断的特定值。随后，在处理中断的过程中使用该变脸，例如当下令进行或禁用中断时。

可将多个中断识别号与相同的软中断程序相连。系统变量INTNO由此可用于软中断程序，以确定所出现中断的类型。

必须始终在模块中声明intnum型变量的全局性。

基本示例

以下示例介绍了数据类型intnum：

例 1

```
VAR intnum feeder_error;
...
PROC main()
  CONNECT feeder_error WITH correct_feeder;
  ISignalDI di1, 1, feeder_error;
```

将输入di1设置为1时，产生中断。此时，调用correct_feeder软中断程序。

例 2

```
VAR intnum feeder1_error;
VAR intnum feeder2_error;
...
PROC init_interrupt()
...
  CONNECT feeder1_error WITH correct_feeder;
  ISignalDI di1, 1, feeder1_error;
  CONNECT feeder2_error WITH correct_feeder;
  ISignalDI di2, 1, feeder2_error;
...
ENDPROC
...
TRAP correct_feeder
  IF INTNO=feeder1_error THEN
  ...
  ELSE
  ...
  ENDIF
...
ENDTRAP
```

将输入di1或di2设置为1时，产生中断。随后，调用correct_feeder软中断程序。将系统变量INTNO用于软中断程序，以查明已出现中断的类型。

下一页继续

限制

任一时间（在 intnum 与 CONNECT 之前） IDelete 类型有效变量的最大数量限制为 100。在任一时间 TRAP 例行程序执行队列中的中断的最大数量限制为 30。

特征

Intnum为有关num的一种别名数据类型，并因此继承其属性。

相关信息

信息，关于	请参阅
中断概述	技术参考手册 - *RAPID*语言概览，*RAPID*概要 - 中断一节
别名数据类型	技术参考手册 - *RAPID*语言概览，基本特征 - 数据类型一节
连接中断	第*124*页的*CONNECT* - 将中断与软中断程序相连

3.40 iodev - 串行通道和文件

手册用法

iodev （*I/O*设备）用于串行通道，例如打印机和文件。

描述

iodev型数据包含对文件或串行通道的引用。其可通过指令Open而与物理单位相关联，随后用于读取和写入。

基本示例

以下示例介绍了数据类型iodev：

例 1

```
VAR iodev file;
...
Open "HOME:/LOGDIR/INFILE.DOC", file\Read;
input := ReadNum(file);
```

打开文件 INFILE.DOC以进行读取。当读取文件时，将file作为参考而非文件名。

特征

iodev是一种非数值的数据类型。

相关信息

信息，关于	请参阅
通过串行通道通信	技术参考手册 - *RAPID*语言概览，*RAPID*概要 - 通信一节
串行通道的配置	技术参考手册 - 系统参数
非值数据类型的特征	技术参考手册 - *RAPID*语言概览，基本特征 - 数据类型一节
文件和串行通道处理	应用手册 - 控制器软件*IRC5*

3.41 iounit_state - I/O单元的状态

手册用法

iounit_state用于反映I/O单元当前所在状态。

描述

当检查函数IOUnitState的返回值时，打算使用iounit_state常量。

基本示例

以下示例介绍了数据类型iounit_state：

例 1

```
IF (IOUnitState ("UNIT1" \Phys) = IOUNIT_PHYS_STATE_RUNNING) THEN
  ! Possible to access some signal on the I/O unit
ELSE
  ! Read/Write some signal on the I/O unit result in error
ENDIF
```

如果I/O单元UNIT1运行正常，则完成测试。

预定义数据

在函数IOUnitState中发现数据类型iounit_state的预定义符号常量。

特征

iounit_state为有关num的一种别名数据类型，并因此继承其特征。

相关信息

信息，关于	请参阅
获取I/O单元的当前状态	第*1123*页的*IOUnitState - 获取I/O单元的当前状态*
输入/输出指令	技术参考手册 - *RAPID*语言概览，*RAPID*概要 - 输入和输出信号一节
输入/输出功能性概述	技术参考手册 - *RAPID*语言概览，运动和*I/O*原则 - *I/O*原则一节
I/O配置	技术参考手册 - 系统参数

3.42 jointtarget - 接头位置数据

手册用法

jointtarget用于确定通过指令MoveAbsJ而将机械臂和外轴移动到的位置。

描述

jointtarget规定机械臂和外轴的各单独轴位置。

组件

robax

robot axes

数据类型：robjoint

机械臂轴的轴位置，以度数计。

将轴位置定义为各轴（臂）从轴校准位置沿正方向或反方向旋转的度数。

extax

external axes

数据类型：extjoint

外轴的位置。

各单独轴（eax_a、eax_b ... eax_f）的位置 is defin如下：

- 对于旋转轴，其位置定义为从校准位置起旋转的度数。
- 对于线性轴，其位置定义为与校准位置的距离（以mm计）。

外轴eax_a ... 为逻辑轴。在系统参数中定义逻辑轴编号与物理轴编号的相互关系。

定义未连接轴的值9E9。如果位置数据中定义的轴不同于程序执行中实际连接的轴，则以下内容适用：

- 如果未在位置数据（值9E9）中定义位置，则当连接且未激活该轴时，将忽略该值。但是如果激活该轴，则将会产生错误。
- 如果在位置数据中定义该位置，则忽略该值，即便未连接该轴。

如果未激活轴，则不会进行任何运动，且不会产生有关轴（含正确位置数据）的错误。

如果某些外轴正以独立模式运行，且应由机械臂及其外轴实施某些新运动，则有关独立模式的外轴的位置数据必须为9E9以外的任意值（未使用，但是供系统处理其他方面）。

基本示例

以下示例介绍了数据类型jointtarget：

例 1

```
CONST jointtarget calib_pos := [ [ 0, 0, 0, 0, 0, 0], [ 0, 9E9,
    9E9, 9E9, 9E9, 9E9] ];
```

通过数据类型jointtarget，在calib_pos定义IRB2400的正常校准位置。同时定义外部逻辑轴a的正常校准位置0（度或毫米）。未定义外轴b到f。

下一页继续

结构

```
< dataobject of jointtarget >
  < robax of robjoint >
    < rax_1 of num >
    < rax_2 of num >
    < rax_3 of num >
    < rax_4 of num >
    < rax_5 of num >
    < rax_6 of num >
  < extax of extjoint >
    < eax_a of num >
    < eax_b of num >
    < eax_c of num >
    < eax_d of num >
    < eax_e of num >
    < eax_f of num >
```

相关信息

信息，关于	请参阅
移动至接头位置	第*335*页的*MoveAbsJ - 移动机械臂至绝对接头位置* 第*364*页的*MoveExtJ - 在没有TCP的情况下，移动一个或数个机械单元*
定位器指令	技术参考手册 - *RAPID*语言概览，*RAPID*概要 - 运动一节
外轴的配置	应用手册 - *Additional axes and stand alone controller*

3.43 listitem - 列表项数据结构

手册用法

listitem用于定义包括用户设备（诸如FlexPendant示教器）上文本以及可选小图标的菜单行。

描述

listitem型数据允许用户定义函数UIListView的菜单行。

基本示例

以下示例介绍了数据类型listitem ：

例 1

```
CONST listitem list {3}:=[[stEmpty, "Item1"], [stEmpty, "Item2"],
     [stEmpty, "Item3"]];
```

在函数UIListView.中使用菜单列表以及项1至项3

组件

此数据类型有以下组件：

image

数据类型：string

路径包括待显示图标图像的文件名（未在本软件版本中执行）。

如果未显示图标，则使用空字符串 ""或stEmpty。

text

数据类型：string

菜单行有待显示的文本。

结构

```
<dataobject of listitem>
  <image of string>
  <text of string>
```

相关信息

信息，关于	请参阅
用户互动列表试图	第*1310*页的*UIListView - 用户列表视图*

3.44 loaddata - 加载数据

手册用法

loaddata用于描述附于机械臂机械界面（机械臂安装法兰）的负载。

负载数据常常定义机械臂的有效负载或支配负载（通过定位器的指令GripLoad或MechUnitLoad来设置），即机械臂夹具所施加的负载。同时将loaddata作为tooldata的组成部分，以描述工具负载。

描述

指定的负载用于设置机械臂的动态模型，以便可以以最佳的方式来控制机械臂运动。

警告

重要的是，始终定义实际工具负载以及使用时机械臂（例如，抓取部分）的有效负载。负载数据定义不正确可能会导致机械臂机械结构过载。

指定不正确的负载数据时，其常常会引起以下后果：

- 机械臂将不会用于其最大容量
- 路径准确性受损，包括过度风险
- 机械结构过载风险

组件

注意

在此描述中，仅将loaddata描述为用于有效负载。当用于工具负载时，请参见第1502页的*tooldata - 工具数据*。

`mass`

数据类型：num

负载的质量，以kg计。

`cog`

center of gravity

数据类型：pos

如果机械臂正夹持着工具，则用工具坐标系表示有效负载的重心，以mm计。如果使用固定工具，则用机械臂所移动工件的坐标系来表示夹具所夹持有效负载的重心。

`aom`

axes of moment

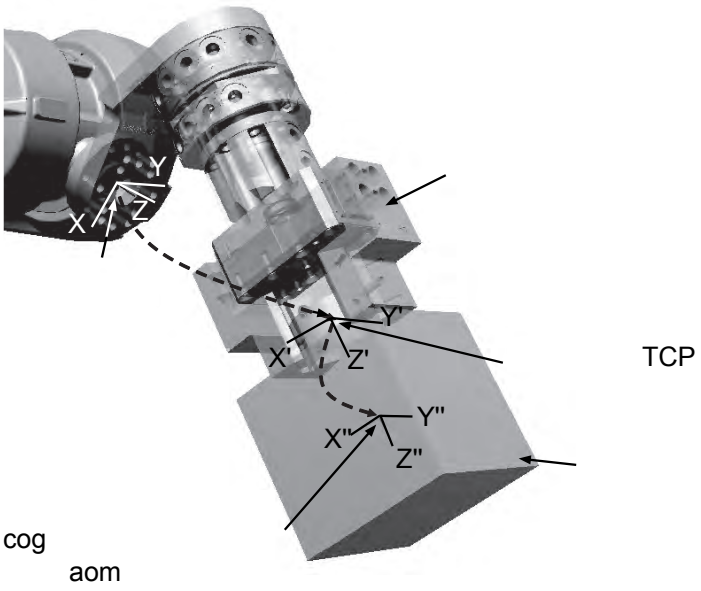
数据类型：orient

矩轴的姿态。存在始于cog的有效负载惯性矩的主轴。如果机械臂正夹持着工具，则用工具坐标系来表示矩轴。

下一页继续

3.44 loaddata - 加载数据
RobotWare - OS
续前页

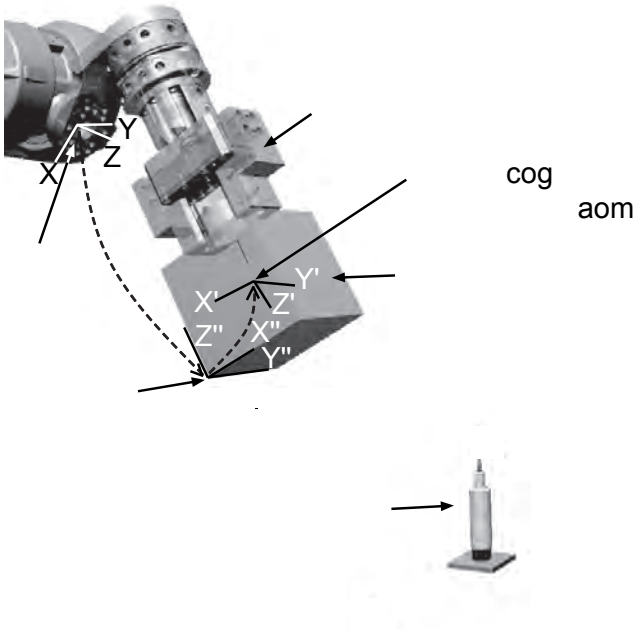
本图显示了有效负载的重心和惯性轴。



xx1100000515

Figure 3.1: 机械臂夹持的工具

如果使用固定工具，则用目标坐标系来表示矩轴。



xx1100000516

Figure 3.2: 固定工具

下一页继续

注意

如果使用StationaryPayLoadMode，则用腕坐标系来表示固定工具的矩轴。

ix

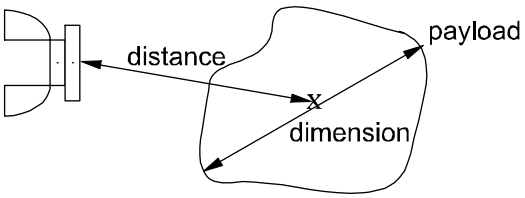
inertia x

数据类型：num

力矩x轴负载的惯性矩，以kgm^{2}计。

惯性矩的正确定义将允许合理利用路径规划器和轴控制器。当处理大张金属板等时，上述规定尤为重要。所有等于0 kgm^{2}的惯性矩ix、iy和iz均意指一个点质量。

通常，仅当安装法兰距重心的距离小于负载的最大维度（参见下图）时，方可定义惯性矩。



xx0500002372

iy

inertia y

数据类型：num

y轴负载的惯性矩，以kgm^{2}计。

有关更多信息，请参见ix。

iz

inertia z

数据类型：num

z轴负载的惯性矩，以kgm^{2}计。

有关更多信息，请参见ix。

基本示例

以下示例介绍了数据类型loaddata：

例 1

```
PERS loaddata piece1 := [ 5, [50, 0, 50], [1, 0, 0, 0], 0, 0, 0];
```

如图所示，通过机械臂所夹持的工具来移动有效负载。使用以下值来描述第*1422*页的机械臂夹持的工具：

- 重量5 kg。
- 重心为工具坐标系中的x=50，y=0和z=50 mm

下一页继续

3.44 loaddata - 加载数据
RobotWare - OS
续前页

- 有效负载为一个点质量

例 2

```
Set gripper;
WaitTime 0.3;
GripLoad piece1;
```

在机械臂抓握负载的同时，指定有效负载的连接，piece1。

例 3

```
Reset gripper;
WaitTime 0.3;
GripLoad load0;
```

在机械臂释放有效负载的同时，规定断开有效负载。

例 4

```
PERS loaddata piece2 := [ 5, [50, 50, 50], [0, 0, 1, 0], 0, 0, 0];
PERS wobjdata wobj2 :=[ TRUE, TRUE, "", [ [0, 0, 0], [1, 0, 0 ,0]
     ], [ [50, -50, 200], [0.5, 0, -0.866 ,0] ] ];
```

根据图中的固定工具，移动有效负载。使用有关loaddata的以下值，描述第*1422*页的固定工具：

- 重量：5 kg
- 工件wobj2的重心为目标坐标系中的x=50，y=50和z=50mm
- 根据目标坐标系，有效负载坐标系/矩轴围绕Y"旋转180°
- 有效负载为一个点质量

以下值用于wobjdata：

- 机械臂正夹持着工件
- 使用固定用户坐标系，即用户坐标系与腕坐标系相同
- 目标坐标系围绕Y旋转-120°，且其在用户坐标系中的原点坐标为x=50，y=-50和z=200mm

限制

仅应当将有效负载定义为永久变量（PERS），且不在程序内。随后，在保存程序时保存当前值，并在加载时进行检索。

GripLoad和MechUnitLoad指令中的loaddata型参数仅应为一个完整的永久数据对象（而非数组元素或记录成分）。

预定义数据

负载load0定义一个有效负载，其质量等于0 kg，即没有负载。将该负载作为指令GripLoad和MechUnitLoad中的参数，以断开有效负载。

始终可以从程序获得负载load0，但是无法进行改变（将其储存在系统程序模块BASE中）。

```
PERS loaddata load0 := [ 0.001, [0, 0, 0.001], [1, 0, 0, 0], 0, 0
     , 0 ];
```

结构

```
< dataobject of loaddata >
  < mass of num >
```

下一页继续

```
< cog of pos >
  < x of num >
  < y of num >
  < z of num >
< aom of orient >
  < q1 of num >
  < q2 of num >
  < q3 of num >
  < q4 of num >
< ix of num >
< iy of num >
< iz of num >
```

相关信息

信息，关于	请参阅
坐标系	技术参考手册 - *RAPID*语言概览
工具负载定义	第*1502*页的*tooldata - 工具数据*
定义机械臂的有效负载	第*223*页的*GripLoad - 定义机械臂的有效负载*
确定机械单元的有效负载	第*327*页的*MechUnitLoad - 确定机械单元的有效负载*
工具负载、有效负载或手臂负载的负载识别	操作员手册 - 带 *FlexPendant* 的 *IRC5*
手臂负载定义	技术参考手册 - 系统参数，题目*Motion*-工作流程-如何定义手臂负载一节
工件数据的定义	第*1523*页的*wobjdata - 工件数据*
StationaryPayLoadMode	技术参考手册 - 系统参数，StationaryPayLoadMode一节。

3.45 loadidnum - 负载识别的类型

手册用法

loadidnum用于代表一个含符号常量的整数。

描述

loadidnum常量旨在用于工具或有效负载（作为指令LoadId中的参数）的负载识别。参见以下例子。

基本示例

以下示例介绍了数据类型loadidnum：

例 1

```
! Load modules into the system
Load \Dynamic, "RELEASE:/system/mockit.sys";
Load \Dynamic, "RELEASE:/system/mockit1.sys";
%"LoadId"% TOOL_LOAD_ID, MASS_WITH_AX3, gun1;
```

通过质量识别、机械臂轴3的移动以及数据类型loadidnum的预定义常量MASS_WITH_AX3的使用，进行工具gun1的负载识别。

预定义数据

预定义数据类型loadidnum的以下符号常量，并作为指令LoadId中的参数。

值	符号常量	备注
1	MASS_KNOWN	工具或有效负载中的已知质量。
2	MASS_WITH_AX3	工具或有效负载中的未知质量。将通过轴3的移动来实现质量识别。

特征

loadidnum 为有关num的一种别名数据类型，并因此继承其特征。

相关信息

信息，关于	请参阅
预定义程序负载识别	操作员手册 - 带 *FlexPendant* 的 *IRC5*，编程和测试-服务程序-*LoadIdentify*、负载识别和服务程序一节
有效的机械臂类型	第*1167*页的*ParIdRobValid* - 用于参数识别的有效机械臂类型
有效的机械臂位置	第*1164*页的*ParIdPosValid* - 用于参数识别的有效机械臂位置
带完整例子的负载识别	第*316*页的*LoadId* - 工具或有效负载的负载识别

3.46 loadsession - 程序加载会话

手册用法

`loadsession`用于定义RAPID程序模块的不同加载会话。

描述

在指令`StartLoad`和`WaitLoad`中使用`loadsession`型数据，以识别加载会话。`loadsession`仅包含对加载会话的引用。

特征

`loadsession`为非值数据类型，且无法用于以值为导向的运算。

相关信息

信息，关于	请参阅
执行期间，加载普通程序模块	第*660*页的*StartLoad* - 执行期间，加载普通程序模块 第*881*页的*WaitLoad* - 将加载的模块与任务相连
非值数据类型的特征	技术参考手册 - *RAPID*语言概览，基本特征 - 数据类型一节

3.47 mecunit - 机械单元

手册用法

mecunit用于定义不同的机械单元，可从程序来控制和访问此类机械单元。

在系统参数中定义机械单元的名称，因此，不得在程序中进行定义。

描述

mecunit类数据仅包含对机械单元的引用。

限制

mecunit类数据不得在程序中定义。但是如果定义了，则会在引用此mecunit的指令或函数执行时立即显示一个错误消息。另外，此数据类型也可以在声明例行程序时用作参数。

预定义数据

在各程序任务中预定义系统参数中定义的机械单元。但是，仅将实际程序任务（在系统参数*Controller/Task/Use Mechanical Unit Group*中定义）控制的机械单元用于进行控制操作。

此外，可在各程序任务中使用数据类型mecunit的预定义变量ROB_ID。如果实际程序控制机械臂，则别名变量ROB_ID包含对机械臂ROB_1到ROB_6之一的引用，其可用于对机械臂进行控制操作。如果实际程序任务未控制任何机械臂，则变量ROB_ID无效。

基本示例

以下示例介绍了数据类型mecunit：

例 1

```
IF TaskRunRob() THEN
  IndReset ROB_ID, 6;
ENDIF
```

如果实际程序任务控制机械臂，则重置机械臂的轴6。

特征

mecunit是一个非值数据类型。这就表示此类数据不允许面向数值的操作。

相关信息

信息，关于	请参阅
检查任务是否使一些机械臂运行	第*1269*页的*TaskRunRob* - 检查任务是否控制一些机械臂
检查任务是否使一些机械单元运行	第*1268*页的*TaskRunMec* - 检查任务是否控制所有机械单元
获得系统中机械单元的名称	第*1092*页的*GetNextMechUnit* - 获取有关机械单元的名称和数据
启用/停用机械单元	第*24*页的*ActUnit* - 启用机械单元 第*140*页的*DeactUnit* - 停用机械单元
机械单元的配置	技术参考手册 - 系统参数

下一页继续

续前页

信息，关于	请参阅
非值数据类型的特征	技术参考手册 - *RAPID*语言概览，基本特征 - 数据类型一节

3.48 motsetdata - 运动设置数据

手册用法

motsetdata用于定义影响程序中所有定位指令的一系列运动设置：

- 最高速率和速率覆盖
- 加速度数据
- 围绕奇异点的行为
- 不同机械臂配置的管理
- 路径分辨率覆盖
- 动作监控
- 加速度/减速度限制
- 圆周路径期间的工具方位调整
- 事件缓冲区的启用和停用

通常无需使用此数据类型，因为仅可通过使用指令VelSet、AccSet、SingArea、ConfJ、ConfL、PathResol、MotionSup、PathAccLim、CirPathMode、WorldAccLim、ActEventBuffer和DeactEventBuffer来进行此类设置。

使用系统变量C_MOTSET，访问此类运动设置的当前值。

描述

当前运动设置（储存在系统变量C_MOTSET中）影响所有运动。

组件

vel.oride

数据类型：veldata/num

速率占编程速率的百分比。

vel.max

数据类型：veldata/num

最大速率，以mm/s计。

acc.acc

数据类型：accdata/num

加速度和减速度占正常值的百分比。

acc.ramp

数据类型：accdata/num

加速度和减速度增加速度占正常值的百分比。

acc.finepramp

数据类型：accdata/num

当机械臂朝一个精确点减少时，减速度减少速度占正常值的百分比。

sing.wrist

数据类型：singdata/bool

允许工具方位出现某种程度的偏差，以防止出现腕奇异点。

下一页继续

sing.lockaxis4

数据类型：singdata/bool

将六轴机械臂上的轴4锁定在0或+-180度，以避免在轴5接近于零时的奇异点问题。

sing.arm

数据类型：singdata/bool

允许工具方位出现某种程度的偏差，以防止出现臂奇异点（未执行）。

sing.base

数据类型：singdata/bool

不允许工具方位出现偏差。

conf.jsup

数据类型：confsupdata/bool

接头运动期间，接头配置监督生效。

conf.lsup

数据类型：confsupdata/bool

线性和圆周运动期间，接头配置监督生效。

conf.ax1

数据类型：confsupdata/num

轴1的最大允许偏差度数（未在本版本中使用）。

conf.ax4

数据类型：confsupdata/num

轴4的最大允许偏差度数（未在本版本中使用）。

conf.ax6

数据类型：confsupdata/num

轴6的最大允许偏差度数（未在本版本中使用）。

pathresol

数据类型： num

当前覆盖占配置路径分辨率的百分比。

motionsup

数据类型：bool

反射运动监控函数的RAPID状态（TRUE=启和FALSE=闭）。

tunevalue

数据类型：num

当前RAPID覆盖占运动监控函数配置调整值的百分比。

acclim

数据类型：bool

路径沿线的工具加速度限制（TRUE=限制和FALSE=未限制）。

下一页继续

3.48 motsetdata - 运动设置数据

RobotWare - OS

续前页

accmax

数据类型：num

TCP加速度限制，以m/s^{2}计。如果acclim为FALSE，则始终将值设置为-1。

decellim

数据类型：bool

路径沿线的工具减速度限制（TRUE=限制和FALSE=未限制）。

decelmax

数据类型：num

TCP减速度限制，以m/s^{2}计。如果decellim为FALSE，则始终将值设置为-1。

cirpathreori

数据类型：num

圆周路径期间的工具方位调整：

0=插补到路径坐标系的标准方法

1=插补到工件坐标系的改良方法

2=使用CirPoint中编程工具方位的改良方法

worldacclim

数据类型：bool

世界坐标系中的加速度限制（TRUE=限制和FALSE=未限制）。

worldaccmax

数据类型：num

世界坐标系中的加速度限制，以m/s^{2}计。如果worldacclim为FALSE，则始终将值设置为-1。

evtbufferact

数据类型：bool

事件缓冲区生效或未生效（TRUE=事件缓冲区生效和FALSE=事件缓冲区未生效）。

限制

仅部件sing.wrist、sing.arm或sing.base 之一可能拥有等于TRUE的值。

基本示例

以下示例介绍了数据类型motsetdata：

例 1

```
IF C_MOTSET.vel.oride > 50 THEN
  ...
ELSE
  ...
ENDIF
```

根据当前速率覆盖，执行程序的不同部分。

下一页继续

预定义数据

C_MOTSET描述机械臂的当前运动设置，且始终可以从程序进行访问。另一方面，仅可通过使用一系列指令而非通过分配来改变C_MOTSET。

设置有关运动参数的以下默认值

- 当使用重启模式重置**RAPID**时
- 当加载一则新程序或一个新模块时
- 当从起点开始执行程序时
- 当将程序指针移动到main时
- 当将程序指针移动到子程序时
- 移动程序指针造成执行顺序丢失时

```
VAR motsetdata C_MOTSET := [
  [ 100, 5000 ],-> veldata
  [ 100, 100, 100 ],-> accdata
  [ FALSE, FALSE, FALSE, TRUE ],-> singdata
  [ TRUE, TRUE, 30, 45, 90 ]-> confsupdata
  100,-> path resolution
  TRUE,-> motionsup
  100,-> tunevalue
  FALSE,-> acclim
  -1,-> accmax
  FALSE,-> decellim
  -1,-> decelmax
  0,-> cirpathreori
  FALSE,-> worldacclim
  -1,-> worldaccmax
  TRUE];-> ActEventBuffer and DeactEventBuffer
```

结构

```
<dataobject of motsetdata>
 <vel of veldata>                    受指令VelSet影响
   <oride of num>
   <max of num>
 <acc of accdata>                    受指令AccSet影响
   <acc of num>
   <ramp of num>
   <finepramp of num>
 <sing of singdata>                  受指令SingArea影响
   <wrist of bool>
   <lockaxis4 of bool>
   <arm of bool>
   <base of bool>
 <conf of confsupdata>               受指令ConfJ和ConfL影响
   <jsup of bool>
   <lsup of bool>
   <ax1 of num>
   <ax4 of num>
   <ax6 of num>
```

下一页继续

3.48 motsetdata - 运动设置数据
RobotWare - OS
续前页

<pathresol of num>	受指令PathResol影响
<motionsup of bool>	受指令MotionSup影响
<tunevalue of num>	受指令MotionSup影响
<acclim of bool>	受指令PathAccLim影响
<accmax of num>	受指令PathAccLim影响
<decellim of bool>	受指令PathAccLim影响
<decelmax of num>	受指令PathAccLim影响
<cirpathreori of num>	受指令CirPathMode影响
<worldacclim of bool>	受指令WorldAccLim影响
<worldaccmax of num>	受指令WorldAccLim影响
<evtbufferact of bool>	受指令ActEventBuffer和DeactEventBuffer影响

相关信息

信息，关于	请参阅
有关设置运动参数的指令	技术参考手册 - *RAPID*语言概览，*RAPID*概要 - 运动设置一节

3.49 num - 数值

手册用法

Num 用于数值；例如，计数器。

描述

num数据类型的值可以为

- 整数；例如，-5
- 小数；例如，3.45

其亦可呈指数地写入；例如，2E3（= 2*10^3 = 2000），2.5E-2（= 0.025）。

始终将-8388607与+8388608之间的整数作为准确的整数储存。

小数仅为近似数字，因此，不得用于等于或不等于对比。若为使用小数的除法和运算，则结果亦将为小数；即并非一个准确的整数。例如：

```
a := 10;
b := 5;
IF a/b=2 THEN

...
```

由于a/b的结果并非一个整数，因此，无须满足此条件。

基本示例

以下示例介绍了数据类型num：

例 1

```
VAR num reg1;
...
reg1 := 3;
```

将reg1指定为值3。

例 2

```
a := 10 DIV 3;
b := 10 MOD 3;
```

整数除法，向 a分配一个整数（= 3），并向b分配余数（= 1）。

预定义数据

系统中存在一些预定义数据。例如，已定义常量pi（π）。

常量pi：=3.1415926；

限制

将分配给一个num变量的介于-8388607到8388608之间的字面值作为准确的整数来储存。

如果将已作为一个dnum解释的字面值作为一个num来分配/使用，其自动转换为一个num。

下一页继续

3.49 num - 数值
RobotWare - OS
续前页

相关信息

信息，关于	请参阅
使用数据类型dnum的数值	第*1385*页的*dnum - 双数值*
数字表达式	技术参考手册 - *RAPID*语言概览，基础*RAPID*编程-表达式一节
使用数值进行运算	技术参考手册 - *RAPID*语言概览，基础*RAPID*编程-表达式一节

3.50 opcalc - Arithmetic Operator

手册用法

opcalc用于代表RAPID函数或指令的参数中的一个算术运算符。

描述

opcalc常量旨在用于确定算术运算的类型。

示例

以下示例介绍了数据类型opcalc：

例 1

```
res := StrDigCalc(str1, OpAdd, str2);
```

向res 分配有关字符串 str1和str2所代表值的加法运算的结果。OpAdd 为数据类型opcalc.

预定义数据

预定义数据类型opcalc的以下符号常量，并将其用于定义所用算术运算的类型，例如，在函数StrDigCalc中。

常量	值	备注
OpAdd	1	加法（+）
OpSub	2	减法（-）
OpMult	3	乘法（*）
OpDiv	4	除法（/）
OpMod	5	模数（%I）

特征

opcalc为有关num的一种别名数据类型，并因此继承其特征。

相关信息

信息，关于	请参阅
一般数据类型、别名数据类型	技术参考手册-*RAPID*概述，基本特征-数据类型一节
有关数字字符串的算术运算。	第*1245*页的*StrDigCalc* - 有关数据类型数字字符串的算术运算

3.51 opnum - 比较运算符

手册用法

opnum用于代表RAPID函数或指令的参数中用作对比的运算符。

描述

opnum常量旨在用于定义在检查通用指令中各值时的对比类型。

基本示例

以下示例介绍了数据类型opnum：

例 1

```
TriggCheckIO checkgrip, 100, airok, EQ, 1, intno1;
```

预定义数据

预定义数据类型opnum的以下符号常量，并将其用于定义所用对比的类型，例如，在函数TriggCheckIO中。

值	符号常量	备注
1	LT	小于
2	LTEQ	小于或等于
3	EQ	等于
4	NOTEQ	不等于
5	GTEQ	大于或等于
6	GT	大于

特征

opnum为有关num的一种别名数据类型，并因此继承其特征。

相关信息

信息，关于	请参阅
一般数据类型、别名数据类型	技术参考手册 - *RAPID*语言概览，基本特征 - 数据类型一节
定义固定位置的I/O检查	第*754*页的*TriggCheckIO* - 定义位于固定位置的*IO*检查

3.52 orient-姿态

手册用法

orient用于姿态（例如，工具方位）和旋转（例如，坐标系旋转）。

描述

以四元数的形式来描述姿态，该四元数包括四个组成部分：q1、q2、q3和q4。

组件

数据类型orient拥有以下组成部分：

q1

数据类型：num

四元数1

q2

数据类型：num

四元数2

q3

数据类型：num

四元数3

q4

数据类型：num

四元数4

基本示例

以下示例介绍了数据类型orient：

例 1

```
VAR orient orient1;
.
orient1 := [1, 0, 0, 0];
```

向orient1姿态分配值q1=1，q2-q4=0；这相当于未旋转。

限制

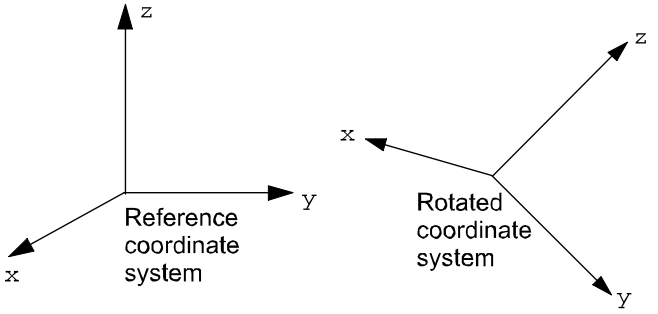
必须使姿态规范化，即平方和必须等于1：

$$q_1^2 + q_2^2 + q_3^2 + q_4^2 = 1$$

下一页继续

什么是四元数？

通过用于描述坐标系各轴相对于参考系统的方向的旋转矩阵，描述坐标系的姿态（诸如工具的姿态）（参见下图）。



xx0500002376

旋转后的坐标系轴（x，y，z）为矢量，其可以用参考坐标系表示如下：

x = (x1, x2, x3)

y = (y1, y2, y3)

z = (z1, z2, z3)

这意味着参考坐标系中x矢量的x轴向分量将为x1，y轴向分量将为x2，以此类推。

这三个矢量可在一个矩阵（旋转矩阵）中组合，各矢量由此构成一栏：

$$\begin{bmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \end{bmatrix}$$

xx0500002381

四元数仅仅是一种描述此旋转矩阵的更为简洁的方式；根据旋转矩阵的各元素，计算四元数：

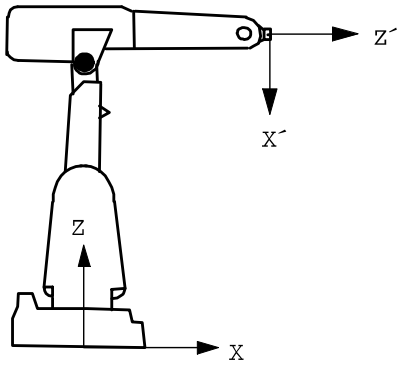
q1 = $\frac{\sqrt{x_1+y_2+z_3+1}}{2}$	
q2 = $\frac{\sqrt{x_1-y_2-z_3+1}}{2}$	符号q2=符号（y3-z2）
q3 = $\frac{\sqrt{y_2-x_1-z_3+1}}{2}$	符号q3=符号（z1-x3）
q4 = $\frac{\sqrt{z_3-x_1-y_2+1}}{2}$	符号q4=符号（x2-y1）

下一页继续

例 1

给工具定位，以便其Z'轴各点一直向前（与基座坐标系X轴的方向相同）。工具的Y'轴相当于基座坐标系的Y轴（参见下图）。位置数据中定义的工具方位如何（robtarget）？

编程位置的工具方位通常与所用工件的坐标系相关。在本例子中，未使用任何工件，且基座坐标系等于世界坐标系。因此，方位与基座坐标系相关。



xx0500002377

随后，各轴将相关，如下：

x'=-z=（0，0，-1）

y'=y=（0，1，0）

z'=x=（1，0，0）

哪一个相当于以下旋转矩阵：

$$\begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{bmatrix}$$

xx0500002388

旋转矩阵提供相应的四元数：

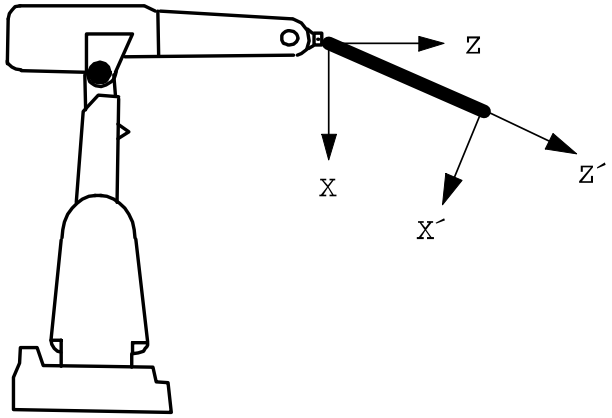
q1 = $\frac{\sqrt{0+1+0+1}}{2}$ = $\frac{\sqrt{2}}{2}$ = 0.707	
q2 = $\frac{\sqrt{0-1-0+1}}{2}$ = 0	
q3 = $\frac{\sqrt{1-0-0+1}}{2}$ = $\frac{\sqrt{2}}{2}$ = 0.707	符号q3=符号（1+1）=+
q4 = $\frac{\sqrt{0-0-1+1}}{2}$ = 0	

下一页继续

3.52 orient-姿态
RobotWare - OS
续前页

例 2

使工具方向相对于有关腕坐标系的X'轴和Z'轴旋转30°（参见下图）。如何在工具数据中定义工具方位？



xx0500002378

随后，各轴将相关，如下：

x'=（cos30°，0，-sin30°）

y'=（0，1，0）

z'=（sin30°，0，cos30°）

哪一个相当于以下旋转矩阵：

$$\begin{bmatrix} \cos 30^\circ & 0 & \sin 30^\circ \\ 0 & 1 & 0 \\ -\sin 30^\circ & 0 & \cos 30^\circ \end{bmatrix}$$

xx0500002393

旋转矩阵提供相应的四元数：

q1 = $\frac{\sqrt{cos30^\circ+1+cos30^\circ+1}}{2}$ = 0.965926	
q2 = $\frac{\sqrt{cos30^\circ-1-cos30^\circ+1}}{2}$ = 0	
q3 = $\frac{\sqrt{1-cos30^\circ-cos30^\circ+1}}{2}$ = 0.258819	符号q3=符号（sin30°+sin30°）=+
q4 = $\frac{\sqrt{cos30^\circ-cos30^\circ-1+1}}{2}$ = 0	

结构

```
< dataobject of orient >
  < q1 of num >
  < q2 of num >
  < q3 of num >
  < q4 of num >
```

下一页继续

相关信息

信息，关于	请参阅
方位运算	技术参考手册 - *RAPID*语言概览，基本特征 - 表达式一节

3.53 paridnum - 参数识别的类型

手册用法

paridnum用于代表一个含符号常量的整数。

描述

paridnum常量旨在用于参数识别，例如，工具、有效负载或外部机械臂负载的负载识别。

基本示例

以下示例介绍了数据类型paridnum：

例 1

```
TEST ParIdRobValid (TOOL_LOAD_ID)
  CASE ROB_LOAD_VAL:
  ! Possible to do load identification of tool in actual robot type
  ...
  CASE ROB_LM1_LOAD_VAL:
  ! Only possible to do load identification of tool with
  ! IRB 6400FHD if actual load < 200 kg
  ...
  CASE ROB_NOT_LOAD_VAL:
  ! Not possible to do load identification of tool in actual robot
       type
  ...
ENDTEST
```

使用数据类型paridnum的预定义常量TOOL_LOAD_ID。

预定义数据

预定义数据类型paridnum的以下符号常量，并将其作为以下指令（ParIdRobValid、ParIdPosValid,LoadId和ManLoadIdProc）的参数。

值	符号常量	备注
1	TOOL_LOAD_ID	确定工具负载
2	PAY_LOAD_ID	确定有效负载（参考指令GripLoad）
3	IRBP_K	识别外机械臂IRBP K负载
4	IRBP_L	识别外机械臂IRBP L负载
4	IRBP_C	识别外机械臂IRBP C负载
4	IRBP_C_INDEX	识别外机械臂IRBP C_INDEX负载
4	IRBP_T	识别外机械臂IRBP T负载
5	IRBP_R	识别外机械臂IRBP R负载
6	IRBP_A	识别外机械臂IRBP A负载
6	IRBP_B	识别外机械臂IRBP B负载
6	IRBP_D	识别外机械臂IRBP D负载

下一页继续

注意

仅在用户针对机械臂的工具或有效负载的负载识别而定义的RAPID程序中使用TOOL_LOAD_ID和PAY_LOAD_ID。

特征

paridnum 为有关num的一种别名数据类型，并因此继承其特征。

相关信息

信息，关于	请参阅
预定义程序Load Identify	操作员手册 - 带 *FlexPendant* 的 *IRC5*，编程和测试-服务程序-*LoadIdentify*、负载识别和服务程序一节
有效的机械臂类型	第*1167*页的*ParIdRobValid* - 用于参数识别的有效机械臂类型
有效的机械臂位置	第*1164*页的*ParIdPosValid* - 用于参数识别的有效机械臂位置
带完整例子的负载识别	第*316*页的*LoadId* - 工具或有效负载的负载识别
外机械臂的负载识别	第*323*页的*ManLoadIdProc* - *IRBP*机械臂的负载识别

3.54 paridvalidnum - ParIdRobValid的结果

手册用法

paridvalidnum用于代表一个含符号常量的整数。

描述

paridvalidnum常量旨在用于参数识别，例如，在检查函数ParIdRobValid的返回值时，工具或有效负载的负载识别。

基本示例

以下示例介绍了数据类型paridvalidnum：

```
TEST ParIdRobValid (PAY_LOAD_ID)
  CASE ROB_LOAD_VAL:
  ! Possible to do load identification of payload in actual robot
  ! type
  ...
  CASE ROB_LM1_LOAD_VAL:
  ! Only possible to do load identification of payload
  ! with IRB 6400FHD if actual load < 200 kg
  ...
  CASE ROB_NOT_LOAD_VAL:
  ! Not possible to do load identification of payload
  ! in actual robot type
  ...
ENDTEST
```

使用数据类型paridvalidnum的预定义常量ROB_LOAD_VAL、ROB_LM1_LOAD_VAL 和 ROB_NOT_LOAD_VAL 。

预定义数据

预定义数据类型paridvalidnum的以下符号常量，并将其用于检查函数 ParIdRobValid的返回值。

值	符号常量	备注
10	ROB_LOAD_VAL	有关当前参数识别的有效机械臂类型
11	ROB_NOT_LOAD_VAL	有关当前参数识别的无效机械臂类型
12	ROB_LM1_LOAD_VAL	如果实际负载<200kg，则有关当前参数识别的有效机械臂类型为IRB 6400FHD。

特征

paridvalidnum为有关num的一种别名数据类型，并继承其特征。

相关信息

信息，关于	请参阅
预定义程序Load Identify	操作员手册 - 带 *FlexPendant* 的 *IRC5*，编程和测试-服务程序-*LoadIdentify*、负载识别和服务程序一节
有效的机械臂类型	第*1167*页的*ParIdRobValid* - 用于参数识别的有效机械臂类型

下一页继续

续前页

信息，关于	请参阅
有效的机械臂位置	第*1164*页的*ParIdPosValid* - 用于参数识别的有效机械臂位置
带完整例子的负载识别	第*316*页的*LoadId* - 工具或有效负载的负载识别

3.55 pathrecid - 路径记录器标识符

手册用法

pathrecid的作用是识别路径记录器的断点。

描述

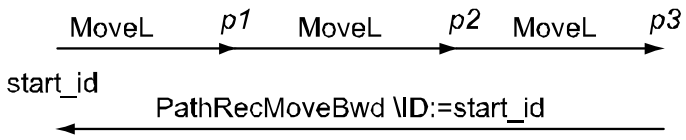
路径记录器是用于记录机械臂执行路径的系统函数。通过指令PathRecStart，可将pathrecid类数据与特定路径位置关联在一起。随后，用户可通过使用指令PathRecMoveBwd，命令记录器运动回路径标识符。

基本示例

以下示例介绍了数据类型pathrecid：

例 1

```
VAR pathrecid start_id;
CONST robtarget p1 := [...];
CONST robtarget p2 := [...];
CONST robtarget p3 := [...];

PathRecStart start_id;
MoveL p1, vmax, z50, tool1;
MoveL p2, vmax, z50, tool1
MoveL p3, vmax, z50, tool1;
IF(PathRecValidBwd (\ID := start_id)) THEN
  StorePath;
  PathRecMoveBwd \ID:=start_id;
  ...
ENDIF
```



pathrecid_Ex

前一例子将启动路径记录器，且将为起点标记路径标识符start_id。因此，使用传统的移动指令，机械臂将向前移动，随后通过记录的路径而返回至开始位置。为了能够运行PathRecorder移动指令，必须用StorePath来改变路径等级。

特征

pathrecid为非值数据类型。

相关信息

信息，关于	请参阅
起动-停止路径记录器	第*440*页的*PathRecStart - 起动路径记录器* 第*442*页的*PathRecStop - 停止路径记录器*

下一页继续

信息，关于	请参阅
检查有效记录的路径	第*1172*页的*PathRecValidBwd* - 是否记录了有效的后退路径 第*1175*页的*PathRecValidFwd* - 是否记录了有效的前进路径
使路径记录器向后	第*431*页的*PathRecMoveBwd* - 将路径记录器向后移动
使路径记录器向前	第*437*页的*PathRecMoveFwd* - 向前移动路径记录器
非值数据类型的特征	技术参考手册 - *RAPID*语言概览，基本特征 - 数据类型一节

3.56 pos - 位置（仅X、Y和Z）

手册用法

将pos用于各位置（仅X、Y和Z）。

将robtarget数据类型用于机械臂位置，包括工具方位和各个轴的配置。

描述

pos类数据描述X、Y和Z位置的坐标。

组件

数据类型pos拥有以下组成部分：

x

数据类型：num

位置的X值。

y

数据类型：num

位置的Y值。

z

数据类型：num

位置的Z值。

基本示例

以下示例介绍了数据类型pos：

例 1

```
VAR pos pos1;
...
pos1 := [500, 0, 940];
```

为pos1位置分配下值：X=500 mm，Y=0 mm，Z=940 mm。

例 2

```
pos1.x := pos1.x + 50;
```

pos1位置沿X方向移动50 mm。

结构

```
< dataobject of pos >
  < x of num >
  < y of num >
  < z of num >
```

相关信息

信息，关于	请参阅
有关各位置的运算	技术参考手册 - *RAPID*语言概览，基本特征 - 表达式一节
机械臂位置，包括姿态	第*1467*页的*robtarget* - 位置数据

3.57 pose - 坐标变换

手册用法

pose用于从一个坐标系改变为另一个坐标系。

描述

pose类数据描述如何替代一个坐标系，并使其围绕另一个坐标系旋转。数据能够描述如何相对于腕坐标系来定位工具坐标系。

组件

此数据类型有以下组件：

trans

translation

数据类型：pos

坐标系位置（x、y和z）的位移。

rot

rotation

数据类型：orient

坐标系的旋转。

基本示例

以下示例介绍了数据类型pose：

```
VAR pose frame1;
...
frame1.trans := [50, 0, 40];
frame1.rot := [1, 0, 0, 0];
```

向frame1坐标变换分配一个值，其相当于位置位移，其中，X=50 mm，Y=0 mm，Z=40 mm；但是并不存在旋转。

结构

```
< dataobject of pose >
  < trans of pos >
  < rot of orient >
```

相关信息

信息，关于	请参阅
什么是四元数？	第1439页的orient-姿态

3.58 processtimes - 进程时间

手册用法

`processtimes`将用于定义CAP所有状态监控阶段的时长，但主阶段除外，主阶段由机器人运动来定义（请参见应用手册 - *Continuous Application Platform*中的监控章节）。

`processtimes`是`capdata`的一个分量，定义了CAP中下列状态监控阶段的超时时间：

- PRE_START
- POST1
- POST2

如果在CAP的相应状态监控阶段中执行监控，那么，指定超时时间必须大于零（请参见应用手册 - *Continuous Application Platform*中的监控和进程阶段章节）。

组件

pre

数据类型：`num`

定义PRE_START阶段的时长，单位为秒。在该时间中，必须满足该阶段的所有定义条件。

post1

数据类型：`num`

定义POST1阶段的时长，单位为秒。在该时间中，必须满足该阶段的所有定义条件。

post2

数据类型：`num`

定义POST2阶段的时长，单位为秒。在该时间中，必须满足该阶段的所有定义条件。

语法

```
< data object of processtimes >
  < pre of num >
  < post1 of num >
  < post2 of num >
```

相关信息

	参见：
`capdata`数据类型	第1355页的*capdata - CAP数据*
Continuous Application Platform	应用手册 - *Continuous Application Platform*

3.59 progdisp - 程序位移

手册用法

progdisp用于储存机械臂和外轴的当前程序位移。

通常无须使用此数据类型，因为通过使用指令PDispSet、PDispOn、PDispOff、EOffsSet、EOffsOn和EOffsOff来设置数据。其仅用于临时储存当前值，以供随后使用。

描述

通过使用系统变量C_PROGDISP，可访问程序位移的当前值。

有关更多的信息，请参见指令PDispSet、PDispOn、EOffsSet和EOffsOn。

组件

pdisp

program displacement

数据类型：pose

通过使用平移和姿态来表示机械臂的程序位移。平移以mm表示。

eoffs

external offset

数据类型：extjoint

各外轴的偏移量。如果轴呈线性，则值以mm表示；如果进行旋转，则值以度表示。

基本示例

以下示例介绍了数据类型progdisp：

例 1

```
VAR progdisp progdisp1;
...
SearchL sen1, psearch, p10, v100, tool1;
PDispOn \ExeP:=psearch, *, tool1;
EOffsOn \ExeP:=psearch, *;
...
progdisp1:=C_PROGDISP;
PDispOff;
EOffsOff;
...
PDispSet progdisp1.pdisp;
EOffsSet progdisp1.eoffs;
```

首先，从搜索位置激活程序位移。然后，将当前程序位移值临时储存在变量progdisp1中，并停用程序位移。随后，通过使用指令PDispSet和EOffsSet，进行重新激活。

预定义数据

系统变量C_PROGDISP描述机械臂和外轴的当前程序位移，且始终可从程序进行访问。另一方面，仅可通过使用一系列指令而非分配来进行改变。

下一页继续

3.59 progdisp - 程序位移

RobotWare - OS

续前页

设置有关程序位移的以下默认值

- 当使用重启模式重置RAPID时
- 当加载一则新程序或一个新模块时
- 当从起点开始执行程序时
- 当将程序指针移动到main时
- 当将程序指针移动到子程序时
- 移动程序指针造成执行顺序丢失时

```
VAR progdisp C_PROGDISP :=
  [ [[ 0, 0, 0], [1, 0, 0, 0]],-> posedata
  [ 0, 0, 0, 0, 0, 0]];-> extjointdata
```

结构

```
< dataobject of progdisp >
  < pdisp of pose >
    < trans of pos >
      < x of num >
      < y of num >
      < z of num >
    < rot of orient >
      < q1 of num >
      < q2 of num >
      < q3 of num >
      < q4 of num >
  < eoffs of extjoint >
    < eax_a of num >
    < eax_b of num >
    < eax_c of num >
    < eax_d of num >
    < eax_e of num >
    < eax_f of num >
```

相关信息

信息，关于	请参阅
用于定义程序位移的指令	技术参考手册 - *RAPID*语言概览，*RAPID*概要 - 运动设置一节
坐标系	技术参考手册 - *RAPID*语言概览，运动和*I/O*原则 - 坐标系一节

3.60 原始数据字节 - 原始数据

手册用法

将rawbytes用作一个通用数据容器。其可以用于同I/O设备进行通信。

描述

通过支持指令/函数，可用任意类型数据-num、byte、string来填充rawbytes数据。在rawbytes的任意变量中，系统同时储存当前有效字节的长度。

基本示例

以下示例介绍了数据类型rawbytes：

例 1

```
VAR rawbytes raw_data;
VAR num integer := 8;
VAR num float := 13.4;

ClearRawBytes raw_data;
PackRawBytes integer, raw_data, 1 \IntX := INT;
PackRawBytes float, raw_data, (RawBytesLen(raw_data)+1) \Float4;
```

在此例子中，首先清除rawbytes类变量raw_data，即将所有字节设置为0（与声明的默认值相同）。随后，在前2个字节放置integer值，在随后4个字节放置float值。

限制

rawbytes变量可能包含0到1024个字节。

结构

rawbytes是一种非数值的数据类型。

声明rawbytes变量时，将rawbytes中的所有字节设置为0，并将变量中的当前有效字节长度设置为0。

相关信息

信息，关于	请参阅
获取rawbytes数据的长度	第*1193*页的*RawBytesLen* - 获取原始数据字节数据的长度
清除rawbytes数据的内容	第*110*页的*ClearRawBytes* - 清除原始数据字节数据的内容
复制rawbytes数据的内容	第*128*页的*CopyRawBytes* - 复制原始数据字节数据的内容
将DeviceNet标题装入rawbytes数据	第*421*页的*PackDNHeader* - 将*DeviceNet*标题装入原始数据字节数据中。
将数据装入rawbytes数据	第*424*页的*PackRawBytes* - 将数据装入原始数据字节数据
写入rawbytes数据	第*923*页的*WriteRawBytes* - 写入原始数据字节数据

下一页继续

3.60 原始数据字节 - 原始数据

RobotWare - OS

续前页

信息，关于	请参阅
读取rawbytes数据	第*499*页的*ReadRawBytes* - 读取原始数据字节数据
打开来自rawbytes数据的数据	第*845*页的*UnpackRawBytes* - 打开来自原始数据字节数据的数据
文件和串行通道处理	应用手册 - 控制器软件*IRC5*

3.61 restartblkdata - 重启用块数据

手册用法

restartblkdata用于定义重启时CAP进程的行为。

restartblkdata是capdata的一个分量，定义了在下列情况下重启时CAP进程的下列条件：

- 在工艺开始或重新启动时（weave_start），机器人执行/禁止定点摆动。
- 相对于进程重启，即可延迟，也可不延迟来重启机器人运动（motion_delay）。
- 应执行/禁止PRE阶段和PRE_START阶段（pre_phase）。
- 进程开始期间，是否应采用不同于主速度的速度（startspeed_phase）。
- 应执行/禁止START_POST1阶段和POST1阶段（post1_phase）。
- 应执行/禁止START_POST2阶段和POST2阶段（post2_phase）。

组件

weave_start

数据类型：bool

值	描述
FALSE	进程开始前，在重启时进行定点摆动
TRUE	进程开始前，在重启时不进行定点摆动

motion_delay

数据类型：bool

值	描述
FALSE	进程开始后，在重启时机器人运动延迟
TRUE	进程开始后，在重启时机器人运动未延迟

pre_phase

数据类型：bool

值	描述
FALSE	在重启时执行PRE阶段和PRE_START阶段
TRUE	在重启时不执行PRE阶段和PRE_START阶段

startspeed_phase

数据类型：bool

值	描述
FALSE	在重启开始时，以启动速度移动机器人
TRUE	在重启开始时，未以启动速度移动机器人，直接采用主速度

下一页继续

3.61 restartblkdata - 重启用块数据

Continuous Application Platform (CAP)

续前页

post1_phase

数据类型：bool

值	描述
FALSE	在重启时执行START_POST1阶段和POST1阶段
TRUE	在重启时不执行START_POST1阶段和POST1阶段

post2_phase

数据类型：bool

值	描述
FALSE	在重启时执行START_POST2阶段和POST2阶段
TRUE	在重启时不执行START_POST2阶段和POST2阶段

语法

```
< data object of restartblkdata >
  < weave_start of bool >
  < motion_delay of bool >
  < pre_phase of bool >
  < startspeed_phase of bool >
  < post1_phase of bool >
  < post2_phase of bool >
```

相关信息

	参见：
capdata数据类型	第1355页的capdata - CAP数据
Continuous Application Platform	应用手册 - *Continuous Application Platform*

3.62 restartdata - 重启关于触发信号的数据

手册用法

`restartdata`反映机械臂运动停止序列中指定I/O信号（过程信号）的预置值和后置值。在指令`TriggStopProc`中规定用于监督的I/O信号。

`TriggStopProc`和`restartdata`旨在用于在RAPID（NOSTEPIN程序）所定义的自身过程指令的程序停止（STOP）或紧急停止（QSTOP）后重启。

定义

本表显示了有关读取I/O信号预置值和后置值的时间点定义。

停止类型	读取有关I/O信号预置值的时间	读取有关I/O信号后置值的时间
在路径上停止	当所有机械臂轴静止不动时	大约在预定时间后400 ms
QSTOP远离路径	尽可能快	大约在预定时间后400 ms

描述

在停止程序执行后，`restartdata` 反映以下数据：

- 有效的重启数据
- 机械臂在路径上停止或未停止
- I/O信号的预置值
- I/O信号的后置值
- 进行过程阴影信号预置时间与后置时间之间的侧面数

组件

`restartstop`

valid restartdata after stop

数据类型：`bool`

TRUE=反映最后STOP或QSTOP

FALSE=无效的重启数据。将所有I/O信号值设置为-1。

`stoponpath`

stop on path

数据类型：`bool`

TRUE=机械臂在路径上停止（STOP）

FALSE=机械臂停止，但是未在路径上停止（QSTOP）

`predo1val`

pre do1 value

数据类型：`dionum`

指令`TriggStopProc`中的参数`DO1`所规定的数字信号“do1”的预置值。

`postdo1val`

post do1 value

数据类型：`dionum`

下一页继续

3.62 restartdata - 重启关于触发信号的数据
RobotWare - OS
续前页

指令TriggStopProc中的参数DO1所规定的数字信号“do1”的后置值。

prego1val

pre go1 value

数据类型：num

指令TriggStopProc中的参数GO1所规定的数字组信号“go1”的预置值。

postgo1val

post go1 value

数据类型：num

指令TriggStopProc中的参数GO1所规定的数字组信号“go1”的后置值。

prego2val

pre go2 value

数据类型：num

指令TriggStopProc中的参数GO2所规定的数字组信号“go2”的预置值。

postgo2val

post go2 value

数据类型：num

指令TriggStopProc中的参数GO2所规定的数字组信号“go2”的后置值。

prego3val

pre go3 value

数据类型：num

指令TriggStopProc中的参数GO3所规定的数字组信号“go3”的预置值。

postgo3val

post go3 value

数据类型：num

指令TriggStopProc中的参数GO3所规定的数字组信号“go3”的后置值。

prego4val

pre go4 value

数据类型：num

指令TriggStopProc中的参数GO4所规定的数字组信号“go4”的预置值。

postgo4val

post go4 value

数据类型：num

指令TriggStopProc中的参数GO4所规定的数字组信号“go4”的后置值。

preshadowval

pre shadow value

数据类型：dionum

指令TriggStopProc中的参数ShadowDO 所规定的数字信号“shadow”的预置值。

下一页继续

`shadowflanks`

number of shadow flanks

数据类型：`num`

预置时间与后置时间之间的数字信号“shadow”的值转换（侧面）数。在指令`TriggStopProc`.的参数`ShadowDO` 中规定信号“shadow”

`postshadowval`

post shadow value

数据类型：`dionum`

指令`TriggStopProc`中的参数`ShadowDO` 所规定的数字信号“shadow”的后置值。

结构

```
< dataobject of restartdata >
  < restartstop of bool >
  < stoponpath of bool >
  < predo1val of dionum >
  < postdo1val of dionum >
  < prego1val of num >
  < postgo1val of num >
  < prego2val of num >
  < postgo2val of num >
  < prego3val of num >
  < postgo3val of num >
  < prego4val of num >
  < postgo4val of num >
  < preshadowval of dionum >
  < shadowflanks of dionum >
  < postshadowval of dionum >
```

相关信息

信息，关于	请参阅
预定义过程指令	第*785*页的*TriggL - 关于事件的机械臂线性运动* 第*747*页的*TriggC - 关于事件的机械臂圆周移动*
安装重启数据的镜像	第*817*页的*TriggStopProc - 产生关于停止时触发信号的重启数据*
沿路径向后移动	第*674*页的*StepBwdPath - 在路径上向后移动一步*
Advanced RAPID	产品规格 - 控制器软件*IRC5*

3.63 rmqheader - RAPID消息序列消息标题
FlexPendant Interface, PC Interface, or Multitasking

3.63 rmqheader - RAPID消息序列消息标题

手册用法

rmqheader（*RAPID Message Queue Header*）用于读取rmqmessage类消息中数据的数据结构。

描述

将非值数据类型rmqmessage的标题部分转换为值数据类型rmqheader。

组件

datatype

数据类型：string

所用数据类型的名称，例如，num、string或其他一些值数据类型。

ndim

Number of Dimensions

数据类型：num

数组维度数量。

dim1

Size of first dimension

数据类型：num

第一个维度的尺寸。未使用时为0。

dim2

Size of second dimension

数据类型：num

第二个维度的尺寸。未使用时为0。

dim3

Size of third dimension

数据类型：num

第三个维度的尺寸。未使用时为0。

示例

有关于数据类型rmqheader的基本例子阐述如下。

例 1

```
VAR rmqmessage message;
VAR rmqheader header;
...
RMQGetMessage message;
RMQGetMsgHeader message \Header:=header;
```

复制和转换来自rmqmessage消息的rmqheader信息。

下一页继续

结构

```
<dataobject of rmqheader>
  <datatype of string>
  <ndim of num>
  <dim1 of num>
  <dim2 of num>
  <dim3 of num>
```

相关信息

信息，关于	请参阅
RAPID消息队列功能描述	应用手册 - 控制器软件*IRC5*，*RAPID*消息序列一节。
从rmqmessage提取标题数据	第*528*页的*RMQGetMsgHeader - 从RMQ*消息获取标题信息
RMQ Message	第*1464*页的*rmqmessage - RAPID*消息序列消息

3.64 rmqmessage - RAPID消息序列消息

手册用法

rmqmessage（*RAPID Message Queue Message*）用于临时储存通信数据。

描述

数据类型rmqmessage为用于储存数据的消息，并同时在不同RAPID任务或机械臂应用开发器客户端之间就RMQ功能进行沟通。其包含有关所发送数据类型、数据维度、发送器识别号和实际数据的信息。

rmqmessage是大型数据类型（大约3000个字节大），并建议其重新使用变量，以保存RAPID内存。

基本示例

以下示例介绍了数据类型rmqmessage：

例 1

```
VAR rmqmessage rmqmessage1;
VAR string myrecdata;
...
RMQGetMsgData rmqmessage1, myrecdata;
```

定义变量rmqmessage1，并可以在RMQ（RAPID消息队列）命令中使用。在本例子中，将rmqmessage1内的数据部分复制到变量myrecdata。

特征

rmqmessage为非值数据类型，且无法用于以值为导向的运算。

相关信息

信息，关于	请参阅
RAPID消息队列功能描述	应用手册 - 控制器软件*IRC5*，*RAPID*消息序列一节。
RMQ Header	第*1462*页的*rmqheader - RAPID*消息序列消息标题
从rmqmessage提取标题数据	第*528*页的*RMQGetMsgHeader - 从RMQ*消息获取标题信息
命令和启用特定数据类型的中断	第*274*页的*IRMQMessage - 下达数据类型的RMQ*中断指令
从RAPID消息队列中获取第一个消息。	第*522*页的*RMQGetMessage - 获取RMQ*消息
将数据发送至RAPID任务或机械臂应用开发器客户端的队列，并等待客户端的回答。	第*537*页的*RMQSendWait - 发送RMQ*数据消息，并等待响应
从rmqmessage提取数据	第*525*页的*RMQGetMsgData - 从RMQ*消息获取数据部分

3.65 rmqslot - RMQ客户端的识别号

手册用法

当同RMQ或机械臂应用开发器客户端进行通信时，使用rmqslot（*RAPID Message Queue Slot*）。

描述

rmqslot为针对RAPID任务而配置的RAPID消息队列识别号或机械臂应用开发器客户端识别号。

基本示例

以下示例介绍了数据类型rmqslot：

例 1

```
VAR rmqslot rmqslot1;
RMQFindSlot rmqslot1, "RMQ_T_ROB1";
...
```

定义变量rmqslot1，并可将其用于指令RMQFindSlot，以获得针对RAPID任务“T_ROB1””而配置的RAPID消息队列“RMQ_T_ROB1”识别号。

特征

rmqslot为非值数据类型，且无法用于以值为导向的运算。

相关信息

信息，关于	请参阅
RAPID消息队列功能描述	应用手册 - 控制器软件*IRC5*，*RAPID*消息序列一节。
寻找RAPID消息队列任务或机械臂应用开发器客户端的识别号。	第*520*页的*RMQFindSlot* - 从槽名中寻找槽识别号
将数据发送至RAPID任务或机械臂应用开发器客户端的队列。	第*534*页的*RMQSendMessage* - 发送*RMQ*数据消息
将数据发送至客户端，并等待客户端的回答。	第*537*页的*RMQSendWait* - 发送*RMQ*数据消息，并等待响应
从指定槽识别号中获取槽名	第*1215*页的*RMQGetSlotName* - 获取*RMQ*客户端的名称

3.66 robjoint - 机械臂轴的接头位置

手册用法

robjoint用于定义机械臂轴的位置，以度计。

描述

robjoint类数据用于储存机械臂轴1到6的轴位置，以度计。将轴位置定义为各轴（臂）从轴校准位置沿正方向或负方向旋转的度数。

组件

rax_1

robot axis 1

数据类型：num

机械臂轴1位置距离校准位置的度数。

...

rax_6

robot axis 6

数据类型：num

机械臂轴6位置距离校准位置的度数。

结构

```
< dataobject of robjoint >
  < rax_1 of num >
  < rax_2 of num >
  < rax_3 of num >
  < rax_4 of num >
  < rax_5 of num >
  < rax_6 of num >
```

相关信息

信息，关于	请参阅
接头位置数据	第*1418*页的*jointtarget - 接头位置数据*
移动至接头位置	第*335*页的*MoveAbsJ - 移动机械臂至绝对接头位置*

3.67 robtarget - 位置数据

手册用法

`robtarget`（*robot target*）用于定义机械臂和附加轴的位置。

描述

位置数据用于定义移动机械臂和附加轴的移动指令中的位置。

当机械臂能够以多种不同的方式达到相同位置时，同时规定轴配置。如果存在任何不明确之处，则定义轴值，例如：

- 如果机械臂位于前进或后退位置，
- 如果轴4各点向下或向上，
- 如果轴6具有负向或正向旋转。

警告

根据工件的坐标系，包括所有程序位移，定义位置。如果通过指令所用工件以外的其他一些工件来编程位置，则机械臂将不会以预期的方式移动。确保使用移动指令编程时用到的相同工件。使用不当会造成人员伤害，或损坏机械臂或其他设备。

组件

`trans`

translation

数据类型：`pos`

用mm来表示工具中心点的位置（x、y和z）。

规定相对于当前目标坐标系的位置，包括程序位移。如果未规定任何工件，则为世界坐标系。

`rot`

rotation

数据类型：`orient`

工具方位以四元数的形式表示（q1、q2、q3和q4）。

规定相对于当前目标坐标系的方位，包括程序位移。如果未规定任何工件，则为世界坐标系。

`robconf`

robot configuration

数据类型：`confdata`

机械臂的轴配置（`cf1`、`cf4`、`cf6`和`cfx`）。以轴1、轴4和轴6当前四分之一旋转的形式进行定义。将第一个正四分之一旋转0到90°定义为0。组件`cfx`的含义取决于机械臂类型。

有关更多的信息，请参见数据类型`confdata`。

`extax`

external axes

数据类型：`extjoint`

下一页继续

3.67 robtarget - 位置数据
RobotWare - OS
续前页

附加轴的位置。

各单个轴（eax_a、eax_b...eax_f）的位置 is defin如下：

- 对于旋转轴，其位置定义为从校准位置起旋转的度数。
- 对于线性轴，其位置定义为与校准位置的距离（以mm计）。

附加轴eax_a ... 为逻辑轴。在系统参数中定义逻辑轴编号与物理轴编号的相互关系。

定义未连接轴的值9E9。如果位置数据中定义的轴不同于程序执行时实际连接的轴，则以下内容适用：

- 如果未在位置数据（值9E9）中定义位置，则当连接且未激活该轴时，将忽略该值。但是如果激活该轴，则将会产生错误。
- 如果在位置数据中定义该位置，则忽略该值，即便未连接该轴。

如果未激活轴，则不会进行任何运动，且不会产生有关轴（含正确位置数据）的错误。

如果一个附加轴以独立模式运行，且应由机械臂及其附加轴进行新的运动，则采用独立模式的附加轴的位置数据不得为9E9。数据必须为系统未使用的任意值。

基本示例

以下示例介绍了数据类型robtarget：

例 1

```
CONST robtarget p15 := [ [600, 500, 225.3], [1, 0, 0, 0], [1, 1,
     0, 0], [ 11, 12.3, 9E9, 9E9, 9E9, 9E9] ];
```

位置p15定义如下：

- 机械臂的位置：在目标坐标系中，x=600、y=500和z=225.3 mm。
- 与目标坐标系方向相同的工具方位。
- 机械臂的轴配置：轴1和轴4位于90-180°，轴6位于0-90°。
- 附加逻辑轴a和b的位置以度或毫米表示（根据轴的类型）。未定义轴c到轴f。

例 2

```
VAR robtarget p20;
...
p20 := CRobT(\Tool:=tool\wobj:=wobjØ);
p20 := Offs(p20,10,0,0);
```

通过调用函数CRobT，将位置p20 设置为同机械臂当前位置相同的位置。随后，将位置沿x方向移动10 mm。

结构

```
< dataobject of robtarget >
    < trans of pos >
      < x of num >
      < y of num >
      < z of num >
    < rot of orient >
      < q1 of num >
      < q2 of num >
      < q3 of num >
      < q4 of num >
```

下一页继续

```
< robconf of confdata >
  < cf1 of num >
  < cf4 of num >
  < cf6 of num >
  < cfx of num >
< extax of extjoint >
  < eax_a of num >
  < eax_b of num >
  < eax_c of num >
  < eax_d of num >
  < eax_e of num >
  < eax_f of num >
```

相关信息

信息，关于	请参阅
移动指令	技术参考手册 - *RAPID*语言概览，*RAPID*概要 - 运动一节
坐标系	技术参考手册 - *RAPID*语言概览，运动和*I/O*原则 - 坐标系一节
处理配置数据	技术参考手册 - *RAPID*语言概览，运动和*I/O*原则-机械臂配置一节
附加轴的配置	应用手册 - *Additional axes and stand alone controller*
什么是四元数？	第*1439*页的*orient*-姿态

3.68 sensor - 外部设备描述符

手册用法

sensor是用于连接外部设备的描述符。

描述

将有关RAPID等级设备的描述符压缩在记录数据类型sensor中。其保存有关传感器设备的信息，例如，id、错误事件号和传感器通信状态。

组件

id

数据类型：num

设备的内部标识符，其将通过RAPID等级设备的第一次运算来进行设置。（尚未执行）。

error

数据类型：num

将参数state设置为STATE_ERROR时，设置error参数。当state从STATE_ERROR到STATE_CONNECTED时，将参数error设置为0。

错误编号	Error
0	无错误。
112600	通信界面初始化失败。
112602	通信界面错误。

state

数据类型：sensorstate

反映设备的实际通信状态。

示例

数据类型sensor的例子如下所示。

例 1

```
PERS sensor AnyDevice;
PERS robdata DataOut := [[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]];
PERS sensdata DataIn :=
     ["No",[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]];
VAR num SampleRate:=64;
...
! Setup Interface Procedure
PROC RRI_Open()
  SiConnect AnyDevice;
  ! Send and receive data cyclic with 64 ms rate
  SiGetCyclic AnyDevice, DataIn, SampleRate;
  SiSetCyclic AnyDevice, DataOut, SampleRate;
ENDPROC
```

下一页继续

调用程序RRI_Open时，首先打设备AnyDevice的连接。随后，开始以速率SampleRate开始进行循环传输。

结构

```
<dataobject of sensor>
  <id of num>
  <error of num>
  <state of sensorstate>
```

相关信息

信息，关于	请参阅
建立与外部系统的连接。	第*602*页的*SiConnect - 传感器接口连接*。
关闭与外部系统的连接。	第*605*页的*SiClose - 传感器接口关闭*。
关于循环传输的登记数据。	第*610*页的*SiSetCyclic - 传感器接口设置循环*。
赞成循环数据传输。	第*606*页的*SiGetCyclic - 传感器接口获得循环*
设备的通信状况。	第*1472*页的*sensorstate - 设备的通信状况*。
Robot Reference Interface	应用手册 - 控制器软件*IRC5*

3.69 sensorstate - 设备的通信状况

手册用法

sensorstate用于代表设备的实际通信状况。

描述

sensorstate常量用于反映设备的实际通信状况。其可以从RAPID进行使用，以评估传感器连接的状况。

预定义数据

预定义数据类型sensorstate的以下符号常量，并可将其用于评估设备所处的通信状态。

常量	值
STATE_ERROR	-1
STATE_UNDEFINED	0
STATE_CONNECTED	1
STATE_OPERATING	2
STATE_CLOSED	3

特征

sensorstate为有关num的一种别名数据类型，并因此继承其特征。

相关信息

信息，关于	请参阅
建立与外部系统的连接。	第*602*页的*SiConnect - 传感器接口连接*。
关闭与外部系统的连接。	第*605*页的*SiClose - 传感器接口关闭*。
关于循环传输的登记数据。	第*610*页的*SiSetCyclic - 传感器接口设置循环*。
赞成循环数据传输。	第*606*页的*SiGetCyclic - 传感器接口获得循环*
外部设备的描述符。	第*1470*页的*sensor - 外部设备描述符*。
Robot Reference Interface	应用手册 - 控制器软件*IRC5*

3.70 shapedata - 全局区域形状数据

手册用法

shapedata的作用是描述一个全局区域的几何形状。

描述

可将全局区域定义为4种不同的几何形状：

- 一个方盒，所有侧面都与世界坐标系平行，并通过WZBoxDef指令进行定义
- 一个球体，通过WZSphDef指令进行定义
- 一个圆柱体，与全局坐标系的z轴平行，并通过WZCylDef指令进行定义
- 机械臂和/或外轴的接头空白区，通过指令WZHomeJointDef或WZLimJointDef进行定义

通过先前指令之一来定义全局区域的几何形状，并通过指令WZLimSup或WZDOSet来定义全局区域的行动。

基本示例

以下示例介绍了数据类型shapedata：

例 1

```
VAR wzstationary pole;
VAR wzstationary conveyor;
...
PROC ...
  VAR shapedata volume;
  ...
  WZBoxDef \Inside, volume, p_corner1, p_corner2;
  WZLimSup \Stat, conveyor, volume;
  WZCylDef \Inside, volume, p_center, 200, 2500;
  WZLimSup \Stat, pole, volume;
ENDPROC
```

将conveyor定义为一个盒子，并激活对该区域的监督。将pole 定义为一个圆柱体，并同时激活对该区域的监督。如果机械臂达到此类区域之一，则停止运动。

特征

shapedata是一种非数值的数据类型。

相关信息

信息，关于	请参阅
全局区域	技术参考手册 - *RAPID*语言概览，*RAPID*概要 - 运动设置一节
定义箱形全局区域	第*929*页的*WZBoxDef - 定义一个箱形全局区域*
定义球形全局区域	第*952*页的*WZSphDef - 定义球形全局区域*
定义圆柱形全局区域	第*931*页的*WZCylDef - 定义圆柱形全局区域*
定义内部接头的全局区域	第*943*页的*WZHomeJointDef - 定义内部接头的全局区域*

下一页继续

3.70 shapedata - 全局区域形状数据

World Zones

续前页

信息，关于	请参阅
定义限制接头的全局区域	第*946*页的*WZLimJointDef* - 定义有关接头内限制的全局区域
启用全局区域限制监控	第*949*页的*WZLimSup* - 启用全局区域限制监控
启用全局区域数字信号输出设置	第*935*页的*WZDOSet* - 启用全局区域，设置数字信号输出

3.71 signalorigin - 介绍I/O信号来源

手册用法

signalorigin用于代表一个含符号常量的整数。

描述

signalorigin类的预定义符号常量可用于检查I/O信号的来源。其旨在于检查函数GetSignalOrigin的返回值时使用。

基本示例

以下示例介绍了数据类型signalorigin：

例 1

```
VAR signalorigin sigorig;
VAR string signalname;
...
sigorig := GetSignalOrigin(mydo, signalname);
IF sigorig = SIGORIG_NONE THEN
  TPWrite "The signal named "+ArgName(mydo)+" can not be used";
  Stop;
ELSEIF (sigorig = SIGORIG_CFG) OR (sigorig = SIGORIG_ALIAS) THEN
  SetDO mydo, 1;
  ...
ELSE
  TPWrite "Unknown origin "+ValToStr(sigorig);
  Stop;
ENDIF
```

将信号来源储存在变量sigorig中。

预定义数据

预定义以下signalorigin型常量：

返回值	符号常量	备注
0	SIGORIG_NONE	在RAPID中声明了I/O信号变量，且没有别名耦合。
1	SIGORIG_CFG	在I/O配置中配置信号。
2	SIGORIG_ALIAS	在RAPID中声明了I/O信号变量，且拥有一个同I/O配置中所配置的I/O信号耦合的别名。

特征

signalorigin为有关num的一种别名数据类型，并因此继承其属性。

相关信息

信息，关于	请参阅
获得有关I/O信号来源的信息	第1100页的*GetSignalOrigin - 获得有关I/O信号来源的信息*

3.72 signalxx - 数字信号和模拟信号

手册用法

将signalxx内的数据类型用于数字和模拟输入和输出信号。

在系统参数中定义信号的名称，因此，不在程序中对其进行定义。

描述

数据类型	用于
signalai	模拟信号输入信号
signalao	模拟信号输出信号
signaldi	数字信号输入信号
signaldo	数字信号输出信号
signalgi	数字信号输入信号组
signalgo	数字信号输出信号组

signalxo类变量仅包含对信号的引用。通过使用指令（例如，DOutput），对值进行设置。

signalxi类变量包含对信号的引用，且可能在用于值范围时直接检索程序中的值。

可直接在程序中读取输入信号的值，例如：

```
! Digital input
IF di1 = 1 THEN ...

! Digital group input
IF gi1 = 5 THEN ...

! Analog input
IF ai1 > 5.2 THEN ...
```

其亦可用于分配中，例如：

```
VAR num current_value;

! Digital input
current_value := di1;

! Digital group input
current_value := gi1;

! Analog input
current_value := ai1;
```

限制

数据类型signalxx的数据不得在程序中定义。但是如果定义了，则会在引用此信号的指令或函数执行时立即显示一个错误消息。另外，此数据类型也可以在声明例行程序时用作参数。

下一页继续

预定义数据

通过使用预定义信号变量（安装的数据），可始终从程序来访问系统参数中定义的信号。但是，应当注意，如果定义了具有相同名称的其他数据，则无法使用此类信号。

特征

signalxx为能够进行数值运算的半值数据类型。

错误处理

系统会生成下列可恢复错误，并在错误处理器中处理这些错误。系统变量ERRNO将被设置成：

如果信号变量是RAPID中声明的变量，则ERR_NO_ALIASIO_DEF。尚未同I/O配置以及指令AliasIO中确定的I/O信号相连。

如果与单元无接触，则ERR_NORUNUNIT。

相关信息

信息，关于	请参阅
输入/输出指令概要	技术参考手册 - *RAPID*语言概览，*RAPID*概要 - 输入和输出信号一节
输入/输出功能性概述	技术参考手册 - *RAPID*语言概览，运动和*I/O*原则 - *I/O*原则一节
I/O配置	技术参考手册 - 系统参数
非值数据类型的特征	技术参考手册 - *RAPID*语言概览，基本特征 - 数据类型一节

3.73 socketdev - 套接字设备

手册用法

socketdev（*socket device*）用于同其他网络计算机通信，或在RAPID任务之间通信。

描述

套接字设备是另一台网络计算机通信链路的处理器。

基本示例

以下示例介绍了数据类型socketdev：

例 1

```
VAR socketdev socket1;
```

定义变量socket1，并可将其用于套接字命令中，例如，SocketCreate。

限制

可声明任意数量的套接字，但是仅可能同时使用32个套接字。

特征

socketdev是一种非数值的数据类型。

相关信息

信息，关于	请参阅
一般的套接字通信	应用手册 - 控制器软件*IRC5*
创建新套接字	第*623*页的*SocketCreate - 创建新套接字*
非值数据类型的特征	技术参考手册 - *RAPID*语言概览，基本特征 - 数据类型一节

3.74 socketstatus - 套接字通信状态

手册用法

socketstatus用于代表套接字通信的状态。

描述

通过函数SocketGetStatus来获得套接字状态，并可将其用于程序流程控制或调试目的。

基本示例

以下示例介绍了数据类型socketstatus：

例 1

```
VAR socketdev socket1;
VAR socketstatus state;
...
SocketCreate socket1;
state := SocketGetStatus( socket1 );
```

将套接字状态SOCKET_CREATED储存在变量状态中。

预定义数据

预定义以下socketstatus型常量：

RAPID常量	值	套接字为...
SOCKET_CREATED	1	创建
SOCKET_CONNECTED	2	同远程主机相连的客户端
SOCKET_BOUND	3	同局部地址和端口绑定的服务器
SOCKET_LISTENING	4	用于接入连接的服务器监听
SOCKET_CLOSED	5	关闭

特征

socketstatus 为有关num的一种别名数据类型，并因此继承其特征。

相关信息

信息，关于	请参阅
一般的套接字通信	应用手册 - 控制器软件*IRC5*
获取套接字状态	第*1230*页的*SocketGetStatus* - 获得当前套接字的状态
一般数据类型、别名数据类型	技术参考手册 - *RAPID*语言概览，基本特征 - 数据类型一节

3.75 speeddata - 速度数据

手册用法

speeddata用于规定机械臂和外轴均开始移动时的速率。

Description

速度数据定义以下速率：

- 工具中心点移动时的速率，
- 工具的重新定位速度，
- 线性或旋转外轴移动时的速率。

当结合多种不同类型的移动时，其中一个速率常常限制所有运动。将减小其他运动的速率，以便所有运动同时停止执行。

同时通过机械臂性能来限制速率。将会根据机械臂类型和运动路径而有所不同。

组件

v_tcp

velocity tcp

数据类型：num

工具中心点的速率，以mm/s计。

如果使用固定工具或协调外轴，则规定相对于工件的速率。

v_ori

velocity orientation

数据类型：num

TCP的重新定位速率，以度/秒表示。

如果使用固定工具或协调外轴，则规定相对于工件的速率。

v_leax

velocity linear external axes

数据类型：num

线性外轴的速率，以mm/s计。

v_reax

velocity rotational external axes

数据类型：num

旋转外轴的速率，以度/秒计。

基本示例

以下示例介绍了数据类型speeddata：

例 1

```
VAR speeddata vmedium := [ 1000, 30, 200, 15 ];
```

使用以下速率，定义速度数据vmedium：

- 对于TCP，速率为1000 mm/s。

下一页继续

- 对于工具的重新定位，速率为30 度/秒。
- 对于线性外轴，速率为200 mm/s。
- 对于旋转外轴，速率为15 度/秒。

```
vmedium.v_tcp := 900;
```

将TCP的速率改变为900 mm/s。

限制

在非常慢的运动下，各运动应足够短，以产生不到240秒的插补时间。

预定义数据

已经在系统中定义了一系列速度数据。

用于移动机械臂和外轴的预定义速度数据：

名称	TCP速度	方向	线性外轴	旋转外轴
v5	5 mm/s	500°/s	5000 mm/s	1000°/s
v10	10 mm/s	500°/s	5000 mm/s	1000°/s
v20	20 mm/s	500°/s	5000 mm/s	1000°/s
v30	30 mm/s	500°/s	5000 mm/s	1000°/s
v40	40 mm/s	500°/s	5000 mm/s	1000°/s
v50	50 mm/s	500°/s	5000 mm/s	1000°/s
v60	60 mm/s	500°/s	5000 mm/s	1000°/s
v80	80 mm/s	500°/s	5000 mm/s	1000°/s
v100	100 mm/s	500°/s	5000 mm/s	1000°/s
v150	150 mm/s	500°/s	5000 mm/s	1000°/s
v200	200 mm/s	500°/s	5000 mm/s	1000°/s
v300	300 mm/s	500°/s	5000 mm/s	1000°/s
v400	400 mm/s	500°/s	5000 mm/s	1000°/s
v500	500 mm/s	500°/s	5000 mm/s	1000°/s
v600	600 mm/s	500°/s	5000 mm/s	1000°/s
v800	800 mm/s	500°/s	5000 mm/s	1000°/s
v1000	1000 mm/s	500°/s	5000 mm/s	1000°/s
v1500	1500 mm/s	500°/s	5000 mm/s	1000°/s
v2000	2000 mm/s	500°/s	5000 mm/s	1000°/s
v2500	2500 mm/s	500°/s	5000 mm/s	1000°/s
v3000	3000 mm/s	500°/s	5000 mm/s	1000°/s
v4000	4000 mm/s	500°/s	5000 mm/s	1000°/s
v5000	5000 mm/s	500°/s	5000 mm/s	1000°/s
v6000	6000 mm/s	500°/s	5000 mm/s	1000°/s
v7000	7000 mm/s	500°/s	5000 mm/s	1000°/s
vmax	*)	500°/s	5000 mm/s	1000°/s

下一页继续

*)有关使用的机械臂类型和正常实用TCP值的最大TCP速度。RAPID函数MaxRobSpeed返回相同的值。如果在工具坐标系中使用非常大的TCP值，则创建自己的速度数据，且TCP速度大于MaxRobSpeed的返回速度。

通过指令MoveExtJ，预定义有待使用的速度数据，以移动旋转外轴。

名称	TCP速度	方向	线性外轴	旋转外轴
vrot1	0 mm/s	0°/s	0 mm/s	1°/s
vrot2	0 mm/s	0°/s	0 mm/s	2°/s
vrot5	0 mm/s	0°/s	0 mm/s	5°/s
vrot10	0 mm/s	0°/s	0 mm/s	10°/s
vrot20	0 mm/s	0°/s	0 mm/s	20°/s
vrot50	0 mm/s	0°/s	0 mm/s	50°/s
vrot100	0 mm/s	0°/s	0 mm/s	100°/s

通过指令MoveExtJ，预定义有待使用的速度数据，以移动线性外轴。

名称	TCP速度	方向	线性外轴	旋转外轴
vlin10	0 mm/s	0°/s	10 mm/s	0°/s
vlin20	0 mm/s	0°/s	20 mm/s	0°/s
vlin50	0 mm/s	0°/s	50 mm/s	0°/s
vlin100	0 mm/s	0°/s	100 mm/s	0°/s
vlin200	0 mm/s	0°/s	200 mm/s	0°/s
vlin500	0 mm/s	0°/s	500 mm/s	0°/s
vlin1000	0 mm/s	0°/s	1000 mm/s	0°/s

结构

```
< dataobject of speeddata >
  < v_tcp of num >
  < v_ori of num >
  < v_leax of num >
  < v_reax of num >
```

相关信息

信息，关于	请参阅
定位器指令	技术参考手册 - *RAPID*语言概览，*RAPID*概要 - 运动一节
一般运动/速度	技术参考手册 - *RAPID*语言概览，运动和*I/O*原则 - 程序执行期间的定位一节
定义最大速率	第*849*页的*VelSet* - 改变编程速率
有关该机械臂的最大TCP速度	第*1140*页的*MaxRobSpeed* - 最大机械臂速度

3.76 stoppointdata - 停止点数据

手册用法

stoppointdata用于规定如何结束一个位置，即在朝下一个位置移动之前，轴必须如何接近编程位置。

描述

可以以飞越点或停止点的形式来终止一个位置。

飞越点意味着从未达到编程位置。在运动指令中规定一个区域，以定义角路径。在达到有关位置之前，使运动方向形成角路径，而非朝向编程位置。请参见数据类型zonedata。

停止点意味着机械臂和外轴必须在机械臂/外轴继续下一次移动之前达到指定位置。当满足有关点的收敛标准时，认为机械臂已达到停止点。收敛标准为速度和位置。同时可能规定时间标准。对于停止点fine，亦请参见数据类型zonedata。

可通过stoppointdata定义三种类型的停止点。

- 将就位类停止点定义为占预定义停止点fine收敛标准（位置和速度）的百分比。就位类型亦使用最短时间和最长时间。机械臂等待时间介于最短时间和最长时间之间，以满足位置和速度标准。
- 停止时间类停止点始终在停止点等待给定的时间。
- 跟随时间类停止点为一种特殊类型的停止点，其用于通过传送带来协调机械臂运动。

stoppointdata同时确定运动应如何与RAPID执行同步。如果运动同步，则RAPID执行等待“就位”，即使在机械臂就位的时候。如果运动不同步，则在物理机械臂达到编程位置之前，RAPID执行获得差不多半秒的“预取”事件。当程序执行获得“就位”或“预取”事件时，继续下一个指令。当出现“预取”事件时，机械臂仍然有一段较长的距离需要移动。当出现“就位”事件时，机械臂接近编程位置。

对于停止时间和跟随时间类型，下一指令分别自停止时间和跟随时间起开始执行。但是对于就位类型，下一指令在满足收敛标准时开始。

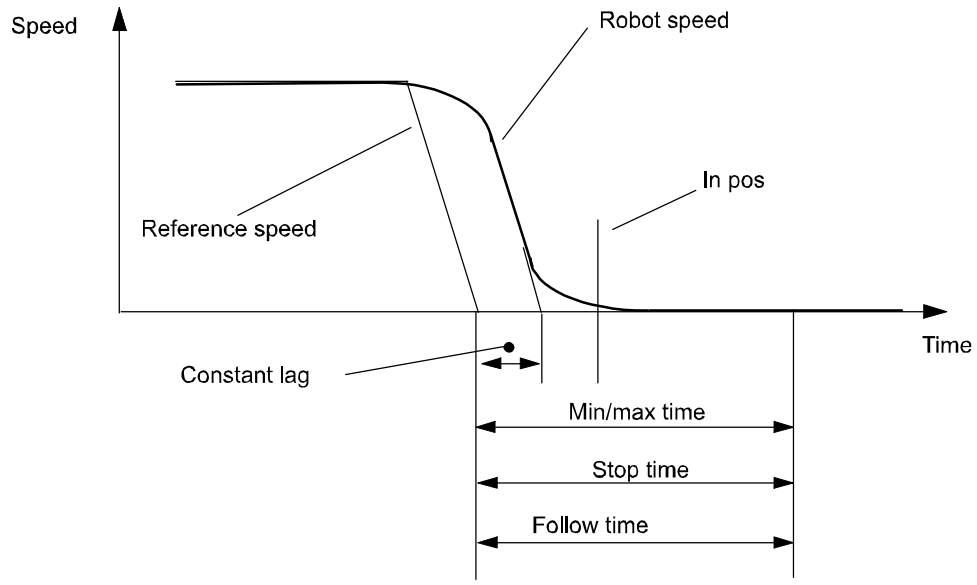
下一页继续

3.76 stoppointdata - 停止点数据

RobotWare - OS

续前页

如果使用移动指令以及参数\Conc，则未完成任何同步，因此，实际移动指令的将立即予以执行。



xx0500002374

在上图中，对停止点的结束进行描述。机械臂速度并不呈线性减少。机械臂伺服常常位于物理机械臂之前。其显示为上图中的常量滞后。常量滞后大约0.1秒。stoppointdata时间元素将参考速度作为触发器。当参考速度为零时，开始时间测量。因此，时间元素中的时间始终包括常量滞后。因此，没道理使用小于常量滞后的值。

组件

type

type of stop point

数据类型：stoppoint

下表定义stoppoint的类型。

1 (inpos)	运动随着一个就位类型的停止点而终止。启用stoppointdata中的inpos元素。未使用指令中的区域数据，使用fine或z0。
2 (stoptime)	运动随着一个停止时间类型的停止点而终止。启用stoppointdata中的stoptime元素。未使用指令中的区域数据，使用fine或z0。
3 (followtime)	运动随着传送带跟随时间类型的精点而终止。当机械臂离开传送带时，使用指令中的区域数据。启用stoppointdata中的followtime元素。

数据类型stoppoint是有关num的一种别名数据类型。其用于选择停止点的类型以及在stoppointdata中使用的数据元素。预定义常量为：

值	符号常量	备注
1	inpos	就位类型编号
2	stoptime	停止时间类型编号
3	fllwtime	跟随时间类型编号

下一页继续

progsynch

program synchronization

数据类型：bool

与RAPID程序执行同步。

- TRUE：运动与RAPID执行同步。直至已达到停止点，程序方才开始执行下一指令。
- FALSE：运动不与RAPID执行同步。在达到停止点之前，程序开始执行下一指令。

如果使用移动指令以及参数\Conc，则在不依赖于progsynch中数据的情况下未完成任何同步，因此，实际移动指令将始终立即就绪。

inpos.position

position condition for TCP

数据类型：num

TCP的位置条件（半径）占正常fine停止点的百分比。

inpos.speed

speed condition for TCP

数据类型：num

TCP的速度条件占正常fine停止点的百分比。

inpos.mintime

minimum wait time

数据类型：num

就位前的最短等待时间，以秒计。用于使机械臂在有关点至少等待规定的时间。最大值为20.0秒。

inpos.maxtime

maximum wait time

数据类型：num

有待满足的收敛标准的最长等待时间，以秒计。用于确保在速度和位置条件设置过紧时，机械臂不会在有关点卡住。最大值为20.0秒。

stoptime

stop time

数据类型：num

时间以秒计，TCP在开始下一次运动之前静止不动。有效范围为0 - 20 s，分辨率为0.001 s。

followtime

follow time

数据类型：num

时间以秒计，TCP随传送带移动。有效范围为0 - 20 s，分辨率为0.001 s。

signal

数据类型：string

下一页继续

3.76 stoppointdata - 停止点数据
RobotWare - OS
续前页

留作未来使用。

relation

数据类型：opnum

留作未来使用。

checkvalue

数据类型：num

留作未来使用。

基本示例

以下示例介绍了数据类型stoppointdata：

Inpos

```
VAR stoppointdata my_inpos := [ inpos, TRUE, [ 25, 40, 0.1, 5], 0,
     0, "", 0, 0];
MoveL *, v1000, fine \Inpos:=my_inpos, grip4;
```

通过以下特征，定义停止点数据my_inpos：

- 停止点为就位类型，inpos。
- 将通过RAPID程序执行来同步停止点，TRUE。
- 停止点距离标准为停止点fine规定距离的25%，25。
- 停止点速度标准为停止点fine规定速度的40%，40。
- 收敛前的最短等待时间为0.1 s，0.1。
- 收敛的最长等待时间为5 s，5。

机械臂朝编程位置移动，直至标准位置或速度之一得以满足。

```
my_inpos.inpos.position := 40;
MoveL *, v1000, fine \Inpos:=my_inpos, grip4;
```

将停止点距离标准调整为40%。

停止时间

```
VAR stoppointdata my_stoptime := [ stoptime, FALSE, [ 0, 0, 0, 0],
     1.45, 0, "", 0, 0];
MoveL *, v1000, fine \Inpos:=my_stoptime, grip4;
```

通过以下特征，定义停止点数据my_stoptime：

- 停止点为停止时间类型，stoptime。
- 将不会通过RAPID程序执行来同步停止点，FALSE。
- 就位等待时间为1.45 s。

机械臂朝编程位置移动，直至出现预取事件。执行下一个RAPID指令。若为移动指令，则在下一个移动开始前，机械臂将停止1.45秒。

```
my_stoptime.stoptime := 6.66;
MoveL *, v1000, fine \Inpos:=my_stoptime, grip4;
```

将停止点停止时间调整为6.66秒。如果下一个RAPID指令为一个移动指令，则机械臂停止6.66 s。

跟随时间

```
VAR stoppointdata my_followtime := [ fllwtime, TRUE, [ 0, 0, 0,
     0], 0, 0.5, "", 0, 0];
MoveL *, v1000, z10 \Inpos:=my_followtime, grip6\wobj:=conveyor1;
```

下一页继续

通过以下特征，定义停止点数据my_followtime：

- 停止点为跟随时间类型，fllwtime。
- 将通过RAPID程序执行来同步停止点，TRUE。
- 停止点跟随时间为0.5 s，0.5。

在留下10 mm的区域之前，z10，机械臂将跟随传送带0.5 s的时间，。

```
my_followtime.followtime := 0.4;
```

将停止点跟随时间调整为0.4 s。

预定义数据

已经在系统中定义了一系列停止点数据。

就位停止点

名称	Progsynch	位置	Speed	最短时间	最长时间	停止时间	跟随时间
inpos20	TRUE	20%	20%	0 s	2 s	-	-
inpos50	TRUE	50%	50%	0 s	2 s	-	-
inpos100	TRUE	100%	100%	0 s	2 s	-	-

（inpos100拥有与停止点fine相同的收敛标准）

停止时间停止点

名称	Progsynch	位置	Speed	最短时间	最长时间	停止时间	跟随时间
stoptime0_5	FALSE	-	-	-	-	0.5 s	-
stoptime1_0	FALSE	-	-	-	-	1.0 s	-
stoptime1_5	FALSE	-	-	-	-	1.5 s	-

跟随时间停止点

名称	Progsynch	位置	Speed	最短时间	最长时间	停止时间	跟随时间
fllwtime0_5	TRUE	-	-	-	-	-	0.5 s
fllwtime1_0	TRUE	-	-	-	-	-	1.0 s
fllwtime1_5	TRUE	-	-	-	-	-	1.5 s

结构

```
< data object of stoppointdata >
  < type of stoppoint >
  < progsynch of bool >
  < inpos of inposdata >
    < position of num >
    < speed of num >
    < mintime of num >
    < maxtime of num >
  < stoptime of num >
  < followtime of num >
  < signal of string >
  < relation of opnum >
```

下一页继续

3.76 stoppointdata - 停止点数据

RobotWare - OS

续前页

```
< checkvalue of num >
```

相关信息

信息，关于	请参阅
定位器指令	技术参考手册 - *RAPID*语言概览，*RAPID*概要 - 运动一节
一般移动/路径	技术参考手册 - *RAPID*语言概览，运动和*I/O*原则 - 程序执行期间的定位一节
停止点或飞越点	第*1531*页的*zonedata* - 区域数据

3.77 string - 字符串

手册用法

string用于字符串。

描述

字符串由一系列附上引号（“”）的字符（最多80个）组成，例如，“这是一个字符串”。

如果字符串中包括引号，则必须保留两个引号，例如，“本字符串包含一个“”字符”。

如果字符串中包括反斜线，则必须保留两个反斜线符号，例如，“本字符串包含一个\\字符”。

基本示例

以下示例介绍了数据类型string：

例 1

```
VAR string text;
...
text := "start welding pipe 1";
TPWrite text;
```

在FlexPendant示教器上写入文本start welding pipe 1。

限制

一个字符串可能拥有0到80个字符；包括额外的引号或反斜线。

一个字符串可能包含ISO 8859-1（Latin-1）规定的任意字符以及控制字符（非ISO 8859-1（Latin-1）字符，且数字代码介于0-255之间）。

预定义数据

可在系统中使用一系列预定义字符串常量，并可同字符串函数一同使用。例如，参见StrMemb。

名称	字符集
STR_DIGIT	<digit> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
STR_UPPER	<upper case letter> ::= A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | À | Á | Â | Ã | Ä | Å | Æ | Ç | È | É | Ê | Ë | Ì | Í | Î | Ï | 1)| Ñ | Ò | Ó | Ô | Õ | Ö | Ø | Ù | Ú | Û | Ü | 2) | 3)
STR_LOWER	<lower case letter> ::= a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p | q | r | s | t | u | v | w | x | y | z | à | á | â | ã | ä | å | æ | ç | è | é | ê | ë | ì | í | î | ï | 1) | ñ | ò | ó | ô | õ | ö | ø | ù | ú | û | ü | 2) | 3) | ß | ÿ-

下一页继续

3.77 string - 字符串
RobotWare - OS
续前页

名称	字符集
STR_WHITE	<blank character> ::=

1）冰岛语字母ð。

2）带重音符的字母Y。

3）冰岛语字母þ。

已经在系统中定义了以下常量：

```
CONST string diskhome := "HOME:";

! For old programs from S4C system
CONST string ram1disk := "HOME:";

CONST string disktemp := "TEMP:";

CONST string flp1 := "flp1:";

CONST string stSpace := " ";

CONST string stEmpty := "";
```

相关信息

信息，关于	请参阅
运用字符串的操作	技术参考手册 - *RAPID*语言概览，基本特征 - 表达式一节
字符串值	技术参考手册 - *RAPID*语言概览，基本特征 - 基本元素一节
运用字符集的指令	第*1256*页的*StrMemb* - 检查字符是否属于一组

3.78 stringdig - 只含数字的字符串

手册用法

stringdig用于代表纯数字字符串中的较大正整数。

因为数据类型num无法通过准确的表达式来处理高于8 388 608的正整数，因而引入该数据类型。

描述

一个stringdig仅可由一系列括有引号（“”）的数字0 ... 9组成，例如，"0123456789"。

数据类型stringdig可处理高达4 294 967 295的正整数。

基本示例

以下示例介绍了数据类型stringdig：

例 1

```
VAR stringdig digits1;
VAR stringdig digits2;
VAR bool flag1;
...
digits1 ="09000000";
digits2 = "9000001";
flag1 := StrDigCmp (digits1, LT, digits2);
```

将数据flag1设置为TRUE，因为09000000小于9000001。

特征

stringdig是string的一种别名数据类型，并因此继承其绝大多数特征。

相关信息

信息，关于	请参阅
字符串值	技术参考手册 - *RAPID*语言概览，基本特征 - 基本元素一节
字符串	第*1489*页的*string* - 字符串
数值	第*1435*页的*num* - 数值
比较运算符	第*1438*页的*opnum* - 比较运算符第*1247*页的*StrDigCmp* - 将仅含数字的两个字符串进行比较
将仅含数字的字符串进行比较	第*1247*页的*StrDigCmp* - 将仅含数字的两个字符串进行比较

3.79 supervtimeouts - 摇手监控超时

手册用法

supervtimeouts将用于为CAP中的摇手监控定义超时时间。

supervtimeouts是capdata的一个分量，定义了CAP中下列摇手监控阶段的超时时间：

- PRE
- END_PRE和START
- END MAIN和START_POST1
- END_POST1和START_POST2
- END_POST2

如果参数设为0，那么，不存在超时。

组件

pre_cond

数据类型：num

需满足的PRE阶段条件的超时时间（单位：秒）。

start_cond

数据类型：num

需满足的END_PRE和START阶段条件的超时时间（单位：秒）。

end_main_cond

数据类型：num

需满足的END_MAIN和START_POST1阶段条件的超时时间（单位：秒）。

end_post1_cond

数据类型：num

需满足的END_POST1和START_POST2阶段条件的超时时间（单位：秒）。

end_post2_cond

数据类型：num

需满足的END_POST2阶段条件的超时时间（单位：秒）。

语法

```
< data object of supervtimeouts >
  < pre_cond of num >
  < start_cond of num >
  < end_main_cond of num >
  < end_post1_cond of num >
  < end_post2_cond of num >
```

相关信息

	参见：
capdata数据类型	第1355页的*capdata - CAP数据*

下一页继续

	参见：
Continuous Application Platform	应用手册 - *Continuous Application Platform*

3.80 switch - Optional parameters

手册用法

switch用于可选参数。

描述

（仅）可将特殊类型switch分配给可选参数，并提供一种使用开关参数（即仅通过名称而非值来规定的参数）的方式。无法将值传送至开关参数。使用开关参数的唯一方式是通过使用预定义函数Present来检查开关参数的存在。

基本示例

以下示例介绍了数据类型switch：

例 1

```
PROC my_routine(\switch on | \switch off)
....
IF Present (off) THEN
....
ENDIF
ENDPROC
```

根据my_routine调用方使用的参数，可控制程序流程。

特征

switch为非值数据类型，且无法用于以值为导向的运算。

相关信息

信息，关于	请参阅
参数	技术参考手册 - *RAPID*语言概览，基本特征 - 程序一节
如何检查是否存在可选参数	第*1189*页的*Present* - 测试是否使用一个可选参数

3.81 symnum - 符号数

手册用法

symnum（*Symbolic Number*）用于代表一个含符号常量的整数。

描述

当检查函数OpMode和RunMode的返回值时，旨在使用symnum常量。

基本示例

以下示例介绍了数据类型symnum：

例 1

```
IF RunMode() = RUN_CONT_CYCLE THEN
..
ELSE
..
ENDIF
```

预定义数据

预定义数据类型symnum的以下符号常量，并且可在检查函数OpMode和RunMode的返回值时使用。

值	符号常量	备注
0	RUN_UNDEF	未定义的运行模式
1	RUN_CONT_CYCLE	持续或循环运行模式
2	RUN_INSTR_FWD	指示向前运行模式
3	RUN_INSTR_BWD	指示向后运行模式
4	RUN_SIM	仿真运行模式
5	RUN_STEP_MOVE	向前运动模式下的移动指令以及持续运行模式下的逻辑指令

值	符号常量	备注
0	OP_UNDEF	未定义的运行模式
1	OP_AUTO	自动的运行模式
2	OP_MAN_PROG	手动运行模式，最快250 mm/s
3	OP_MAN_TEST	手动全速运行模式，100%

特征

Symnum为有关num的一种别名数据类型，并因此继承其特征。

相关信息

信息，关于	请参阅
一般数据类型、别名数据类型	技术参考手册 - *RAPID*语言概览，基本特征 - 数据类型一节

3.82 syncident - 同步点的识别号

手册用法

syncident（*synchronization identity*）用于指定同步点的名称。同步点的名称将与syncident类申报数据的名称（识别号）相同。

描述

syncident用于识别程序中的点，实际程序任务将于此等待合作程序任务达到相同的同步点。

在所有合作程序任务中，syncident类数据名称（识别号）必须相同。

在指令WaitSyncTask、SyncMoveOn和SyncMoveOff中使用数据类型syncident。

基本示例

以下示例介绍了数据类型syncident：

例 1

位于程序任务ROB1中的程序实例

```
PERS tasks task_list{3} := [ ["STN1"], ["ROB1"], ["ROB2"] ];
VAR syncident sync1;

WaitSyncTask sync1, task_list;
```

在执行程序任务ROB1中的指令WaitSyncTask时，该程序任务的执行将进入等待，直至其他程序任务STN1和ROB2已通过相同的同步（交会）点sync1而达到其相应的WaitSyncTask。

结构

syncident是一种非数值的数据类型。

相关信息

信息，关于	请参阅
指定协作的程序任务	第*1500*页的*tasks - RAPID*程序任务
等待同步点及其他任务	第*890*页的*WaitSyncTask - 在同步点等待其他程序任务*
起动协调同步移动	第*709*页的*SyncMoveOn - 起动协调同步移动*
结束协调同步移动	第*704*页的*SyncMoveOff - 结束协调同步移动*

3.83 系统数据-当前RAPID系统数据设置

手册用法

System data反映RAPID系统的当前设置，例如，当前模型运动设置、当前错误恢复数ERRNO、当前中断数INTNO等等。

可通过程序来访问和读取此类数据。其可以用于读取当前状态，例如，当前程序位移。

C_MOTSET

数据类型motsetdata的变量C_MOTSET反映当前运动设置：

描述	数据类型	由以下进行修改	请参阅 第*1497*页的系统数据-当前RAPID系统数据设置。
当前运动设置，即：	motsetdata	指令：	第*1430*页的*motsetdata* - 运动设置数据
速率覆盖和最大速率		VelSet	第*849*页的*VelSet* - 改变编程速率
加速度覆盖		AccSet	第*19*页的*AccSet* – 降低加速度
围绕奇异点的运动		SingArea	第*608*页的*SingArea* - 确定奇点周围的插补
线性配置控制 接头配置控制		ConfL ConfJ	第*122*页的*ConfL* - 线性运动期间,监测配置 第*120*页的*ConfJ* - 接头移动期间,控制配置
路径分辨率		PathResol	第*445*页的*PathResol* - 覆盖路径分辨率
调节运动监控		MotionSup	第*333*页的*MotionSup* - 禁用/启用运动监控
降低TCP沿运动路径的加速度/减速度		PathAccLim	第*428*页的*PathAccLim* - 降低路径沿线的*TCP*加速度
修改圆弧插补期间的工具方位		CirPathMode	第*98*页的*CirPathMode* - 圆周路径期间的工具方位调整
降低世界坐标系中的有效负载加速度		WorldAccLim	第*908*页的*WorldAccLim* - 控制世界坐标系中的加速度

C_PROGDISP

数据类型progdisp的变量C_PROGDISP反映当前程序位移和外轴偏移量：

描述	数据类型	由以下进行修改	请参阅 第*1497*页的系统数据-当前RAPID系统数据设置。
机械臂各轴的当前程序位移	progdisp	指令：	第*1453*页的*progdisp* - 程序位移
		PDispSet	第*452*页的*PDispSet* - 启用使用已知坐标系的程序位移
		PDispOn	第*448*页的*PDispOn* - 启用程序位移
		PDispOff	第*447*页的*PDispOff* - 停用程序位移
当前外轴偏移量		EOffsSet	第*189*页的*EOffsSet* - 启用附加轴（使用已知值）的偏移量
		EOffsOn	第*187*页的*EOffsOn* - 启用附加轴的偏移量
		EOffsOff	第*186*页的*EOffsOff* - 停用附加轴的偏移量

下一页继续

3.83 系统数据-当前RAPID系统数据设置
RobotWare - OS
续前页

ERRNO

数据类型errnum的变量ERRNO反映当前错误恢复数：

描述	数据类型	由以下进行修改	请参阅第*1497*页的系统数据-当前*RAPID*系统数据设置。
出现的最新错误	errnum	系统	技术参考手册 - *RAPID*语言概览，*RAPID*概要 - 错误恢复一节 第*1414*页的*intnum* - 中断识别号

INTNO

数据类型intnum的变量INTNO反映当前中断号：

描述	数据类型	由以下进行修改	请参阅 第*1497*页的系统数据-当前*RAPID*系统数据设置。
出现的最新中断	intnum	系统	技术参考手册 - *RAPID*语言概览，*RAPID*概要 - 中断一节 第*1414*页的*intnum* - 中断识别号

ROB_ID

数据类型mecunit的变量ROB_ID包含了对实际程序任务中TCP机械臂（如有）的引用。

描述	数据类型	由以下进行修改	请参阅 第*1497*页的系统数据-当前*RAPID*系统数据设置。
对实际程序任务中机械臂（如有）的引用。在通过TaskRunRob ()进行使用之前，始终进行检查	mecunit	系统	第*1428*页的*mecunit* - 机械单元

3.84 taskid - 任务标识

手册用法

taskid用于识别系统中的可用程序任务。

在系统参数中定义程序任务的名称，因此，不能在程序中对其进行定义。

描述

taskid类数据仅包含对程序任务的引用。

限制

不得在程序中定义taskid类数据。另一方面，在声明一个程序时，可将数据类型用作一个参数。

预定义数据

始终可以从程序（安装数据）来访问系统参数中定义的程序任务。

对于系统中的所有程序任务，可获得数据类型taskid的预定义变量。变量识别号将为“任务名”+“Id”，例如，对于T_ROB1任务，变量识别号将为T_ROB1Id、T_ROB2 - T_ROB2Id等。

特征

taskid为非值数据类型。这意味着此类数据不允许以值为导向的运算。

相关信息

信息，关于	请参阅
保存普通程序模块	第*542*页的*Save* - 保存普通程序模块
程序任务的配置	技术参考手册 - 系统参数
非值数据类型的特征	技术参考手册 - *RAPID*语言概览，基本特征 - 数据类型一节

3.85 tasks - RAPID程序任务

手册用法

tasks用于规定多个RAPID程序任务。

描述

为规定多个RAPID程序任务，可将各任务的名称作为一个字符串。随后，数据类型tasks的数组可保存所有任务名称。

随后，可将此任务列表用于指令WaitSyncTask和SyncMoveOn中。

> **注意**
>
> 上述指令要求将数据定义为可在所有合作任务中获得的系统全局PERS变量。

组件

此数据类型有以下组件。

taskname

数据类型：string

在字符串中规定RAPID程序任务的名称。

基本示例

以下示例介绍了数据类型tasks：

例 1

位于程序任务T_ROB1中的程序实例

```
PERS tasks task_list{3} := [ ["T_STN1"], ["T_ROB1"], ["T_ROB2"] ];
VAR syncident sync1;

WaitSyncTask sync1, task_list;
```

在执行程序任务T_ROB1中的指令WaitSyncTask时，该程序任务的执行将进入等待，直至其他所有程序任务T_STN1和T_ROB2已通过相同的同步（交会）点sync1而达到其相应的WaitSyncTask。

结构

```
<dataobject of tasks>
  <taskname of string>
```

相关信息

信息，关于	请参阅
同步点的识别号	第*1496*页的*syncident - 同步点的识别号*
等待同步点及其他任务	第*890*页的*WaitSyncTask - 在同步点等待其他程序任务*
起动协调同步移动	第*709*页的*SyncMoveOn - 起动协调同步移动*
结束协调同步移动	第*704*页的*SyncMoveOff - 结束协调同步移动*

3.86 testsignal - 测试信号

手册用法

当对机械臂运动系统进行试验时，使用数据类型testsignal。

描述

可在机械臂系统中获得一系列预定义试验信号。可获得testsignal数据类型，从而简化指令TestSignDefine的编程。

基本示例

以下示例介绍了数据类型testsignal：

例 1

```
TestSignDefine 2, speed, Orbit, 2, 0;
```

预定义常量speed用于读取机械臂orbit上轴2的实际速度。

预定义数据

在系统中预定义有关外机械臂各轴的以下试验信号。所有数据均位于SI单元中，并在轴的电动机侧进行测量。

符号常量	值	单位
speed	6	rad/s
torque_ref	9	Nm
resolver_angle	1	rad
speed_ref	4	rad/s
dig_input1	102	0或1
dig_input2	103	0或1

特征

testsignal为有关num的一种别名数据类型，并因此继承其特征。

相关信息

信息，关于	请参阅
定义测试信号	第727页的*TestSignDefine - 定义测试信号*
读取测试信号	第1276页的*TestSignRead - 读取测试信号值*
重置测试信号	第729页的*TestSignReset - 重置所有测试信号定义*

3.87 tooldata - 工具数据

手册用法

tooldata用于描述工具（例如，焊枪或夹具）的特征。此类特征包括工具中心点（TCP）的位置和方位以及工具负载的物理特征。

如果工具得以固定在空间中（固定工具），则工具数据首先定义空间中该工具的位置和方位以及TCP。随后，描述机械臂所移动夹具的负载。

描述

工具数据在以下方面影响机械臂运动：

- 工具中心点（TCP）指的是一个将满足指定路径和速率性能的点。如果重新定位工具或使用经协调的外轴，则仅该点将以编程速率沿所期望路径移动。
- 如果使用固定工具，则编程速度和路径将与机械臂所夹持的工件相关。
- 编程位置指的是当前TCP位置以及相对于工具坐标系的方位。这意味着，如果更换受损工具，则在重新定义工具坐标系的情况下，仍可使用旧程序。

当机械臂点动时，同时使用工具数据，从而：

- 定义在重新定位工具时未发生移动的TCP。
- 定义工具坐标系，从而便于朝工具坐标方向移动或旋转。

警告

重要的是，始终定义实际工具负载以及使用时机械臂（例如，抓取部分）的有效负载。负载数据定义不正确可能会导致机械臂机械结构过载。

指定不正确的负载数据时，其常常会引起以下后果：

- 机械臂将不会用于其最大容量
- 路径准确性受损，包括过度风险
- 机械结构过载风险

组件

`robhold`

robot hold

数据类型：`bool`

定义机械臂是否夹持工具：

- TRUE：机械臂正夹持着工具。
- FALSE：机械臂未夹持工具，即为固定工具。

`tframe`

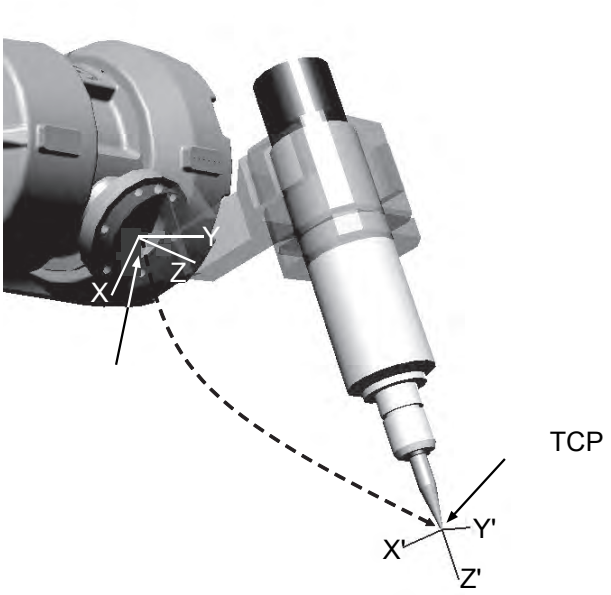
tool frame

数据类型：`pose`

工具坐标系，即：

- TCP的位置（x、y和z）以mm计，并用腕坐标系来表示（tool0）（参见下图）。
- 工具坐标系的方位，用腕坐标系来表示（参见下图）。

下一页继续

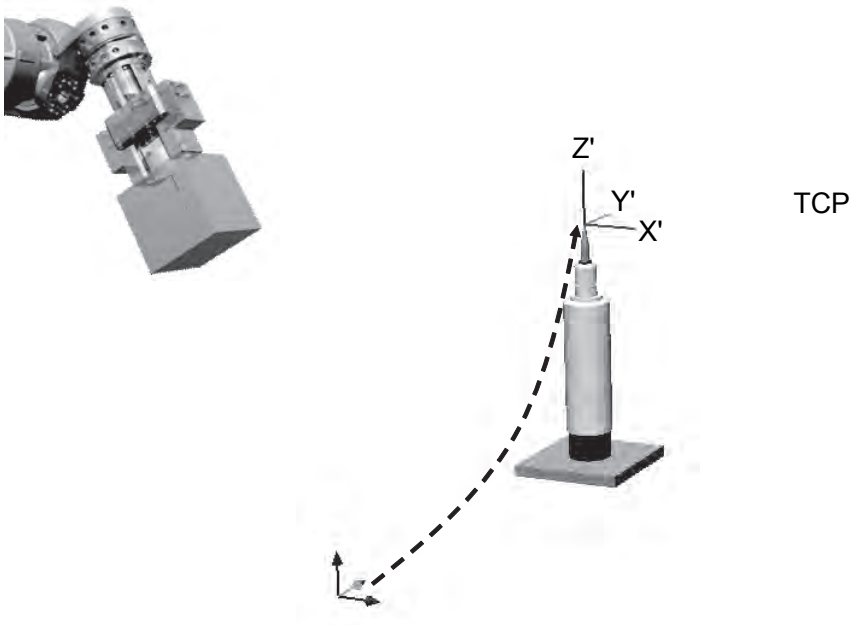


xx1100000517

Figure 3.3: 机械臂夹持的工具

注意

如果使用固定工具，则相对于世界坐标系来定义工具坐标系。



xx1100000518

Figure 3.4: 固定工具

`tload`

tool load

下一页继续

3.87 tooldata - 工具数据
RobotWare - OS
续前页

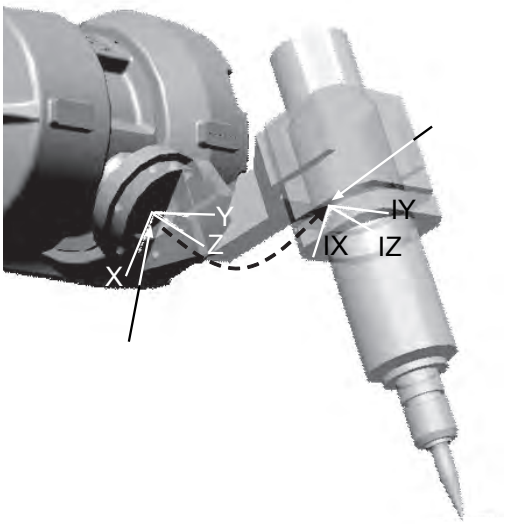
数据类型：loaddata

> **注意**
>
> 此数据用于机械臂夹持工具和固定工具。对于机械臂夹持工具，数据描述工具负载。对于固定工具，数据描述机械臂所持夹具的负载。

机械臂所持工具：

工具的负载，即：

- 工具的质量（重量），以kg计。
- 工具负载的重心（x、y和z），以mm计，并以腕坐标系来表示。
- 工具力矩主惯性轴的方位，用腕坐标系表示。
- 围绕力矩惯性轴的惯性矩，以kgm^2计。如果将所有惯性部件定义为0 kgm^2，则将工具作为一个点质量来处理。



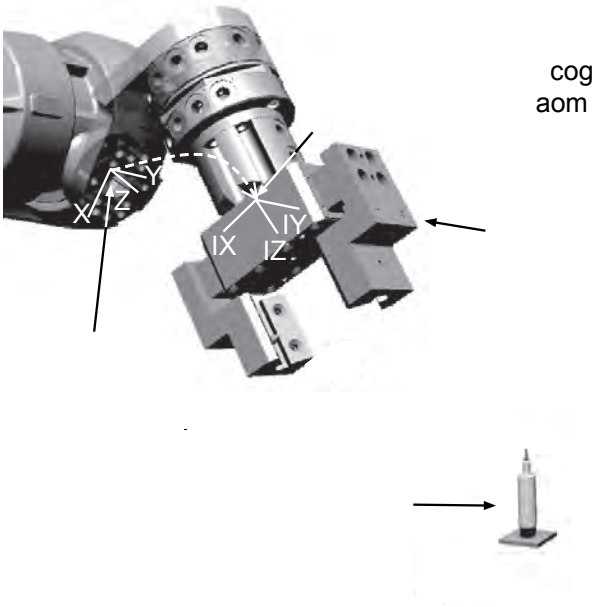
xx1100000519

固定工具：

用于夹持工件的夹具的负载：

- 所移动夹具的质量（重量），以kg计
- 所移动夹具的重心（x、y和z），以mm计，并以腕坐标系来表示
- 所移动夹具力矩主惯性轴的方位，用腕坐标系表示
- 围绕力矩惯性轴的惯性矩，以kgm^2计。如果将所有惯性部件定义为0 kgm^2，则将夹具作为一个点质量来处理。

下一页继续



xx1100000520

注意

仅在tooldata中规定工具/夹具的负载。通过指令GripLoad，连接和断开夹具所能处理的有效负载，并使用loaddata来进行确定。

可能针对带夹持工件的夹具和不带工件的夹具，定义和使用不同的tooldata，而非使用指令GripLoad。

概述

针对机械臂所持工具，在腕坐标系中定义tooldata中TCP的位置和方位。

针对固定工具，在世界坐标系中定义tooldata中TCP的位置和方位。

在所有情况下，tooldata中的loaddata部分均与腕坐标系相关，无论是否使用机械臂所持工具（用以描述工具）或固定工具（用以描述夹具）。

基本示例

以下示例介绍了数据类型tooldata：

例 1

```
PERS tooldata gripper := [ TRUE, [[97.4, 0, 223.1], [0.924, 0,
     0.383 ,0]], [5, [23, 0, 75], [1, 0, 0, 0], 0, 0, 0]];
```

使用以下值来描述工具：

- 机械臂正夹持着工具。
- TCP所在点与安装法兰的直线距离为223.1 mm，且沿腕坐标系X轴97.4 mm。
- 工具的X'方向和Z'方向相对于腕坐标系Y方向旋转45°。
- 工具质量为5 kg。
- 重心所在点与安装法兰的直线距离为75 mm，且沿腕坐标系X轴23 mm。
- 可将负载视为一个点质量，即不带任何惯性矩。

下一页继续

3.87 tooldata - 工具数据
RobotWare - OS
续前页

例 2

```
gripper.tframe.trans.z := 225.2;
```

将工具、gripper的TCP调整至沿z方向225.2处。

限制

应将工具数据定义为一个永久变量（PERS），且不得在程序内进行定义。随后，在保存程序时保存当前值，并在有载时恢复当前值。

任意运动指令中的工具数据类参数应仅为一个完整的永久数据对象（而非数组元素或记录成分）。

预定义数据

工具tool0定义腕坐标系，其原点为安装法兰的中心。始终可以从程序来访问tool0，但是不得进行改变（储存在系统程序模块BASE中）。

```
PERS tooldata tool0 := [ TRUE, [ [0, 0, 0], [1, 0, 0 ,0] ], [0.001,
     [0, 0, 0.001], [1, 0, 0, 0], 0, 0, 0] ];
```

结构

```
< dataobject of tooldata >
  < robhold of bool >
  < tframe of pose >
    < trans of pos >
      < x of num >
      < y of num >
      < z of num >
    < rot of orient >
      < q1 of num >
      < q2 of num >
      < q3 of num >
      < q4 of num >
  < tload of loaddata >
    < mass of num >
    < cog of pos >
      < x of num >
      < y of num >
      < z of num >
    < aom of orient >
      < q1 of num >
      < q2 of num >
      < q3 of num >
      < q4 of num >
    < ix of num >
    < iy of num >
    < iz of num >
```

相关信息

信息，关于	请参阅
定位器指令	技术参考手册 - *RAPID*语言概览，*RAPID*概要 - 运动一节

下一页继续

信息，关于	请参阅
坐标系	技术参考手册 - *RAPID*语言概览，运动和*I/O*原则 - 坐标系一节
定义机械臂的有效负载	第*223*页的*GripLoad* - 定义机械臂的有效负载
负载数据的定义	第*1421*页的*loaddata* - 加载数据
工件数据的定义	第*1523*页的*wobjdata* - 工件数据

3.88 tpnum - FlexPendant示教器窗口编号

手册用法

tpnum用于代表FlexPendant示教器窗口号以及符号常量。

描述

tpnum常量旨在用于指令TPShow中。

基本示例

以下示例介绍了数据类型tpnum：

例 1

```
TPShow TP_LATEST;
```

执行本指令后，将启用于当前FlexPendant示教器窗口前最后使用的FlexPendant示教器窗口。

预定义数据

预定义数据类型tpnum的以下符号常量，并可用于指令TPShow中：

值	符号常量	备注
2	TP_LATEST	最新使用的FlexPendant示教器窗口

特征

tpnum为有关num的一种别名数据类型，并因此继承其特征。

相关信息

信号关于	请参阅
一般数据类型、别名数据类型	技术参考手册 - *RAPID*语言概览，基本特征 - 数据类型一节
使用FlexPendant示教器进行通信	技术参考手册 - *RAPID*语言概览，*RAPID*概要 - 通信一节
位于FlexPendant示教器上的开关窗口	第*743*页的*TPShow - 位于FlexPendant示教器上的开关窗口*

3.89 trapdata - 当前TRAP的中断数据

手册用法

trapdata（软中断数据）用于容纳可使得当前TRAP程序开始执行的中断数据。

在使用指令ReadErrData之前，于指令IError所产生的TRAP程序中使用。

描述

trapdata 类数据代表与中断相关的内部信息，此中断使得当前软中断程序开始执行。其内容取决于中断类型。

基本示例

以下示例介绍了数据类型trapdata：

例 1

```
VAR errdomain err_domain;
VAR num err_number;
VAR errtype err_type;
VAR trapdata err_data;
...
TRAP trap_err
  GetTrapData err_data;
  ReadErrData err_data, err_domain, err_number, err_type;
ENDTRAP
```

当错误被困于软中断程序trap_err时，将错误域、错误编号和错误类型保存在适当的trapdata型非值变量中。

特征

trapdata是一种非数值的数据类型。

相关信息

信息，关于	请参阅
中断概述	技术参考手册 - *RAPID*语言概览，*RAPID*概要 - 中断一节
更多关于中断管理的信息	技术参考手册 - *RAPID*语言概览，基本特征 - 中断一节
非值数据类型	技术参考手册 - *RAPID*语言概览，基本特征 - 数据类型一节
调整关于错误的中断	第*234*页的*IError - 调整关于错误的中断*
获取当前TRAP的中断数据	第*219*页的*GetTrapData - 获取当前TRAP的中断数据*
获取关于错误的信息	第*496*页的*ReadErrData - 获取关于错误的信息*
Advanced RAPID	产品规格 - 控制器软件*IRC5*

3.90 触发数据 - 定位事件，触发

手册用法

triggdata用于储存有关机械臂移动期间定位事件的数据。

一起定位事件的具体形式既可以是设置一个输出信号，也可以是在机器人移动路径上的某特定位置处运行一则中断例程。

描述

为了在发生定位事件时定义各度量的条件，将采用triggdata类型的变量。将采用TriggIO、TriggEquip、TriggCheckIO、TriggInt、TriggSpeed或TriggRampAO指令之一来在程序中形成该变量的数据成分，并凭借TriggL、TriggC或TriggJ指令之一来运用该变量的数据成分。

基本示例

以下示例介绍了数据类型triggdata ：

例 1

```
VAR triggdata gunoff;
TriggIO gunoff, 0,5 \DOp:=gun, 0;
TriggL p1, v500, gunoff, fine, gun1;
```

当TCP位于点p1之前0,5 mm时，将数字信号输出信号gun设置为值0。

特征

triggdata 是一种非数值的数据类型。

相关信息

信息，关于	请参阅
triggs的定义	第*773*页的*TriggIO* - 定义停止点附近的固定位置或时间*I/O*事件 第*763*页的*TriggEquip* - 定义路径上的固定位置和时间*I/O*事件 第*754*页的*TriggCheckIO* - 定义位于固定位置的*IO*检查 第*769*页的*TriggInt* - 定义与位置相关的中断
triggs的使用	第*785*页的*TriggL* - 关于事件的机械臂线性运动 第*747*页的*TriggC* - 关于事件的机械臂圆周移动 第*778*页的*TriggJ* - 关于事件的轴式机械臂运动
非值数据类型的特征	技术参考手册 - *RAPID*语言概览，基本特征 - 数据类型一节

3.91 triggios - Positioning events, trigg

手册用法

triggios用于储存有关机械臂移动期间定位事件的数据。当定位事件分布于路径上的特定位置时，将输出信号设置为指定值。

描述

triggios用于定义有关设置机械臂移动路径沿线固定位置处的数字信号输出信号、数字信号输出信号组或模拟信号输出信号的条件和行动。

组件

used

数据类型：bool

定义是否应当使用数组元素。

distance

数据类型：num

定义路径上应出现I/O事件的位置。如果将组件start设置为FALSE，则规定为距移动路径终点的距离，以mm计（正值）。

start

数据类型：bool

当距离始于移动起点而非终点时，设置为TRUE。

equiplag

Equipment Lag

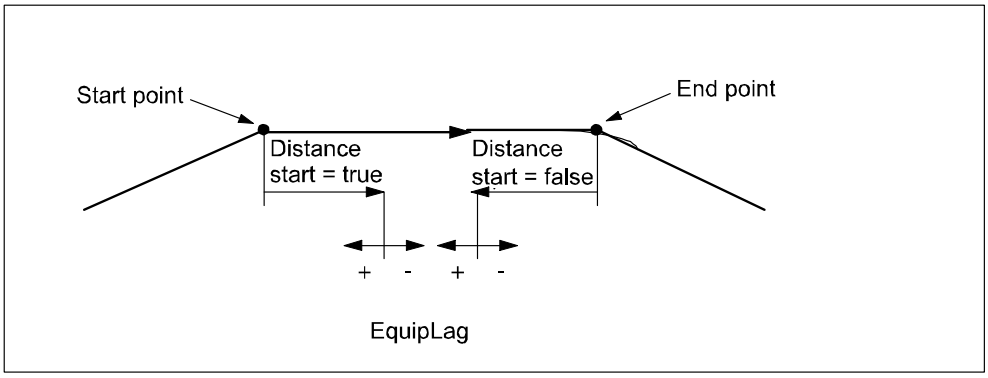
数据类型：num

指定外部设备的滞后，以s计。

关于外部设备滞后补偿，使用正参数值。正值意味着I/O信号由TCP达到相对于移动起点或终点的指定距离前一指定时间的机器人系统来设置。

负值意味着I/O信号由TCP已通过相对于移动起点或终点的指定距离后一指定时间的机器人系统来设置。

本图显示了组件equiplag的使用。



xx0800000173

下一页继续

3.91 triggios - Positioning events, trigg

RobotWare - OS

续前页

`signalname`

数据类型：`string`

所应更改的信号的名称。其必须为数字信号输出信号、数字信号输出信号组或模拟信号输出信号。

`setvalue`

数据类型：`num`

输出信号的期望值（处于当前信号的容许范围内）。

`xxx`

数据类型：`num`

当前未使用组件。增加以便能够增强未来版本的功能，并且仍然能够兼容。

示例

以下示例介绍了数据类型`triggios`：

例 1

```
VAR triggios gunon{1};

gunon{1}.used:=TRUE;
gunon{1}.distance:=3;
gunon{1}.start:=TRUE;
gunon{1}.signalname:="gun";
gunon{1}.equiplag:=0;
gunon{1}.setvalue:=1;

MoveJ p1, v500, z50, gun1;
TriggLIOs p2, v500, \TriggData1:=gunon, z50, gun1;
MoveL p3, v500, z50, gun1;
```

当TCP位于点`p1`后3 mm时，设置信号`gun`。

结构

```
<dataobject of triggios>
  <used of bool>
  <distance of num>
  <start of bool>
  <equiplag of num>
  <signalname of string>
  <setvalue of num>
  <xxx of num>
```

相关信息

信息，关于	请参阅
定位事件，trigg	第*1513*页的*triggiosdnum - Positioning events, trigg*
机械臂线性移动以及I/O事件	第*798*页的*TriggLIOs - 机械臂线性移动以及I/O事件*

3.92 triggiosdnum - Positioning events, trigg

手册用法

triggiosdnum用于储存有关机械臂移动期间定位事件的数据。当定位事件分布于路径上的特定位置时，将输出信号设置为指定值。

描述

triggiosdnum用于定义有关设置机械臂移动路径沿线固定位置处的数字信号输出信号、数字信号输出信号组或模拟信号输出信号的条件和行动。

组件

used

数据类型：bool

定义是否应当使用数组元素。

distance

数据类型：num

定义路径上应出现I/O事件的位置。如果将组件start设置为FALSE，则规定为距移动路径终点的距离，以mm计（正值）。

start

数据类型：bool

当距离始于移动起点而非终点时，设置为TRUE。

equiplag

Equipment Lag

数据类型：num

指定外部设备的滞后，以s计。

关于外部设备滞后补偿，使用正参数值。正值意味着I/O信号由TCP达到相对于移动起点或终点的指定距离前一指定时间的机器人系统来设置。

负值意味着I/O信号由TCP已通过相对于移动起点或终点的指定距离后一指定时间的机器人系统来设置。

signalname

数据类型：string

所应更改的信号的名称。其必须为数字信号输出信号、数字信号输出信号组或模拟信号输出信号。

setvalue

数据类型：dnum

输出信号的期望值（处于当前信号的容许范围内）。

xxx

数据类型：num

当前未使用组件。增加以便能够增强未来版本的功能，并且仍然能够兼容。

下一页继续

3.92 triggiosdnum - Positioning events, trigg

RobotWare - OS

续前页

示例

以下示例介绍了数据类型triggiosdnum：

例 1

```
VAR triggiosdnum gunon{1};

gunon{1}.used:=TRUE;
gunon{1}.distance:=3;
gunon{1}.start:=TRUE;
gunon{1}.signalname:="go_gun";
gunon{1}.equiplag:=0;
gunon{1}.setvalue:=123456789;

MoveJ p1, v500, z50, gun1;
TriggLIOs p2, v500, \TriggData3:=gunon, z50, gun1;
MoveL p3, v500, z50, gun1;
```

当TCP位于点p1后3 mm时，设置信号go_gun。

结构

```
<dataobject of triggiosdnum>
  <used of bool>
  <distance of num>
  <start of bool>
  <equiplag of num>
  <signalname of string>
  <setvalue of dnum>
  <xxx of num>
```

相关信息

信息，关于	请参阅
定位事件，trigg	第*1511*页的*triggios - Positioning events, trigg*
机械臂线性移动以及I/O事件	第*798*页的*TriggLIOs - 机械臂线性移动以及I/O事件*

3.93 triggmode - 触发行动模式

手册用法

triggmode用于在定义触发器时指定不同的行动模式。

描述

triggmode常量将用于定义触发器定义中采用的指令的模式。

基本示例

下列示例展示了triggmode数据类型：

例 1

```
CONNECT intno1 WITH trap1;
TriggInt trigg1, Distance:=17, intno1 \Inhib:=inhibit
     \Mode:=TRIGG_MODE1;
TriggL p1, v500, trigg1, z50, gun1;
```

中断程序trap1在运行中，当TCP处于点p1，距离这个点17 mm；如果永久变量inhibit标志为TRUE （模式TRIGG_MODE1 从inhibit标志读取）。

例 2

```
TriggEquip trigg1, 17, 0 \GOp:=go1, SetValue:=5 \Inhib:=inhibit
     \Mode:=TRIGG_MODE2;
TriggL p1, v500, trigg1, z50, gun1;
```

如果永久变量标记inhibit为FALSE，那么，当TCP处于p1点前17 mm位置时，I/O信号go1被设为SetValue中指定的值。如果永久变量inhibit为TRUE，那么完全不执行任何行动（保持I/O信号go1的值）。

例 3

```
TriggEquip trigg1, 17, 0 \GOp:=go1, SetValue:=0 \Inhib:=inhibit
     \InhibSetValue:=setDnum \Mode:=TRIGG_MODE3;
TriggL p1, v500, trigg1, z50, gun1;
```

如果永久变量标记inhibit为TRUE，那么，当TCP处于p1点前17 mm位置时，I/O信号go1被设为从dnum永久变量setDnum中读取的值。如果inhibit为FALSE，那么完全不执行任何行动（保持I/O信号go1的值）。

预定义数据

triggmode数据类型的下列符号常量被预定义，可用于在定义触发器时指定不同的行动模式。

值	符号常量	备注
1	TRIGG_MODE1	可凭借TriggCheckIO、TriggEquip、TriggIO、TriggInt、TriggSpeed和TriggRampAO指令来进行使用。 反转从可选参数Inhib采用的永久变量读取的值。

下一页继续

3.93 triggmode - 触发行动模式
RobotWare - OS
续前页

值	符号常量	备注
2	TRIGG_MODE2	如果采用可选参数Inhib，那么，可凭借TriggEquip、TriggIO、TriggSpeed和TriggRampAO指令来进行使用。 如果在到达设置I/O信号的位置-时间时，Inhib中采用的指定标记的实际值为TRUE，那么不会更新指定的I/O信号（无行动）。
3	TRIGG_MODE3	仅可与TriggEquip和TriggIO指令的可选参数Inhib和InhibSetValue结合使用。 完全不考虑参数SetValue和SetDvalue。 如果Inhib中采用的指定标记的实际值为TRUE，那么，I/O信号设为可选参数InhibSetValue采用的永久变量在该位置具备的值。 如果Inhib中采用的指定标记的实际值为FALSE，那么，I/O信号不更新（无行动）。

特征

triggmode是num的别名数据类型，因此继承了其特征。

相关信息

信息，关于	请参阅
触发的使用	第*785*页的*TriggL* - 关于事件的机械臂线性运动 第*747*页的*TriggC* - 关于事件的机械臂圆周移动 第*778*页的*TriggJ* - 关于事件的轴式机械臂运动
触发的定义	第*763*页的*TriggEquip* - 定义路径上的固定位置和时间*I/O*事件 第*769*页的*TriggInt* - 定义与位置相关的中断 第*773*页的*TriggIO* - 定义停止点附近的固定位置或时间*I/O*事件 第*804*页的*TriggRampAO* - 定义路径上的固定位置斜坡*AO*事件 第*810*页的*TriggSpeed* - 定义与固定位置-时间尺度事件成比例的*TCP*速度模拟信号输出
定义固定位置的I/O检查	第*754*页的*TriggCheckIO* - 定义位于固定位置的*IO*检查
触发数据的储存	第*1510*页的触发数据 - 定位事件，触发
一般数据类型、别名数据类型	技术参考手册 - *RAPID*语言概览，基本特征 - 数据类型一节

3.94 triggstrgo - Positioning events, trigg

手册用法

`triggstrgo`*(trigg stringdig group output)*用于储存有关机械臂移动期间定位事件的数据。当定位事件分布于路径上的特定位置时，将数字信号输出信号组设置为指定值。

描述

`triggstrgo`用于定义有关设置机械臂移动路径沿线固定位置处的数字信号输出信号组的条件和行动。

组件

`used`

数据类型：`bool`

定义是否应当使用数组元素。

`distance`

数据类型：`num`

定义路径上应出现I/O事件的位置。如果将组件`start`设置为FALSE，则规定为距移动路径终点的距离，以mm计（正值）。

`start`

数据类型：`bool`

当距离始于移动起点而非终点时，设置为TRUE。

`equiplag`

Equipment Lag

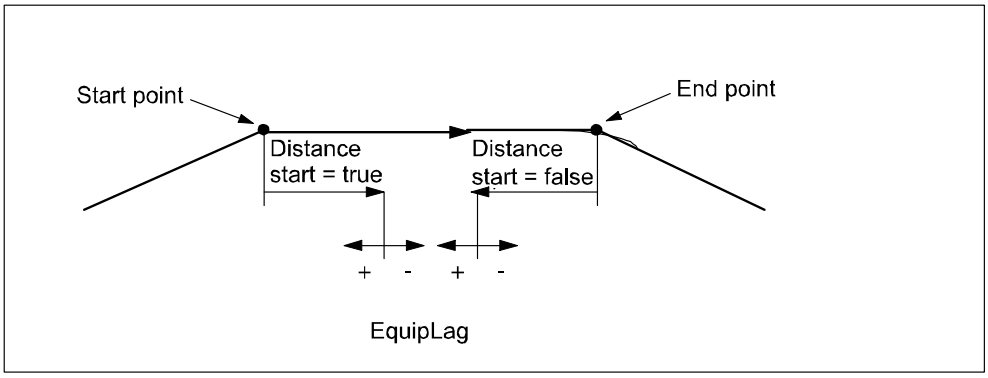
数据类型：`num`

指定外部设备的滞后，以s计。

关于外部设备滞后补偿，使用正参数值。正值意味着I/O信号由TCP达到相对于移动起点或终点的指定距离前一指定时间的机器人系统来设置。

负值意味着I/O信号由TCP已通过相对于移动起点或终点的指定距离后一指定时间的机器人系统来设置。

本图显示了组件`equiplag`的使用。



xx0800000173

下一页继续

3.94 triggstrgo - Positioning events, trigg
RobotWare - OS
续前页

signalname

数据类型：string

所应更改的信号的名称。其必须为组输出信号的名称。

setvalue

数据类型：stringdig

输出信号的期望值（在当前数字组输出信号的容许范围内）。使用stringdig数据类型，使其可能使用高达4294967295的值，其为数字信号组所能拥有的最大值（本系统组信号中最多包含32个信号）。

xxx

数据类型：num

当前未使用组件。增加以便能够增强未来版本的功能，并且仍然能够兼容。

示例

以下示例介绍了数据类型triggstrgo：

例 1

```
VAR triggstrgo gunon{1};

gunon{1}.used:=TRUE;
gunon{1}.distance:=3;
gunon{1}.start:=TRUE;
gunon{1}.signalname:="gun";
gunon{1}.equiplag:=0;
gunon{1}.setvalue:="4294967295";

MoveJ p1, v500, z50, gun1;
TriggLIOs p2, v500, \TriggData2:=gunon, z50, gun1;
MoveL p3, v500, z50, gun1;
```

当TCP位于点p1后方3 mm时，将信号gun设置为值4294967295。

结构

```
<dataobject of triggstrgo>
  <used of bool>
  <distance of num>
  <start of bool>
  <equiplag of num>
  <signalname of string>
  <setvalue of stringdig>
  <xxx of num>
```

相关信息

信息，关于	请参阅
机械臂线性移动以及I/O事件	第*798*页的*TriggLIOs - 机械臂线性移动以及I/O事件*
将仅含数字的两个字符串进行比较	第*1247*页的*StrDigCmp - 将仅含数字的两个字符串进行比较*

下一页继续

信息，关于	请参阅
有关stringdig数据类型的算术运算	第*1245*页的*StrDigCalc - 有关数据类型数字字符串的算术运算*

3.95 tunetype - 伺服调节类型

手册用法

tunetype用于代表一个含不同伺服调节类型符号常量的整数。

描述

tunetype常量旨在用作指令TuneServo的参数。

基本示例

以下示例介绍了数据类型tunetype：

例 1

```
TuneServo MHA160R1, 1, 110 \Type:= TUNE_KP;
```

预定义数据

预定义数据类型tunetype的以下符号常量，并可用作指令TuneServo的参数。

值	符号常量	备注
0	TUNE_DF	减少过界
1	TUNE_KP	影响位置控制增益
2	TUNE_KV	影响速度控制增益
3	TUNE_TI	影响速度控制整合时间
4	TUNE_FRIC_LEV	影响摩擦力补偿等级
5	TUNE_FRIC_RAMP	影响摩擦力补偿斜坡
6	TUNE_DG	减少过界
7	TUNE_DH	减少与重负载的摩擦力
8	TUNE_DI	减少路径错误
9	TUNE_DK	仅供ABB内部使用
10	TUNE_DL	仅供ABB内部使用

特征

tunetype 为有关num的一种别名数据类型，并因此继承其特征。

相关信息

信息，关于	请参阅
一般数据类型、别名数据类型	技术参考手册 - *RAPID*语言概览，基本特征 - 数据类型一节
数据类型tunetype的使用	第826页的*TuneServo - 调节伺服*

3.96 uishownum - UIShow的实例标识符

手册用法

uishownum为用于指令UIShow中参数InstanceId的数据类型。其用于确定FlexPendant示教器上的视图。

描述

当uishownum类永久变量与指令UIShow一同使用时，给定一个特定值，以识别在FlexPendant示教器上启动的视图。随后，将此永久变量用于涉及该视图的所有处理过程，例如，再次启动视图、修改视图等。

示例

以下示例介绍了数据类型uishownum：

例 1

```
CONST string Name:="TpsViewMyAppl.gtpu.dll";
CONST string Type:="ABB.Robotics.SDK.Views.TpsViewMyAppl";
CONST string Cmd1:="Init data string passed to the view";
PERS uishownum myinstance:=0;
VAR num mystatus:=0;
...
! Launch one view of the application MyAppl
UIShow Name, Type \InitCmd:=Cmd1 \InstanceID:=myinstance
     \Status:=mystatus;
```

通过init命令Cmd1.，上述事件号将启动一个应用MyAppl的视图。本记号用于识别保存在参数myinstance中的视图。

特征

uishownum为有关num的一种别名数据类型，并因此继承其属性。

相关信息

信息，关于	请参阅
UIShow	第839页的*UIShow - 用户界面显示*

3.97 weavestartdata - 摆动启动数据

手册用法

weavestartdata用于在一个CAP进程启动和重启时控制定点摆动。

weavestartdata是capdata的一个分量，定义了在CAP进程启动或重启时定点摆动的属性：

- 在启动时应当进行定点摆动的情况下（active）
- 定点摆动宽度（width）
- 相对于路径方向的方向（dir）
- 定点摆动频率（cycle_time）

定点摆动一直采用“之”字形几何摆动，请参见第*1366*页的*capweavedata - CAP*摆动数据。

组件

active

数据类型：bool

值	描述
TRUE	在CAP进程开始时进行定点摆动
FALSE	在CAP进程开始时不进行定点摆动

width

数据类型：num

定义定点摆动幅度（毫米）。

dir

数据类型：num

定义定点摆动相对于路径方向的方向（度）。零度表示摆动同时垂直于路径和工具z坐标。

cycle_time

数据类型：num

定义整个定点摆动周期的总时间（单位：秒）

语法

```
< data object of weavestartdata >
  < active of bool >
  < width of num >
  < dir of num >
  < cycle_time of num >
```

相关信息

	参见：
capdata数据类型	第*1355*页的*capdata - CAP*数据
Continuous Application Platform	应用手册 - *Continuous Application Platform*

3.98 wobjdata - 工件数据

手册用法

wobjdata用于描述机械臂焊接、处理、于其内部移动等的工件。

描述

如果在定位指令中定义工件，则位置将基于工件的坐标。本过程的优势如下：

- 如果手动输入位置数据，例如离线编程，则常常可从图纸获得有关值。
- 在机械臂装置改变后，可迅速重新使用程序。例如，如果移动固定装置，则仅须重新定义用户坐标系。
- 可对工件附着过程中的变化进行补偿。为此，将需要某种传感器来定位工件。

如果使用固定工具或协调外轴，则必须定义工件，因为路径和速率随后将与工件而非TCP相关。

工件数据亦可用于点动：

- 可使机械臂朝工件方向点动。
- 根据工件的坐标系，显示当前位置。

组件

robhold

robot hold

数据类型：bool

规定实际程序任务中的机械臂是否正夹持着工件：

- TRUE:机械臂正夹持着工件，即使用一个固定工具。
- FALSE:机械臂未夹持着工件，即机械臂正夹持着工具。

ufprog

user frame programmed

数据类型：bool

规定是否使用固定的用户坐标系：

- TRUE:固定的用户坐标系。
- FALSE:可移动的用户坐标系，即使用协调外轴。同时以半协调或同步协调模式用于MultiMove系统。

ufmec

user frame mechanical unit

数据类型：string

用于协调机械臂移动的机械单元。仅在可移动的用户坐标系中进行规定（ufprog为FALSE）。

规定系统参数中所定义的机械单元名称，例如，orbit_a。

uframe

user frame

数据类型：pose

下一页继续

3.98 wobjdata - 工件数据
RobotWare - OS
续前页

用户坐标系，即当前工作面或固定装置的位置（参见下图）：

- 坐标系原点的位置（x、y和z），以mm计。
- 坐标系的旋转，表示为一个四元数（q1、q2、q3和q4）。

如果机械臂正夹持着工具，则在世界坐标系中定义用户坐标系（如果使用固定工具，则在腕坐标系中定义）。

对于可移动的用户坐标系（ufprog为FALSE），由本系统对用户坐标系进行持续定义。

`oframe`

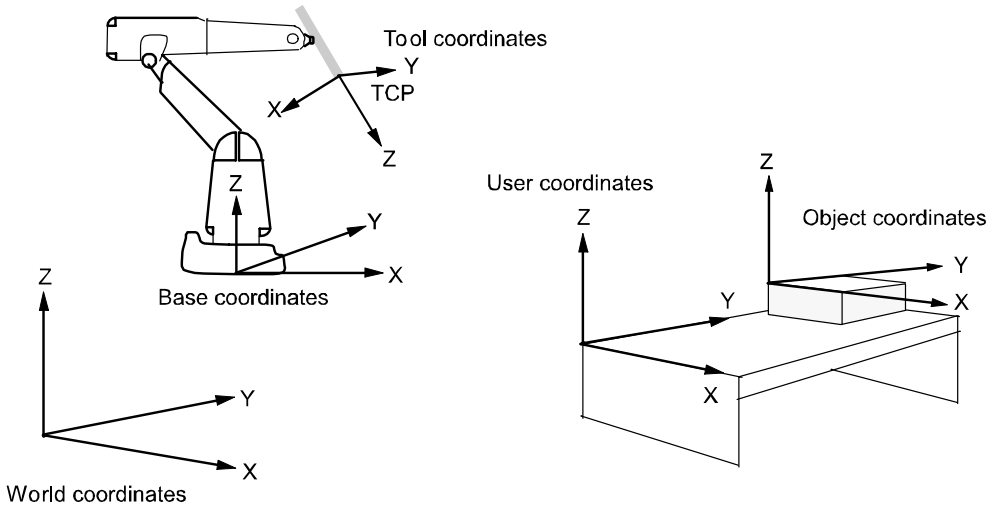
object frame

数据类型：pose

目标坐标系，即当前工件的位置（参见下图）：

- 坐标系原点的位置（x、y和z），以mm计。
- 坐标系的旋转，表示为一个四元数（q1、q2、q3和q4）。

在用户坐标系中定义目标坐标系。



xx0500002369

基本示例

以下示例介绍了数据类型wobjdata：

例 1

```
PERS wobjdata wobj2 :=[ FALSE, TRUE, "", [ [300, 600, 200], [1, 0,
     0 ,0] ], [ [0, 200, 30], [1, 0, 0 ,0] ] ];
```

使用以下值来描述上图中的工件：

- 机械臂未夹持着工件。
- 使用固定的用户坐标系。
- 用户坐标系不旋转，且其在世界坐标系中的原点坐标为x=300、y=600和z=200 mm。
- 目标坐标系不旋转，且其在用户坐标系中的原点坐标为x=0、y=200和z=30 mm。

```
wobj2.oframe.trans.z := 38.3;
```

- 将工件wobj2的位置调整为沿z方向38.3 mm处。

下一页继续

限制

应将工件数据定义为一个永久变量（PERS），且不得在程序内进行定义。随后，在保存程序时保存当前值，并在有载时恢复当前值。

任意运动指令中的工件数据类参数应仅为一个完整的永久数据对象（而非数组元素或记录成分）。

预定义数据

定义工件数据wobj0，以便目标坐标系符合世界坐标系。机械臂未夹持工件。

始终可从程序访问Wobj0，但是不得进行改变（将其储存在系统程序模块BASE中）。

```
PERS wobjdata wobj0 := [ FALSE, TRUE, "", [ [0, 0, 0], [1, 0, 0
     ,0] ], [ [0, 0, 0], [1, 0, 0 ,0] ] ];
```

结构

```
< dataobject of wobjdata >
  < robhold of bool >
  < ufprog of bool >
  < ufmec of string >
  < uframe of pose >
    < trans of pos >
      < x of num >
      < y of num >
      < z of num >
    < rot of orient >
      < q1 of num >
      < q2 of num >
      < q3 of num >
      < q4 of num >
  < oframe of pose >
    < trans of pos >
      < x of num >
      < y of num >
      < z of num >
    < rot of orient >
      < q1 of num >
      < q2 of num >
      < q3 of num >
      < q4 of num >
```

相关信息

信息，关于	请参阅
定位器指令	技术参考手册 - *RAPID*语言概览，*RAPID*概要 - 运动一节
坐标系	技术参考手册 - *RAPID*语言概览，运动和*I/O*原则 - 坐标系一节
协调的外轴	技术参考手册 - *RAPID*语言概览，运动和*I/O*原则 - 坐标系一节

下一页继续

3.98 wobjdata - 工件数据
RobotWare - OS
续前页

信息，关于	请参阅
协调轴的校准	应用手册 - *Additional axes and stand alone controller* 应用手册 - *MultiMove*

3.99 wzstationary - 固定式全局区域数据

手册用法

wzstationary（*world zone stationary*）用于识别固定全局区域，且仅可用于同事件POWER ON相连的事件程序中。

在程序执行期间和点动期间监控机器人移动时的一个全局区域。如果相关机器人的TCP触及该全局区域，或机械臂/外轴触及了关节上的全局区域，那么就停止相关移动，并设置或重置一个数字信号输出信号。

描述

通过WZLimSup或WZDOSet指令，定义和激活wzstationary全局区域。

WZLimSup或WZDOSet给予数据类型wzstationary的变量或永久数据对象一个数值。该值确定全局区域。

在电动机开启的状态下，固定全局区域始终有效，且仅通过重启来擦除。不可能通过RAPID指令来禁用、启用或擦除固定全局区域。

应当由通电来启用固定全局区域，并且应当在POWER ON事件程序或半静态任务中进行定义。

基本示例

以下示例介绍了数据类型wzstationary：

例 1

```
VAR wzstationary conveyor;
...
PROC ...
  VAR shapedata volume;
  ...
  WZBoxDef \Inside, volume, p_corner1, p_corner2;
  WZLimSup \Stat, conveyor, volume;
ENDPROC
```

将conveyor定义为一个方盒（皮带下方的体积）。如果机械臂达到此体积，则停止移动。

限制

可将wzstationary 数据定义为一个变量（**VAR**）或一个永久数据对象（**PERS**）。其可以位于全局任务中或位于模块内，但是不得位于程序内。

wzstationary类参数应仅为整体数据集（而非数组元素或记录成分）。

控制系统未使用wzstationary 类数据的初始值。当需要使用多任务系统中的永久变量时，在上述两个任务中，将初始值设置为0，例如PERS wzstationary share_workarea := [0];

更多示例

有关一个完整的例子，请参见指令WZLimSup。

特征

wzstationary为wztemporary的一种别名数据类型，并继承其特征。

下一页继续

3.99 wzstationary - 固定式全局区域数据
World Zones
续前页

相关信息

信息，关于	请参阅
全局区域	技术参考手册 - *RAPID*语言概览，运动和*I/O*原则 - 全局区域一节
全局区域形状	第*1473*页的*shapedata* - 全局区域形状数据
临时全局区域	第*1529*页的*wztemporary* - 临时全局区域数据
启用全局区域限制监控	第*949*页的*WZLimSup* - 启用全局区域限制监控
启用全局区域数字信号输出设置	第*935*页的*WZDOSet* - 启用全局区域，设置数字信号输出

3.100 wztemporary - 临时全局区域数据

手册用法

wztemporary（*world zone temporary*）用于识别一个临时的全局区域，并可用于有关任意运动任务的RAPID程序中。

在程序执行期间和点动期间监控机器人移动时的一个全局区域。如果相关机器人的TCP触及该全局区域，或机械臂/外轴触及了关节上的全局区域，那么就停止相关移动，并设置或重置一个数字信号输出信号。

描述

通过WZLimSup或WZDOSet指令，定义和激活wztemporary全局区域。

WZLimSup或WZDOSet给予数据类型wztemporary的变量或永久数据对象一个数值。该值确定全局区域。

一旦定义和启用，临时的全局区域可通过WZDisable禁用，通过WZEnable再次启用以及通过WZFree擦除。

自动擦除运动任务中的所有临时全局区域，并将运动任务中的所有wztemporary类数据对象设置为0。

- 在motion 任务中加载新程序时
- 从motion任务起点开始程序执行时

基本示例

以下示例介绍了数据类型wztemporary：

例 1

```
VAR wztemporary roll;
...
PROC
  VAR shapedata volume;
  CONST pos t_center := [1000, 1000, 1000];
  ...
  WZCylDef \Inside, volume, t_center, 400, 1000;
  WZLimSup \Temp, roll, volume;
ENDPROC
```

将wztemporary变量、roll定义为一个圆柱体。如果机械臂达到此体积，则停止移动。

限制

可将wztemporary数据定义为一个变量（VAR）或一个永久数据对象（PERS）。其可以成为任务中的全局数据或模块内的局部数据，但是不得为模块内的局部数据。

wztemporary类参数必须为整体数据集而非数组元素或记录成分。

必须在运动任务中将临时全局区域定义为（WZLimSup或WZDOSet）以及自由（WZFree）。在任意背景下，全局区域的定义均未获得认可，因为其将影响相关运动任务中的程序执行。可将指令WZDisable 和WZEnable 用于背景任务。当需要在多任务系统中使用永久变量时，在上述两种任务中，均将初始值设置为0，例如，PERS wztemporary share_workarea := [0] ;

下一页继续

3.100 wztemporary - 临时全局区域数据

RobotWare - OS

续前页

更多示例

有关一个完整的例子，请参见指令WZDOSet。

结构

```
< dataobject of wztemporary >
  < wz of num >
```

相关信息

信息，关于	请参阅
全局区域	技术参考手册 - *RAPID*语言概览，运动和*I/O*原则 - 全局区域一节
全局区域形状	第*1473*页的*shapedata* - 全局区域形状数据
固定式全局区域	第*1527*页的*wzstationary* - 固定式全局区域数据
启用全局区域限制监控	第*949*页的*WZLimSup* - 启用全局区域限制监控
启用全局区域数字信号输出设置	第*935*页的*WZDOSet* - 启用全局区域，设置数字信号输出
停用全局区域	第*933*页的*WZDisable* - 停用临时全局区域监控
启用全局区域	第*939*页的*WZEnable* - 启用临时全局区域监控
擦除全局区域	第*941*页的*WZFree* - 擦除临时全局区域监控

3.101 zonedata - 区域数据

手册用法

zonedata用于规定如何结束一个位置，即在朝下一个位置移动之前，轴必须如何接近编程位置。

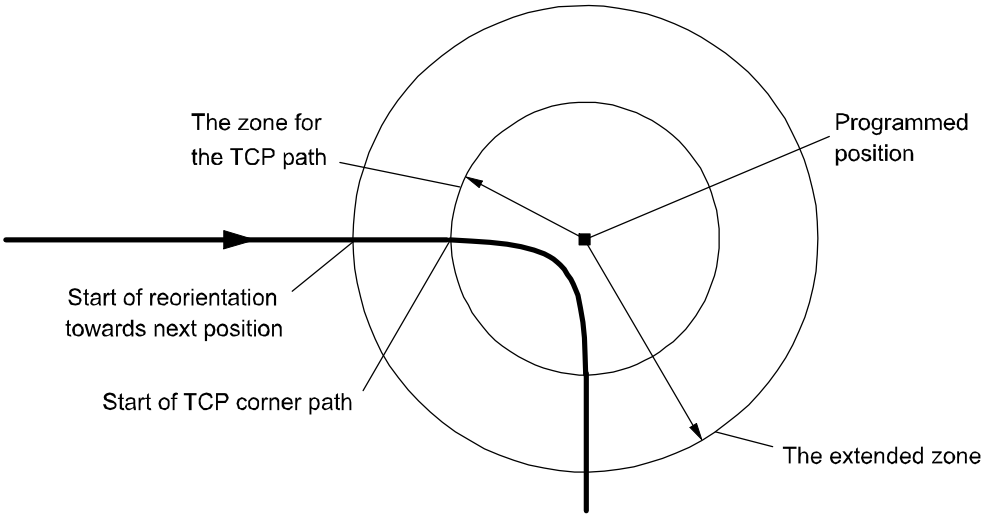
描述

可以以停止点或飞越点的形式来终止一个位置。

停止点意味着机械臂和外轴必须在使用下一个指令来继续程序执行之前达到指定位置（静止不动）。同时可能定义除预定义fine以外的停止点。可通过使用stoppointdata来操作停止标准，该停止标准说明是否认为机械臂已达到有关点。

飞越点意味着从未达到编程位置，而是在达到该位置之前改变运动方向。可针对各位置定义两个不同的区域（范围）：

- TCP路径区域。
- 有关工具重新定位和外轴的扩展区。



xx0500002357

接头移动期间，区域函数相同，但是区域半径可能与编程区域有所不同。

区域半径不得大于距最近位置（向前或向后）距离的一半。如果指定了较大的区域，则机械臂会自动缩小该区域。

TCP路径区域

一旦达到区域边缘，随即产生角路径（抛物线）（参见上图）。

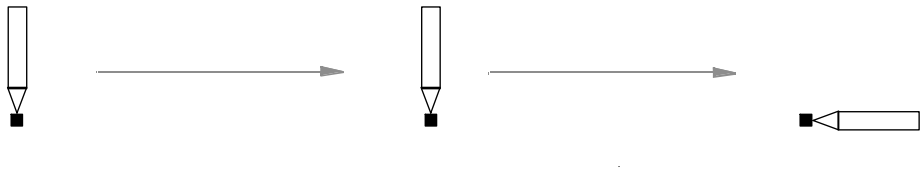
下一页继续

3.101 zonedata - 区域数据
RobotWare - OS
续前页

有关工具重新定位的区域

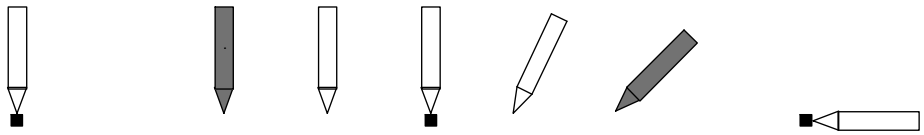
一旦TCP达到扩展区域，随即开始重新定位。重新定位工具，以便方位相同，使得区域在停止点编程完毕时位于相同位置。如果区域半径有所增加，则重新定位将更为顺利，且降低速率以实施重新定位的风险会变低。

下图显示了三处编程位置，最后一处具有不同的工具方位。



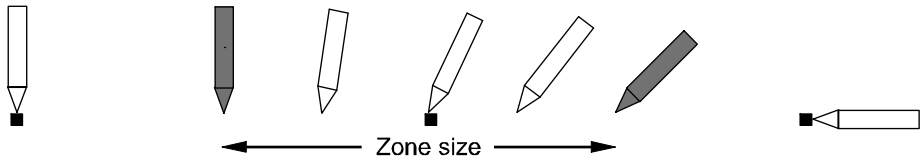
xx0500002358

下图显示了所有位置均为停止点时的程序执行情况。



xx0500002359

下图显示了中间位置为飞越点时的程序执行情况。



xx0500002360

外轴区域

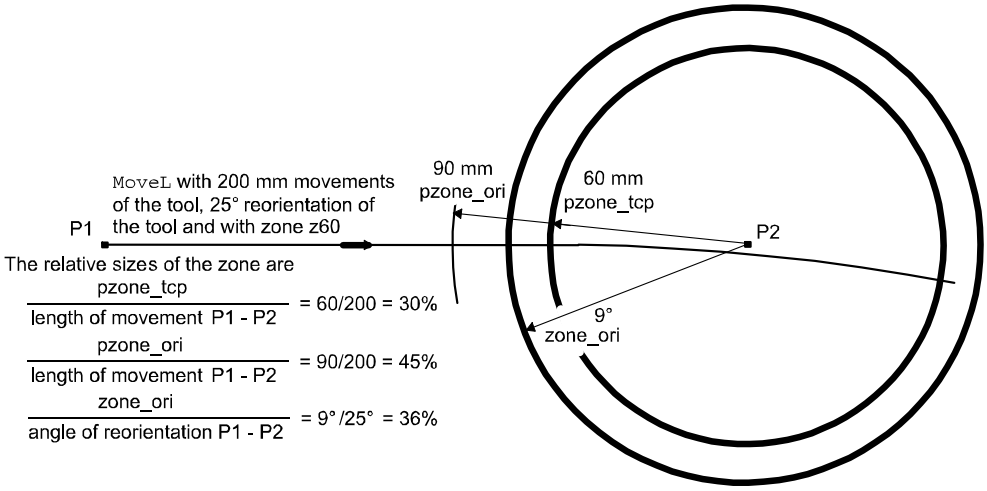
一旦TCP达到扩展区域，外轴随即开始朝下一个位置移动。在这种情况下，慢轴可在早期阶段开始加速，并由此更为顺利地执行。

下一页继续

缩小区域

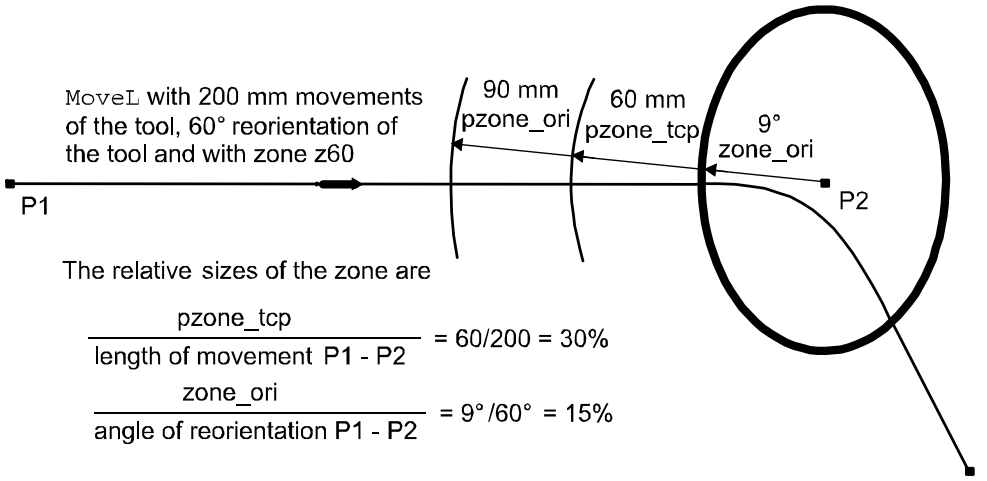
通过工具的大幅度重新定位或外轴的长距离移动，可由机械臂来缩小扩展区域甚至TCP区域。根据区域组件（参见第*1534*页的组件）和编程移动，将区域定义为区域的最小相对尺寸。

下图显示了缩小区域的例子，以便工具重新定位为zone_ori所致运动的36%。



xx0500002362

下图显示了缩小区域的例子，以便工具和TCP路径重新定位为zone_ori所致运动的15%。



xx0500002363

下一页继续

3.101 zonedata - 区域数据
RobotWare - OS
续前页

当外轴启用时，其会按照此类公式，对区域的相对尺寸造成影响：

$$\frac{\text{pzone\_eax}}{\text{length of movement P1 - P2}}$$

$$\frac{\text{zone\_leax}}{\text{length of max linear ext. axis movement P1 - P2}}$$

$$\frac{\text{zone\_reax}}{\text{angle of max reorientation of rotating ext. axis P1 - P2}}$$

xx0500002364

注意

如果TCP区域由于zone_ori、zone_leax或zone_reax而缩小，则路径规划器进入能够处理无TCP运动的模式。如果在此模式下存在TCP运动，则不会针对角区域中的路径曲率而对速度进行补偿。例如，这将导致90度角中的速度降低30%。如果这是一个问题，则增加限制区域组件。

组件

finep

fine point

数据类型：bool

规定运动是否随着停止点（fine点）或飞越点而结束。

- TRUE：运动随停止点而结束，且程序执行将不再继续，直至机械臂达到停止点。未使用区域数据中的剩余组件。
- FALSE：运动随飞越点而结束，且程序执行在机械臂达到有关区域之前继续进行大约100 ms。

pzone_tcp

path zone TCP

数据类型：num

TCP区域的尺寸（半径），以mm计。

根据以下组件pzone_ori...zone_reax 和编程运动，将扩展区域定义为区域的最小相对尺寸。

pzone_ori

path zone orientation

数据类型：num

有关工具重新定位的区域半径。将半径定义为TCP距编程点的距离，以mm计。

尺寸必须大于有关pzone_tcp的对应值。如果规定更低的值，则尺寸自动增加，以使其与pzone_tcp相同。

pzone_eax

path zone external axes

数据类型：num

下一页继续

有关外轴的区域半径。将半径定义为TCP距编程点的距离，以mm计。

尺寸必须大于有关pzone_tcp的对应值。如果规定更低的值，则尺寸自动增加，以使其与pzone_tcp相同。

zone_ori

zone orientation

数据类型：num

有关工具重新定位的区域半径，以度计。如果机械臂正夹持着工件，则意味着有关工件的旋转角。

zone_leax

zone linear external axes

数据类型：num

有关线性外轴的区域半径，以mm计。

zone_reax

zone rotational external axes

数据类型：num

有关旋转外轴的区域半径，以度计。

基本示例

以下示例介绍了数据类型zonedata：

例 1

```
VAR zonedata path := [ FALSE, 25, 40, 40, 10, 35, 5 ];
```

通过以下特征，定义区域数据path：

- TCP路径的区域半径为25 mm。
- 工具重新定位的区域半径为40 mm（TCP运动）。
- 外轴的区域半径为40 mm（TCP运动）。

如果TCP静止不动，或存在大幅度重新定位，或存在有关该区域的外轴大幅度运动，则应用以下规定：

- 工具重新定位的区域半径为10度。
- 线性外轴的区域半径为35 mm。
- 旋转外轴的区域半径为5度。

```
path.pzone_tcp := 40;
```

将TCP路径的区域半径调整为40 mm。

预定义数据

已经在系统中定义了一系列区域数据。

停止点

使用zonedata命名的fine。

下一页继续

3.101 zonedata - 区域数据

RobotWare - OS

续前页

飞越点

路径区域				Zone		
名称	TCP路径	方向	外轴	方向	线性轴	旋转轴
z0	0.3 mm	0.3 mm	0.3 mm	0.03°	0.3 mm	0.03°
z1	1 mm	1 mm	1 mm	0.1°	1 mm	0.1°
z5	5 mm	8 mm	8 mm	0.8°	8 mm	0.8°
z10	10 mm	15 mm	15 mm	1.5°	15 mm	1.5°
z15	15 mm	23 mm	23 mm	2.3°	23 mm	2.3°
z20	20 mm	30 mm	30 mm	3.0°	30 mm	3.0°
z30	30 mm	45 mm	45 mm	4.5°	45 mm	4.5°
z40	40 mm	60 mm	60 mm	6.0°	60 mm	6.0°
z50	50 mm	75 mm	75 mm	7.5°	75 mm	7.5°
z60	60 mm	90 mm	90 mm	9.0°	90 mm	9.0°
z80	80 mm	120 mm	120 mm	12°	120 mm	12°
z100	100 mm	150 mm	150 mm	15°	150 mm	15°
z150	150 mm	225 mm	225 mm	23°	225 mm	23°
z200	200 mm	300 mm	300 mm	30°	300 mm	30°

结构

```
< data object of zonedata >
  < finep of bool >
  < pzone_tcp of num >
  < pzone_ori of num >
  < pzone_eax of num >
  < zone_ori of num >
  < zone_leax of num >
  < zone_reax of num >
```

相关信息

信息，关于	请参阅
定位器指令	技术参考手册 - *RAPID*语言概览，*RAPID*概要 - 运动一节
一般移动/路径	技术参考手册 - *RAPID*语言概览，运动和*I/O*原则 - 程序执行期间的定位一节
外轴的配置	应用手册 - *Additional axes and stand alone controller*
其他停止点	第*1483*页的*stoppointdata* - 停止点数据

4 编程类型实例

4.1 关于运动的错误处理器

手册用法

此类例子描述如何在异步提升过程或运动错误之后，于错误处理器中使用运动指令。

本函数仅可用于主任务T_ROB1，或者如果在MultiMove系统中，则可用于运动任务中。

描述

ERROR 处理器可开始新的临时运动，并最终重启原来的中断和停止运动。例如，其可用于在异步提升过程或运动错误之后，转至服务位置或清洁焊枪。

为实现此功能，必须在ERROR处理器中使用指令StorePath - RestoPath。为重启运动和继续程序执行，可使用多个RAPID指令。

类型实例

功能的类型实例阐述如下。

准则

```
...
ERROR
  IF ERRNO = ERR_PATH_STOP THEN
    StorePath;
    ! Move away and back to the interrupted position
    ...
    RestoPath;
    StartMoveRetry;
  ENDIF
ENDPROC
```

执行StartMoveRetry时，机械臂恢复其运动，重启所有有效过程，并再次尝试程序执行。在一次不可分割的操作中，StartMoveRetry与StartMove和RETRY相同。

自动重启执行

```
CONST robtarget service_pos := [...];
VAR robtarget stop_pos;
...
ERROR
  IF ERRNO = AW_WELD_ERR THEN
    ! Current movement on motion base path level
    ! is already stopped.
    ! New motion path level for new movements in the ERROR handler
    StorePath;
    ! Store current position from motion base path level
    stop_pos := CRobT(\Tool:=tool1, \WObj:=wobj1);
    ! Do the work to fix the problem
    MoveJ service_pos, v50, fine, tool1, \WObj:=wobj1;
    ...
    ! Move back to the position on the motion base path level
    MoveJ stop_pos, v50, fine, tool1, \WObj:=wobj1;
```

下一页继续

4.1 关于运动的错误处理器

Path Recovery

续前页

```
    ! Go back to motion base path level
    RestoPath;
    ! Restart the stopped movements on motion base path level,
    ! restart the process and retry program execution
    StartMoveRetry;
  ENDIF
ENDPROC
```

此为如何在机械臂运动期间的某类过程错误之后使用自动异步错误恢复的类型实例。

手动重启执行

```
...
ERROR
  IF ERRNO = PROC_ERR_XXX THEN
    ! Current movement on motion base path level
    ! is already stopped and in stop move state.
    ! This error must be handle manually.
    ! Reset the stop move state on motion base path level.
    StopMoveReset;
  ENDIF
ENDPROC
```

此为如何在机械臂运动期间的某类过程错误之后手动恢复异步错误的类型实例。

在上述ERROR之后，处理器已执行到最后，程序执行停止且程序指针位于过程错误指令开头（同时位于所用NOSTEPIN程序开头）。下一个程序开始从出现原始过程错误的位置重启程序和运动。

程序执行

执行行为：

- 在ERROR处理器开始执行时，程序离开其基础执行等级
- 在StorePath执行时，运动系统离开其基础执行等级
- 在RestoPath执行时，运动系统返回其基础执行等级
- 在StartMoveRetry执行时，程序返回至其基础执行等级

限制

必须在含移动指令的ERROR处理器中使用以下RAPID指令，以便在异步提升过程或路径错误之后自动进行错误恢复：

指令	描述
StorePath	输入新的运动路径等级
RestoPath	返回至运动基路径等级
StartMoveRetry	重启运动基路径等级上已中断的运动。同时重启有关过程，并重新尝试程序执行。 与StartMove+RETRY的功能相同。

必须在ERROR处理器中使用以下RAPID指令，以便在异步提升过程或路径错误之后手动进行错误恢复：

指令	描述
StopMoveReset	输入新的运动路径等级

下一页继续

相关信息

信息，关于	请参阅
输入新的运动路径等级	第*695*页的*StorePath* - 发生中断时，存储路径
返回至运动基路径等级	第*512*页的*RestoPath* - 中断之后，恢复路径
重启已中断的运动、过程并重新尝试程序执行。	第*665*页的*StartMoveRetry* - 重启机械臂移动和执行

4.2 包含运动或不包含运动的服务程序

手册用法

此类实例描述如何在服务程序中使用运动指令。有关于StopMove、StartMove和StopMoveReset的相同原则亦适用于不含运动的服务程序（仅逻辑指令）。

可手动启动不含参数的服务程序或其他程序（无返回值程序），并按照此类实例来执行运动。

本功能仅可用于主任务T_ROB1，或者如果在MultiMove系统中，则可以以独立或半协调模式用于运动任务中。

描述

服务程序可启动新的临时运动，并在随后的程序启动中重启原始运动。例如，其可以用于转至服务位置，或开始手动清洁焊枪。

为实现此功能，必须将指令StorePath - RestoPath和StopMoveReset用于服务程序。

类型实例

功能的类型实例阐述如下。

准则

```
PROC xxxx()
  StopMove;
  StorePath;
  ! Move away and back to the interrupted position
  ...
  RestoPath;
  StopMoveReset;
ENDPROC
```

服务程序执行期间，需要StopMove，从而确保未根据手动“停止程序-重启程序”顺序而重启最初停止的运动。

在路径上停止

```
VAR robtarget service_pos := [...];
...
PROC proc_stop_on_path()
  VAR robtarget stop_pos;
  ! Current stopped movements on motion base path level
  ! must not be restarted in the service routine.
  StopMove;
  ! New motion path level for new movements in the service routine.
  StorePath;
  ! Store current position from motion base path level
  stop_pos := CRobT(\Tool:=tool1 \WObj:=wobj1);
  ! Do the work
  MoveJ service_pos, v50, fine, tool1 \WObj:=wobj1;
  ...
  ! Move back to interrupted position on the motion base path level
  MoveJ stop_pos, v50, fine, tool1, \WObj:=wobj1;
  ! Go back to motion base path level
```

下一页继续

```
  RestoPath;
  ! Reset the stop move state for the interrupted movement
  ! on motion base path level
  StopMoveReset;
ENDPROC
```

在此类型实例中，服务程序中的运动起止于程序在路径上停止的位置。

同时注意，在编程时，已知晓所用工具和工件。

在下一个停止点停止

```
TASK PERS tooldata used_tool := [...];
TASK PERS wobjdata used_wobj := [...];
...
PROC proc_stop_in_stop_point()
  VAR robtarget stop_pos;
  ! Current move instruction on motion base path level continue to
       its ToPoint and will be finished in a stop point.
  StartMove;
  ! New motion path level for new movements in the service routine
  StorePath;
  ! Get current tool and work object data
  GetSysData used_tool;
  GetSysData used_wobj;
  ! Store current position from motion base path level
  stop_pos := CRobT(\Tool:=used_tool \WObj:=used_wobj);
  ! Do the work
  MoveJ Offs(stop_pos,0,0,20),v50,fine,used_tool\WObj:=used_wobj;
  ...
  ! Move back to interrupted position on the motion base path level
  MoveJ stop_pos, v50, fine, used_tool,\WObj:=used_wobj;
  ! Go back to motion base path level
  RestoPath;
  ! Reset the stop move state for any new movement
  ! on motion base path level
  StopMoveReset;
ENDPROC
```

在本类型实例中，服务程序中的运动继续进行，并在指令 StorePath就绪之前的中断移动指令中的ToPoint处结束。

同时注意，在编程时，未知晓所用工具和工件。

程序执行

执行行为：

- 在服务程序开始执行时，程序离开其基础执行等级
- 在StorePath执行时，运动系统离开其基础执行等级
- 在RestoPath执行时，运动系统返回其基础执行等级
- 在ENDPROC执行时，程序返回至其基础执行等级

下一页继续

续前页

限制

必须通过移动指令，在服务程序中使用以下RAPID指令，以使其奏效：

指令	描述
StorePath	输入新的运动路径等级
RestoPath	返回至运动基路径等级
StopMoveReset	重置有关运动基路径等级上已中断运动的停止移动状态

相关信息

信息，关于	请参阅
未重启运动基路径等级上已停止的运动	第*690*页的*StopMove* - 停止机械臂的移动
重启运动基路径等级上已停止的运动	第*690*页的*StopMove* - 停止机械臂的移动
输入新的运动路径等级	第*695*页的*StorePath* - 发生中断时，存储路径
返回至运动基路径等级	第*512*页的*RestoPath* - 中断之后，恢复路径
重置有关运动基路径等级上已中断运动的停止移动状态	第*693*页的*StopMoveReset* - 重置系统停止移动状态

4.3 包含运动或不包含运动的系统I/O中断

手册用法

此类实例描述如何在系统I/O中断程序中使用运动指令。有关于StopMove、StartMove和StopMoveReset的相同原则亦适用于不含运动的系统I/O中断（仅逻辑指令）。

本功能仅可用于主任务T_ROB1，或者如果在MultiMove系统中，则可以以独立或半协调模式用于运动任务中。

描述

系统I/O中断程序可启动新的临时运动，并在随后的程序启动中重启原始运动。例如，其可以用于转至服务位置，或在出现中断时清洁焊枪。

为实现此功能，必须将指令StorePath - RestoPath和StopMoveReset用于系统I/O中断程序。

类型实例

功能的类型实例阐述如下。

准则

```
PROC xxxx()
  StopMove;
  StorePath;
  ! Move away and back to the interrupted position
  ...
  RestoPath;
  StopMoveReset;
ENDPROC
```

需要StopMove ，从而确保在启动I/O中断程序时，未重启最初停止的运动。

不包含StopMove或包含StartMove，I/O中断程序中的运动将立即继续，并于中断移动指令中的ToPoint处结束。

在路径上停止

```
VAR robtarget service_pos := [...];
...
PROC proc_stop_on_path()
  VAR robtarget stop_pos;
  ! Current stopped movements on motion base path level is nnt
       restarted in the system I/O routine.
  StopMove \Quick;
  ! New motion path level for new movements in the system
  ! I/O routine.
  StorePath;
  ! Store current position from motion base path level
  stop_pos := CRobT(\Tool:=tool1 \WObj:=wobj1);
  ! Do the work
  MoveJ service_pos, v50, fine, tool1 \WObj:=wobj1;
  ...
  ! Move back to interrupted position on the motion base path level
  MoveJ stop_pos, v50, fine, tool1, \WObj:=wobj1;
  ! Go back to motion base path level
```

下一页继续

```
  RestoPath;
  ! Reset the stop move state for the interrupted movement
  ! on motion base path level
  StopMoveReset;
ENDPROC
```

在本类实例中，已中断运动立即停止，并在完成系统I/O中断程序后的程序启动时重启。

同时注意，在编程时，已知晓所用工具和工件。

在下一个停止点停止

```
TASK PERS tooldata used_tool := [...];
TASK PERS wobjdata used_wobj := [...];
...
PROC proc_stop_in_stop_point()
  VAR robtarget stop_pos;
  ! Current move instruction on motion base path level continue to
       its ToPoint and will be finished in a stop point.
  StartMove;
  ! New motion path level for new movements in the system
  ! I/O routine
  StorePath;
  ! Get current tool and work object data
  GetSysData used_tool;
  GetSysData used_wobj;
  ! Store current position from motion base path level
  stop_pos := CRobT(\Tool:=used_tool \WObj:=used_wobj);
  ! Do the work
  MoveJ Offs(stop_pos,0,0,20),v50,fine,used_tool\WObj:=used_wobj;
  ...
  ! Move back to interrupted position on the motion base path level
  MoveJ stop_pos, v50, fine, used_tool,\WObj:=used_wobj;
  ! Go back to motion base path level
  RestoPath;
  ! Reset the stop move state for new movement
  ! on motion base path level
  StopMoveReset;
ENDPROC
```

在本类实例中，立即继续系统I/O程序中的运动，并在中断移动指令中的ToPoint处结束。

同时注意，在编程时，未知晓所用工具和工件。

程序执行

执行行为：

- 在系统I/O程序开始执行时，程序离开其基础执行等级
- 在StorePath执行时，运动系统离开其基础执行等级
- 在RestoPath执行时，运动系统返回其基础执行等级
- 在ENDPROC执行时，程序返回至其基础执行等级

下一页继续

限制

必须通过移动指令，在系统I/O程序中使用以下RAPID指令，以使其奏效：

指令	描述
StorePath	输入新的运动路径等级
RestoPath	返回至运动基路径等级
StopMoveReset	重置有关运动基路径等级上已中断运动的停止移动状态

相关信息

信息，关于	请参阅
未重启运动基路径等级上已停止的运动	第*690*页的*StopMove - 停止机械臂的移动*
重启运动基路径等级上已停止的运动	第*663*页的*StartMove - 重启机械臂移动*
输入新的运动路径等级	第*695*页的*StorePath - 发生中断时，存储路径*
返回至运动基路径等级	第*512*页的*RestoPath - 中断之后，恢复路径*
重置有关运动基路径等级上已中断运动的停止移动状态	第*693*页的*StopMoveReset - 重置系统停止移动状态*

4.4 关于运动的软中断程序

手册用法

此类实例描述如何在出现中断后，于TRAP程序中使用移动指令。

本功能仅可用于主任务T_ROB1，或者如果在MultiMove系统中，则可用于运动任务中。

描述

TRAP程序可启动新的临时运动，并最终重启原始运动。例如，在出现中断时，其可以用于转至服务位置，或者清洁焊枪。

为实现此功能，必须将指令StorePath - RestoPath和StartMove 用于TRAP程序。

类型实例

功能的类型实例阐述如下。

准则

```
TRAP xxxx
  StopMove;
  StorePath;
  ! Move away and back to the interrupted position
  ...
  RestoPath;
  StartMove;
ENDTRAP
```

如果使用 StopMove，则立即在持续路径上停止运动；否则，继续运动至实际移动指令中的ToPoint处。

在下一个停止点停止

```
VAR robtarget service_pos := [...];
...
TRAP trap_in_stop_point
  VAR robtarget stop_pos;
  ! Current move instruction on motion base path level continue
  ! to it’s ToPoint and will be finished in a stop point.
  ! New motion path level for new movements in the TRAP
  StorePath;
  ! Store current position from motion base path level
  stop_pos := CRobT(\Tool:=tool1 \WObj:=wobj1);
  ! Do the work
  MoveJ service_pos, v50, fine, tool1 \WObj:=wobj1;
  ...
  ! Move back to interrupted position on the motion base path level
  MoveJ stop_pos, v50, fine, tool1, \WObj:=wobj1;
  ! Go back to motion base path level
  RestoPath;
  ! Restart the interupted movements on motion base path level
  StartMove;
ENDTRAP
```

下一页继续

在本类实例中，软中断程序中的运动起止于中断移动指令中的ToPoint。同时注意，在编程时已知晓工具和工件。

立即停止在路径上

```
TASK PERS tooldata used_tool := [...];
TASK PERS wobjdata used_wobj := [...];
...
TRAP trap_stop_at_once
  VAR robtarget stop_pos;
  ! Current move instruction on motion base path level stops
  ! at once
  StopMove;
  ! New motion path level for new movements in the TRAP
  StorePath;
  ! Get current tool and work object data
  GetSysData used_tool;
  GetSysData used_wobj;
  ! Store current position from motion base path level
  stop_pos := CRobT(\Tool:=used_tool \WObj:=used_wobj);
  ! Do the work
  MoveJ Offs(stop_pos,0,0,20),v50,fine,used_tool\WObj:=used_wobj;
  ...
  ! Move back to interrupted position on the motion base path level
  MoveJ stop_pos, v50, fine, used_tool,\WObj:=used_wobj;
  ! Go back to motion base path level
  RestoPath;
  ! Restart the interupted movements on motion base path level
  StartMove;
ENDTRAP
```

在本类实例中，软中断程序中的运动起止于路径上中断移动指令停止的位置。同时注意，在编程时，尚未知晓所用工具和工件。

程序执行

执行行为：

- 在TRAP程序开始执行时，程序离开其基础执行等级
- 在StorePath执行时，运动系统离开其基础执行等级
- 在RestoPath执行时，运动系统返回其基础执行等级
- 在ENDTRAP执行时，程序返回至其基础执行等级

限制

必须通过移动指令，在TRAP程序中使用以下RAPID指令，以使其奏效：

指令	描述
StorePath	输入新的运动路径等级
RestoPath	返回至运动基路径等级
StartMove	重启运动基路径等级上的中断运动

下一页继续

续前页

相关信息

信息，关于	请参阅
立即停止当前运动	第*690*页的*StopMove* - 停止机械臂的移动
输入新的运动路径等级	第*695*页的*StorePath* - 发生中断时，存储路径
返回至运动基路径等级	第*512*页的*RestoPath* - 中断之后，恢复路径
重启中断的运动	第*663*页的*StartMove* - 重启机械臂移动

索引

Contact us

ABB AB
Discrete Automation and Motion
Robotics
S-721 68 VÄSTERÅS, Sweden
Telephone +46 (0) 21 344 400

ABB AS, Robotics
Discrete Automation and Motion
Nordlysvegen 7, N-4340 BRYNE, Norway
Box 265, N-4349 BRYNE, Norway
Telephone: +47 51489000

ABB Engineering (Shanghai) Ltd.
No. 4528 Kangxin Hingway
PuDong District
SHANGHAI 201319, China
Telephone: +86 21 6105 6666

www.abb.com/robotics

3HAC050917-010, Rev C, zh-cn