

ROBOTICS

# Manual de referencia técnica

Instrucciones, funciones y tipos de datos de RAPID



Trace back information:  
Workspace 21A version a10  
Checked in 2021-03-16  
Skribenta version 5.4.005

**Manual de referencia técnica**  
**Instrucciones, funciones y tipos de datos de RAPID**

RobotWare 6.12

ID de documento: 3HAC050917-005

Revisión: N

La información de este manual puede cambiar sin previo aviso y no puede entenderse como un compromiso por parte de ABB. ABB no se hace responsable de ningún error que pueda aparecer en este manual.

Excepto en los casos en que se indica expresamente en este manual, ninguna parte del mismo debe entenderse como una garantía por parte de ABB por las pérdidas, lesiones, daños materiales, idoneidad para un fin determinado ni garantías similares.

ABB no será en ningún caso responsable de los daños accidentales o consecuentes que se produzcan como consecuencia del uso de este manual o de los productos descritos en el mismo.

Se prohíbe la reproducción o la copia de este manual o cualquiera de sus partes si no se cuenta con una autorización escrita de ABB.

Guardar para futuras referencias.

Usted puede obtener copias adicionales de este manual a través de ABB.

Traducción del manual original.

© Copyright 2004-2021 ABB. Reservados todos los derechos.  
Las especificaciones están sujetas a cambios sin previo aviso.



# Contenido

|  |           |
|--|-----------|
| Descripción general de este manual .....   | 19        |
| <b>1 Instrucciones</b> .....   | <b>25</b> |
| 1.1 AccSet - Reduce la aceleración .....   | 25        |
| 1.2 ActEventBuffer - Activación de búfer de eventos .....  | 28        |
| 1.3 ActUnit - Activa una unidad mecánica .....   | 30        |
| 1.4 Add - Suma un valor numérico .....   | 32        |
| 1.5 AliasCamera - Define un dispositivo de cámara con un nombre de alias .....                               | 34        |
| 1.6 AliasIO - Define una señal de E/S con un nombre de alias .....   | 37        |
| 1.7 AliasIOReset - Restablecer una señal de E/S con un nombre de alias .....                                 | 40        |
| 1.8 ":= " - Asigna un valor .....  | 42        |
| 1.9 BitClear - Desactiva un bit específico de un dato byte o dnum .....                                      | 44        |
| 1.10 BitSet - Activa un bit específico de un dato byte o dnum .....  | 47        |
| 1.11 BookErrNo - Registra un número de error de sistema de RAPID .....                                       | 50        |
| 1.12 Break - Interrumpe la ejecución del programa .....  | 52        |
| 1.13 CallByVar - Llama a un procedimiento mediante una variable .....  | 53        |
| 1.14 CamFlush - Elimina los datos de colección de la cámara .....  | 55        |
| 1.15 CamGetParameter - Obtiene distintos parámetros designados de la cámara .....                            | 56        |
| 1.16 CamGetResult - Obtiene un objetivo de cámara de la colección .....                                      | 58        |
| 1.17 CamLoadJob - Carga una tarea de cámara en una cámara .....  | 60        |
| 1.18 CamReqlImage - Ordena a la cámara la adquisición de una imagen .....                                    | 62        |
| 1.19 CamSetExposure - Establece parámetros específicos de la cámara .....                                    | 64        |
| 1.20 CamSetParameter - Establece distintos parámetros designados de la cámara .....                          | 66        |
| 1.21 CamSetProgramMode - Ordena a la cámara que cambie al modo de programación .....                         | 68        |
| 1.22 CamSetRunMode - Ordena a la cámara que cambie al modo de ejecución .....                                | 70        |
| 1.23 CamStartLoadJob - Inicia la carga de una tarea de cámara en una cámara .....                            | 72        |
| 1.24 CamStartSetParameter - Iniciar la operación de ajuste de un parámetro .....                             | 74        |
| 1.25 CamWaitLoadJob - Esperar hasta que una tarea de cámara esté cargada .....                               | 77        |
| 1.26 CamWaitSetParameter - Esperar hasta que una operación de ajuste esté lista .....                        | 79        |
| 1.27 CancelLoad - Cancela la carga de un módulo .....  | 81        |
| 1.28 CapAPTrSetup - Configuración de un seguimiento de punto .....   | 83        |
| 1.29 CapAPTrSetupAI - Configurar un seguimiento del punto controlado por señales de entrada analógicas ..... | 86        |
| 1.30 CapAPTrSetupAO - Configurar un seguimiento del punto controlado por señales de salida analógicas .....  | 89        |
| 1.31 CapAPTrSetupPERS - Configurar un seguimiento del punto controlado por variables persistentes .....      | 92        |
| 1.32 CapC - Instrucción de movimiento CAP circular .....   | 95        |
| 1.33 CapCondSetDO - Activar una señal digital de salida en la parada de TCP .....                            | 105       |
| 1.34 CapEquiDist - Generar evento equidistante .....   | 107       |
| 1.35 CapL - Instrucción de movimiento CAP lineal .....   | 109       |
| 1.36 CapLATrSetup - Configurar un sensor de seguimiento anticipatorio .....                                  | 118       |
| 1.37 CapNoProcess - Ejecutar CAP sin proceso .....   | 123       |
| 1.38 CapRefresh - Actualizar datos de CAP .....  | 125       |
| 1.39 CAPSetStopMode - Establecer el modo de paro para errores de ejecución .....                             | 127       |
| 1.40 CapWeaveSync - Configurar señales y niveles para la sincronización de oscilación .....                  | 128       |
| 1.41 CheckProgRef - Comprobar referencias de programa .....  | 131       |
| 1.42 CirPathMode - Reorientación de la herramienta durante trayectorias circulares .....                     | 133       |
| 1.43 Clear - Borra el valor .....  | 139       |
| 1.44 ClearIOBuff - Vacía el búfer de entrada de un canal serie .....   | 140       |
| 1.45 ClearPath - Elimina la trayectoria actual .....   | 142       |
| 1.46 ClearRawBytes - Borra el contenido de un dato de tipo rawbytes .....                                    | 146       |
| 1.47 ClkReset - Pone a cero un reloj utilizado como temporizador .....                                       | 148       |
| 1.48 ClkStart - Pone en marcha un reloj utilizado para la temporización .....                                | 149       |
| 1.49 ClkStop - Detiene un reloj utilizado para la temporización .....  | 151       |
| 1.50 Cerrar - Cierra un archivo o dispositivo de E/S .....   | 152       |
| 1.51 CloseDir - Cierra un directorio .....   | 153       |

|       |   |     |
|-------|---|-----|
| 1.52  | Comentario - Comentario .....   | 154 |
| 1.53  | Compact IF (IF compacto) - Si se cumple una condición, entonces... (una instrucción) ....             | 155 |
| 1.54  | ConfJ - Controla la configuración durante el movimiento de los ejes .....                             | 156 |
| 1.55  | ConfL - Monitoriza la configuración durante el movimiento lineal .....                                | 158 |
| 1.56  | CONNECT - Conecta una interrupción a una rutina TRAP .....  | 161 |
| 1.57  | ContactL - Movimiento de contacto lineal .....  | 163 |
| 1.58  | CopyFile - Copia un archivo .....   | 169 |
| 1.59  | CopyRawBytes - Copia el contenido de un dato de tipo rawbytes .....                                   | 171 |
| 1.60  | CornerPathWarning - Mostrar u ocultar avisos de trayectoria de esquina .....                          | 174 |
| 1.61  | CorrClear - Elimina todos los generadores de correcciones .....                                       | 176 |
| 1.62  | CorrCon - Establece una conexión con un generador de correcciones .....                               | 177 |
| 1.63  | CorrDiscon - Cierra la conexión con un generador de correcciones .....                                | 182 |
| 1.64  | CorrWrite - Escribe en un generador de correcciones .....   | 183 |
| 1.65  | DeactEventBuffer - Desactivación de búfer de eventos .....  | 185 |
| 1.66  | DeactUnit - Desactiva una unidad mecánica .....   | 187 |
| 1.67  | Decr - Reduce un número en 1 .....  | 189 |
| 1.68  | DropSensor - Colocación de un objeto en el sensor .....   | 191 |
| 1.69  | DropWObj - Suelta un objeto de trabajo sobre un transportador .....                                   | 193 |
| 1.70  | EOffsOff - Desactiva un offset de ejes adicionales .....  | 194 |
| 1.71  | EOffsOn - Activa un offset de ejes adicionales .....  | 195 |
| 1.72  | EOffsSet - Activa un offset de ejes adicionales a partir de valores conocidos .....                   | 197 |
| 1.73  | EraseModule - Elimina un módulo .....   | 199 |
| 1.74  | ErrLog - Escribe un mensaje de error .....  | 201 |
| 1.75  | ErrRaise - Escribe un aviso y llama a un gestor de errores .....                                      | 205 |
| 1.76  | ErrWrite - Escribe un mensaje de error .....  | 210 |
| 1.77  | EXIT - Finaliza la ejecución del programa .....   | 212 |
| 1.78  | ExitCycle - Interrumpe el ciclo actual y pasa al siguiente .....                                      | 213 |
| 1.79  | FitCircle: Se ajusta a un círculo con puntos 3D .....   | 215 |
| 1.80  | FOR - Repite un número determinado de veces .....   | 219 |
| 1.81  | FricIdInit - Iniciar identificación de fricción .....   | 222 |
| 1.82  | FricIdEvaluate - Evaluar identificación de fricción .....   | 223 |
| 1.83  | FricIdSetFricLevels - Establecimiento de niveles de fricción tras la identificación de fricción ..... | 226 |
| 1.84  | GetDataVal - Obtiene el valor de un objeto de datos .....   | 228 |
| 1.85  | GetGroupSignalInfo - Leer información sobre una señal digital de grupo .....                          | 231 |
| 1.86  | GetJointData: Obtener datos conjuntos específicos .....   | 233 |
| 1.87  | GetSysData - Obtiene datos del sistema .....  | 235 |
| 1.88  | GetTorqueMargin - Lee el menor margen de par .....  | 238 |
| 1.89  | GetTrapData - Obtiene datos de interrupción para la rutina TRAP actual .....                          | 240 |
| 1.90  | GOTO - Salta a otra instrucción .....   | 242 |
| 1.91  | GripLoad - Define la carga útil de un robot .....   | 244 |
| 1.92  | HollowWristReset - Restablecer la muñeca hueca .....  | 246 |
| 1.93  | ICap - Conectar eventos CAP a rutinas TRAP .....  | 248 |
| 1.94  | IDelete - Cancela una interrupción .....  | 254 |
| 1.95  | IDisable - Desactiva todas las interrupciones .....   | 255 |
| 1.96  | IEnable - Habilita el uso de interrupciones .....   | 256 |
| 1.97  | IError - Solicita una interrupción para errores .....   | 257 |
| 1.98  | IF - Si se cumple una condición, ...; de lo contrario, ... ..   | 260 |
| 1.99  | Incr - Aumenta en 1 un valor .....  | 262 |
| 1.100 | IndAMove - Movimiento independiente de posición absoluta .....  | 264 |
| 1.101 | IndCMove - Movimiento independiente continuo .....  | 268 |
| 1.102 | IndDMove - Movimiento independiente de posición delta .....   | 272 |
| 1.103 | IndReset - Restablecimiento independiente .....   | 276 |
| 1.104 | IndRMove - Movimiento independiente de posición relativa .....  | 281 |
| 1.105 | InitSuperv - Restablecer toda la supervisión para CAP .....   | 286 |
| 1.106 | InvertDO - Invierte el valor de una señal de salida digital .....                                     | 287 |
| 1.107 | IOBusStart - Start of I/O network .....   | 289 |
| 1.108 | IOBusState - Obtener el estado actual de una red de E/S .....   | 290 |
| 1.109 | IODisable - Desactivar un dispositivo de E/S .....  | 293 |

|       |  |     |
|-------|--|-----|
| 1.110 | IOEnable - Activar un dispositivo de E/S .....   | 296 |
| 1.111 | IOEventMessage: activar/desactivar mensajes de eventos de E/S desde el dispositivo ...                               | 299 |
| 1.112 | IPathPos - Obtener valor robtarget de la línea central en la oscilación .....  | 301 |
| 1.113 | IPers - Interrupción en caso de cambio de valor de una variable persistente .....                                    | 303 |
| 1.114 | IRMQMessage - Ordenar interrupciones de RMQ para un tipo de dato .....   | 305 |
| 1.115 | ISignalAI - Interrupciones a partir de una señal analógica de entrada .....  | 309 |
| 1.116 | ISignalAO - Interrupciones a partir de una señal analógica de salida .....   | 319 |
| 1.117 | ISignalDI - Solicita interrupciones a partir de una señal digital de entrada .....                                   | 323 |
| 1.118 | ISignalDO - Interrupciones a partir de una señal digital de salida .....   | 326 |
| 1.119 | ISignalGI - Solicita interrupciones de un grupo de señales digitales de entrada .....                                | 329 |
| 1.120 | ISignalGO - Solicita interrupciones de un grupo de señales digitales de salida .....                                 | 332 |
| 1.121 | ISleep - Desactiva una interrupción .....  | 335 |
| 1.122 | ITimer - Solicita una interrupción temporizada .....   | 337 |
| 1.123 | IVarValue - Solicita una interrupción a partir del valor de una variable .....                                       | 340 |
| 1.124 | IWatch - Activar una interrupción .....  | 344 |
| 1.125 | Etiqueta - Nombre de línea .....   | 346 |
| 1.126 | Load - Carga un módulo de programa durante la ejecución .....  | 347 |
| 1.127 | LoadId - Identificación de carga de la herramienta o la carga útil .....   | 352 |
| 1.128 | MakeDir - Crea un nuevo directorio .....   | 358 |
| 1.129 | ManLoadIdProc - Identificación de carga de los manipuladores IRBP .....  | 360 |
| 1.130 | MatrixSolve - Soluciona un sistema de ecuaciones lineales .....  | 364 |
| 1.131 | MatrixSolveQR - Calcula una factorización QR .....   | 367 |
| 1.132 | MatrixSVD - Calcula una descomposición en valores singulares .....   | 369 |
| 1.133 | MechUnitLoad - Define una carga útil para una unidad mecánica .....  | 372 |
| 1.134 | MotionProcessModeSet - Configuración del modo de proceso de movimientos .....  | 377 |
| 1.135 | MotionSup - Desactiva/activa la supervisión del movimiento .....   | 379 |
| 1.136 | MoveAbsJ - Mueve el robot a una posición de ejes absoluta .....  | 382 |
| 1.137 | MoveC - Mueve el robot en círculo .....  | 389 |
| 1.138 | MoveCAO - Mueve el robot en una trayectoria circular y establece una salida analógica en la esquina .....            | 397 |
| 1.139 | MoveCDO - Mueve el robot en una trayectoria circular y establece una salida digital en la esquina .....              | 402 |
| 1.140 | MoveCGO - Mueve el robot en una trayectoria circular y establece una señal de salida de grupo en la esquina .....    | 407 |
| 1.141 | MoveCSync - Mueve el robot en una trayectoria circular y ejecuta un procedimiento de RAPID .....                     | 412 |
| 1.142 | MoveExtJ - Mueve una o varias unidades mecánicas sin TCP .....   | 417 |
| 1.143 | MoveJ - Mueve el robot mediante un movimiento de ejes .....  | 421 |
| 1.144 | MoveJAO - Mueve el robot mediante el movimiento de los ejes y activa una salida analógica en la esquina .....        | 427 |
| 1.145 | MoveJDO - Mueve el robot mediante el movimiento de los ejes y activa una salida digital en la esquina .....          | 432 |
| 1.146 | MoveJGO - Mueve el robot mediante un movimiento de ejes y establece una señal de salida de grupo en la esquina ..... | 437 |
| 1.147 | MoveJSync - Mueve el robot con un movimiento de ejes y ejecuta un procedimiento de RAPID .....                       | 442 |
| 1.148 | MoveL - Mueve el robot siguiendo una trayectoria lineal .....  | 447 |
| 1.149 | MoveLAO - Mueve el robot siguiendo una trayectoria lineal y establece una salida analógica en la esquina .....       | 453 |
| 1.150 | MoveLDO - Mueve el robot linealmente y establece una salida digital en la esquina .....                              | 458 |
| 1.151 | MoveLGO - Mueve el robot linealmente y establece una señal de salida de grupo en la esquina .....                    | 463 |
| 1.152 | MoveLSync - Mueve el robot de forma lineal y ejecuta un procedimiento de RAPID .....                                 | 468 |
| 1.153 | MovePnP: Mueve el robot a lo largo de una trayectoria de elección y colocación. ....                                 | 473 |
| 1.154 | MToolRotCalib - Calibración de la rotación de una herramienta móvil .....  | 483 |
| 1.155 | MToolTCPCalib - Calibración del TCP de una herramienta móvil .....   | 486 |
| 1.156 | Open - Abre un archivo o dispositivo de E/S .....  | 489 |
| 1.157 | OpenDir - Abre un directorio .....   | 494 |
| 1.158 | PackDNHeader - Empaqueta un encabezado de DeviceNet en datos rawbytes .....  | 496 |

|       |  |     |
|-------|--|-----|
| 1.159 | PackRawBytes - Empaqueta datos en un dato de tipo rawbytes .....                                     | 499 |
| 1.160 | PathAccLim - Reduce la aceleración del TCP a lo largo de la trayectoria .....                        | 504 |
| 1.161 | PathLengthReset - Restablece el valor de longitud de trayectoria actual del contador ....            | 508 |
| 1.162 | PathLengthStart - Activa el contador que monitoriza la longitud de trayectoria .....                 | 510 |
| 1.163 | PathLengthStop - Detiene el contador que monitoriza la longitud de trayectoria .....                 | 512 |
| 1.164 | PathRecMoveBwd - Hace retroceder la grabadora de trayectorias .....                                  | 514 |
| 1.165 | PathRecMoveFwd - Hace avanzar la grabadora de trayectorias .....                                     | 521 |
| 1.166 | PathRecStart - Inicia la grabadora de trayectorias .....   | 524 |
| 1.167 | PathRecStop - Detiene la grabadora de trayectorias .....   | 527 |
| 1.168 | PathResol - Ajusta la resolución de la trayectoria .....   | 530 |
| 1.169 | PDispOff - Desactiva el desplazamiento de programa .....   | 532 |
| 1.170 | PDispOn - Activa el desplazamiento de programa .....   | 533 |
| 1.171 | PDispSet - Activa un desplazamiento de programa a partir de una base de coordenadas conocida .....   | 538 |
| 1.172 | ProcCall - Llama a un nuevo procedimiento .....  | 541 |
| 1.173 | ProcErrRecovery - Genera errores de movimiento de proceso y permite la recuperación tras ellos ..... | 543 |
| 1.174 | PrxActivAndStoreRecord - Activación y almacenamiento de los datos de perfil grabados                 | 549 |
| 1.175 | PrxActivRecord - Activación de los datos de perfil grabados .....                                    | 551 |
| 1.176 | PrxDBgStoreRecord - Almacenamiento y depuración de los datos de perfil grabados ....                 | 553 |
| 1.177 | PrxDeactRecord - Desactivación de un registro .....  | 554 |
| 1.178 | PrxResetPos - Restablecimiento de la posición cero del sensor .....                                  | 555 |
| 1.179 | PrxResetRecords - Restablecimiento y desactivación de todos los registros .....                      | 556 |
| 1.180 | PrxSetPosOffset - Establecimiento de una posición de referencia para el sensor .....                 | 557 |
| 1.181 | PrxSetRecordSampleTime - Establecimiento del tiempo de muestreo para la grabación de un perfil ..... | 558 |
| 1.182 | PrxSetSyncalarm - Establecimiento del comportamiento de alarma de sincronización ....                | 559 |
| 1.183 | PrxStartRecord - Grabación de un nuevo perfil .....  | 561 |
| 1.184 | PrxStopRecord - Parada de la grabación de un perfil .....  | 563 |
| 1.185 | PrxStoreRecord - Almacenamiento de los datos de perfil grabados .....                                | 564 |
| 1.186 | PrxUseFileRecord - Utilización de los datos de perfil grabados .....                                 | 566 |
| 1.187 | PulseDO - Genera un pulso en una señal digital de salida .....                                       | 568 |
| 1.188 | RAISE - Llamada a un gestor de errores .....   | 571 |
| 1.189 | RaiseToUser - Propaga un error al nivel del usuario .....  | 574 |
| 1.190 | ReadAnyBin - Leer datos de un dispositivo de E/S o un archivo binario .....                          | 577 |
| 1.191 | ReadBlock - Lee un bloque de datos de un dispositivo .....   | 580 |
| 1.192 | ReadCfgData - Lee un atributo de un parámetro del sistema .....                                      | 582 |
| 1.193 | ReadErrData - Obtiene información sobre un error .....   | 586 |
| 1.194 | ReadRawBytes - Lee datos de tipo rawbytes .....  | 589 |
| 1.195 | ReadVarArr - Lee múltiples variables de un dispositivo sensor .....                                  | 592 |
| 1.196 | RemoveAllCyclicBool - Eliminar todas las condiciones de Cyclic bool .....                            | 594 |
| 1.197 | RemoveCyclicBool - Eliminar una condición de Cyclic bool .....                                       | 596 |
| 1.198 | RemoveDir - Elimina un directorio .....  | 598 |
| 1.199 | RemoveFile - Elimina un archivo .....  | 600 |
| 1.200 | RemoveSuperv - Eliminar la condición de una señal .....  | 602 |
| 1.201 | RenameFile - Cambia el nombre de un archivo .....  | 604 |
| 1.202 | Reset - Pone a cero una señal digital de salida .....  | 606 |
| 1.203 | ResetAxisDistance - Restablece la información de distancia recorrida para el eje .....               | 608 |
| 1.204 | ResetAxisMoveTime - Restablece el cronómetro de movimiento del eje .....                             | 610 |
| 1.205 | ResetPPMoved - Restablecer el estado del puntero de programa movido en el modo manual. ....          | 612 |
| 1.206 | ResetRetryCount - Restablecer el número de reintentos .....  | 613 |
| 1.207 | ResetTorqueMargin - Restablecer el menor margen de par .....   | 614 |
| 1.208 | RestoPath - Restablece la trayectoria después de una interrupción .....                              | 615 |
| 1.209 | RETRY - Reanudar la ejecución después de un error .....  | 617 |
| 1.210 | RETURN - Finaliza la ejecución de una rutina .....   | 619 |
| 1.211 | Rewind - Rebobina la posición del archivo .....  | 621 |
| 1.212 | RMQEmptyQueue - Vacía la cola de mensajes de RAPID .....   | 622 |
| 1.213 | RMQFindSlot - Buscar una identidad de ranura para el nombre de ranura .....                          | 624 |

|       |  |     |
|-------|--|-----|
| 1.214 | RMQGetMessage - Obtener un mensaje de RMQ .....  | 626 |
| 1.215 | RMQGetMsgData - Obtener la parte de datos de un mensaje de RMQ .....                           | 629 |
| 1.216 | RMQGetMsgHeader - Obtener información de encabezado de un mensaje de RMQ .....                 | 632 |
| 1.217 | RMQReadWait - Devuelve un mensaje de una cola RMQ .....  | 635 |
| 1.218 | RMQSendMessage - Enviar un mensaje de datos de RMQ .....                                       | 638 |
| 1.219 | RMQSendWait - Enviar un mensaje de datos de RMQ y esperar una respuesta .....                  | 642 |
| 1.220 | SafetyControllerSyncRequest - Inicio del procedimiento de sincronización del hardware .....    | 647 |
| 1.221 | Save - Guarda un módulo de programa .....  | 648 |
| 1.222 | SaveCfgData - Guardar parámetros del sistema a un archivo .....                                | 651 |
| 1.223 | SCWrite - Envía los datos de la variable a una aplicación cliente .....                        | 653 |
| 1.224 | SearchC - Realiza una búsqueda en círculo usando el robot .....                                | 656 |
| 1.225 | SearchExtJ - Busca con una o varias unidades mecánicas sin TCP .....                           | 667 |
| 1.226 | SearchL - Realiza una búsqueda lineal usando el robot .....                                    | 675 |
| 1.227 | SenDevice - Establece una conexión a un dispositivo de sensor .....                            | 687 |
| 1.228 | Set - Activa una señal digital de salida .....   | 689 |
| 1.229 | SetAllDataVal - Establece un valor en todos los objetos de datos de un conjunto definido ..... | 691 |
| 1.230 | SetAO - Cambia el valor de una señal analógica de salida .....                                 | 694 |
| 1.231 | SetDataSearch - Definir el conjunto de símbolos de una secuencia de búsqueda .....             | 696 |
| 1.232 | SetDataVal - Establece el valor de un objeto de datos .....                                    | 701 |
| 1.233 | SetDO - Cambia el valor de una señal digital de salida .....                                   | 704 |
| 1.234 | SetGO - Cambia el valor de un grupo de señales digitales de salida .....                       | 707 |
| 1.235 | SetLeadThrough - Activar y desactivar proceso de guiado .....                                  | 711 |
| 1.236 | SetSysData - Establece datos del sistema .....   | 714 |
| 1.237 | SetupCyclicBool - Configurar una condición de Cyclic bool .....                                | 716 |
| 1.238 | SetupSuperv - Configurar las condiciones para la supervisión de señales en CAP .....           | 719 |
| 1.239 | SiConnect - Conexión a Sensor Interface .....  | 722 |
| 1.240 | SiClose - Cierre de Sensor Interface .....   | 725 |
| 1.241 | SiGetCyclic - Obtención de modo cíclico de Sensor Interface .....                              | 727 |
| 1.242 | SimCollision: simular una colisión .....   | 729 |
| 1.243 | SingArea - Define el método de interpolación alrededor de puntos singulares .....              | 730 |
| 1.244 | SiSetCyclic - Establecimiento de modo cíclico de Sensor Interface .....                        | 733 |
| 1.245 | SkipWarn - Omitir el último aviso .....  | 735 |
| 1.246 | SocketAccept - Aceptar una conexión entrante .....   | 736 |
| 1.247 | SocketBind - Enlazar un zócalo a mi dirección IP y puerto .....                                | 739 |
| 1.248 | SocketClose - Cerrar un zócalo .....   | 742 |
| 1.249 | SocketConnect - Establece una conexión a un ordenador remoto .....                             | 744 |
| 1.250 | SocketCreate - Crea un nuevo zócalo .....  | 747 |
| 1.251 | SocketListen - Permanece a la escucha de conexiones entrantes .....                            | 749 |
| 1.252 | SocketReceive - Recibe datos de un ordenador remoto .....                                      | 751 |
| 1.253 | SocketReceiveFrom - Recepción de datos desde un ordenador remoto .....                         | 756 |
| 1.254 | SocketSend - Envía datos a un ordenador remoto .....   | 761 |
| 1.255 | SocketSendTo - Envío de datos a un ordenador remoto .....                                      | 765 |
| 1.256 | SoftAct - Activa el servo suave .....  | 770 |
| 1.257 | SoftDeact - Desactiva el servo suave .....   | 772 |
| 1.258 | SoftElbow - Hacer el codo para las fuerzas externas .....                                      | 773 |
| 1.259 | SpeedLimAxis - Establecer la limitación de velocidad de un eje .....                           | 775 |
| 1.260 | SpeedLimCheckPoint - Establecer la limitación de velocidad de los puntos de control ....       | 779 |
| 1.261 | SpeedRefresh - La redefinición de velocidad para el movimiento en curso .....                  | 785 |
| 1.262 | SpyStart - Comienza la grabación de los datos de tiempo de ejecución .....                     | 788 |
| 1.263 | SpyStop - Detiene la grabación de los datos de tiempo de ejecución .....                       | 791 |
| 1.264 | StartLoad - Carga de programa durante la ejecución .....                                       | 792 |
| 1.265 | StartMove - Reanuda el movimiento del robot .....  | 796 |
| 1.266 | StartMoveRetry - Reanuda el movimiento y la ejecución del robot .....                          | 799 |
| 1.267 | STCalib - Calibra una herramienta servo .....  | 802 |
| 1.268 | STClose - Cierra una herramienta servo .....   | 807 |
| 1.269 | StepBwdPath - Retrocede un paso a lo largo de la trayectoria .....                             | 811 |
| 1.270 | STIndGun: establece la herramienta servo en el modo independiente .....                        | 813 |
| 1.271 | STIndGunReset: restablece la herramienta servo del modo independiente .....                    | 815 |
| 1.272 | SToolRotCalib - Calibración del TCP y de la rotación de una herramienta estacionaria ....      | 816 |

|       |   |      |
|-------|---|------|
| 1.273 | SToolTCPCalib - Calibración del TCP de una herramienta estacionaria .....   | 819  |
| 1.274 | Stop - Detención de la ejecución del programa .....   | 822  |
| 1.275 | STOpen - Abre una herramienta servo .....   | 825  |
| 1.276 | StopMove - Detiene el movimiento del robot .....  | 828  |
| 1.277 | StopMoveReset - Restablece el estado de movimiento de paro de sistema .....   | 832  |
| 1.278 | StorePath - Almacena la trayectoria cuando se produce una interrupción .....  | 835  |
| 1.279 | StrFormat: dar formato a una cadena .....   | 838  |
| 1.280 | STTune - Ajusta una herramienta servo .....   | 840  |
| 1.281 | STTuneReset - Restablece el ajuste de la herramienta servo .....  | 844  |
| 1.282 | SupSyncSensorOff - Parada de la supervisión de sensor sincronizada .....  | 845  |
| 1.283 | SupSyncSensorOn - Inicio de la supervisión de sensor sincronizada .....   | 846  |
| 1.284 | SyncMoveOff - Finaliza los movimientos sincronizados coordinados .....  | 848  |
| 1.285 | SyncMoveOn - Inicia los movimientos sincronizados coordinados .....   | 854  |
| 1.286 | SyncMoveResume - Activa el modo de movimientos sincronizados coordinados .....  | 860  |
| 1.287 | SyncMoveSuspend - Activa el movimiento independiente-semicoordinado .....   | 862  |
| 1.288 | SyncMoveUndo - Activa los movimientos independientes .....  | 864  |
| 1.289 | SyncToSensor - Sincronización con un sensor .....   | 866  |
| 1.290 | SystemStopAction - Para el sistema de robot .....   | 868  |
| 1.291 | TEST - En función del valor de una expresión... ..  | 870  |
| 1.292 | TestSignDefine - Define una señal de prueba .....   | 872  |
| 1.293 | TestSignReset - Restablece todas las definiciones de señales de prueba .....  | 874  |
| 1.294 | TextTabInstall - Instalación de una tabla de textos .....   | 875  |
| 1.295 | TPERase - Borra el texto mostrado en el FlexPendant .....   | 877  |
| 1.296 | TPReadDnum - Lee un número del FlexPendant .....  | 878  |
| 1.297 | TPReadFK - Lee las teclas de función .....  | 882  |
| 1.298 | TPReadNum - Lee un número del FlexPendant .....   | 887  |
| 1.299 | TPShow - Cambia de ventana en el FlexPendant .....  | 891  |
| 1.300 | TPWrite - Escribe en el FlexPendant .....   | 892  |
| 1.301 | TriggC - Movimiento circular del robot con eventos .....  | 895  |
| 1.302 | TriggCheckIO - Define una comprobación de E/S en una posición fija .....  | 904  |
| 1.303 | TriggDataCopy - Copiar el contenido de una variable de tipo triggdata .....   | 910  |
| 1.304 | TriggDataReset - Restablecer el contenido en una variable de tipo triggdata .....   | 912  |
| 1.305 | TriggEquip - Define un evento de E/S basado en la posición y el tiempo en la trayectoria .....                                      | 914  |
| 1.306 | TriggInt - Define una interrupción dependiente de una posición .....  | 921  |
| 1.307 | TriggIO - Define un evento de E/S de posición o tiempo fijos cerca de un punto de paro .....  | 926  |
| 1.308 | TriggJ - Movimientos de ejes del robot a partir de eventos .....  | 932  |
| 1.309 | TriggL - Movimiento lineal del robot con eventos .....  | 940  |
| 1.310 | TriggJIOs - Movimientos de ejes del robot con eventos de E/S .....  | 949  |
| 1.311 | TriggLIOs - Movimientos lineales del robot con eventos de E/S .....   | 957  |
| 1.312 | TriggRampAO - Define un evento AO de rampa de posición fija en la trayectoria .....   | 965  |
| 1.313 | TriggSpeed - Define la velocidad del TCP en proporción a una salida analógica con un evento de escala fija de posición-tiempo ..... | 973  |
| 1.314 | TriggStopProc - Genera datos de reinicio para las señales de disparo ante paros .....   | 984  |
| 1.315 | TryInt - Comprobar si un objeto de dato es un entero válido .....   | 990  |
| 1.316 | TRYNEXT - Salta sobre una instrucción que ha provocado un error .....   | 992  |
| 1.317 | TuneReset - Restablecimiento del ajuste del servo .....   | 993  |
| 1.318 | TuneServo - Ajuste de servos .....  | 994  |
| 1.319 | UIMsgBox - Cuadro de mensaje de usuario de tipo básico .....  | 1002 |
| 1.320 | UIMsgWrite - Cuadro de diálogo de mensaje de usuario tipo sin espera .....  | 1011 |
| 1.321 | UIMsgWriteAbort - Cancelar cuadro de diálogo de mensaje de usuario tipo sin espera ...  | 1016 |
| 1.322 | UIShow - Visualización de interfaz de usuario .....   | 1017 |
| 1.323 | UnLoad - Descargar un módulo de programa durante la ejecución .....   | 1021 |
| 1.324 | UnpackRawBytes - Desempaqueta datos de un dato de tipo rawbytes .....   | 1024 |
| 1.325 | VelSet - Cambia la velocidad programada .....   | 1029 |
| 1.326 | WaitAI - Espera hasta que se establece un valor de señal analógica de entrada .....   | 1032 |
| 1.327 | WaitAO - Espera hasta que se establece un valor de señal analógica de salida .....  | 1037 |
| 1.328 | WaitDI - Espera hasta que se activa una señal digital de entrada .....  | 1042 |
| 1.329 | WaitDO - Espera hasta que se activa una señal digital de salida .....   | 1047 |
| 1.330 | WaitGI - Espera hasta que se activa un grupo de entradas digitales .....  | 1052 |

|          |   |             |
|----------|---|-------------|
| 1.331    | WaitGO - Espera hasta que se activa un grupo de salidas digitales .....                     | 1058        |
| 1.332    | WaitLoad - Conectar un módulo cargado a una tarea .....                                     | 1064        |
| 1.333    | WaitRob - Esperar hasta un punto de paro o una velocidad cero .....                         | 1069        |
| 1.334    | WaitSensor - Espera a la conexión de un sensor .....  | 1071        |
| 1.335    | WaitSyncTask - Esperar en un punto de sincronización con otras tareas de programa ....      | 1075        |
| 1.336    | WaitTestAndSet - Esperar a que la variable cambie a FALSE y activarla a continuación ..     | 1079        |
| 1.337    | WaitTime - Esperar una cantidad de tiempo determinada .....                                 | 1082        |
| 1.338    | WaitUntil - Esperar hasta que se cumple una condición .....                                 | 1084        |
| 1.339    | WaitWObj - Esperar a un objeto de trabajo en un transportador .....                         | 1090        |
| 1.340    | WarmStart - Reinicio del controlador .....  | 1093        |
| 1.341    | WHILE - Repetir siempre y cuando se cumpla una condición .....                              | 1094        |
| 1.342    | WorldAccLim - Control de aceleración en el sistema de coordenadas mundo .....               | 1096        |
| 1.343    | Write - Escribe en un archivo o dispositivo de E/S alfanumérico .....                       | 1098        |
| 1.344    | WriteAnyBin - Escribe datos en un archivo o dispositivo de E/S binario .....                | 1101        |
| 1.345    | WriteBin - Escribe en un dispositivo de E/S binario .....                                   | 1103        |
| 1.346    | WriteBlock - Escribir un bloque de datos en un dispositivo .....                            | 1105        |
| 1.347    | WriteCfgData - Escribe un atributo de un parámetro del sistema .....                        | 1107        |
| 1.348    | WriteRawBytes - Escribe un dato de tipo rawbytes .....                                      | 1111        |
| 1.349    | WriteStrBin - Escribe una cadena en un dispositivo de E/S binario .....                     | 1114        |
| 1.350    | WriteVar - Escribir una variable .....  | 1116        |
| 1.351    | WriteVarArr - Escribe múltiples variables en un dispositivo sensor .....                    | 1119        |
| 1.352    | WZBoxDef - Define una zona mundo con forma de prisma .....                                  | 1121        |
| 1.353    | WZCylDef - Define una zona mundo con forma cilíndrica .....                                 | 1123        |
| 1.354    | WZDisable - Desactiva la supervisión de las zonas mundo temporales .....                    | 1126        |
| 1.355    | WZDOSet - Activación de salidas digitales basadas en zonas mundo .....                      | 1128        |
| 1.356    | WZEnable - Activa la supervisión de las zonas mundo temporales .....                        | 1133        |
| 1.357    | WZFree - Elimina la supervisión de las zonas mundo temporales .....                         | 1135        |
| 1.358    | WZHomeJointDef - Define una zona mundo para las posiciones iniciales de los ejes .....      | 1137        |
| 1.359    | WZLimJointDef - Define una zona mundo para la limitación de los ejes .....                  | 1141        |
| 1.360    | WZLimSup - Activa la supervisión de límites de las zonas mundo .....                        | 1145        |
| 1.361    | WZSphDef - Define una zona mundo con forma esférica .....                                   | 1148        |
| <b>2</b> | <b>Funciones</b>  | <b>1151</b> |
| 2.1      | Abs - Obtiene el valor absoluto .....   | 1151        |
| 2.2      | AbsDnum - Obtiene el valor absoluto de un dnum .....  | 1153        |
| 2.3      | ACos - Calcula el valor de arco coseno .....  | 1155        |
| 2.4      | ACosDnum - Calcula el valor de arco coseno .....  | 1156        |
| 2.5      | AINput - Lee el valor de una señal analógica de entrada .....                               | 1157        |
| 2.6      | AND - Evalúa un valor lógico .....  | 1159        |
| 2.7      | AOutput - Lee el valor de una señal analógica de salida .....                               | 1161        |
| 2.8      | ArgName - Obtiene el nombre de un argumento .....   | 1163        |
| 2.9      | ASin - Calcula el valor de arco seno .....  | 1167        |
| 2.10     | ASinDnum - Calcula el valor de arco seno .....  | 1168        |
| 2.11     | ATan - Calcula el valor de arco tangente .....  | 1169        |
| 2.12     | ATanDnum - Calcula el valor de arco tangente .....  | 1170        |
| 2.13     | ATan2 - Calcula el valor de arco tangente 2 .....   | 1171        |
| 2.14     | ATan2Dnum - Calcula el valor de arco tangente 2 .....                                       | 1172        |
| 2.15     | BitAnd - AND lógico bit a bit - Operación con datos de byte .....                           | 1173        |
| 2.16     | BitAndDnum - Operación lógica AND bit a bit en un dato dnum .....                           | 1175        |
| 2.17     | BitCheck - Comprueba si un bit especificado de un dato de byte está activado .....          | 1177        |
| 2.18     | BitCheckDnum - Comprueba si un bit especificado de un dato dnum está activado .....         | 1179        |
| 2.19     | BitLSh - DESPLAZAMIENTO A LA IZQUIERDA lógico bit a bit - Operación de byte .....           | 1181        |
| 2.20     | BitLShDnum - Operación lógica DESPLAZAMIENTO A LA IZQUIERDA bit a bit en dato<br>dnum ..... | 1183        |
| 2.21     | BitNeg - NEGACIÓN lógica bit a bit - Operación con datos de byte .....                      | 1186        |
| 2.22     | BitNegDnum - Operación lógica NEGACIÓN bit a bit - operación en datos dnum .....            | 1188        |
| 2.23     | BitOr - OR lógico bit a bit - Operación con datos de byte .....                             | 1190        |
| 2.24     | BitOrDnum - Operación lógica OR bit a bit en datos dnum .....                               | 1192        |
| 2.25     | BitRSh - DESPLAZAMIENTO A LA DERECHA lógico bit a bit - Operación de byte .....             | 1194        |

|      |   |      |
|------|---|------|
| 2.26 | BitRShDnum - Operación lógica DESPLAZAMIENTO A LA DERECHA bit a bit en un dnum .....    | 1196 |
| 2.27 | BitXOr - XOR lógico bit a bit - Operación con datos de byte .....                       | 1198 |
| 2.28 | BitXOrDnum - Operación lógica XOR bit a bit en datos dnum .....                         | 1200 |
| 2.29 | ByteToStr - Convierte un byte en un dato de cadena de caracteres .....                  | 1202 |
| 2.30 | CalcJointT - Calcula los ángulos de las articulaciones a partir de un robtarget .....   | 1204 |
| 2.31 | CalcRobT - Calcula el valor de robtarget a partir de jointtarget .....                  | 1209 |
| 2.32 | CalcRotAxFrameZ - Calcular la base de coordenadas de un eje de rotación .....           | 1211 |
| 2.33 | CalcRotAxisFrame - Calcular la base de coordenadas de un eje de rotación .....          | 1216 |
| 2.34 | CamGetExposure - Obtiene parámetros específicos de la cámara .....                      | 1220 |
| 2.35 | CamGetLoadedJob - Obtiene el nombre de la tarea de cámara cargada .....                 | 1222 |
| 2.36 | CamGetMode: obtener el modo actual de la cámara .....                                   | 1224 |
| 2.37 | CamGetName - Obtiene el nombre de la cámara utilizada .....                             | 1225 |
| 2.38 | CamNumberOfResults - Obtiene el número de resultados disponibles .....                  | 1226 |
| 2.39 | CapGetFailSigs - Obtener señales de E/S con fallos .....                                | 1228 |
| 2.40 | CDate - Lee la fecha actual como una cadena .....                                       | 1230 |
| 2.41 | CJointT - Lee los ángulos actuales de los ejes .....                                    | 1231 |
| 2.42 | ClkRead - Lee un reloj utilizado para la temporización .....                            | 1233 |
| 2.43 | CorrRead - Lee los offsets totales actuales .....                                       | 1235 |
| 2.44 | Cos - Calcula el valor de coseno .....  | 1236 |
| 2.45 | CosDnum - Calcula el valor de coseno .....  | 1237 |
| 2.46 | CPos - Lee los datos de posición (pos) actuales .....                                   | 1238 |
| 2.47 | CRobT - Lee los datos de posición (robtarget) actuales .....                            | 1240 |
| 2.48 | CrossProd - Producto cruzado de dos vectores pos .....                                  | 1243 |
| 2.49 | CSpeedOverride - Lee el ajuste de velocidad actual .....                                | 1246 |
| 2.50 | CTime - Lee la hora actual en forma de una cadena .....                                 | 1248 |
| 2.51 | CTool - Lee los datos actuales de la herramienta .....                                  | 1249 |
| 2.52 | CWObj - Lee los datos del objeto de trabajo actual .....                                | 1251 |
| 2.53 | DecToHex - Convierte de decimal a hexadecimal .....                                     | 1253 |
| 2.54 | DefAccFrame - Define una base de coordenadas exacta .....                               | 1254 |
| 2.55 | DefDFrame - Define una base de coordenadas de desplazamiento .....                      | 1257 |
| 2.56 | DefFrame - Define una base de coordenadas .....   | 1260 |
| 2.57 | Dim - Obtiene las dimensiones de una matriz .....                                       | 1263 |
| 2.58 | DInput - Lee el valor de una señal digital de entrada .....                             | 1265 |
| 2.59 | Distance - Distancia entre dos puntos .....   | 1267 |
| 2.60 | DIV - Evalúa una división entera .....  | 1269 |
| 2.61 | DnumToNum - Convierte dnum a num .....  | 1270 |
| 2.62 | DnumToStr - Convierte un valor numérico en una cadena de caracteres .....               | 1272 |
| 2.63 | DotProd - Producto escalar de dos vectores pos .....                                    | 1274 |
| 2.64 | DOutput - Lee el valor de una señal digital de salida .....                             | 1276 |
| 2.65 | EulerZYX - Obtiene ángulos Euler a partir de una orientación .....                      | 1278 |
| 2.66 | EventType - Obtiene el tipo de evento actual dentro de cualquier rutina de evento ..... | 1280 |
| 2.67 | ExecHandler - Obtener el tipo de gestor de ejecución .....                              | 1282 |
| 2.68 | ExecLevel - Obtener el nivel de ejecución .....   | 1283 |
| 2.69 | Exp - Calcula el valor exponencial .....  | 1284 |
| 2.70 | FileSize - Obtiene el tamaño de un archivo .....  | 1285 |
| 2.71 | FileTimeDnum - Obtener información de tiempo sobre un archivo .....                     | 1288 |
| 2.72 | FSSize - Obtiene el tamaño de un sistema de archivos .....                              | 1291 |
| 2.73 | GetAxisDistance - Proporciona la información de distancia recorrida por el eje .....    | 1294 |
| 2.74 | GetAxisMoveTime - Obtiene el valor del cronómetro de movimiento del eje .....           | 1296 |
| 2.75 | GetMaxNumberOfCyclicBool - Obtener el número máximo de condiciones de Cyclic bool. .... | 1298 |
| 2.76 | GetMecUnitName - Obtener el nombre de la unidad mecánica .....                          | 1299 |
| 2.77 | GetModalPayLoadMode - Obtener el valor de ModalPayLoadMode .....                        | 1300 |
| 2.78 | GetMotorTorque - Lee el par motor actual .....  | 1301 |
| 2.79 | GetNextCyclicBool - Obtener los nombres de todos los Cyclic bools .....                 | 1304 |
| 2.80 | GetNextMechUnit - Obtener el nombre y los datos de las unidades mecánicas .....         | 1306 |
| 2.81 | GetNextSym - Obtiene el siguiente símbolo coincidente .....                             | 1309 |
| 2.82 | GetNumberOfCyclicBool - Obtener el número de condiciones de Cyclic bool .....           | 1311 |



|       |   |      |
|-------|---|------|
| 2.83  | GetServiceInfo - Obtener información de servicio del sistema .....                                    | 1312 |
| 2.84  | GetSignalOrigin - Obtención de información acerca del origen de una señal de E/S .....                | 1314 |
| 2.85  | GetSysInfo - Obtener información acerca del sistema .....   | 1316 |
| 2.86  | GetTaskName - Obtiene el nombre y el número de la tarea actual .....                                  | 1319 |
| 2.87  | GetTime - Lee la hora actual como un valor numérico .....   | 1321 |
| 2.88  | GetTSPStatus: Obtener el estado del panel de selección de tareas actuales .....                       | 1323 |
| 2.89  | GetUASUserName - Obtener el nombre de usuario del usuario .....                                       | 1325 |
| 2.90  | GInput - Lee el valor de una señal de entrada de grupo .....  | 1327 |
| 2.91  | GInputDnum - Lee el valor de una señal de entrada de grupo .....                                      | 1329 |
| 2.92  | GOutput - Lee el valor de un grupo de señales digitales de salida .....                               | 1332 |
| 2.93  | GOutputDnum - Lee el valor de una señal de salida de grupo .....                                      | 1334 |
| 2.94  | HexToDec - Convierte de hexadecimal a decimal .....   | 1337 |
| 2.95  | IndInpos - Estado de posición de un eje independiente .....   | 1338 |
| 2.96  | IndInpos - Estado de velocidad de un eje independiente .....  | 1340 |
| 2.97  | IOUnitState - Obtener el estado actual de un dispositivo de E/S .....                                 | 1342 |
| 2.98  | IsBrakeCheckActive: Comprobar si la prueba de frenos está en marcha .....                             | 1345 |
| 2.99  | IsCollFree: comprueba si la posición colisionaría .....   | 1346 |
| 2.100 | IsCyclicBool - Comprueba si una variable persistente es un Cyclic bool .....                          | 1348 |
| 2.101 | IsFile - Comprobar el tipo de un archivo .....  | 1351 |
| 2.102 | IsLeadThrough - Comprobar el estado del proceso de guiado .....                                       | 1355 |
| 2.103 | IsMechUnitActive - Indica si una unidad mecánica está activa .....                                    | 1357 |
| 2.104 | IsPers - Determina si es una variable persistente .....   | 1358 |
| 2.105 | IsStopMoveAct - Está activo el indicador de movimiento de paro .....                                  | 1360 |
| 2.106 | IsStopStateEvent - Comprueba si se ha movido el puntero de programa .....                             | 1362 |
| 2.107 | IsSyncMoveOn - Comprueba si el modo de movimiento sincronizado está activado .....                    | 1364 |
| 2.108 | IsSysID - Comprobar la identidad del sistema .....  | 1366 |
| 2.109 | IsVar - Determina si un dato es una variable .....  | 1367 |
| 2.110 | Max - Obtener el mayor de dos valores .....   | 1369 |
| 2.111 | MaxExtLinearSpeed - Velocidad máxima de eje adicional .....   | 1370 |
| 2.112 | MaxExtReorientSpeed - Velocidad de giro máxima de eje adicional .....                                 | 1371 |
| 2.113 | MaxRobReorientSpeed - Velocidad de reorientación máxima del robot .....                               | 1372 |
| 2.114 | MaxRobSpeed - Velocidad máxima del robot .....  | 1373 |
| 2.115 | Min - Obtener el menor de dos valores .....   | 1374 |
| 2.116 | MirPos - Obtención de la posición espejo de una posición .....  | 1375 |
| 2.117 | MOD - Evalúa un módulo de entero .....  | 1377 |
| 2.118 | ModExist - Comprobar si un módulo de programa existe .....  | 1378 |
| 2.119 | ModTimeDnum - Obtener la hora de modificación del módulo cargado .....                                | 1379 |
| 2.120 | MotionPlannerNo - Obtiene el número de planificador de movimientos conectado .....                    | 1381 |
| 2.121 | NonMotionMode - Lee el modo de ejecución sin movimiento .....   | 1383 |
| 2.122 | NOT - Invierte un valor lógico .....  | 1385 |
| 2.123 | NOrient - Normaliza la orientación .....  | 1386 |
| 2.124 | NumToDnum - Convierte num a dnum .....  | 1388 |
| 2.125 | NumToStr - Convierte un valor numérico en una cadena de caracteres .....                              | 1389 |
| 2.126 | Offs - Desplaza una posición del robot .....  | 1391 |
| 2.127 | OpMode - Lee el modo de funcionamiento .....  | 1393 |
| 2.128 | OR - Evalúa un valor lógico .....   | 1394 |
| 2.129 | OrientZYX - Genera una orientación a partir de ángulos Euler .....                                    | 1396 |
| 2.130 | ORobT - Elimina el desplazamiento de programa de una posición .....                                   | 1398 |
| 2.131 | ParIdPosValid - Posición de robot válida para la identificación de parámetros .....                   | 1400 |
| 2.132 | ParIdRobValid - Tipo de robot válido para la identificación de parámetros .....                       | 1403 |
| 2.133 | PathLengthGet - Lee el valor de longitud de trayectoria actual del contador .....                     | 1406 |
| 2.134 | PathLevel - Obtiene el nivel de trayectoria actual .....  | 1408 |
| 2.135 | PathRecValidBwd - Comprueba si existe una trayectoria de retroceso válida guardada .....              | 1410 |
| 2.136 | PathRecValidFwd - Comprueba si existe una trayectoria de avance válida guardada .....                 | 1413 |
| 2.137 | PFRestart - Comprueba si se ha interrumpido una trayectoria después de un fallo de alimentación ..... | 1417 |
| 2.138 | PosInv - Invierte los datos de pose .....   | 1418 |
| 2.139 | PoseMult - Multiplica datos de pose .....   | 1420 |
| 2.140 | PoseVect - Aplica una transformación a un vector .....  | 1422 |

|       |  |      |
|-------|--|------|
| 2.141 | Pow - Calcula el resultado de elevar un valor a una potencia .....   | 1424 |
| 2.142 | PowDnum - Calcula el resultado de elevar un valor a una potencia .....                                       | 1425 |
| 2.143 | PPMovedInManMode - Comprobar si el puntero de programa se ha movido en el modo manual. ....                  | 1426 |
| 2.144 | Present - Comprueba si se está usando un parámetro opcional .....  | 1427 |
| 2.145 | ProgMemFree - Obtiene el tamaño de memoria libre del programa .....  | 1429 |
| 2.146 | PrxGetMaxRecordpos - Obtención de la posición máxima de sensor .....   | 1430 |
| 2.147 | Rand - Genera un número aleatorio .....  | 1431 |
| 2.148 | RawBytesLen - Obtiene la longitud de un dato de tipo rawbytes .....  | 1432 |
| 2.149 | ReadBin - Lee un byte de un archivo o dispositivo de E/S .....   | 1434 |
| 2.150 | ReadDir - Lee la siguiente entrada de un directorio .....  | 1437 |
| 2.151 | ReadMotor - Lee los ángulos actuales de los motores .....  | 1440 |
| 2.152 | ReadNum - Lee un número de un archivo o dispositivo de E/S .....   | 1442 |
| 2.153 | ReadStr - Lee una cadena de un archivo o dispositivo de E/S .....  | 1445 |
| 2.154 | ReadStrBin - Lee una cadena de un dispositivo de E/S o archivo binario .....                                 | 1449 |
| 2.155 | ReadVar - Lee una variable de un dispositivo .....   | 1451 |
| 2.156 | RelTool - Hace un desplazamiento respecto de la herramienta .....  | 1453 |
| 2.157 | RemainingRetries - Reintentos restantes aún pendientes .....   | 1455 |
| 2.158 | RMQGetSlotName - Obtener el nombre de un cliente de RMQ .....  | 1456 |
| 2.159 | RobName - Obtiene el nombre del robot del TCP .....  | 1458 |
| 2.160 | RobOS - Comprueba si el programa se está ejecutando en RC o VC .....   | 1460 |
| 2.161 | Round - Redondear un valor numérico .....  | 1461 |
| 2.162 | RoundDnum - Redondear un valor numérico .....  | 1463 |
| 2.163 | RunMode - Obtiene el modo de ejecución .....   | 1465 |
| 2.164 | SafetyControllerGetChecksum - Obtener la suma de comprobación del archivo de configuración de usuarios ..... | 1467 |
| 2.165 | SafetyControllerGetOpModePinCode - Obtener el código pin del modo de funcionamiento .....                    | 1468 |
| 2.166 | SafetyControllerGetSWVersion - Obtener la versión del firmware del Safety Controller ...                     | 1469 |
| 2.167 | SafetyControllerGetUserChecksum - Obtener la suma de comprobación de los parámetros protegidos .....         | 1470 |
| 2.168 | Sin - Calcula el valor del seno .....  | 1471 |
| 2.169 | SinDnum - Calcula el valor del seno .....  | 1472 |
| 2.170 | SocketGetStatus - Obtiene el estado actual de un zócalo .....  | 1473 |
| 2.171 | SocketPeek - Prueba para comprobar la presencia de datos en un zócalo .....                                  | 1476 |
| 2.172 | Sqrt - Calcula el valor de la raíz cuadrada .....  | 1478 |
| 2.173 | SqrtDnum - Calcula el valor de la raíz cuadrada .....  | 1479 |
| 2.174 | STCalcForce - Calcula fuerza de la punta de una herramienta servo .....                                      | 1480 |
| 2.175 | STCalcTorque - Calcula el par motor de una herramienta servo .....   | 1482 |
| 2.176 | STIsCalib - Compruebe si una herramienta servo está calibrada .....  | 1484 |
| 2.177 | STIsClosed - Comprueba si una herramienta servo está cerrada .....   | 1486 |
| 2.178 | STIsIndGun - Comprueba si una herramienta servo se encuentra en el modo independiente .....                  | 1488 |
| 2.179 | STIsOpen - Comprueba si una herramienta servo está abierta .....   | 1489 |
| 2.180 | StrDigCalc - Operaciones aritméticas con el tipo de dato stringdig .....                                     | 1491 |
| 2.181 | StrDigCmp - Comparar dos cadenas que sólo contienen dígitos .....  | 1494 |
| 2.182 | StrFind - Busca un carácter en una cadena de caracteres .....  | 1496 |
| 2.183 | StrLen - Obtiene la longitud de una cadena .....   | 1498 |
| 2.184 | StrMap - Mapea una cadena de caracteres .....  | 1499 |
| 2.185 | StrMatch - Busca un patrón dentro de una cadena de caracteres .....  | 1501 |
| 2.186 | StrMemb - Comprueba si un carácter pertenece a un conjunto .....   | 1503 |
| 2.187 | StrOrder - Comprueba si dos cadenas de caracteres están ordenadas .....                                      | 1505 |
| 2.188 | StrPart - Busca una parte de una cadena .....  | 1507 |
| 2.189 | StrToByte - Convierte una cadena en un byte .....  | 1509 |
| 2.190 | StrToVal - Convierte una cadena de caracteres en un valor .....  | 1511 |
| 2.191 | Tan - Calcula la tangente .....  | 1513 |
| 2.192 | TanDnum - Calcula el valor de la tangente .....  | 1514 |
| 2.193 | TaskRunMec - Comprueba si una tarea controla alguna unidad mecánica .....                                    | 1515 |
| 2.194 | TaskRunRob - Comprueba si una tarea controla algún robot .....   | 1516 |

|          |  |             |
|----------|--|-------------|
| 2.195    | TasksInSync - Devuelve el número de tareas sincronizadas .....                             | 1517        |
| 2.196    | TasksActive: Comprobar si una tarea normal está activa .....                               | 1519        |
| 2.197    | TasksExecuting: Comprobar si la tarea se está ejecutando .....                             | 1521        |
| 2.198    | TestAndSet - Comprueba una variable y la establece si no está establecida .....            | 1523        |
| 2.199    | TestDI - Se comprueba si una entrada digital está activada .....                           | 1526        |
| 2.200    | TestSignRead - Obtiene el valor de una señal de test .....                                 | 1528        |
| 2.201    | TextGet - Obtener un texto de las tablas de textos del sistema .....                       | 1530        |
| 2.202    | TextTabFreeToUse - Comprueba si una tabla de textos está libre para su uso .....           | 1533        |
| 2.203    | TextTabGet - Obtiene el número de una tabla de textos .....                                | 1535        |
| 2.204    | TriggDataValid - Comprobar si el contenido de una variable de tipo triggdata es válido ... | 1537        |
| 2.205    | Trunc - Trunca un valor numérico .....   | 1539        |
| 2.206    | TruncDnum - Trunca un valor numérico .....   | 1541        |
| 2.207    | Type - Obtiene el nombre del tipo de dato de una variable .....                            | 1543        |
| 2.208    | UIAlphaEntry - Introducción alfanumérica del usuario .....                                 | 1545        |
| 2.209    | UIClientExist - Existe cliente de usuario .....  | 1552        |
| 2.210    | UIDnumEntry - Introducción de número de usuario .....                                      | 1554        |
| 2.211    | UIDnumTune - Ajuste de número de usuario .....   | 1561        |
| 2.212    | UIListView - Vista de lista de usuario .....   | 1568        |
| 2.213    | UIMessageBox - Cuadro de mensaje de usuario de tipo avanzado .....                         | 1577        |
| 2.214    | UINumEntry - Introducción de número de usuario .....                                       | 1585        |
| 2.215    | UINumTune - Ajuste de número de usuario .....  | 1592        |
| 2.216    | ValidIO - Señal de E/S válida para su uso .....  | 1599        |
| 2.217    | ValToStr - Convierte un valor en una cadena .....  | 1601        |
| 2.218    | VectMagn - Magnitud de un vector pos .....   | 1603        |
| 2.219    | XOR - Evalúa un valor lógico .....   | 1605        |
| <b>3</b> | <b>Tipos de datos</b> .....  | <b>1607</b> |
| 3.1      | aiotrigg - Condición de disparo con E/S analógica .....                                    | 1607        |
| 3.2      | ALIAS - Asignación de un tipo de dato de alias .....                                       | 1609        |
| 3.3      | bool - Valores lógicos .....   | 1610        |
| 3.4      | btnres - Datos de resultado de pulsador .....  | 1611        |
| 3.5      | busstate - Estado de la red de E/S .....   | 1613        |
| 3.6      | buttondata - Datos de botón .....  | 1614        |
| 3.7      | byte - Valores enteros 0-255 .....   | 1616        |
| 3.8      | cameradev - dispositivo de cámara .....  | 1617        |
| 3.9      | camerastatus: estado de comunicación de la cámara .....                                    | 1618        |
| 3.10     | cameratarget - datos de cámara .....   | 1619        |
| 3.11     | capaptrreferencedata - Datos de configuración variable para seguimiento del punto .....    | 1622        |
| 3.12     | capdata - Datos CAP .....  | 1624        |
| 3.13     | caplatrackdata - Datos de seguimiento de sensor de seguimiento anticipatorio de CAP ..     | 1628        |
| 3.14     | capspeeddata - Datos de velocidad para CAP .....   | 1632        |
| 3.15     | capstopmode - Define modos de paro para CAP .....  | 1634        |
| 3.16     | captrackdata - Datos de seguimiento de CAP .....   | 1635        |
| 3.17     | capweavedata - Datos de oscilación para CAP .....  | 1638        |
| 3.18     | cfgdomain - Dominio de configuración .....   | 1646        |
| 3.19     | clock - Medición de tiempo .....   | 1647        |
| 3.20     | confdata - Datos de configuración del robot .....  | 1648        |
| 3.21     | corrdescr - Descriptor de generador de correcciones .....                                  | 1655        |
| 3.22     | datapos - Inclusión de un bloque para un objeto de datos .....                             | 1657        |
| 3.23     | dionum - Valores digitales (0-1) .....   | 1658        |
| 3.24     | dir - Estructura de directorio de archivos .....   | 1659        |
| 3.25     | dnum - Valores numéricos dobles .....  | 1660        |
| 3.26     | errdomain - Dominio del error .....  | 1662        |
| 3.27     | errnum - Número de error .....   | 1664        |
| 3.28     | errstr - Cadena de error .....   | 1672        |
| 3.29     | errtype - Tipo de error .....  | 1673        |
| 3.30     | event_type - Tipo de rutina de evento .....  | 1675        |
| 3.31     | exec_level - Nivel de ejecución .....  | 1676        |
| 3.32     | extjoint - Posición de los ejes externos .....   | 1677        |

|      |   |      |
|------|---|------|
| 3.33 | flypointdata - Datos para inicio/fin sobre la marcha .....  | 1679 |
| 3.34 | handler_type - Tipo de gestor de ejecución .....  | 1682 |
| 3.35 | icondata - Datos de visualización de iconos .....   | 1683 |
| 3.36 | identno - Identidad para las instrucciones de movimiento .....                                      | 1685 |
| 3.37 | intnum - Identidad de interrupción .....  | 1687 |
| 3.38 | iodev - Dispositivo de E/S .....  | 1689 |
| 3.39 | iounit_state - Estado del dispositivo de E/S .....  | 1690 |
| 3.40 | jointtarget - Datos de posición de eje .....  | 1691 |
| 3.41 | listitem - Estructura de datos de elementos de lista .....  | 1693 |
| 3.42 | loaddata - Datos de carga .....   | 1694 |
| 3.43 | loadidnum - Tipo de identificación de carga .....   | 1700 |
| 3.44 | loadsession - Sesión de carga de programa .....   | 1702 |
| 3.45 | mecunit - Unidad mecánica .....   | 1703 |
| 3.46 | motsetdata - Datos de parámetros de movimiento .....  | 1705 |
| 3.47 | num - Valores numéricos .....   | 1712 |
| 3.48 | opcalc - Operador aritmético .....  | 1714 |
| 3.49 | opnum - Operador de comparación .....   | 1715 |
| 3.50 | orient - Orientación .....  | 1716 |
| 3.51 | paridnum - Tipo de identificación de parámetro .....  | 1721 |
| 3.52 | paridvalidnum - Resultado de ParldRobValid .....  | 1723 |
| 3.53 | pathrecid - Identificador de grabadora de trayectorias .....  | 1725 |
| 3.54 | pnpdata: Configurar las trayectorias de elección y colocación .....                                 | 1727 |
| 3.55 | pos - Posiciones (sólo X, Y y Z) .....  | 1729 |
| 3.56 | pose - Transformaciones de coordenadas .....  | 1731 |
| 3.57 | processtimes - Tiempos de proceso .....   | 1732 |
| 3.58 | progdisp - Desplazamiento de programa .....   | 1733 |
| 3.59 | rawbytes - Datos sin formato .....  | 1736 |
| 3.60 | restartblkdata - bloque de datos para el reinicio .....   | 1738 |
| 3.61 | restartdata - Datos de reinicio de señales de disparo .....   | 1740 |
| 3.62 | rmqheader - Encabezado de mensaje de RAPID Message Queue .....                                      | 1744 |
| 3.63 | rmqmessage - Mensaje de RAPID Message Queue .....   | 1746 |
| 3.64 | rmqslot - Número de identidad de un cliente de RMQ .....  | 1748 |
| 3.65 | robjoint - Posición de eje de los ejes del robot .....  | 1749 |
| 3.66 | robtarget - Datos de posición .....   | 1750 |
| 3.67 | sensor - Descriptor de dispositivo externo .....  | 1754 |
| 3.68 | sensorstate - Estado de comunicación del dispositivo .....  | 1756 |
| 3.69 | sensorvardata - Configuración de múltiples variables de datos para la interfaz de<br>sensores ..... | 1757 |
| 3.70 | shapedata - Datos de forma de zonas mundo .....   | 1759 |
| 3.71 | signalorigin - Describe el origen de la señal de E/S .....  | 1761 |
| 3.72 | signalxx - Señales digitales y analógicas .....   | 1763 |
| 3.73 | socketdev - Dispositivo de zócalo .....   | 1765 |
| 3.74 | socketstatus - Estado de comunicación de zócalo .....   | 1766 |
| 3.75 | speeddata - Datos de velocidad .....  | 1767 |
| 3.76 | stoppointdata - Datos de punto de paro .....  | 1771 |
| 3.77 | string - Cadenas .....  | 1778 |
| 3.78 | stringdig - Cadena de caracteres con sólo dos dígitos .....   | 1780 |
| 3.79 | supervtimeouts - Tiempos límite de supervisión de intercambio .....                                 | 1781 |
| 3.80 | switch - Parámetros opcionales .....  | 1783 |
| 3.81 | symnum - Número simbólico .....   | 1784 |
| 3.82 | syncident - Identidad de punto de sincronización .....  | 1786 |
| 3.83 | System data - Ajustes de datos del sistema RAPID actual .....                                       | 1787 |
| 3.84 | taskid - Identificación de tarea .....  | 1790 |
| 3.85 | tasks - Tareas de programa RAPID .....  | 1791 |
| 3.86 | testsignal - Señal de prueba .....  | 1793 |
| 3.87 | tooldata - Datos de herramienta .....   | 1795 |
| 3.88 | tpnum - Número de ventana del FlexPendant .....   | 1802 |
| 3.89 | trapdata - Datos de interrupción para la rutina TRAP actual .....                                   | 1803 |
| 3.90 | triggdata - Eventos de posicionamiento, trigg .....   | 1805 |

|               |  |             |
|---------------|--|-------------|
| 3.91          | triggios - Eventos de posicionamiento, trigg .....           | 1806        |
| 3.92          | triggiosdnum - Eventos de posicionamiento, trigg .....       | 1809        |
| 3.93          | triggmode - Disparar modo de acción .....                    | 1811        |
| 3.94          | triggstrgo - Positioning events, trigg .....                 | 1814        |
| 3.95          | tsp_status: Estado de panel de selección de tareas .....     | 1817        |
| 3.96          | tunetype - Tipo de ajuste de servo .....                     | 1819        |
| 3.97          | uishownum - ID de instancia para UIShow .....                | 1820        |
| 3.98          | weavestartdata - Datos de inicio de oscilación .....         | 1821        |
| 3.99          | wobjdata - Datos del objeto de trabajo .....                 | 1823        |
| 3.100         | wzstationary - Datos de zona mundo estacionaria .....        | 1827        |
| 3.101         | wztemporary - Datos de zona mundo temporal .....             | 1829        |
| 3.102         | zonedata - Datos de zonas .....                              | 1831        |
| <b>4</b>      | <b>Ejemplos de tipos de programación</b> .....               | <b>1837</b> |
| 4.1           | Gestor de ERROR con movimientos .....                        | 1837        |
| 4.2           | Rutinas de servicio con o sin movimientos .....              | 1840        |
| 4.3           | Interrupciones de E/S del sistema con o sin movimiento ..... | 1843        |
| 4.4           | Rutinas TRAP con movimientos .....                           | 1846        |
| <b>Índice</b> |  | <b>1849</b> |

**Esta página se ha dejado vacía intencionadamente**

# Descripción general de este manual

## Acerca de este manual

Éste es un manual de referencia técnica destinado a los programadores de RAPID. En este manual se detallan las instrucciones, las funciones y tipos de datos básicos de RAPID.

Este manual describe RobotWare 6.

## Utilización

Este manual debe leerse durante la programación y siempre que se necesite información específica acerca de una instrucción, función o tipo de dato de RAPID.

## ¿A quién va destinado este manual?

Este manual está destinado a personas que tengan cierta experiencia anterior con la programación, por ejemplo un programador de robots.

## Requisitos previos

El lector debe tener cierta experiencia en programación y haber estudiado:

- *Manual de referencia técnica - RAPID Overview*

## Organización de los capítulos

Este manual está organizado en los capítulos siguientes:


| Capítulo                          | Contenido   |
|-----------------------------------|---|
| Instrucciones                     | Descripciones detalladas de todas las instrucciones básicas de RAPID, con ejemplos de cómo usarlas.   |
| Funciones                         | Descripciones detalladas de todas las funciones básicas de RAPID, con ejemplos de cómo usarlas.   |
| Tipos de datos                    | Descripciones detalladas de todos los tipos de datos básicos de RAPID, con ejemplos de cómo usarlas.  |
| Ejemplos de tipos de programación | Una visión general de cómo escribir código de programa con instrucciones/funciones/tipos de datos diferentes. Este capítulo también contiene sugerencias y explicaciones de programación. |

## Referencias

| Referencia   | ID de documento |
|--|-----------------|
| <i>Manual de referencia técnica - RAPID Overview</i> | 3HAC050947-005  |
| <i>Technical reference manual - RAPID kernel</i>     | 3HAC050946--001 |
| <i>Application manual - Controller software IRC5</i> | 3HAC050798--001 |
| <i>Manual del operador - RobotStudio</i>             | 3HAC032104-005  |

*Continúa en la página siguiente*

## Revisiones

| Revisión | Descripción  |
|----------|--|
| -        | <p>Publicado con RobotWare 6.0.</p> <p> <b>Nota</b></p> <p>En el caso del IRC5 con RobotWare 5, consulte el manual 3HAC16581-001.</p>   |
| A        | <p>Publicado con RobotWare 6.01.</p> <ul style="list-style-type: none"> <li>• Añadidos los siguientes tipos de datos, instrucciones y funciones:           <ul style="list-style-type: none"> <li><a href="#">AliasIOReset - Restablecer una señal de E/S con un nombre de alias en la página 40</a>, <a href="#">TriggJIOs - Movimientos de ejes del robot con eventos de E/S en la página 949</a></li> </ul> </li> <li>• Añadida información acerca de los robots de 7 ejes al tipo de dato <a href="#">confdata - Datos de configuración del robot en la página 1648</a>.</li> </ul>  |
| B        | <p>Publicado con RobotWare 6.02.</p> <ul style="list-style-type: none"> <li>• Añadidos los siguientes tipos de datos, instrucciones y funciones comunes:           <ul style="list-style-type: none"> <li><a href="#">SaveCfgData - Guardar parámetros del sistema a un archivo en la página 651</a>, <a href="#">cfgdomain - Dominio de configuración en la página 1646</a>, <a href="#">TriggDataCopy - Copiar el contenido de una variable de tipo triggdata en la página 910</a>, <a href="#">TriggDataReset - Restablecer el contenido en una variable de tipo triggdata en la página 912</a>, <a href="#">TriggDataValid - Comprobar si el contenido de una variable de tipo triggdata es válido en la página 1537</a></li> <li><a href="#">Alnput - Lee el valor de una señal analógica de entrada en la página 1157</a>, <a href="#">Dlnput - Lee el valor de una señal digital de entrada en la página 1265</a>, <a href="#">Glnput - Lee el valor de una señal de entrada de grupo en la página 1327</a></li> </ul> </li> <li>• Añadidos todos los tipos de datos, instrucciones y funciones para la opción de RobotWare <i>Integrated Vision</i>.</li> <li>• Añadido un aviso acerca de la distancia de rotura a <a href="#">SoftAct - Activa el servo suave en la página 770</a>.</li> <li>• Añadidas funciones trigonométricas para el tipo de dato <code>dnum</code>:<br/><code>ACosDnum</code>, <code>ASinDnum</code>, <code>ATanDnum</code>, <code>ATan2Dnum</code>, <code>CosDnum</code>, <code>TanDnum</code>, <code>SinDnum</code></li> <li>• Añadidas las instrucciones RAPID para la funcionalidad EGM Path Correction.</li> <li>• Correcciones menores.</li> </ul> |
| C        | <p>Publicado con RobotWare 6.03.</p> <ul style="list-style-type: none"> <li>• Añadida nueva funcionalidad a la instrucción <a href="#">MotionProcessModeSet - Configuración del modo de proceso de movimientos en la página 377</a>.</li> <li>• Añadidas instrucciones, funciones y tipos de datos de <i>CAP</i>.</li> <li>• Añadidas instrucciones y funciones de <i>Cyclic bool</i>.</li> <li>• Se añadieron instrucciones y funciones relacionadas con la seguridad funcional.</li> <li>• <code>signalxx</code> es ahora un tipo de dato de semivalor que permite operaciones orientadas a valores; consulte <a href="#">signalxx - Señales digitales y analógicas en la página 1763</a>.</li> <li>• Correcciones menores.</li> </ul>   |

Continúa en la página siguiente



| Revisión | Descripción   |
|----------|---|
| D        | <p>Publicado con RobotWare 6.04.</p> <ul style="list-style-type: none"> <li>• Añadida la sección <a href="#">SafetyControllerGetOpModePinCode - Obtener el código pin del modo de funcionamiento en la página 1468</a>.</li> <li>• Se añadieron nuevas instrucciones para visualizar condiciones de espera; consulte <a href="#">UIMsgWrite - Cuadro de diálogo de mensaje de usuario tipo sin espera en la página 1011</a>.</li> <li>• Se añadió recuperación de errores <code>ERR_TASKNAME</code> donde faltaba.</li> <li>• Añadidos <a href="#">SetLeadThrough - Activar y desactivar proceso de guiado en la página 711</a> y <a href="#">IsLeadThrough - Comprobar el estado del proceso de guiado en la página 1355</a>.</li> <li>• Correcciones menores.</li> </ul>  |
| E        | <p>Publicado con RobotWare 6.05.</p> <ul style="list-style-type: none"> <li>• Se han añadido nuevas instrucciones para <i>Integrated Vision</i>, consulte <a href="#">AliasCamera - Define un dispositivo de cámara con un nombre de alias en la página 34</a>.</li> <li>• Se han añadido nuevas instrucciones matemáticas para cálculo de matrices, consulte <a href="#">MatrixSolve - Soluciona un sistema de ecuaciones lineales en la página 364</a>.</li> <li>• Se han añadido nuevas instrucciones para la distancia recorrida y el tiempo ocupado para Track Motion, consulte <a href="#">ResetAxisDistance - Restablece la información de distancia recorrida para el eje en la página 608</a>.</li> <li>• Se ha añadido la posibilidad de interrumpir todas las instrucciones <code>UIxx</code> y <code>TPxx</code> con un booleano persistente.</li> <li>• Se han añadido nuevas instrucciones para manejar múltiples variables desde un dispositivo, consulte <a href="#">WriteVarArr - Escribe múltiples variables en un dispositivo sensor en la página 1119</a>.</li> <li>• Se han eliminado las instrucciones <code>DitherAct</code> y <code>DitherDeact</code>.</li> <li>• Correcciones menores.</li> </ul> |
| F        | <p>Publicado con RobotWare 6.06.</p> <ul style="list-style-type: none"> <li>• Añadida la sección <a href="#">FitCircle: Se ajusta a un círculo con puntos 3D en la página 215</a>.</li> <li>• Añadida la sección <a href="#">GetJointData: Obtener datos conjuntos específicos en la página 233</a>.</li> <li>• Añadidos <a href="#">GetTSPStatus: Obtener el estado del panel de selección de tareas actuales en la página 1323</a> y <a href="#">tsp_status: Estado de panel de selección de tareas en la página 1817</a>.</li> <li>• Añadida la sección <a href="#">IsBrakeCheckActive: Comprobar si la prueba de frenos está en marcha en la página 1345</a>.</li> <li>• Añadidos <a href="#">TaskIsActive: Comprobar si una tarea normal está activa en la página 1519</a> y <a href="#">TaskIsExecuting: Comprobar si la tarea se está ejecutando en la página 1521</a>.</li> <li>• Correcciones menores.</li> </ul>  |

Continúa en la página siguiente

| Revisión | Descripción   |
|----------|---|
| G        | <p>Publicado con RobotWare 6.07.</p> <ul style="list-style-type: none"> <li>Se añadieron las funciones <a href="#">MaxExtLinearSpeed - Velocidad máxima de eje adicional en la página 1370</a>, <a href="#">MaxExtReorientSpeed - Velocidad de giro máxima de eje adicional en la página 1371</a> y <a href="#">MaxRobReorientSpeed - Velocidad de reorientación máxima del robot en la página 1372</a>. Para <a href="#">speeddata - Datos de velocidad en la página 1767</a>, el valor predefinido <math>v_{max}</math> se cambió para usar los valores máximos devueltos por estas funciones.</li> <li>Se añadió la función <a href="#">CrossProd - Producto cruzado de dos vectores pos en la página 1243</a>.</li> <li>Se añadió información sobre cuadrantes para robots de 7 ejes en la sección <a href="#">confdata - Datos de configuración del robot en la página 1648</a>.</li> <li>Se actualizó información sobre limitaciones para <a href="#">MoveC - Mueve el robot en círculo en la página 389</a>.</li> <li>Se añadieron limitaciones para <a href="#">Comentario - Comentario en la página 154</a>.</li> <li>Se aclaró la ejecución del programa al utilizar <code>\LockAxis4</code> en la instrucción <code>SingArea</code>. También se retiró nota indicando que <code>\LockAxis4</code> solo puede utilizarse en robots de eslabones en serie. Consulte <a href="#">SingArea - Define el método de interpolación alrededor de puntos singulares en la página 730</a>.</li> <li>Actualizada la sección <a href="#">PathResol - Ajusta la resolución de la trayectoria en la página 530</a>.</li> <li>Cambios menores en las limitaciones en <a href="#">WriteCfgData - Escribe un atributo de un parámetro del sistema en la página 1107</a>.</li> <li>Se añadieron las instrucciones <a href="#">CapAPTrSetupAI - Configurar un seguimiento del punto controlado por señales de entrada analógicas en la página 86</a>, <a href="#">CapAPTrSetupAO - Configurar un seguimiento del punto controlado por señales de salida analógicas en la página 89</a>, <a href="#">CapAPTrSetupPERS - Configurar un seguimiento del punto controlado por variables persistentes en la página 92</a> y el tipo de datos <a href="#">capaptrreferencedata - Datos de configuración variable para seguimiento del punto en la página 1622</a>.</li> <li>Se añadieron modos de seguimiento 13, 14 y 15 al tipo de datos <code>trackdata</code>.</li> <li>Se aclaró el orden de giro en <a href="#">OrientZYX - Genera una orientación a partir de ángulos Euler en la página 1396</a> y <a href="#">RelTool - Hace un desplazamiento respecto de la herramienta en la página 1453</a>.</li> <li>Se añadieron las instrucciones <code>EGMStreamStart</code>, <code>EGMStreamStop</code> y <code>EGMWaitCond</code>.</li> </ul> |
| H        | <p>Publicado con RobotWare 6.08.</p> <ul style="list-style-type: none"> <li>Correcciones menores.</li> <li>Actualizadas las limitaciones con información sobre la declaración de variable de señal para <a href="#">AliasIO - Define una señal de E/S con un nombre de alias en la página 37</a>.</li> <li>Añadidos los datos predefinidos de <code>PRESS_TENDING_MODE</code> a <a href="#">MotionProcessModeSet - Configuración del modo de proceso de movimientos en la página 377</a>.</li> <li>Añadida la instrucción <code>MovePnP</code> y los tipos de datos <code>pnpdata</code>.</li> <li>Añadida la instrucción <code>SoftElbow</code>.</li> <li>Cambios menores en <a href="#">zonedata - Datos de zonas en la página 1831</a>. Referencia a <i>Manual de referencia técnica - RAPID Overview</i>, donde se describe de manera más detallada los cambios en las trayectorias de esquinas.</li> <li>Añadida la función <code>GetUASUserName</code>.</li> </ul>  |

| Revisión | Descripción  |
|----------|--|
| J        | <p>Publicado con RobotWare 6.09.</p> <ul style="list-style-type: none"> <li>• Cambios realizados en texto y ejemplos para comunicación, centrándose menos en los puertos serie y más en los dispositivos de E/S en general.</li> <li>• Descripción de <code>ERR_UNKINO</code> actualizada en la sección <i>errnum - Número de error en la página 1664</i>.</li> <li>• Descripción actualizada sobre el nivel de acceso de las señales en <i>WZDOSet - Activación de salidas digitales basadas en zonas mundo en la página 1128</i>.</li> <li>• Añadido el argumento faltante <code>\StreamStart</code> a las instrucciones <code>EGMActJoing</code> y <code>EGMActPose</code>.</li> <li>• Descripción actualizada del argumento <code>RestartDist</code> en la instrucción <code>CapRefresh</code>.</li> <li>• Información de limitación actualizada para varias instrucciones.</li> <li>• Las instrucciones <code>WaitSensor</code>, <code>WaitWObj</code> y <code>Errnum</code> han sido actualizadas con información acerca de <code>ERR_CNV_OBJ_LOST</code>.</li> <li>• Ejemplo de código corregido en <i>switch - Parámetros opcionales en la página 1783</i>.</li> </ul> <p>Instrucciones añadidas:</p> <ul style="list-style-type: none"> <li>• <i>CAPSetStopMode - Establecer el modo de paro para errores de ejecución en la página 127</i></li> <li>• <i>GetTorqueMargin - Lee el menor margen de par en la página 238</i></li> <li>• <i>PathLengthReset - Restablece el valor de longitud de trayectoria actual del contador en la página 508</i></li> <li>• <i>PathLengthStart - Activa el contador que monitoriza la longitud de trayectoria en la página 510</i></li> <li>• <i>PathLengthStop - Detiene el contador que monitoriza la longitud de trayectoria en la página 512</i></li> <li>• <i>ResetTorqueMargin - Restablecer el menor margen de par en la página 614</i></li> </ul> <p>Funciones añadidas:</p> <ul style="list-style-type: none"> <li>• <i>PathLengthGet - Lee el valor de longitud de trayectoria actual del contador en la página 1406</i></li> </ul> <p>Tipos de datos añadidos:</p> <ul style="list-style-type: none"> <li>• <i>capstopmode - Define modos de paro para CAP en la página 1634</i></li> </ul> |
| K        | <p>Publicado con RobotWare 6.10.</p> <ul style="list-style-type: none"> <li>• Corrección en la función <code>PathLengthGet</code>.</li> <li>• Añadida información relacionada con IRB 5500 en las secciones <i>ConfJ - Controla la configuración durante el movimiento de los ejes en la página 156</i> y <i>ConfL - Monitoriza la configuración durante el movimiento lineal en la página 158</i></li> <li>• La información relacionada con <i>Externally Guided Motion</i> se ha movido a un manual independiente, <i>3HAC073319--001</i>.</li> </ul>  |
| L        | <p>Publicado con RobotWare 6.10.02.</p> <ul style="list-style-type: none"> <li>• Información actualizada las para instrucciones <code>SetAllDataVal</code> y <code>SetDataSearch</code>.</li> </ul>  |

Continúa en la página siguiente

| Revisión | Descripción  |
|----------|--|
| M        | <p>Publicado con RobotWare 6.11.</p> <ul style="list-style-type: none"> <li>• Información actualizada para la función <code>TextGet</code>.</li> <li>• Se ha añadido una nueva recuperación en caso de error para enum <code>ERR_CAM_NOT_ON_NETWORK</code>.</li> <li>• Instrucciones añadidas: <ul style="list-style-type: none"> <li>- <i>SimCollision: simular una colisión en la página 729</i></li> <li>- <i>StrFormat: dar formato a una cadena en la página 838</i></li> </ul> </li> <li>• Funciones añadidas: <ul style="list-style-type: none"> <li>- <i>CamGetMode: obtener el modo actual de la cámara en la página 1224</i></li> </ul> </li> <li>• Tipos de datos añadidos: <ul style="list-style-type: none"> <li>- <i>camerastatus: estado de comunicación de la cámara en la página 1618</i></li> </ul> </li> <li>• <code>STIndGun</code> y <code>STIndGunReset</code> actualizados con información de la herramienta servo.</li> <li>• Se ha añadido una nueva instrucción <code>IOEventMessage</code> en <i>IOEventMessage: activar/desactivar mensajes de eventos de E/S desde el dispositivo en la página 299</i>.</li> <li>• Se ha añadido una nueva función <code>IsCollFree</code> en <i>IsCollFree: comprueba si la posición colisionaría en la página 1346</i>.</li> <li>• Función <code>EGMGetState</code> y tipos de datos <code>egmframetype</code> y <code>egmident</code> retirados del manual y trasladados a AM Externally Guided Motion.</li> </ul> |
| N        | <p>Publicado con RobotWare 6.12.</p> <ul style="list-style-type: none"> <li>• Instrucciones añadidas: <ul style="list-style-type: none"> <li>- <i>CamStartSetParameter - Iniciar la operación de ajuste de un parámetro en la página 74</i></li> <li>- <i>CamWaitSetParameter - Esperar hasta que una operación de ajuste esté lista en la página 79</i></li> </ul> </li> <li>• Funciones añadidas: <ul style="list-style-type: none"> <li>- <i>Rand - Genera un número aleatorio en la página 1431</i></li> </ul> </li> <li>• Se ha actualizado la información sobre los archivos .xml en la función <i>TextTabGet - Obtiene el número de una tabla de textos en la página 1535</i>.</li> <li>• Se han añadido nuevos argumentos (de <code>Arg3</code> a <code>Arg6</code>) para <code>StrFormat</code> y <code>TextGet</code>.</li> <li>• Añadidas limitaciones para la instrucción <code>StartLoad</code>, consulte <i>StartLoad - Carga de programa durante la ejecución en la página 792</i>.</li> <li>• Se ha actualizado la información sobre el uso de la instrucción <code>Stop</code> en rutinas de evento, consulte <i>Stop - Detención de la ejecución del programa en la página 822</i>.</li> </ul>   |

# 1 Instrucciones

## 1.1 AccSet - Reduce la aceleración

### Utilización

`AccSet` se utiliza en el caso de manejar cargas frágiles o para reducir las vibraciones y errores en la trayectoria. Permite obtener aceleraciones y deceleraciones más lentas, lo que da lugar a movimientos más suaves en el robot. Esta instrucción sólo puede usarse en la tarea principal `T_ROB1` o, si se cuenta con un sistema MultiMove, en las tareas de movimiento.

### Ejemplos básicos

Los ejemplos siguientes ilustran la instrucción `AccSet`.

#### Ejemplo 1

```
AccSet 50, 100;
```

La aceleración se limita al 50% del valor normal.

#### Ejemplo 2

```
AccSet 100, 50;
```

La rampa de aceleración está limitada al 50 % del valor normal, lo que significa que el tiempo en alcanzar la aceleración se duplica.

#### Ejemplo 3

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

La rampa de deceleración al decelerar hacia un punto fino se limita al 50% del valor normal.

### Argumentos

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

Acc

Tipo de dato: num

Aceleración y deceleración como porcentaje de los valores normales. El 100 % corresponde a la aceleración máxima. Un valor de entrada < 20 % da lugar a un 20 % de la aceleración máxima.

Ramp

Tipo de dato: num

La velocidad con que aumentan la aceleración y la deceleración como porcentaje los valores normales. La reducción de este valor permite limitar la vibración. El 100 % corresponde a la velocidad máxima. Un valor de entrada < 10 % proporciona un 10 % de la velocidad máxima.

*Continúa en la página siguiente*

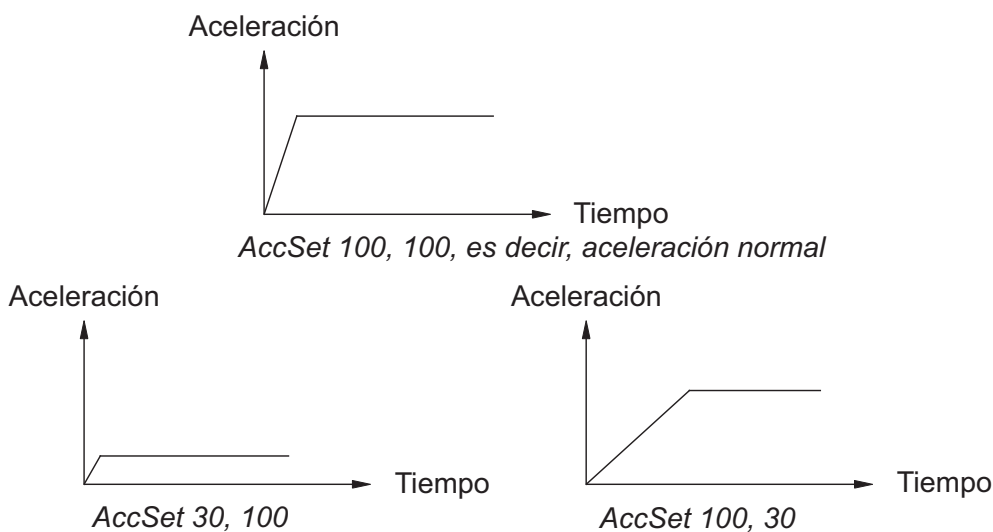
# 1 Instrucciones

## 1.1 AccSet - Reduce la aceleración

RobotWare - OS

Continuación

Las figuras muestran cómo la reducción de la aceleración da lugar a unos movimientos más suaves.



xx0500002146

[ \FinePointRamp ]

Tipo de dato: num

La velocidad a la que la deceleración disminuye como porcentaje de los valores normales. El parámetro sólo afecta a la rampa cuando el robot decelera hacia un punto fino. En un punto fino, el valor de rampa de deceleración es una combinación de este parámetro y el valor Ramp,  $\text{Ramp} * \text{FinePointRamp}$ . El parámetro debe ser mayor que 0 y estar dentro del intervalo de 0 a 100%.

Si no se usa este argumento opcional, el valor `FinePointRamp` se cambia al valor predeterminado, 100%.

## Ejecución de programas

La aceleración se aplica a la siguiente instrucción de movimiento ejecutada, tanto en el robot como en los ejes externos hasta que se ejecute una nueva instrucción `AccSet`.

Los valores predeterminados (`AccSet 100, 100`) se establecen automáticamente

- cuando se utiliza el modo de reinicio **Restablecer RAPID**
- al cargar un nuevo programa o un nuevo módulo
- al iniciar la ejecución del programa desde el principio
- al mover el puntero del programa a `main`
- al mover el puntero del programa a una rutina
- al mover el puntero de programa de una forma que se pierde el orden de la ejecución.

## Sintaxis

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

Continúa en la página siguiente

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

### Información relacionada

| Para obtener más información sobre                               | Consulte  |
|--|---|
| Datos de parámetros de movimiento                                | <a href="#">motsetdata - Datos de parámetros de movimiento en la página 1705</a>                          |
| Reducción de la aceleración del TCP a lo largo de la trayectoria | <a href="#">PathAccLim - Reduce la aceleración del TCP a lo largo de la trayectoria en la página 504</a>  |
| Definición de la velocidad máxima                                | <a href="#">VelSet - Cambia la velocidad programada en la página 1029</a>                                 |
| Control de aceleración en el sistema de coordenadas mundo        | <a href="#">WorldAccLim - Control de aceleración en el sistema de coordenadas mundo en la página 1096</a> |
| Instrucciones de posicionamiento                                 | <a href="#">Manual de referencia técnica - RAPID Overview</a>   |

# 1 Instrucciones

---

## 1.2 ActEventBuffer - Activación de búfer de eventos

RobotWare - OS

## 1.2 ActEventBuffer - Activación de búfer de eventos

---

### Descripción

`ActEventBuffer` no se utiliza para activar el uso del búfer de eventos en la tarea actual de programa de movimiento.

Las instrucciones `ActEventBuffer` y `DeactEventBuffer` deben usarse al combinar una aplicación con puntos finos y una aplicación continua en la que las señales deben activarse de antemano debido al uso de equipos de proceso lentos.

Esta instrucción sólo puede usarse en la tarea principal `T_ROB1` o, si se cuenta con un sistema `MultiMove`, en las tareas de movimiento.

---

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción `ActEventBuffer`:

#### Ejemplo 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` desactiva el búfer de eventos configurado. Al utilizar una aplicación con puntos finos, el arranque del robot desde el punto fino será más rápido. Al activar el búfer de eventos con `ActEventBuffer`, es posible activar las señales de antemano para una aplicación que usa equipos de proceso lentos.

---

### Ejecución de programas

El uso de un búfer de eventos se aplica a la siguiente instrucción de movimiento del robot ejecutada, de cualquier tipo, y se mantiene vigente hasta la ejecución de una instrucción `DeactEventBuffer`.

La instrucción esperará hasta que el robot y los ejes externos hayan alcanzado el punto de paro (`ToPoint` de la instrucción de movimiento actual) antes de la activación del búfer de eventos. Por tanto, se recomienda programar con un punto fino la instrucción de movimiento precedente a `ActEventBuffer`.

El valor predeterminado (`ActEventBuffer`) se establece automáticamente

- cuando se utiliza el modo de reinicio **Restablecer RAPID**
- al cargar un nuevo programa o un nuevo módulo
- al iniciar la ejecución del programa desde el principio
- al mover el puntero del programa a `main`
- al mover el puntero del programa a una rutina
- al mover el puntero de programa de una forma que se pierde el orden de la ejecución.

*Continúa en la página siguiente*



### Limitaciones

ActEventBuffer no puede ejecutarse en rutinas de RAPID que estén conectadas a los siguientes eventos especiales del sistema: PowerOn, Stop, QStop, Restart o Step.

### Sintaxis

```
ActEventBuffer ' ; '
```

### Información relacionada

| Para obtener más información sobre | Consulte  |
|------------------------------------|---|
| Desactivación de búfer de eventos  | <a href="#">DeactEventBuffer - Desactivación de búfer de eventos en la página 185</a> |
| Configuración de Event preset time | <i>Manual de referencia técnica - Parámetros del sistema</i>                          |
| Datos de parámetros de movimiento  | <a href="#">motsetdata - Datos de parámetros de movimiento en la página 1705</a>      |

# 1 Instrucciones

---

## 1.3 ActUnit - Activa una unidad mecánica RobotWare - OS

### 1.3 ActUnit - Activa una unidad mecánica

---

#### Utilización

ActUnit se utiliza para activar una unidad mecánica.

Puede usarse para determinar qué unidad debe estar activa, por ejemplo cuando se utilizan unidades de accionamiento comunes.

Esta instrucción sólo puede usarse en la tarea principal T\_ROB1 o, si se cuenta con un sistema MultiMove, en las tareas de movimiento.

#### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción ActUnit:

#### Ejemplo 1

```
ActUnit orbit_a;
```

Activación de la unidad mecánica orbit\_a.

#### Argumentos

```
ActUnit MechUnit
```

MechUnit

**Mechanical Unit**

Tipo de dato: mecunit

El nombre de la unidad mecánica a activar.

#### Ejecución de programas

Cuando la trayectoria real de los robots y de los ejes externos ha sido completada, la trayectoria del nivel de trayectoria actual se borra y se activa la unidad mecánica especificada. Esto significa que es controlada y monitorizada por el robot.

Si varias unidades mecánicas comparten una misma unidad de accionamiento, la activación de una de estas unidades mecánicas también conecta la unidad a la unidad de accionamiento común.

#### Limitaciones

Si esta instrucción va precedida de una instrucción de movimiento, ésta última debe programarse con un punto de paro (zonedata fine), no un punto de paso. De lo contrario, no será posible reanudar la ejecución tras una caída de suministro eléctrico.

ActUnit no puede ejecutarse en rutinas de RAPID que estén conectadas a los siguientes eventos especiales del sistema: PowerOn, Stop, QStop, Restart, Reset o Step.

Es posible usar ActUnit - DeactUnit en el nivel StorePath, pero al ejecutar RestoPath deben estar activadas las mismas unidades mecánicas que al ejecutar StorePath. Esta operación en la grabadora de trayectorias y la trayectoria del nivel básico estarán intactas, pero la trayectoria del nivel StorePath se borrará.

*Continúa en la página siguiente*

### Sintaxis

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

### Información relacionada

| Para obtener más información sobre                   | Consulte   |
|--|--|
| Desactivación de unidades mecánicas                  | <a href="#">DeactUnit - Desactiva una unidad mecánica en la página 187</a>                     |
| Unidades mecánicas                                   | <a href="#">mecunit - Unidad mecánica en la página 1703</a>                                    |
| Más ejemplos   | <a href="#">DeactUnit - Desactiva una unidad mecánica en la página 187</a>                     |
| Comprobar si una unidad mecánica está activada o no. | <a href="#">IsMechUnitActive - Indica si una unidad mecánica está activa en la página 1357</a> |
| Grabadora de trayectorias                            | <a href="#">PathRecMoveBwd - Hace retroceder la grabadora de trayectorias en la página 514</a> |

# 1 Instrucciones

---

## 1.4 Add - Suma un valor numérico

RobotWare - OS

## 1.4 Add - Suma un valor numérico

---

### Utilización

Add se utiliza para sumar o restar un valor a o de una variable o una variable persistente de tipo numérico.

---

### Ejemplos básicos

Los ejemplos siguientes ilustran la instrucción Add.

#### Ejemplo 1

```
Add reg1, 3;  
3 se suma a reg1, es decir reg1:=reg1+3.
```

#### Ejemplo 2

```
Add reg1, -reg2;  
El valor de reg2 se resta de reg1, es decir,reg1:=reg1-reg2.
```

#### Ejemplo 3

```
VAR dnum mydnum:=5;  
Add mydnum, 500000000;  
500000000 se suma a mydnum, es decir mynum:=mynum+500000000.
```

#### Ejemplo 4

```
VAR dnum mydnum:=5000;  
VAR num mynum:=6000;  
Add mynum, DnumToNum(mydnum \Integer);  
5000 se suma a mynum, es decir mynum:=mynum+5000. Debe usar DnumToNum para obtener un valor numérico num que puede usar junto con la variable nummynum.
```

---

### Argumentos

Add Name | Dname AddValue | AddDvalue

Name

**Tipo de dato:** num  
El nombre de la variable o de la variable persistente que se desea cambiar.

Dname

**Tipo de dato:** dnum  
El nombre de la variable o de la variable persistente que se desea cambiar.

AddValue

**Tipo de dato:** num  
El valor a sumar.

AddDvalue

**Tipo de dato:** dnum  
El valor a sumar.

*Continúa en la página siguiente*

### Limitaciones

Si el valor añadir es del tipo `dnum` y la variable/persistente que debe cambiarse es de tipo `num`, se generará un error de tiempo de ejecución. La combinación de argumentos no es posible (consulte el Ejemplo 4 anterior acerca de cómo resolver esto).

### Sintaxis

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 > ';' 
```

### Información relacionada

| Para obtener más información sobre   | Consulte  |
|--|---|
| Incremento de una variable en 1  | <a href="#">Incr - Aumenta en 1 un valor en la página 262</a> |
| Decremento de una variable en 1  | <a href="#">Decr - Reduce un número en 1 en la página 189</a> |
| Cambio de un dato mediante una expresión arbitraria, por ejemplo, una multiplicación | <a href="#">":=" - Asigna un valor en la página 42</a>        |

# 1 Instrucciones

---

## 1.5 AliasCamera - Define un dispositivo de cámara con un nombre de alias *Integrated Vision*

## 1.5 AliasCamera - Define un dispositivo de cámara con un nombre de alias

---

### Utilización

**AliasCamera** Se utiliza para definir una cámara con un alias o para usar cámaras en los módulos de tarea integrados. Las cámaras que tengan un alias pueden usarse en programas genéricos predefinidos. La instrucción **AliasCamera** debe ejecutarse antes de utilizar la cámara real.

---

### Ejemplos básicos

Los siguientes ejemplos ilustran la instrucción **AliasCamera**.

#### Ejemplo 1

```
VAR cameradev mycamera;  
...  
PROC prog_start()  
  AliasCamera "CAMERA1", mycamera;  
  ...  
  CamReqImage mycamera;
```

La rutina `prog_start` se ejecuta al principio del programa de RAPID. La instrucción **AliasCamera** busca la variable predefinida del dispositivo cámara de RAPID con el nombre `CAMERA1`, y su contenido se copia a `mycamera`. Desde ahora, es posible acceder a la cámara con un dispositivo cámara `mycamera`.

#### Ejemplo 2

```
VAR cameradev mycamera;  
PROC procl()  
  IF GetTaskName() = "T_ROB_L" THEN  
    AliasCamera CAMERA_L, mycamera;  
  ELSE  
    AliasCamera CAMERA_R, mycamera;  
  ENDIF  
  ...  
  CamReqImage mycamera;
```

La rutina `procl` está conectada al evento `START` en los parámetros del sistema. El programa que define el dispositivo cámara `mycamera` se conecta a la cámara configurada `CAMERA_L` o `CAMERA_R` al inicio del programa.

---

### Argumentos

`AliasCamera CameraName | FromCamera ToCamera`

CameraName

**Tipo de dato:** string

El identificador de la cámara según la configuración `Communication` de los parámetros del sistema. La instrucción **AliasCamera** busca la variable predefinida del dispositivo cámara de RAPID (datos instalados) con el nombre utilizado en `CameraName`, y copia su contenido.

FromCamera

**Tipo de dato:** cameradev

*Continúa en la página siguiente*

---

## 1.5 AliasCamera - Define un dispositivo de cámara con un nombre de alias

*Integrated Vision*

*Continuación*

El identificador de cámara en la configuración Communication de los parámetros del sistema desde la que se copia el dispositivo cámara. La cámara debe estar definida en los parámetros del sistema.

ToCamera

Tipo de dato: cameradev

El identificador de la cámara según el programa al que se copia el dispositivo cámara. En el programa RAPID hay que declarar cameradev.

### Ejecución de programas

El contenido del dispositivo cámara se copia desde la cámara indicada en el argumento CameraName o FromCamera al dispositivo cámara indicado en el argumento ToCamera.

### Gestión de errores

Se generan los siguientes errores recuperables que pueden gestionarse con un gestor de errores. La variable del sistema ERRNO se establecerá en:

|                  |  |
|------------------|--|
| ERR_ALIASCAM_DEF | La cámara en el argumento CameraName o el cameradev utilizado en el argumento FromCamera no está definida en la configuración Communication de los parámetros del sistema. O la ToCamera no está declarada en el programa de RAPID o ya está definida en la configuración Communication de los parámetros del sistema. |
|------------------|--|

### Limitación

Al iniciar el programa, no se puede usar el alias de la cámara sino hasta después de ejecutar la instrucción AliasCamera.

La instrucción AliasCamera **debe** estar situada:

- en la rutina de evento ejecutada al iniciar el programa (evento START).
- o bien en la parte del programa que se ejecuta tras cada inicio del programa (antes de usar la cámara).

Para evitar errores, no se recomienda utilizar la reconexión dinámica de una cámara AliasCamera a varias cámaras físicas diferentes.

### Sintaxis

```
AliasCamera
  [ CameraName ':=' ] < expression (IN) of string >
  | FromCamera ':=' < variable (VAR) of cameradev > ','
  [ ToCamera ':=' < variable (VAR) of cameradev > ] ';'

```

### Información relacionada

| Para obtener más información sobre | Consulte  |
|------------------------------------|---|
| Definición de dispositivos cámara  | <a href="#">cameradev - dispositivo de cámara en la página 1617</a> |
| Configuración de cámaras           | <i>Manual de referencia técnica - Parámetros del sistema</i>        |
| Definición de rutinas de evento    | <i>Manual de referencia técnica - Parámetros del sistema</i>        |

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.5 AliasCamera - Define un dispositivo de cámara con un nombre de alias

*Integrated Vision*

*Continuación*

| Para obtener más información sobre | Consulte  |
|------------------------------------|---|
| Integrated Vision                  | <i>Manual de aplicaciones - Integrated Vision</i> |



---

## 1.6 AliasIO - Define una señal de E/S con un nombre de alias

---

### Utilización

AliasIO se utiliza para definir una señal de cualquier tipo con un nombre de alias o para usar señales en los módulos de tarea incorporados.

Las señales con nombres de alias pueden usarse en los programas genéricos predefinidos, sin ninguna modificación del programa antes de ejecutarlo en distintas instalaciones de robot.

Es necesario ejecutar la instrucción AliasIO antes de cualquier uso de la señal en sí. Consulte [Ejemplos básicos en la página 37](#) para los módulos cargados y [Más ejemplos en la página 38](#) para los módulos instalados.

---

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción AliasIO:

Consulte también [Más ejemplos en la página 38](#).

#### Ejemplo 1

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

La rutina prog\_start está conectada al evento START en los parámetros del sistema. La señal digital de salida definida por el programa alias\_do está conectada a la señal digital de salida configurada config\_do en el momento del inicio del programa.

---

### Argumentos

AliasIO FromSignal ToSignal

#### FromSignal

Tipo de dato: signalxx o string

#### Módulos cargados:

El identificador de la señal, con un nombre que se corresponde con la configuración (tipo de dato signalxx) de la que se copia el descriptor de la señal. La señal debe estar definida en la configuración de E/S.

#### Módulos instalados o módulos cargados:

Una referencia (CONST, VAR o un parámetro de este tipo) que contenga el nombre de la señal (tipo de dato string) de la que se copia el descriptor de la señal tras buscarlo en el sistema. La señal debe estar definida en la configuración de E/S.

#### ToSignal

Tipo de dato: signalxx

El identificador de la señal de acuerdo con el programa (tipo de dato signalxx) del que se copia el descriptor de la señal. La señal debe estar declarada en el programa de RAPID.

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.6 AliasIO - Define una señal de E/S con un nombre de alias

RobotWare - OS

Continuación

Debe usarse (o buscarse) el mismo tipo de datos para los argumentos `FromSignal` y `ToSignal` y debe ser uno de los tipos `signalxx` (`signalai`, `signalao`, `signaldi`, `signaldo`, `signalgi` o `signalgo`).

### Ejecución de programas

El valor del descriptor de la señal se copia de la señal indicada en el argumento `FromSignal` a la señal indicada en el argumento `ToSignal`.

### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                          | Causa del error  |
|---------------------------------|--|
| <code>ERR_ALIASIO_DEF</code>    | <code>FromSignal</code> no está definido en la configuración de E/S ni <code>ToSignal</code> está declarado en el programa de RAPID, o bien <code>ToSignal</code> no está definido en la configuración de E/S. |
| <code>ERR_ALIASIO_TYPE</code>   | Los tipos de datos de los argumentos <code>FromSignal</code> y <code>ToSignal</code> no son del mismo tipo.  |
| <code>ERR_NO_ALIASIO_DEF</code> | La variable de señal es una variable declarada en RAPID. No se ha conectado a una señal E/S definida en la configuración de E/S con la instrucción <code>AliasIO</code> .                                      |

### Más ejemplos

A continuación aparecen más ejemplos de la instrucción `AliasIO`.

#### Ejemplo 1

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

La rutina `prog_start` se conecta al evento `START` en los parámetros del sistema. La señal digital de entrada definida por el programa `alias_di` es conectada a la señal digital de entrada configurada `config_di` (a través de la constante `config_string`) en el momento del inicio del programa.

### Limitaciones

Al iniciar el programa, no es posible utilizar la señal de alias hasta ejecutar la instrucción `AliasIO`.

La variable de la señal debe declararse globalmente en el módulo. No debe formar parte de un componente de REGISTRO, ni declararse localmente en un procedimiento (de lo contrario, la actualización de la señal después del reinicio tras la caída de la alimentación no funcionará como es debido).

La instrucción `AliasIO` debe estar situada en uno de los lugares siguientes:

- En la rutina de evento ejecutada al iniciar el programa (evento `START`)
- O bien en la parte del programa que se ejecuta después de cada inicio del programa (antes de usar la señal).

Continúa en la página siguiente

Para evitar errores, no se recomienda utilizar la reconexión dinámica de una señal AliasIO a varias señales físicas diferentes.

### Sintaxis

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

```

### Información relacionada

| Para obtener más información sobre                             | Consulte   |
|--|--|
| Restablecer una señal de E/S con un alias                      | <a href="#">AliasIOReset - Restablecer una señal de E/S con un nombre de alias en la página 40</a>                 |
| Instrucciones de entrada/salida                                | <i>Manual de referencia técnica - RAPID Overview</i>   |
| Funcionalidad de entrada/salida en general                     | <i>Manual de referencia técnica - RAPID Overview</i>   |
| Obtención de información acerca del origen de una señal de E/S | <a href="#">GetSignalOrigin - Obtención de información acerca del origen de una señal de E/S en la página 1314</a> |
| Configuración de E/S   | <i>Manual de referencia técnica - Parámetros del sistema</i>   |
| Definición de rutinas de evento                                | <i>Manual de referencia técnica - Parámetros del sistema</i>   |
| Módulos de tarea cargados o instalados                         | <i>Manual de referencia técnica - Parámetros del sistema</i>   |
| <i>Advanced RAPID</i>  | <i>Application manual - Controller software IRC5</i>   |

# 1 Instrucciones

---

## 1.7 AliasIOReset - Restablecer una señal de E/S con un nombre de alias

RobotWare - OS

## 1.7 AliasIOReset - Restablecer una señal de E/S con un nombre de alias

---

### Utilización

AliasIOReset se utiliza para restablecer una señal que se ha utilizado en una llamada anterior a AliasIO.

---

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción AliasIOReset:

#### Ejemplo 1

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

La señal de salida digital `alias_do` definida por el programa se conecta a la señal configurada de salida digital `config_do` del principio del procedimiento `myproc`. La señal `config_do` se define en la configuración de E/S. posteriormente, cuando ya no deba seguir utilizándose `alias_do`, el alias asociado se elimina.

---

### Argumentos

AliasIOReset Signal

Signal

Tipo de dato: `signalxx`

El identificador de señal, de acuerdo con el programa (tipo de dato `signalxx`) que debe restablecerse. La señal debe declararse en el programa de RAPID.

---

### Ejecución de programas

Se elimina todo el alias asociado. La señal no puede utilizarse hasta que se cree un alias asociado con AliasIO.

---

### Limitación

Las señales definidas en la configuración de E/S no pueden restablecerse. Solo pueden utilizarse las señales que se hayan utilizado en una instrucción AliasIO y estén declaradas en el programa de RAPID.

---

### Sintaxis

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

---

Continúa en la página siguiente

## 1.7 AliasIOReset - Restablecer una señal de E/S con un nombre de alias RobotWare - OS Continuación

### Información relacionada

| Para obtener más información sobre                        | Consulte   |
|---|--|
| Definir una señal de E/S con un nombre de alias           | <a href="#">AliasIO - Define una señal de E/S con un nombre de alias en la página 37</a>                           |
| Instrucciones de entrada/salida                           | <i>Manual de referencia técnica - RAPID Overview</i>   |
| Obtener información acerca del origen de una señal de E/S | <a href="#">GetSignalOrigin - Obtención de información acerca del origen de una señal de E/S en la página 1314</a> |
| Funcionalidad de entrada/salida en general                | <i>Manual de referencia técnica - RAPID Overview</i>   |
| Configuración de E/S                                      | <i>Manual de referencia técnica - Parámetros del sistema</i>   |
| Definición de rutinas de evento                           | <i>Manual de referencia técnica - Parámetros del sistema</i>   |
| Módulos de tarea cargados o instalados                    | <i>Manual de referencia técnica - Parámetros del sistema</i>   |
| <i>Advanced RAPID</i>                                     | <i>Application manual - Controller software IRC5</i>   |

# 1 Instrucciones

---

## 1.8 " := " - Asigna un valor RobotWare - OS

### 1.8 " := " - Asigna un valor

---

#### Utilización

La instrucción " := " se utiliza para asignar un nuevo valor a un dato. Este valor puede ser desde un valor constante hasta una expresión aritmética, por ejemplo, `reg1+5*reg3..`

---

#### Ejemplos básicos

Los siguientes ejemplos ilustran la función " := ":  
Consulte también [Más ejemplos en la página 42](#).

##### Ejemplo 1

```
reg1 := 5;
```

reg1 recibe el valor 5.

##### Ejemplo 2

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

Se asigna a reg1 el valor que devuelve el cálculo reg2-reg3.

##### Ejemplo 3

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

counter aumenta en 1.

---

#### Argumentos

```
Data := Value
```

Data

**Tipo de dato:** All  
El dato al que se desea asignar un nuevo valor.

Value

**Tipo de dato:** Same as Data  
El valor deseado.

---

#### Más ejemplos

A continuación aparecen más ejemplos de la instrucción " := ".

##### Ejemplo 1

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

El TCP de tool1 se desplaza 20 mm en la dirección X.

##### Ejemplo 2

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

Se asigna a un elemento de la matriz pallet un valor igual al valor absoluto de la variable value.

---

#### Limitaciones

El dato (cuyo valor se desea cambiar) no debe ser ninguno de los siguientes:

- Una constante

*Continúa en la página siguiente*

- Un tipo de dato sin valor

El dato y el valor deben tener tipos de dato similares (el mismo tipo o el mismo alias).

---

## Sintaxis

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

---

## Información relacionada

| Para obtener más información sobre       | Consulte   |
|--|--|
| Expresiones                              | <i>Manual de referencia técnica - RAPID Overview</i> |
| Tipos de datos sin valor                 | <i>Manual de referencia técnica - RAPID Overview</i> |
| Asignación de un valor inicial a un dato | <i>Manual del operador - IRC5 con FlexPendant</i>    |

# 1 Instrucciones

## 1.9 BitClear - Desactiva un bit específico de un dato byte o dnum RobotWare - OS

### 1.9 BitClear - Desactiva un bit específico de un dato byte o dnum

#### Utilización

BitClear se utiliza para desactivar (poner a 0) un bit especificado de un dato byte o dnum definido.

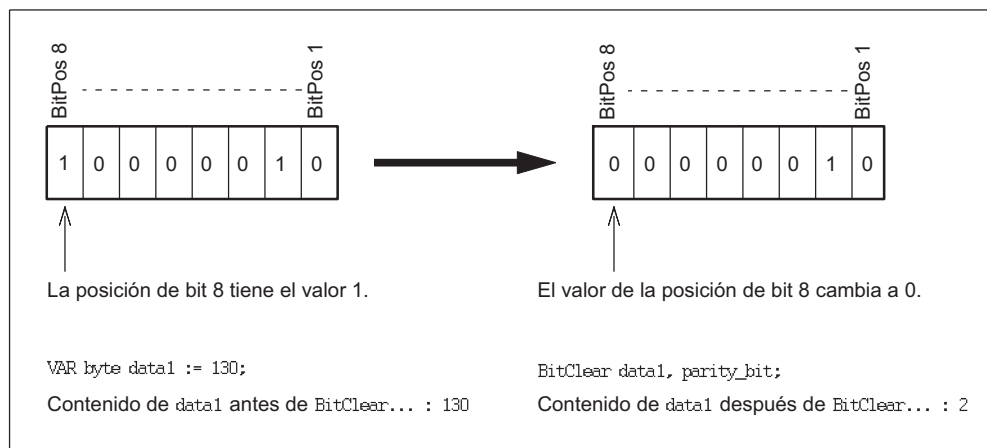
#### Ejemplos básicos

Los ejemplos siguientes ilustran la instrucción BitClear.

##### Ejemplo 1

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

El bit número 8 (parity\_bit) de la variable data1 cambiará a 0; por ejemplo, el contenido de la variable data1 cambiará de 130 a 2 (representación entera). La manipulación de bits de un tipo de dato byte al utilizar BitClear se muestra en la figura que aparece a continuación.



xx0500002147

##### Ejemplo 2

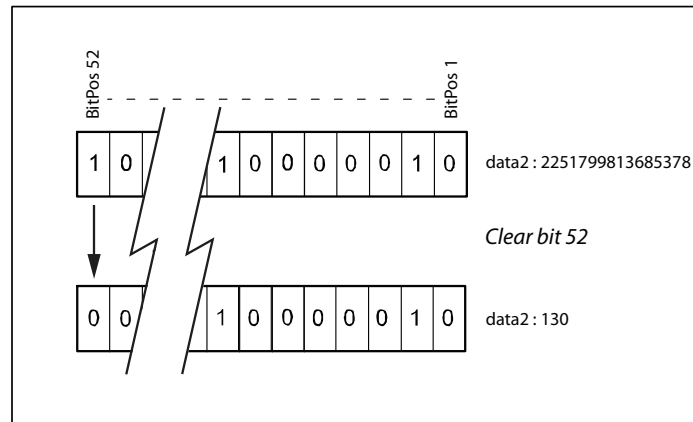
```
CONST num parity_bit := 52;  
VAR dnum data2 := 2251799813685378;  
BitClear data2, parity_bit;
```

El bit número 52 (parity\_bit) de la variable data2 cambia a 0, con lo que el contenido de la variable data2 cambia de 2251799813685378 a 130 (representación

Continúa en la página siguiente



entera). La manipulación de bits de un dato de tipo `dnum` al utilizar `BitClear` se muestra en la figura siguiente.



xx120000014

### Argumentos

```
BitClear BitData | DnumData BitPos
```

BitData

Tipo de dato: `byte`

Los datos de bits a cambiar, en representación entera.

DnumData

Tipo de dato: `dnum`

Los datos de bits de `dnum` a cambiar, en representación entera.

BitPos

**Bit Position**

Tipo de dato: `num`

El valor de bit (1-8) de `BitData`, o la posición de bit (1-52) de `DnumData`, a cambiar a 0.

### Limitaciones

El rango de los tipos de datos `byte` es de 0 a 255 en decimal.

Las posiciones de bit válidas son de la 1 a la 8 para el tipo de dato `byte`.

El rango de los tipos de datos `dnum` es de 0 - 4503599627370495 en decimal.

Las posiciones de bit válidas son de la 1 a la 52 para el tipo de dato `dnum`.

### Sintaxis

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

```

Continúa en la página siguiente

# 1 Instrucciones

---

## 1.9 BitClear - Desactiva un bit específico de un dato byte o dnum

RobotWare - OS

Continuación

---

### Información relacionada

| Para obtener más información sobre                                    | Consulte  |
|---|---|
| Activa un bit específico de un dato byte o dnum                       | <a href="#">BitSet - Activa un bit específico de un dato byte o dnum en la página 47</a>                        |
| Comprobación de si un bit específico de un dato de byte está activado | <a href="#">BitCheck - Comprueba si un bit especificado de un dato de byte está activado en la página 1177</a>  |
| Comprueba si un bit especificado de un dato dnum está activado        | <a href="#">BitCheckDnum - Comprueba si un bit especificado de un dato dnum está activado en la página 1179</a> |
| Otras funciones de bits   | <i>Manual de referencia técnica - RAPID Overview</i>  |
| <i>Advanced RAPID</i>   | <i>Application manual - Controller software IRC5</i>  |

## 1.10 BitSet - Activa un bit específico de un dato byte o dnum

## Utilización

BitSet se utiliza para cambiar a 1 un bit especificado dentro de un dato byte o dnum definido.

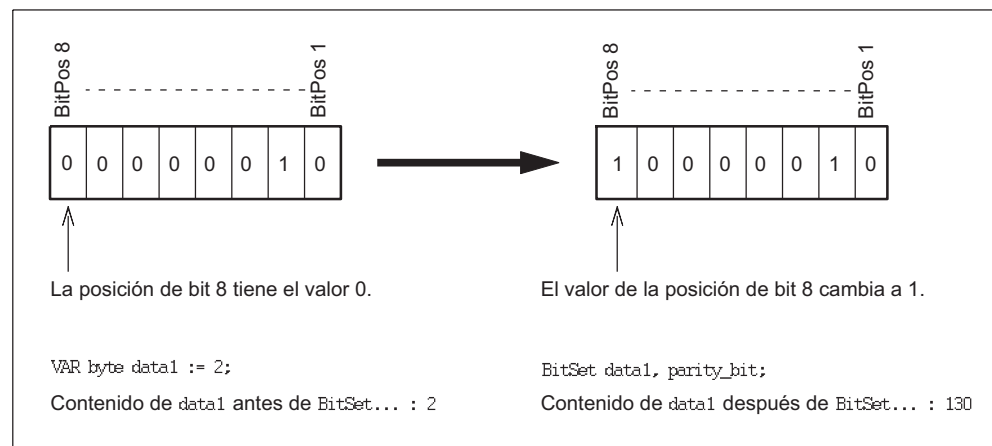
## Ejemplos básicos

Los ejemplos siguientes ilustran la instrucción BitSet.

## Ejemplo 1

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

El bit número 8 (parity\_bit) de la variable data1 cambia a 1; por ejemplo, el contenido de la variable data1 cambia de 2 a 130 (representación entera). La manipulación de bits de un tipo de dato byte cuando se utiliza BitSet se muestra en la figura que aparece a continuación.



xx0500002148

## Ejemplo 2

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

El bit número 52 (parity\_bit) de la variable data2 cambia a 1, con lo que el contenido de la variable data2 cambia de 130 a 2251799813685378 (representación

*Continúa en la página siguiente*

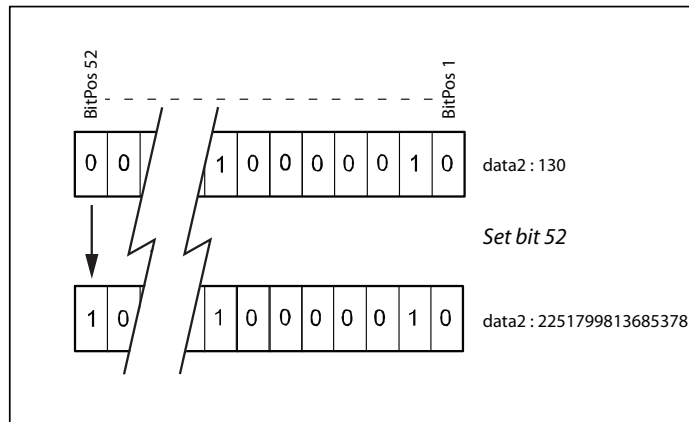
# 1 Instrucciones

## 1.10 BitSet - Activa un bit específico de un dato byte o dnum

RobotWare - OS

Continuación

entera). La manipulación de bits de un dato de tipo `dnum` al utilizar `BitSet` se muestra en la figura siguiente.



xx1200000015

### Argumentos

`BitSet BitData | DnumData BitPos`

`BitData`

Tipo de dato: `byte`

Los datos de bits a cambiar, en representación entera.

`DnumData`

Tipo de dato: `dnum`

Los datos de bits a cambiar, en representación entera.

`BitPos`

*Bit Position*

Tipo de dato: `num`

El valor de bit (1-8) de `BitData`, o la posición de bit (1-52) de `DnumData`, a cambiar a 1.

### Limitaciones

El rango de los tipos de datos `byte` es de 0 a 255 en su representación de número entero.

Las posiciones de bit válidas son de la 1 a la 8 para el tipo de dato `byte`.

El rango de los tipos de datos `dnum` es de 0 a 4503599627370495 en su representación de número entero.

Las posiciones de bit válidas son de la 1 a la 52 para el tipo de dato `dnum`.

### Sintaxis

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

Continúa en la página siguiente

### Información relacionada

| Para obtener más información sobre                                    | Consulte  |
|---|---|
| Desactiva un bit específico de un dato byte o dnum                    | <a href="#">BitClear - Desactiva un bit específico de un dato byte o dnum en la página 44</a>                   |
| Comprobación de si un bit específico de un dato de byte está activado | <a href="#">BitCheck - Comprueba si un bit especificado de un dato de byte está activado en la página 1177</a>  |
| Comprueba si un bit especificado de un dato dnum está activado        | <a href="#">BitCheckDnum - Comprueba si un bit especificado de un dato dnum está activado en la página 1179</a> |
| Otras funciones de bits   | <i>Manual de referencia técnica - RAPID Overview</i>  |
| <i>Advanced RAPID</i>   | <i>Application manual - Controller software IRC5</i>  |

# 1 Instrucciones

---

## 1.11 BookErrNo - Registra un número de error de sistema de RAPID

RobotWare - OS

## 1.11 BookErrNo - Registra un número de error de sistema de RAPID

---

### Utilización

BookErrNo se utiliza para registrar un nuevo número de error de sistema de RAPID.

---

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción BookErrNo:

#### Ejemplo 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;
```

La variable ERR\_GLUEFLOW se asignará a un número de error de sistema que esté libre en el código de RAPID.

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

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

Si la entrada digital di1 contiene 0, el nuevo número de error registrado será elevado y la variable de error de sistema ERRNO cambiará al nuevo número de error registrado. De esta forma, el manejo de errores de los errores generados por el usuario puede realizarse con el gestor de errores, de la forma habitual.

---

### Argumentos

BookErrNo ErrorName

ErrorName

Tipo de dato: errnum

El nombre de la variable del nuevo número de error de sistema de RAPID.

---

### Limitaciones

La variable del nuevo error no debe estar declarada como una variable de rutina.

---

*Continúa en la página siguiente*

## 1.11 BookErrNo - Registra un número de error de sistema de RAPID

*RobotWare - OS*

*Continuación*

La variable del nuevo error debe estar declarada con un valor inicial de -1, lo que indica que este error debe ser un error de sistema de RAPID.

### Sintaxis

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

### Información relacionada

| Para obtener más información sobre | Consulte  |
|------------------------------------|---|
| Gestión de errores                 | <i>Manual de referencia técnica - RAPID Overview</i>                    |
| Número de error                    | <a href="#">errnum - Número de error en la página 1664</a>              |
| Llamada a un gestor de errores     | <a href="#">RAISE - Llamada a un gestor de errores en la página 571</a> |
| <i>Advanced RAPID</i>              | <i>Application manual - Controller software IRC5</i>                    |

# 1 Instrucciones

## 1.12 Break - Interrumpe la ejecución del programa

RobotWare - OS

## 1.12 Break - Interrumpe la ejecución del programa

### Utilización

`Break` se utiliza para provocar la interrupción inmediata de la ejecución de un programa con fines de depuración del código de un programa de RAPID. El movimiento del robot se detiene.

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción `Break`:

#### Ejemplo 1

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

La ejecución del programa se detiene y es posible analizar variables, valores, etc. con fines de depuración.

### Ejecución de programas

La instrucción detiene inmediatamente la ejecución del programa, sin esperar a que los ejes del robot o los ejes externos alcancen sus puntos de destino programados para el movimiento que se está realizando. Posteriormente es posible reanudar la ejecución del programa a partir de la instrucción siguiente.

Si hay una instrucción `Break` en alguna rutina de evento, la ejecución de la rutina se interrumpirá y no se ejecutará ninguna rutina de evento STOP. La rutina de evento se ejecutará desde el principio la próxima vez que tenga lugar el mismo evento.

### Sintaxis

```
Break ; '
```

### Información relacionada

| Para obtener más información sobre           | Consulte   |
|--|--|
| Paro para acciones de programa               | <a href="#">Stop - Detención de la ejecución del programa en la página 822</a> |
| Paro después de un error no recuperable      | <a href="#">EXIT - Finaliza la ejecución del programa en la página 212</a>     |
| Finalización de la ejecución del programa    | <a href="#">EXIT - Finaliza la ejecución del programa en la página 212</a>     |
| Paro de los movimientos del robot únicamente | <a href="#">StopMove - Detiene el movimiento del robot en la página 828</a>    |



## 1.13 CallByVar - Llama a un procedimiento mediante una variable

### Utilización

CallByVar (*Call By Variable*) puede usarse para llamar a procedimientos que tienen nombres específicos, por ejemplo, `proc_name1`, `proc_name2`, `proc_name3` ... `proc_nameX` a través de una variable.

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción CallByVar:  
Consulte también [Más ejemplos en la página 53](#).

#### Ejemplo 1

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

Se llama al procedimiento `proc2`.

### Argumentos

CallByVar Name Number

Name

Tipo de dato: string

La primera parte del nombre del procedimiento, por ejemplo, `proc_name`.

Number

Tipo de dato: num

El valor numérico del número del procedimiento. Este valor se convierte en una cadena y constituye la segunda parte del nombre del procedimiento, por ejemplo, 1. El valor debe ser número un entero positivo.

### Más ejemplos

Más ejemplos sobre cómo hacer selecciones estáticas y dinámicas de llamadas a procedimiento.

#### Ejemplo 1: Selección estática de llamadas a procedimientos

```
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
```

En función de si el valor del registro `reg1` es 1, 2, 3 ó 4, se llama a procedimientos distintos que realizan el tipo adecuado de trabajo para la puerta seleccionada. La ubicación de la puerta se indica en el argumento `door_loc`.

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.13 CallByVar - Llama a un procedimiento mediante una variable

RobotWare - OS

Continuación

Ejemplo 2: Selección dinámica de llamadas a procedimientos con sintaxis de RAPID

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

Se llama al procedimiento `proc2` con el argumento `door_loc`.

Limitación: Todos los procedimientos deben tener un nombre específico, por ejemplo, `proc1`, `proc2`, `proc3`.

Ejemplo 3: Selección dinámica de llamadas a procedimientos con CallByVar

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

Se llama al procedimiento `proc2`.

Limitación: Todos los procedimientos deben tener un nombre específico, como `proc1`, `proc2`, `proc3` y no puede usarse ningún argumento.

### Limitaciones

Sólo puede usarse para llamar a procedimientos que no utilizan parámetros.

No puede usarse para llamar a procedimientos de tipo LOCAL.

La ejecución de `CallByVar` requiere un poco más de tiempo que la ejecución de una llamada normal a un procedimiento.

### Gestión de errores

Pueden generarse los errores recuperables enumerados a continuación. Estos errores pueden ser gestionados en un gestor de errores. El valor de la variable de sistema `ERRNO` cambia a:

| Nombre                     | Causa del error  |
|----------------------------|--|
| <code>ERR_ARGVALERR</code> | El argumento <code>Number</code> es <code>&lt; 0</code> o no es un entero. |
| <code>ERR_REFUNKPRC</code> | Se hace referencia a un procedimiento desconocido.                         |
| <code>ERR_CALLPROC</code>  | Error de llamada a procedimiento (no es un procedimiento).                 |

### Sintaxis

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

### Información relacionada

| Para obtener más información sobre | Consulte  |
|------------------------------------|---|
| Llamadas a procedimientos          | <i>Manual de referencia técnica - RAPID Overview</i><br><i>Manual del operador - IRC5 con FlexPendant</i> |

### 1.14 CamFlush - Elimina los datos de colección de la cámara

#### Utilización

CamFlush se utiliza para vaciar (eliminar) la colección de cameratarget de la cámara.

#### Ejemplos básicos

El ejemplo siguiente ilustra la instrucción CamFlush.

#### Ejemplo 1

```
CamFlush mycamera;
```

Los datos de colección de la cámara mycamera se eliminan.

#### Argumentos

```
CamFlush Camera
```

Camera

**Tipo de dato:** cameradev

**El nombre de la cámara.**

#### Sintaxis

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

#### Información relacionada

| Para obtener más información sobre | Consulte  |
|------------------------------------|---|
| Integrated Vision                  | <i>Manual de aplicaciones - Integrated Vision</i> |

# 1 Instrucciones

---

## 1.15 CamGetParameter - Obtiene distintos parámetros designados de la cámara *Integrated Vision*

## 1.15 CamGetParameter - Obtiene distintos parámetros designados de la cámara

---

### Utilización

`CamGetParameter` se utiliza para obtener parámetros designados que la cámara puede exponer. El usuario debe conocer el nombre del parámetro y su tipo de devolución para poder recuperar su valor.

---

### Ejemplos básicos

El ejemplo siguiente ilustra la instrucción `CamGetParameter`.

#### Ejemplo 1

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

Obtención del parámetro booleano con nombre `Pattern_1.Tool_Enabled_Status` y escritura del valor en el FlexPendant.

---

### Argumentos

```
CamGetParameter Camera ParName [\NumVar] | [\BoolVar] | [\StrVar]
```

Camera

Tipo de dato: cameradev  
El nombre de la cámara.

ParName

**Parameter Name**  
Tipo de dato: string  
El nombre del parámetro en la cámara.

[\NumVar]

Tipo de dato: num  
Una variable (VAR) para almacenar el valor numérico del objeto de datos recuperado.

[\BoolVar]

Tipo de dato: bool  
Una variable (VAR) para almacenar el valor booleano del objeto de datos recuperado.

[\StrVar]

Tipo de dato: string  
Una variable (VAR) para almacenar el valor de cadena del objeto de datos recuperado.

*Continúa en la página siguiente*

## 1.15 CamGetParameter - Obtiene distintos parámetros designados de la cámara

*Integrated Vision*  
*Continuación*

### Ejecución de programas

La instrucción lee directamente el parámetro especificado cuando se ejecuta la instrucción y devuelve el valor.

Si la instrucción se utiliza para leer un resultado del análisis de imagen, asegúrese de que la cámara haya terminado de procesar la imagen antes de obtener los datos.

### Gestión de errores

Pueden generarse los errores recuperables enumerados a continuación. Estos errores pueden ser gestionados en un gestor de errores. El valor de la variable de sistema `ERRNO` cambia a:

| Nombre                 | Causa del error   |
|------------------------|---|
| ERR_CAM_BUSY           | La cámara está ocupada con otra petición y no puede ejecutar la orden actual.   |
| ERR_CAM_COM_TIMEOUT    | Error de comunicación con la cámara. Es probable que la cámara esté desconectada.                                     |
| ERR_CAM_GET_MISMATCH   | El parámetro capturado de la cámara con la instrucción <code>CamGetParameter</code> tiene un tipo de dato incorrecto. |
| ERR_CAM_NOT_ON_NETWORK | La cámara no está conectada.  |

### Sintaxis

```
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 > ] ';'

```

### Información relacionada

| Para obtener más información sobre | Consulte  |
|------------------------------------|---|
| Integrated Vision                  | <i>Manual de aplicaciones - Integrated Vision</i> |

## 1 Instrucciones

---

### 1.16 CamGetResult - Obtiene un objetivo de cámara de la colección *Integrated Vision*

### 1.16 CamGetResult - Obtiene un objetivo de cámara de la colección

---

#### Utilización

`CamGetResult` (*Camera Get Result*) se utiliza para obtener un objetivo de cámara de la colección de resultados de visión.

---

#### Ejemplos básicos

El ejemplo siguiente ilustra la instrucción `CamGetResult`.

#### Ejemplo 1

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

Ordenar a la cámara `mycamera` la adquisición de una imagen. Obtener un resultado de visión comenzando por la imagen que tiene la `SceneId` indicada.

---

#### Argumentos

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

Camera

**Tipo de dato:** `cameradev`

El nombre de la cámara.

CamTarget

**Posición de cámara**

**Tipo de dato:** `cameratarget`

La variable en la que se almacena el resultado de visión.

[\SceneId]

**Identificación de escena**

**Tipo de dato:** `num`

`SceneId` es un identificador que especifica a partir de qué imagen ha sido generado el `cameratarget`.

[\MaxTime]

**Tiempo máximo**

**Tipo de dato:** `num`

El intervalo de tiempo máximo, en segundos, que debe esperar la ejecución del programa. El valor máximo permitido es de 120 segundos.

*Continúa en la página siguiente*

### Ejecución de programas

`CamGetResult` obtiene un objetivo de cámara de la colección de resultados de visión. Si no se utiliza `SceneId` ni `MaxTime` y no hay ningún resultado que pueda capturarse, la instrucción se quedará colgada indefinidamente. Si se utiliza una `SceneId` en `CamGetResult`, es necesario que haya sido generada en una instrucción `CamReqImage` precedente.

La `SceneId` sólo puede usarse si la imagen ha sido ordenada mediante la instrucción `CamReqImage`. Si las imágenes son generadas por una señal de E/S externa, la `SceneId` no puede usarse en la instrucción `CamGetResult`.

### Gestión de errores

Pueden generarse los errores recuperables enumerados a continuación. Estos errores pueden ser gestionados en un gestor de errores. El valor de la variable de sistema `ERRNO` cambia a:

| Nombre                            | Causa del error  |
|-----------------------------------|--|
| <code>ERR_CAM_BUSY</code>         | La cámara está ocupada con otra petición y no puede ejecutar la orden actual.  |
| <code>ERR_CAM_MAXTIME</code>      | No fue posible capturar ningún resultado dentro del tiempo límite.   |
| <code>ERR_CAM_NO_MORE_DATA</code> | No es posible capturar más resultados de visión para la <code>SceneId</code> utilizada, o el resultado no pudo ser capturado dentro del tiempo límite. |

### Sintaxis

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

```

### Información relacionada

| Para obtener más información sobre | Consulte  |
|------------------------------------|---|
| Integrated Vision                  | <i>Manual de aplicaciones - Integrated Vision</i> |

# 1 Instrucciones

---

## 1.17 CamLoadJob - Carga una tarea de cámara en una cámara *Integrated Vision*

## 1.17 CamLoadJob - Carga una tarea de cámara en una cámara

---

### Utilización

CamLoadJob (*Camera Load Job*) carga una tarea de cámara, *job*, que describe los parámetros de exposición, la calibración y qué herramientas de visión se deben aplicar.

---

### Ejemplos básicos

El ejemplo siguiente ilustra la instrucción CamLoadJob.

#### Ejemplo 1

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

El trabajo myjob se carga en una cámara con el nombre mycamera.

---

### Argumentos

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

Camera

Tipo de dato: cameradev

El nombre de la cámara.

Name

Tipo de dato: string

El nombre del trabajo a cargar en la cámara.

[\KeepTargets]

Tipo de dato: switch

Este argumento se utiliza para especificar si se debe conservar cualquiera de los objetivos de cámara existentes generados por la cámara.

[\MaxTime]

Tipo de dato: num

El intervalo de tiempo máximo, en segundos, que debe esperar la ejecución del programa. El valor máximo permitido es de 120 segundos.

---

### Ejecución de programas

La ejecución de CamLoadJob esperará hasta que el trabajo esté cargado o fallará con un error de tiempo límite. Si se utiliza el argumento opcional *KeepTargets*, se conservarán los datos de colección anteriores de la cámara especificada. El comportamiento predeterminado es la eliminación (vaciado) de los datos de colección anteriores.

*Continúa en la página siguiente*



### Gestión de errores

Pueden generarse los errores recuperables enumerados a continuación. Estos errores pueden ser gestionados en un gestor de errores. El valor de la variable de sistema `ERRNO` cambia a:

| Nombre                              | Causa del error   |
|-------------------------------------|---|
| <code>ERR_CAM_BUSY</code>           | La cámara está ocupada con otra petición y no puede ejecutar la orden actual.     |
| <code>ERR_CAM_COM_TIMEOUT</code>    | Error de comunicación con la cámara. Es probable que la cámara esté desconectada. |
| <code>ERR_CAM_MAXTIME</code>        | El trabajo de cámara no fue cargado antes de alcanzar el tiempo límite.           |
| <code>ERR_CAM_NOT_ON_NETWORK</code> | La cámara no está conectada   |
| <code>ERR_CAM_NO_PROGMODE</code>    | La cámara no está en el modo de programa  |

### Limitaciones

Sólo es posible ejecutar `CamLoadJob` cuando la cámara se encuentra en el modo de programa. Utilice la instrucción `CamSetProgramMode` para poner la cámara en el modo de programación.

Para poder cargar el trabajo, el archivo del trabajo debe estar almacenado en el disco flash de la cámara.

### Sintaxis

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

### Información relacionada

| Para obtener más información sobre | Consulte  |
|------------------------------------|---|
| Integrated Vision                  | <i>Manual de aplicaciones - Integrated Vision</i> |

# 1 Instrucciones

---

## 1.18 CamReqImage - Ordena a la cámara la adquisición de una imagen *Integrated Vision*

### 1.18 CamReqImage - Ordena a la cámara la adquisición de una imagen

---

#### Utilización

CamReqImage (*Camera Request Image*) ordena a la cámara la adquisición de una imagen.

---

#### Ejemplos básicos

El ejemplo siguiente ilustra la instrucción CamReqImage.

#### Ejemplo 1

```
CamReqImage mycamera;
```

Ordenar a la cámara mycamera la adquisición de una imagen.

---

#### Argumentos

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

Camera

Tipo de dato: cameradev

El nombre de la cámara.

[\SceneId]

*Identificación de escena*

Tipo de dato: num

El argumento opcional SceneId es un identificador de la imagen adquirida. Se genera con cada instrucción CamReqImage ejecutada con el argumento opcional SceneId. El identificador es un entero de entre 1 y 8388608. Si no se utiliza SceneId, el valor de identificador se cambia a 0.

[\KeepTargets]

Tipo de dato: switch

Este argumento se utiliza para especificar si se deben conservar los datos de colección usados de la cámara especificada.

[\AwaitComplete]

Tipo de dato: switch

Si se especifica el argumento opcional \AwaitComplete, la instrucción espera hasta que se hayan recibido los resultados de la imagen.

Si se utiliza \AwaitComplete, el tipo de disparador de la cámara debe cambiarse a Externo.

---

#### Ejecución de programas

CamReqImage ordena a la adquisición de una imagen. Si se utiliza el argumento opcional SceneId, los resultados de visión disponibles de una imagen adquirida aparecen marcados con el número exclusivo generado por la instrucción.

Si se utiliza el argumento opcional KeepTargets, se conservarán los datos de colección anteriores de la cámara especificada. El comportamiento predeterminado es la eliminación (vaciado) de todos los datos de colección anteriores.

---

*Continúa en la página siguiente*

## 1.18 CamReqImage - Ordena a la cámara la adquisición de una imagen

*Integrated Vision*

*Continuación*

### Gestión de errores

Pueden generarse los errores recuperables enumerados a continuación. Estos errores pueden ser gestionados en un gestor de errores. El valor de la variable de sistema `ERRNO` cambia a:

| Nombre                              | Causa del error   |
|-------------------------------------|---|
| <code>ERR_CAM_BUSY</code>           | La cámara está ocupada con otra petición y no puede ejecutar la orden actual.     |
| <code>ERR_CAM_COM_TIMEOUT</code>    | Error de comunicación con la cámara. Es probable que la cámara esté desconectada. |
| <code>ERR_CAM_NO_RUNMODE</code>     | La cámara no está en el modo de funcionamiento                                    |
| <code>ERR_CAM_NOT_ON_NETWORK</code> | La cámara no está conectada.  |

### Limitaciones

Sólo es posible ejecutar `CamReqImage` cuando la cámara se encuentra en el modo de ejecución. Utilice la instrucción `CamSetRunMode` para poner la cámara en el modo de ejecución.

### Sintaxis

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

### Información relacionada

| Para obtener más información sobre | Consulte  |
|------------------------------------|---|
| Integrated Vision                  | <i>Manual de aplicaciones - Integrated Vision</i> |

## 1 Instrucciones

---

### 1.19 CamSetExposure - Establece parámetros específicos de la cámara *Integrated Vision*

### 1.19 CamSetExposure - Establece parámetros específicos de la cámara

---

#### Utilización

`CamSetExposure` (*Camera Set Exposure*) establece parámetros específicos de la cámara y permite adaptar los parámetros de la imagen a las condiciones de iluminación del ambiente.

---

#### Ejemplos básicos

El ejemplo siguiente ilustra la instrucción `CamSetExposure`.

#### Ejemplo 1

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

Ordenar a la cámara `mycamera` que cambie el tiempo de exposición a 10 ms.

---

#### Argumentos

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

Camera

Tipo de dato: `cameradev`

El nombre de la cámara.

`[\ExposureTime]`

Tipo de dato: `num`

Si se utiliza este argumento opcional, se actualiza el tiempo de exposición de la cámara. El valor se indica en milisegundos (ms).

`[\Brightness]`

Tipo de dato: `num`

Si se utiliza este argumento opcional, se actualiza el ajuste de claridad de la cámara. Este valor se expresa normalmente en una escala de 0 a 1.

`[\Contrast]`

Tipo de dato: `num`

Si se utiliza este argumento opcional, se actualiza el ajuste de contraste de la cámara. Este valor se expresa normalmente en una escala de 0 a 1.

---

#### Ejecución de programas

Esta instrucción ajusta el tiempo de exposición, la claridad y el contraste si es posible actualizar estos parámetros en la cámara en cuestión. Si la cámara no admite algún ajuste en concreto, se mostrará un mensaje de error al usuario y la ejecución del programa se detiene.

*Continúa en la página siguiente*

### Gestión de errores

Pueden generarse los errores recuperables enumerados a continuación. Estos errores pueden ser gestionados en un gestor de errores. El valor de la variable de sistema `ERRNO` cambia a:

| Nombre                              | Causa del error   |
|-------------------------------------|---|
| <code>ERR_CAM_COM_TIMEOUT</code>    | Error de comunicación con la cámara. Es probable que la cámara esté desconectada. |
| <code>ERR_CAM_NOT_ON_NETWORK</code> | La cámara no está conectada.  |

### Sintaxis

```
CamSetExposure
[ Camera ':= ' ] < variable (VAR) of cameradev > ', '
[ '\ExposureTime ':= ' < variable (IN) of num > ]
[ '\Brightness ':= ' < variable (IN) of num > ]
[ '\Contrast ':= ' < variable (IN) of num > ] ';'

```

### Información relacionada

| Para obtener más información sobre | Consulte  |
|------------------------------------|---|
| Integrated Vision                  | <i>Manual de aplicaciones - Integrated Vision</i> |

# 1 Instrucciones

---

## 1.20 CamsetParameter - Establece distintos parámetros designados de la cámara *Integrated Vision*

## 1.20 CamsetParameter - Establece distintos parámetros designados de la cámara

---

### Utilización

`CamsetParameter` se utiliza para establecer distintos parámetros designados que una cámara puede exponer. Esta instrucción permite cambiar distintos parámetros de la cámara en tiempo de ejecución. El usuario debe conocer el nombre del parámetro y su tipo de devolución para poder establecer su valor.

---

### Ejemplos básicos

El ejemplo siguiente ilustra la instrucción `CamsetParameter`.

#### Ejemplo 1

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

En este ejemplo, se cambia a falso el parámetros denominado "Pattern\_1.Tool\_Enabled", lo que significa que la herramienta de visión especificada no debe ejecutarse al adquirir una imagen.

De esta forma se consigue una ejecución más rápida de la herramienta de visión. Sin embargo, la herramienta sigue produciendo resultados con los valores de la ejecución activa más reciente. Para que estos objetivos no se utilicen, exclúyalos del programa de RAPID.

---

### Argumentos

```
CamsetParameter Camera ParName [\NumVal] | [\BoolVal] | [\StrVal]
```

Camera

Tipo de dato: `cameradev`

El nombre de la cámara.

ParName

Tipo de dato: `string`

El nombre del parámetro en la cámara.

[\NumVal]

Tipo de dato: `num`

El valor numérico a establecer para el parámetro de cámara denominado con el nombre establecido en el argumento `ParName`.

[\BoolVal]

Tipo de dato: `bool`

El valor booleano a establecer para el parámetro de cámara denominado con el nombre establecido en el argumento `ParName`.

[\StrVal]

Tipo de dato: `string`

El valor de cadena a establecer para el parámetro de cámara denominado con el nombre establecido en el argumento `ParName`.

---

*Continúa en la página siguiente*

### Limitaciones

Los parámetros creados por el usuario sólo pueden ser del tipo `EditString`, `EditInt`, o `EditFloat`.

### Gestión de errores

Pueden generarse los errores recuperables enumerados a continuación. Estos errores pueden ser gestionados en un gestor de errores. El valor de la variable de sistema `ERRNO` cambia a:

| Nombre                              | Causa del error  |
|-------------------------------------|--|
| <code>ERR_CAM_BUSY</code>           | La cámara está ocupada con otra petición y no puede ejecutar la orden actual.  |
| <code>ERR_CAM_COM_TIMEOUT</code>    | Error de comunicación con la cámara. Es probable que la cámara esté desconectada.  |
| <code>ERR_CAM_NOT_ON_NETWORK</code> | La cámara no está conectada.   |
| <code>ERR_CAM_SET_MISMATCH</code>   | El parámetro escrito en la cámara con la instrucción <code>CamGetParameter</code> tiene un tipo de dato incorrecto o el valor está fuera de rango. |

### Sintaxis

```
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 > ] ';'

```

### Información relacionada

| Para obtener más información sobre | Consulte  |
|------------------------------------|---|
| Integrated Vision                  | <i>Manual de aplicaciones - Integrated Vision</i> |

# 1 Instrucciones

---

## 1.21 CamSetProgramMode - Ordena a la cámara que cambie al modo de programación *Integrated Vision*

### 1.21 CamSetProgramMode - Ordena a la cámara que cambie al modo de programación

---

#### Utilización

CamSetProgramMode (*Camera Set Program Mode*) ordena a la cámara que cambie al modo de programación (fuera de línea).

---

#### Ejemplos básicos

El ejemplo siguiente ilustra la instrucción CamSetProgramMode.

#### Ejemplo 1

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

En primer lugar, cambiar la cámara al modo de programación. A continuación, cargar myjob en la cámara. Posteriormente, ordenar a la cámara que cambie al modo de ejecución.

---

#### Argumentos

CamSetProgramMode Camera

Camera

Tipo de dato: cameradev  
El nombre de la cámara.

---

#### Ejecución de programas

Al ordenar a una cámara que cambie al modo de programación con CamSetProgramMode, será posible cambiar los ajustes y cargar un trabajo en la cámara.

---

#### Gestión de errores

Pueden generarse los errores recuperables enumerados a continuación. Estos errores pueden ser gestionados en un gestor de errores. El valor de la variable de sistema ERRNO cambia a:

| Nombre                 | Causa del error   |
|------------------------|---|
| ERR_CAM_BUSY           | La cámara está ocupada con otra petición y no puede ejecutar la orden actual.     |
| ERR_CAM_COM_TIMEOUT    | Error de comunicación con la cámara. Es probable que la cámara esté desconectada. |
| ERR_CAM_NOT_ON_NETWORK | La cámara no está conectada.  |

---

#### Sintaxis

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

---

*Continúa en la página siguiente*



1.21 CamSetProgramMode - Ordena a la cámara que cambie al modo de programación

*Integrated Vision*

*Continuación*

## Información relacionada

| Para obtener más información sobre | Consulte  |
|------------------------------------|---|
| Integrated Vision                  | <i>Manual de aplicaciones - Integrated Vision</i> |

# 1 Instrucciones

---

## 1.22 CamSetRunMode Ordena a la cámara que cambie al modo de ejecución *Integrated Vision*

## 1.22 CamSetRunMode Ordena a la cámara que cambie al modo de ejecución

---

### Utilización

CamSetRunMode (*Camera Set Running Mode*) ordena a la cámara que cambie al modo de ejecución (en línea) y pone al día al controlador acerca de la configuración actual de salida a RAPID.

---

### Ejemplos básicos

El ejemplo siguiente ilustra la instrucción CamSetRunMode.

#### Ejemplo 1

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

En primer lugar, cambiar la cámara al modo de programación. A continuación, cargar myjob en la cámara. Posteriormente, ordenar a la cámara que cambie al modo de ejecución con la instrucción CamSetRunMode.

---

### Argumentos

CamSetRunMode Camera

Camera

Tipo de dato: cameradev  
El nombre de la cámara.

---

### Ejecución de programas

Al pedir a una cámara que cambie al modo de ejecución con CamSetRunMode, es posible comenzar a tomar imágenes.

---

### Gestión de errores

Pueden generarse los errores recuperables enumerados a continuación. Estos errores pueden ser gestionados en un gestor de errores. El valor de la variable de sistema ERRNO cambia a:

| Nombre                 | Causa del error   |
|------------------------|---|
| ERR_CAM_BUSY           | La cámara está ocupada con otra petición y no puede ejecutar la orden actual.     |
| ERR_CAM_COM_TIMEOUT    | Error de comunicación con la cámara. Es probable que la cámara esté desconectada. |
| ERR_CAM_NOT_ON_NETWORK | La cámara no está conectada.  |

---

### Sintaxis

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

---

*Continúa en la página siguiente*

1.22 CamSetRunMode Ordena a la cámara que cambie al modo de ejecución  
*Integrated Vision*  
*Continuación*

---

## Información relacionada

| Para obtener más información sobre | Consulte  |
|------------------------------------|---|
| Integrated Vision                  | <i>Manual de aplicaciones - Integrated Vision</i> |

## 1 Instrucciones

---

### 1.23 CamStartLoadJob - Inicia la carga de una tarea de cámara en una cámara *Integrated Vision*

### 1.23 CamStartLoadJob - Inicia la carga de una tarea de cámara en una cámara

---

#### Utilización

CamStartLoadJob comenzará la carga de un trabajo en una cámara, mientras que la ejecución continúa por la instrucción siguiente. Mientras se está realizando la carga, otras instrucciones pueden ejecutarse en paralelo.

---

#### Ejemplos básicos

El ejemplo siguiente ilustra la instrucción CamStartLoadJob.

#### Ejemplo 1

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

En primer lugar se inicia la carga del trabajo en la cámara y, mientras la carga se está realizando, se realiza un movimiento a la posición p1. Cuando el movimiento esté listo y la carga haya finalizado, se adquiere una imagen.

---

#### Argumentos

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

Camera

Tipo de dato: cameradev

El nombre de la cámara.

Name

Tipo de dato: string

El nombre del trabajo a cargar en la cámara.

[\KeepTargets]

Tipo de dato: switch

Este argumento se utiliza para especificar si se deben conservar los datos de colección usados de la cámara especificada.

---

#### Ejecución de programas

La ejecución de CamStartLoadJob sólo solicita la carga y continúa directamente en la instrucción siguiente, sin esperar a que se complete la carga. Si se utiliza el argumento opcional \KeepTargets, no se eliminan los datos de colección anteriores de la cámara especificada. El comportamiento predeterminado es la eliminación (vaciado) de los datos de colección anteriores.

La cámara estará ocupada ejecutando la operación de carga y no aceptará ninguna nueva solicitud de cámara antes de que la operación se complete con

CamWaitLoadJob, con la excepción de que las solicitudes de CamStartSetParameter se pueden poner en cola.

*Continúa en la página siguiente*

## 1.23 CamStartLoadJob - Inicia la carga de una tarea de cámara en una cámara

*Integrated Vision*

*Continuación*

### Gestión de errores

Pueden generarse los errores recuperables enumerados a continuación. Estos errores pueden ser gestionados en un gestor de errores. El valor de la variable de sistema `ERRNO` cambia a:

| Nombre                              | Causa del error   |
|-------------------------------------|---|
| <code>ERR_CAM_BUSY</code>           | La cámara está ocupada con otra petición y no puede ejecutar la orden actual. |
| <code>ERR_CAM_NOT_ON_NETWORK</code> | La cámara no está conectada   |

### Limitaciones

Sólo es posible ejecutar `CamStartLoadJob` cuando la cámara se encuentra en el modo de programa. Utilice la instrucción `CamSetProgramMode` para poner la cámara en el modo de programación.

Mientras se ejecuta la carga en curso de un trabajo, no es posible acceder a esa cámara concreta con ninguna otra instrucción o función. La siguiente instrucción o función de la cámara debe ser una instrucción `CamWaitLoadJob`.

Para poder cargar el trabajo, el archivo del trabajo debe estar almacenado en el disco flash de la cámara.

### Sintaxis

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

### Información relacionada

| Para obtener más información sobre | Consulte  |
|------------------------------------|---|
| Integrated Vision                  | <i>Manual de aplicaciones - Integrated Vision</i>   |
| <code>CamWaitLoadJob</code>        | <a href="#">CamWaitLoadJob – Esperar hasta que una tarea de cámara esté cargada en la página 77</a>   |
| <code>CamStartSetParameter</code>  | <a href="#">CamStartSetParameter - Iniciar la operación de ajuste de un parámetro en la página 74</a> |

# 1 Instrucciones

---

## 1.24 CamStartSetParameter - Iniciar la operación de ajuste de un parámetro

RobotWare - OS

## 1.24 CamStartSetParameter - Iniciar la operación de ajuste de un parámetro

---

### Utilización

`CamStartSetParameter` se utiliza para iniciar la operación de ajuste de un parámetro en la cámara. Cuando la operación de ajuste está en curso, se pueden ejecutar en paralelo otras instrucciones y funciones de RAPID. La cámara estará ocupada ejecutando la operación de ajuste de parámetros y no responderá a ninguna otra solicitud antes de que se complete esta operación con `CamWaitSetParameter`.

---

### Ejemplos básicos

Los siguientes ejemplos ilustran la instrucción `CamStartSetParameter`.

#### Ejemplo 1

```
CamStartSetParameter mycamera, "Pattern_1.Tool_Enabled"  
  \BoolVal:=FALSE;  
MoveL p1, v1000, fine, tool2;  
CamWaitSetParameter mycamera;
```

Primero se ordena un ajuste de parámetro, y mientras se realiza el ajuste, se efectúa un movimiento a la posición p1. Cuando el movimiento está listo y el ajuste del parámetro está listo, la ejecución de RAPID continúa.

#### Ejemplo 2

```
CamStartSetParameter mycamera,  
  "Pattern_1.Description"\StrVal:="mydescription";  
CamStartSetParameter mycamera,  
  "Pattern_1.Rotation_Tolerance"\NumVal:=15;  
MoveL p1, v1000, fine, tool2;  
CamWaitSetParameter mycamera;  
CamWaitSetParameter mycamera;
```

Se ordenan dos ajustes de parámetros y durante el tiempo en que se manejan se efectúa un movimiento. Las dos instrucciones `CamStartSetParameter` deben coincidir con dos instrucciones `CamWaitSetParameter` para poder realizar cualquier otra solicitud respecto a la cámara.

---

### Argumentos

```
CamStartSetParameter Camera ParName [\NumVal] | [\BoolVal] |  
  [\StrVal]
```

Camera

Tipo de dato: cameradev  
El nombre de la cámara.

ParName

Tipo de dato: string  
El nombre del parámetro en la cámara.

[\NumVal]

Tipo de dato: num

*Continúa en la página siguiente*

## 1.24 CamStartSetParameter - Iniciar la operación de ajuste de un parámetro

RobotWare - OS

Continuación

El valor numérico a establecer para el parámetro de cámara denominado con el nombre establecido en el argumento `ParName`.

`[\BoolVal]`

Tipo de dato: `bool`

El valor booleano a establecer para el parámetro de cámara denominado con el nombre establecido en el argumento `ParName`.

`[\StrVal]`

Tipo de dato: `string`

El valor de cadena a establecer para el parámetro de cámara denominado con el nombre establecido en el argumento `ParName`.

### Ejecución de programas

`CamStartSetParameter` iniciará la operación de ajuste de un parámetro en la cámara. Cuando la operación de ajuste está en curso, se pueden ejecutar en paralelo otras instrucciones y funciones de RAPID.

La cámara estará ocupada ejecutando la operación de ajuste de parámetros y no responderá a ninguna otra solicitud antes de que se complete esta operación con `CamWaitSetParameter`. Cada instrucción `CamStartSetParameter` debe ir acompañada de la correspondiente instrucción `CamWaitSetParameter`.

Si hay varias solicitudes `StartSetParameter` en cola, las correspondientes `WaitSetParameter` reflejarán el mismo orden que el de la solicitud y, por tanto, devolverán el estado para dicha `StartSetParameter`.



#### Nota

Los parámetros definidos por el usuario en la cámara tienen que estar asociados a una de las siguientes funciones de la cámara:

- `EditFloat`
- `EditInt`
- `EditString`

### Limitaciones

El controlador puede tener 10 operaciones `CamStartSetParameter` pendientes. Cuando el controlador tiene 10 solicitudes pendientes, es necesario confirmar todas las solicitudes con instrucciones `CamWaitSetParameter` antes de que se pida una nueva `CamStartSetParameter`.

### Gestión de errores

Se generan los siguientes errores recuperables que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` se establecerá en:

|                                     |   |
|-------------------------------------|---|
| <code>ERR_CAM_BUSY</code>           | La cámara está ocupada con otra petición y no puede ejecutar la orden actual. |
| <code>ERR_CAM_NOT_ON_NETWORK</code> | La cámara no está conectada.  |
| <code>ERR_CAM_NO_PROGMode</code>    | La cámara no está en el modo de programa.                                     |

Continúa en la página siguiente

# 1 Instrucciones

## 1.24 CamStartSetParameter - Iniciar la operación de ajuste de un parámetro

RobotWare - OS

Continuación

### Sintaxis

```
CamStartSetParameter
  [ 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 > ] ';'

```

### Información relacionada

| Para obtener más información sobre | Consulte   |
|------------------------------------|--|
| Integrated Vision                  | <i>Manual de aplicaciones - Integrated Vision</i>  |
| CamWaitSetParameter                | <a href="#">CamWaitSetParameter - Esperar hasta que una operación de ajuste esté lista en la página 79</a> |
| CamSetParameter                    | <a href="#">CamSetParameter - Establece distintos parámetros designados de la cámara en la página 66</a>   |



## 1.25 CamWaitLoadJob – Esperar hasta que una tarea de cámara esté cargada *Integrated Vision*

### 1.25 CamWaitLoadJob – Esperar hasta que una tarea de cámara esté cargada

#### Utilización

CamWaitLoadJob (*Camera Wait Load Job*) espera hasta que la carga de un trabajo en una cámara se haya completado.

#### Ejemplos básicos

El ejemplo siguiente ilustra la instrucción CamWaitLoadJob.

#### Ejemplo 1

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

En primer lugar se inicia la carga del trabajo en la cámara y, mientras la carga se está realizando, se realiza un movimiento a la posición p1. Cuando el movimiento esté listo y la carga haya finalizado, se adquiere una imagen.

#### Argumentos

CamWaitLoadJob Camera

Camera

Tipo de dato: cameradev

El nombre de la cámara.

#### Gestión de errores

Pueden generarse los errores recuperables enumerados a continuación. Estos errores pueden ser gestionados en un gestor de errores. El valor de la variable de sistema ERRNO cambia a:

| Nombre                 | Causa del error   |
|------------------------|---|
| ERR_CAM_COM_TIMEOUT    | Error de comunicación con la cámara. Es probable que la cámara esté desconectada. |
| ERR_CAM_NOT_ON_NETWORK | La cámara no está conectada   |

#### Limitaciones

Sólo es posible ejecutar CamWaitLoadJob cuando la cámara se encuentra en el modo de programa. Utilice la instrucción CamSetProgramMode para poner la cámara en el modo de programación.

Mientras se ejecuta la carga en curso de un trabajo, no es posible acceder a esa cámara concreta con ninguna otra instrucción o función. La siguiente instrucción o función de la cámara debe ser una instrucción CamWaitLoadJob.

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.25 CamWaitLoadJob – Esperar hasta que una tarea de cámara esté cargada

*Integrated Vision*

*Continuación*

---

### Sintaxis

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

---

### Información relacionada

| Para obtener más información sobre | Consulte   |
|------------------------------------|--|
| Integrated Vision                  | <i>Manual de aplicaciones - Integrated Vision</i>  |
| CamStartLoadJob                    | <a href="#">CamStartLoadJob - Inicia la carga de una tarea de cámara en una cámara en la página 72</a> |

## 1.26 CamWaitSetParameter - Esperar hasta que una operación de ajuste esté lista

RobotWare - OS

### 1.26 CamWaitSetParameter - Esperar hasta que una operación de ajuste esté lista

#### Utilización

CamWaitSetParameter esperará hasta que la operación de ajuste del parámetro se realice en la cámara y regresará con el estado.

#### Ejemplos básicos

El ejemplo siguiente ilustra la instrucción CamWaitSetParameter.

#### Ejemplo 1

```
CamStartSetParameter mycamera, "Pattern_1.Tool_Enabled"
  \BoolVal:=FALSE;
MoveL p1, v1000, fine, tool2;
CamWaitSetParameter mycamera;
```

Primero se ordena un ajuste de parámetro, y mientras se realiza el ajuste, se efectúa un movimiento a la posición p1. Cuando el movimiento está listo y el ajuste del parámetro está listo, la ejecución de RAPID continúa.

#### Argumentos

CamWaitSetParameter Camera

Camera

Tipo de dato: cameradev

El nombre de la cámara.

#### Gestión de errores

Se generan los siguientes errores recuperables que pueden gestionarse con un gestor de errores. La variable del sistema ERRNO se establecerá en:

|                                |   |
|--------------------------------|---|
| ERR_CAM_BUSY                   | La cámara está ocupada con otra petición y no puede ejecutar la orden actual.   |
| ERR_CAM_COM_TIMEOUT            | Error de comunicación con la cámara. Es probable que la cámara esté desconectada.   |
| ERR_CAM_NOT_ON_NETWORK         | La cámara no está conectada.  |
| ERR_CAM_NO_START_SET_PARAMETER | No hay ninguna solicitud en curso para ajustar un parámetro para la cámara.   |
| ERR_CAM_SET_MISMATCH           | El parámetro escrito en la cámara con la instrucción CamSetParameter tiene un tipo de dato incorrecto o el valor está fuera de rango. |
| ERR_CAM_SET_PARAMETER_REJECTED | La instrucción anterior ha fallado y también ha hecho que falle la configuración de parámetro para la cámara.                         |

#### Sintaxis

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

#### Información relacionada

| Para obtener más información sobre | Consulte                                   |
|------------------------------------|--|
| Integrated Vision                  | Manual de aplicaciones - Integrated Vision |

Continúa en la página siguiente

# 1 Instrucciones

---

## 1.26 CamWaitSetParameter - Esperar hasta que una operación de ajuste esté lista

RobotWare - OS

Continuación

| Para obtener más información sobre | Consulte  |
|------------------------------------|---|
| CamStartSetParameter               | <a href="#">CamStartSetParameter - Iniciar la operación de ajuste de un parámetro en la página 74</a> |

## 1.27 CancelLoad - Cancela la carga de un módulo

### Utilización

CancelLoad puede usarse para cancelar la operación de carga generada por la instrucción StartLoad.

CancelLoad sólo puede usarse entre instrucciones StartLoad y WaitLoad.

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción CancelLoad:

Consulte también [Más ejemplos en la página 81](#).

#### Ejemplo 1

```
CancelLoad load1;
```

Se cancela la sesión de carga load1.

### Argumentos

```
CancelLoad LoadNo
```

LoadNo

Tipo de dato: loadsession

Una referencia a la sesión de carga, creada por la instrucción StartLoad.

### Más ejemplos

A continuación aparecen más ejemplos de cómo usar la instrucción CancelLoad.

#### Ejemplo 1

```
VAR loadsession load1;

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

La instrucción CancelLoad cancela la carga en curso del módulo PART\_B.MOD y permite cargar en su lugar el módulo PART\_C.MOD.

### Gestión de errores

Pueden generarse los errores recuperables enumerados a continuación. Estos errores pueden ser gestionados en un gestor de errores. El valor de la variable de sistema ERRNO cambia a:

| Nombre           | Causa del error   |
|------------------|---|
| ERR_LOADNO_NOUSE | La variable especificada en el argumento LoadNo no se está utilizando, lo que significa que no hay ninguna sesión en uso. |

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.27 CancelLoad - Cancela la carga de un módulo

RobotWare - OS

Continuación

---

### Limitación

CancelLoad sólo puede usarse en la secuencia una vez que la instrucción StartLoad esté lista y antes del inicio de la instrucción WaitLoad.

---

### Sintaxis

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

---

### Información relacionada

| Para obtener más información sobre                | Consulte  |
|---|---|
| Carga de módulos de programa durante la ejecución | <a href="#">StartLoad - Carga de programa durante la ejecución en la página 792</a>             |
| Conexión de un módulo cargado a una tarea         | <a href="#">WaitLoad - Conectar un módulo cargado a una tarea en la página 1064</a>             |
| Sesión de carga                                   | <a href="#">loadsession - Sesión de carga de programa en la página 1702</a>                     |
| Carga de un módulo de programa                    | <a href="#">Load - Carga un módulo de programa durante la ejecución en la página 347</a>        |
| Descarga de un módulo de programa                 | <a href="#">UnLoad - Descargar un módulo de programa durante la ejecución en la página 1021</a> |
| Comprobar referencias de programa                 | <a href="#">CheckProgRef - Comprobar referencias de programa en la página 131</a>               |

## 1.28 CapAPTrSetup - Configuración de un seguimiento de punto Continuous Application Platform (CAP)

### 1.28 CapAPTrSetup - Configuración de un seguimiento de punto

#### Utilización

CapAPTrSetup (*Configuración de un seguimiento de punto*) se utiliza para configurar un tipo de seguimiento de punto del sensor, por ejemplo, *WeldGuide* o *AWC*.

La interfaz de sensores se comunica con un máximo de un sensor a través de canales serie, utilizando el protocolo de transporte RTP1.

#### Ejemplo básico

##### SIO.cfg:

```
COM_TRP:
-Name "wg:" -Type "SOCKETDEV" -RemoteAddress "192.168.1255.101"
-RemotePort "6344"
```

##### Código de RAPID:

```
! 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 "wg:", do_left, 80, do_right, 80;
```

#### Argumentos

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

##### device

Tipo de dato: `string`

El nombre del dispositivo de E/S configurado en `sio.cfg` para el sensor utilizado.

##### DoLeft

Tipo de dato: `signaldo`

La señal digital de salida para la sincronización de la oscilación en el ciclo de oscilación izquierda.

##### LevelLeft

Tipo de dato: `num`

La posición de coordinación en el lado izquierdo del patrón de oscilación. El valor especificado es un porcentaje de la anchura a la izquierda del centro de oscilación. Si la oscilación va más allá de este punto, se cambia automáticamente al nivel alto una señal de salida digital (siempre y cuando la señal esté definida).

*Continúa en la página siguiente*

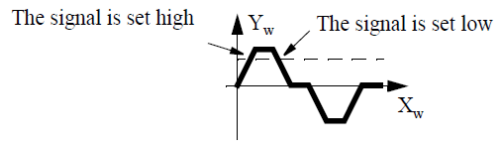
# 1 Instrucciones

## 1.28 CapAPTrSetup - Configuración de un seguimiento de punto

### Continuous Application Platform (CAP)

#### Continuación

El tipo de coordinación puede utilizarse para el seguimiento de cordones mediante el seguimiento de a través del arco.



xx1200000178

#### DoRight

Tipo de dato: `signaldo`

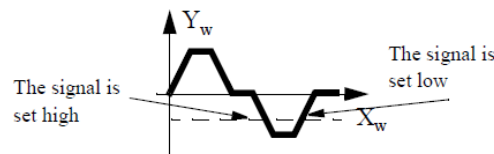
La señal digital de salida para la sincronización en el ciclo de oscilación derecha.

#### LevelRight

Tipo de dato: `num`

La posición de coordinación en el lado derecho del patrón de oscilación. El valor especificado es un porcentaje de la anchura a la derecha del centro de oscilación. Si la oscilación va más allá de este punto, se cambia automáticamente al nivel alto una señal de salida digital (siempre y cuando la señal esté definida).

El tipo de coordinación puede utilizarse para el seguimiento de cordones mediante el seguimiento de a través del arco.



xx1200000179

#### [LogFile]

Tipo de dato: `string`

Nombre del archivo de registro de Tracklog.

#### [LogSize]

Tipo de dato: `num`

Tamaño del búfer de anillo de Tracklog, que es el número de mediciones de sensor que pueden almacenarse en el búfer durante el seguimiento.

Valor predeterminado: 1000.

## Sintaxis

```
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 > ';'

```

Continúa en la página siguiente



## 1.28 CapAPTrSetup - Configuración de un seguimiento de punto *Continuous Application Platform (CAP)* Continuación

### Información relacionada

|  | Descrito en:  |
|--|---|
| <i>Continuous Application Platform</i> | <i>Application manual - Continuous Application Platform</i> |
| <i>Sensor Interface</i>                | <i>Application manual - Controller software IRC5</i>        |

## 1 Instrucciones

---

### 1.29 CapAPTrSetupAI - Configurar un seguimiento del punto controlado por señales de entrada analógicas *Continuous Application Platform (CAP)*

### 1.29 CapAPTrSetupAI - Configurar un seguimiento del punto controlado por señales de entrada analógicas

---

#### Utilización

CapAPTrSetupAI se usa para configurar un seguimiento del punto controlado por señales de entrada analógicas.

---

#### Ejemplos básicos

El ejemplo siguiente ilustra la instrucción CapAPTrSetupAI.

#### Ejemplo 1

```
TASK PERS capdata cData:=[.....];
TASK PERS weavestartdata wsData:=[.....];
TASK PERS capweavedata wData:=[.....];
TASK PERS captrackdata trackData:["ANALOG_TRACKER",.....];

VAR capaptrreferencedata referenceData:=[2,2,1,1,0.1,0.1];
VAR signalai ai_y;
VAR signalai ai_z;

AliasIO realsignal_y, ai_y;
AliasIO realsignal_z, ai_z;
CapAPTrSetupAI ai_y, ai_z, referenceData;

CapL p1, v200, cData, wsData, wData , fine, tWeldGun
  \Track:=trackData;
```

#### Argumentos

```
CapAPTrSetupAI ai_y, ai_z, ReferenceData [\MaxIncrCorr]
  [\WarnMaxCorr] [\Filter] [\SampleTime] [\Logfile] [\LogSize]
  [\LatestCorr] [\AccCorr]
```

ai\_y

**Tipo de dato:** signalai

Señal de entrada analógica usada como posición de proceso para la dirección y.

ai\_z

**Tipo de dato:** signalai

Señal de entrada analógica usada como posición de proceso para la dirección z.

ReferenceData

**Tipo de dato:** capaptrreferencedata

Configure datos usados para el bucle del regulador de corrección.

MaxIncrCorr

**Tipo de dato:** num

Corrección incremental máxima permitida (en mm).

Si la corrección incremental del TCP es superior a \MaxIncrCorr y \WarnMaxCorr, el robot continuará su trayectoria, aunque la corrección incremental aplicada no

*Continúa en la página siguiente*

---

## 1.29 CapAPTrSetupAI - Configurar un seguimiento del punto controlado por señales de entrada analógicas Continuous Application Platform (CAP) Continuación

superará `\MaxIncCorr`. Si no se especificó `\WarnMaxCorr`, se informa de un error de seguimiento y se detiene la ejecución del programa.

`WarnMaxCorr`

Tipo de dato: `switch`

Si este modificador está presente, la ejecución del programa no se interrumpe cuando se supera el límite de corrección máxima especificado en `\MaxIncCorr`. Solo se enviará un aviso.

`Filter`

Tipo de dato: `num`

Tamaño del filtro de datos de muestreo de referencia. Se permite un valor entre 1 y 15; el valor predeterminado es 1.

`SampleTime`

Tipo de dato: `num`

Tiempo de muestreo en milisegundos para el bucle de corrección. El valor se redondea a un múltiplo de 24. El valor mínimo permitido es 24, y el valor predeterminado es 24.

`LogFile`

Tipo de dato: `string`

El nombre del archivo de registro tracklog. El archivo de registro se ubica en el directorio HOME del sistema.

`LatestCorr`

Tipo de dato: `pos`

Tamaño de la última corrección añadida (en mm).

`AccCorr`

Tipo de dato: `pos`

Tamaño de la corrección total acumulada añadida (en mm).

`LogSize`

Tipo de dato: `num`

El tamaño del búfer de anillo de tracklog, que es el número de mediciones de sensor que pueden almacenarse en el búfer durante el seguimiento.

Valor predeterminado: 1000.

### Sintaxis

```
CapAPTrSetupAI
  [aoi_y ':='] <expression (IN) of signalai> ', '
  [ai_z ':='] <expression (IN) of signalai> ', '
  [ReferenceData ':='] <expression (IN) of capaptrreferencedata>
    ', '
  [\MaxIncrCorr ':='] <expression (IN) of num> ', '
  [\WarnMaxCorr ':='] <expression (IN) of switch> ', '
  [\Filter ':='] <expression (IN) of num> ', '
  [\SampleTime ':='] <expression (IN) of num> ', '
```

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.29 CapAPTrSetupAI - Configurar un seguimiento del punto controlado por señales de entrada analógicas Continuous Application Platform (CAP)

### Continuación

```
[\LogFile ':='] <expression (IN) of string> ', '  
[\LogSize ':='] <expression (IN) of num> ', '  
[\LatestCorr ':='] <expression (PERS) of pos> ', '  
[\AccCorr ':='] <expression (PERS) of pos> ';' 
```

### Información relacionada

| Para obtener más información sobre     | Consulte   |
|--|--|
| Instrucción CapAPTrSetupAO             | <a href="#">CapAPTrSetupAO - Configurar un seguimiento del punto controlado por señales de salida analógicas en la página 89</a> |
| Instrucción CapAPTrSetupPERS           | <a href="#">CapAPTrSetupPERS - Configurar un seguimiento del punto controlado por variables persistentes en la página 92</a>     |
| Tipo de dato capaptrreferencedata      | <a href="#">capaptrreferencedata - Datos de configuración variable para seguimiento del punto en la página 1622</a>              |
| <i>Continuous Application Platform</i> | <i>Application manual - Continuous Application Platform</i>  |
| <i>Sensor Interface</i>                | <i>Application manual - Controller software IRC5</i>   |

## 1.30 CapAPTrSetupAO - Configurar un seguimiento del punto controlado por señales de salida analógicas *Continuous Application Platform (CAP)*

### 1.30 CapAPTrSetupAO - Configurar un seguimiento del punto controlado por señales de salida analógicas

#### Utilización

CapAPTrSetupAO se usa para configurar un seguimiento del punto controlado por señales de salida analógicas.

#### Ejemplos básicos

El ejemplo siguiente ilustra la instrucción CapAPTrSetupAO.

#### Ejemplo 1

```
TASK PERS capdata cData:=[.....];
TASK PERS weavestartdata wsData:=[.....];
TASK PERS capweavedata wData:=[.....];
TASK PERS captrackdata trackData:["ANALOG_TRACKER",.....];

VAR capaptrreferencedata referenceData:[2,2,1,1,0.1,0.1];
VAR signalao ao_y;
VAR signalao ao_z;

AliasIO realsignal_y, ao_y;
AliasIO realsignal_z, ao_z;
CapAPTrSetupAO ao_y, ao_z, referenceData;

CapL p1, v200, cData, wsData, wData , fine, tWeldGun
  \Track:=trackData;
```

#### Argumentos

```
CapAPTrSetupAO ao_y, ao_z, ReferenceData [\MaxIncrCorr]
  [\WarnMaxCorr] [\Filter] [\SampleTime] [\Logfile] [\LogSize]
  [\LatestCorr] [\AccCorr]
```

ao\_y

Tipo de dato: signalao

Señal de salida analógica usada como posición de proceso para la dirección y.

ao\_z

Tipo de dato: signalao

Señal de salida analógica usada como posición de proceso para la dirección z.

ReferenceData

Tipo de dato: capaptrreferencedata

Configure datos usados para el bucle del regulador de corrección.

MaxIncrCorr

Tipo de dato: num

Corrección incremental máxima permitida (en mm).

Si la corrección incremental del TCP es superior a \MaxIncrCorr y \WarnMaxCorr, el robot continuará su trayectoria, aunque la corrección incremental aplicada no

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.30 CapAPTrSetupAO - Configurar un seguimiento del punto controlado por señales de salida analógicas Continuous Application Platform (CAP)

### Continuación

superará `\MaxIncCorr`. Si no se especificó `\WarnMaxCorr`, se informa de un error de seguimiento y se detiene la ejecución del programa.

WarnMaxCorr

Tipo de dato: `switch`

Si este modificador está presente, la ejecución del programa no se interrumpe cuando se supera el límite de corrección máxima especificado en `\MaxIncCorr`. Solo se enviará un aviso.

Filter

Tipo de dato: `num`

Tamaño del filtro de datos de muestreo de referencia. Se permite un valor entre 1 y 15; el valor predeterminado es 1.

SampleTime

Tipo de dato: `num`

Tiempo de muestreo en milisegundos para el bucle de corrección. El valor se redondea a un múltiplo de 24. El valor mínimo permitido es 24, y el valor predeterminado es 24.

LogFile

Tipo de dato: `string`

El nombre del archivo de registro tracklog. El archivo de registro se ubica en el directorio HOME del sistema.

LogSize

Tipo de dato: `num`

El tamaño del búfer de anillo de tracklog, que es el número de mediciones de sensor que pueden almacenarse en el búfer durante el seguimiento.

Valor predeterminado: 1000.

LatestCorr

Tipo de dato: `pos`

Tamaño de la última corrección añadida (en mm).

AccCorr

Tipo de dato: `pos`

Tamaño de la corrección total acumulada añadida (en mm).

---

### Sintaxis

```
CapAPTrSetupAO
[ao_y ':='] <expression (IN) of signalao> ', '
[ao_z ':='] <expression (IN) of signalao> ', '
[ReferenceData ':='] <expression (IN) of capaptrreferencedata>
', '
[\MaxIncrCorr ':='] <expression (IN) of num> ', '
[\WarnMaxCorr ':='] <expression (IN) of switch> ', '
[\Filter ':='] <expression (IN) of num> ', '
[\SampleTime ':='] <expression (IN) of num> ', '

```

*Continúa en la página siguiente*

## 1.30 CapAPTrSetupAO - Configurar un seguimiento del punto controlado por señales de salida analógicas Continuous Application Platform (CAP) Continuación

```
[\LogFile ':='] <expression (IN) of string> ','
[\LogSize ':='] <expression (IN) of num> ','
[\LatestCorr ':='] <expression (PERS) of pos> ','
[\AccCorr ':='] <expression (PERS) of pos> ';'

```

### Información relacionada

| Para obtener más información sobre     | Consulte  |
|--|---|
| Instrucción CapAPTrSetupAI             | <a href="#">CapAPTrSetupAI - Configurar un seguimiento del punto controlado por señales de entrada analógicas en la página 86</a> |
| Instrucción CapAPTrSetupPERS           | <a href="#">CapAPTrSetupPERS - Configurar un seguimiento del punto controlado por variables persistentes en la página 92</a>      |
| Tipo de dato capaptrreferencedata      | <a href="#">capaptrreferencedata - Datos de configuración variable para seguimiento del punto en la página 1622</a>               |
| <i>Continuous Application Platform</i> | <i>Application manual - Continuous Application Platform</i>   |
| <i>Sensor Interface</i>                | <i>Application manual - Controller software IRC5</i>  |

## 1 Instrucciones

---

### 1.31 CapAPTrSetupPERS - Configurar un seguimiento del punto controlado por variables persistentes *Continuous Application Platform (CAP)*

### 1.31 CapAPTrSetupPERS - Configurar un seguimiento del punto controlado por variables persistentes

---

#### Utilización

CapAPTrSetupPERS se usa para configurar un seguimiento del punto controlado por variables persistentes.

---

#### Ejemplos básicos

El ejemplo siguiente ilustra la instrucción CapAPTrSetupPERS.

#### Ejemplo 1

```
TASK PERS capdata cData:=[.....];
TASK PERS weavestartdata wsData:=[.....];
TASK PERS capweavedata wData:=[.....];
TASK PERS captrackdata trackData:=["ANALOG_TRACKER",.....];
PERS pos corr:=[0,-0.05,-0.025];

VAR capaptrreferencedata referenceData:=[2,2,1,1,0.1,0.1];

IDelete intnol;
CONNECT intnol WITH trOffset;
CapAPTRSetupPERS corr.y, corr.z, referenceData;

ITimer 1,intnol;
CapL p1, v200, cData, wsData, wData , fine,
    tWeldGun\Track:=trackData;

TRAP trOffset
    corr.y := referenceData.reference_y +- .....;
    corr.z := referenceData.reference_z +- .....;
ENDTRAP
```

#### Argumentos

```
CapAPTrSetupPERS var_y, var_z, ReferenceData [\MaxIncrCorr]
[\WarnMaxCorr] [\Filter] [\SampleTime] [\Logfile] [\LogSize]
[\LatestCorr] [\AccCorr]
```

var\_y

**Tipo de dato:** num

Señal de entrada analógica usada como posición de proceso para la dirección y.

var\_z

**Tipo de dato:** signalai

Señal de entrada analógica usada como posición de proceso para la dirección z.

ReferenceData

**Tipo de dato:** capaptrreferencedata

Configure datos usados para el bucle del regulador de corrección.

*Continúa en la página siguiente*



## 1.31 CapAPTrSetupPERS - Configurar un seguimiento del punto controlado por variables persistentes Continuous Application Platform (CAP) Continuación

MaxIncCorr

Tipo de dato: num

Corrección incremental máxima permitida (en mm).

Si la corrección incremental del TCP es superior a `\MaxIncCorr` y `\WarnMaxCorr`, el robot continuará su trayectoria, aunque la corrección incremental aplicada no superará `\MaxIncCorr`. Si no se especificó `\WarnMaxCorr`, se informa de un error de seguimiento y se detiene la ejecución del programa.

WarnMaxCorr

Tipo de dato: switch

Si este modificador está presente, la ejecución del programa no se interrumpe cuando se supera el límite de corrección máxima especificado en `\MaxIncCorr`. Solo se enviará un aviso.

Filter

Tipo de dato: num

Tamaño del filtro de datos de muestreo de referencia. Se permite un valor entre 1 y 15; el valor predeterminado es 1.

SampleTime

Tipo de dato: num

Tiempo de muestreo en milisegundos para el bucle de corrección. El valor se redondea a un múltiplo de 24. El valor mínimo permitido es 24, y el valor predeterminado es 24.

LogFile

Tipo de dato: string

El nombre del archivo de registro tracklog. El archivo de registro se ubica en el directorio HOME del sistema.

LatestCorr

Tipo de dato: pos

Tamaño de la última corrección añadida (en mm).

AccCorr

Tipo de dato: pos

Tamaño de la corrección total acumulada añadida (en mm).

LogSize

Tipo de dato: num

El tamaño del búfer de anillo de tracklog, que es el número de mediciones de sensor que pueden almacenarse en el búfer durante el seguimiento.

Valor predeterminado: 1000.

---

### Sintaxis

```
CapAPTrSetupPERS
  [var_y ':='] <expression (PERS) of num> ', '
  [var_z ':='] <expression (PERS) of vnum> ', '

```

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.31 CapAPTrSetupPERS - Configurar un seguimiento del punto controlado por variables persistentes Continuous Application Platform (CAP)

### Continuación

```
[ReferenceData ':='] <expression (IN) of capaptrreferencedata>
', '
[\MaxIncrCorr ':='] <expression (IN) of num> ', '
[\WarnMaxCorr ':='] <expression (IN) of switch> ', '
[\Filter ':='] <expression (IN) of num> ', '
[\SampleTime ':='] <expression (IN) of num> ', '
[\LogFile ':='] <expression (IN) of string> ', '
[\LogSize ':='] <expression (IN) of num> ', '
[\LatestCorr ':='] <expression (PERS) of pos> ', '
[\AccCorr ':='] <expression (PERS) of pos> ';'
```

### Información relacionada

| Para obtener más información sobre | Consulte  |
|------------------------------------|---|
| Instrucción CapAPTrSetupAI         | <a href="#">CapAPTrSetupAI - Configurar un seguimiento del punto controlado por señales de entrada analógicas en la página 86</a> |
| Instrucción CapAPTrSetupAO         | <a href="#">CapAPTrSetupAO - Configurar un seguimiento del punto controlado por señales de salida analógicas en la página 89</a>  |
| Tipo de dato capaptrreferencedata  | <a href="#">capaptrreferencedata - Datos de configuración variable para seguimiento del punto en la página 1622</a>               |
| Continuous Application Platform    | Application manual - Continuous Application Platform  |
| Sensor Interface                   | Application manual - Controller software IRC5   |

## 1.32 CapC - Instrucción de movimiento CAP circular

### Utilización

CapC se utiliza para mover el punto central de la herramienta (TCP) a lo largo de una trayectoria circular hasta un destino dado y a la vez controlar un proceso continuo. Además, es posible conectar hasta ocho eventos a CapC. Los eventos se definen mediante las instrucciones TriggRampAO, TriggIO, TriggEquip, TriggInt, TriggCheckIO o TriggSpeed.

### Ejemplos básicos

#### Ejemplo 1

**Movimientos circulares con CapC.**

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

El TCP de la herramienta, `gun1`, se mueve en círculo hacia un punto fino `p1` con velocidad definida en `cdata`.

#### Ejemplo 2

**Movimiento circular con evento de usuario y evento CAP.**

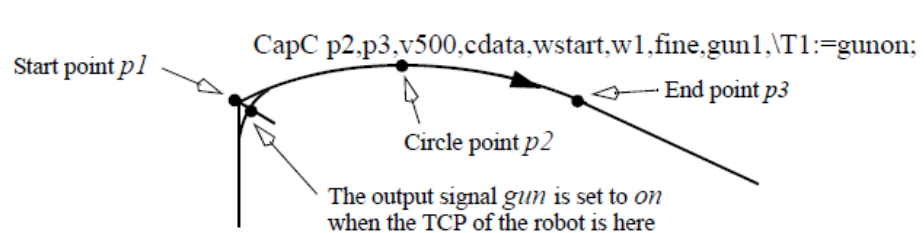
```
VAR intnum start_intno;
...
PROC main()
  VAR triggdata gunon;

  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
```

La señal digital de salida `gun` se activa cuando el TCP del robot atraviesa el punto central de la trayectoria de esquina del punto `p1`. La rutina TRAP `start_trap` se ejecuta cuando se inicia el proceso CAP.



xx120000174

Continúa en la página siguiente

# 1 Instrucciones

---

## 1.32 CapC - Instrucción de movimiento CAP circular

### Continuous Application Platform (CAP)

#### Continuación

---

#### Argumentos

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

Cirpoint

**Tipo de dato:** robtarget

El punto de círculo del robot. El punto de círculo es un punto del círculo entre el punto de inicio y el punto de destino. Para conseguir la máxima exactitud, debe estar situado a mitad de camino entre los puntos inicial y de destino. Si lo sitúa cerca del punto de inicio o del punto final, es posible que el robot genere un aviso. El punto de círculo se define como una posición con nombre o se almacena directamente en la instrucción (marcada con un asterisco \* en la instrucción).

ToPoint

**Tipo de dato:** robtarget

El punto de destino de los ejes del robot y de los ejes adicionales. Se define como una posición con nombre o se almacena directamente en la instrucción (marcada con un asterisco \* en la instrucción).

[\Id]

*(Identidad de sincronización)*

**Tipo de dato:** identno

La identidad de sincronización para las instrucciones de RAPID de movimiento en un sistema MultiMove en modo sincronizado.

Speed

**Tipo de dato:** speeddata

Los datos de velocidad que se aplican a los movimientos sin el proceso CAP activo. Los datos de velocidad definen la velocidad del punto central de la herramienta, los ejes adicionales y la reorientación de la herramienta. Si hay un proceso CAP activo (no bloqueado), el argumento Cdata define la velocidad del TCP.

Cdata

*(Datos de proceso CAP)*

**Tipo de dato:** capdata

Datos de proceso CAP; consulte [capdata - Datos CAP en la página 1624](#) para obtener una descripción detallada.

[\Movestart\_timer]

*(Tiempo en s)*

**Tipo de dato:** num

Límite superior para la diferencia de tiempo entre la orden de inicio del proceso y el inicio real del movimiento del TCP de los robots en un sistema MultiMove en modo sincronizado.

Weavestart

*(Datos de inicio de oscilación)*

*Continúa en la página siguiente*

---

Tipo de dato: weavestartdata

Datos de inicio de oscilación para el proceso CAP; consulte [weavestartdata - Datos de inicio de oscilación en la página 1821](#) para obtener una descripción detallada.

Weave

*(Datos de oscilación)*

Tipo de dato: capweavedata

Datos de inicio de oscilación para el proceso CAP; consulte [capweavedata - Datos de oscilación para CAP en la página 1638](#) para obtener una descripción detallada.

Zone

Tipo de dato: zonedata

Los datos de zona del movimiento. Los datos de zona describen el tamaño de la trayectoria de esquina generada.

[ \Inpos ]

*(En posición)*

Tipo de dato: stoppointdata

Este argumento se utiliza para especificar los criterios de convergencia para la posición del TCP del robot en el punto de paro. Los datos de puntos de paro sustituyen a la zona especificada en el parámetro `Zone`.

Tool

Tipo de dato: tooldata

La herramienta en uso durante el movimiento del robot. El punto central de la herramienta (TCP) es el punto que se mueve hacia la posición de destino especificada.

[ \WObj ]

Tipo de dato: wobjdata

El objeto de trabajo (sistema de coordenadas) con el que está relacionada la posición de robot indicada en la instrucción.

Es posible omitir este argumento. Si se omite, la posición depende del sistema de coordenadas mundo. Si, por otro lado, se usa un TCP estacionario o ejes externos adicionales, es necesario especificar este argumento para ejecutar un movimiento lineal respecto del objeto de trabajo.

[ \Track ]

*(Datos de sensor de track)*

Tipo de dato: captrackdata

Esta estructura de datos contiene los datos necesarios para el uso de la corrección de trayectoria que generan los sensores junto con `CapC`; consulte [captrackdata - Datos de seguimiento de CAP en la página 1635](#). Este argumento no se permite junto con el argumento `\Corr`.

[ \Corr ]

*(Utilizar generador de correcciones)*

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.32 CapC - Instrucción de movimiento CAP circular

### *Continuous Application Platform (CAP)*

#### *Continuación*

Tipo de dato: switch

Este argumento dice a CapC que lea las correcciones de trayectoria de un generador de correcciones, CorrCon. Este argumento no se admite junto con el argumento \Track.

Se requiere RobotWare, opción *Path Offset*, cuando se utiliza este argumento.

[\PreProcessTracking]

Tipo de dato: switch

Este argumento solo es eficaz si *first\_instruction* cambia a TRUE y se ha incluido el argumento \Track.

Este argumento activa el *seguimiento del proceso previo*, lo que significa que durante la instrucción CapX el robot solo hará el seguimiento, sin proceso. Por lo tanto, los datos del sensor están disponibles para el seguimiento desde el inicio de la trayectoria con proceso, p. ej. de soldadura.

Para obtener más información, consulte *Operating manual - Tracking and searching with optical sensors*.

[\Time]

Tipo de dato: num

Este argumento se utiliza para especificar el tiempo total en segundos que dura el movimiento del robot y de los ejes adicionales. A continuación, se sustituye por los datos de velocidad correspondientes.

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

(Trigg x)

Tipo de dato: trigndata

Las variables que hacen referencia a las condiciones de disparo y la actividad de disparo, definidas anteriormente en el programa usando las instrucciones

TriggRampAO, TriggIO, TriggEquip o TriggInt.

[\TLoad]

Tipo de dato: loaddata

El argumento \TLoad describe la carga total usada durante el movimiento. La carga total es la carga de la herramienta más la carga útil transportada por la herramienta. Si se utiliza el argumento \TLoad, no se tiene en cuenta el valor de loaddata en los tooldata actuales.

Si el argumento \TLoad se configura como load0, entonces no se toma en consideración el argumento y en su lugar se utiliza el loaddata en el tooldata actual. Para una descripción completa del argumento TLoad, consulte MoveL.

---

## Gestión de errores

Existen varios tipos diferentes de errores que pueden gestionarse en el gestor de errores para las instrucciones CapC/CapL:

- errores de supervisión
- errores específicos de sensores
- error específico para un sistema MultiMove

*Continúa en la página siguiente*

- errores heredados de la funcionalidad `TriggX`
- otros errores CAP

Si una de las señales que se supone que se va supervisar no tiene el valor correcto o si cambia el valor durante la supervisión, se establece la variable del sistema `ERRNO`.

Si no se pueden leer los valores del sensor de track, se establece la variable de sistema `ERRNO`.

Para un sistema MultiMove que se ejecuta en modo sincronizado, el gestor de errores debe ocuparse de otros dos errores. Uno se utiliza para informar de que alguna otra aplicación ha detectado un error recuperable. Esto activa la gestión de errores recuperables en las tareas RAPID sincronizadas. Se informa del otro error, `CAP_MOV_WATCHDOG`, si vence el tiempo entre la petición de inicio de proceso y el inicio real del movimiento del TCP de los robots en un sistema MultiMove en modo sincronizado. El tiempo utilizado se especifica en el parámetro opcional `Movestart_timer` de la instrucción `CapC`.

Si se detecta cualquier situación anormal, la ejecución del programa se detendrá. Sin embargo, si se ha programado un gestor de errores, los errores definidos a continuación pueden solucionarse sin detener la producción. Aunque se recomienda que algunos de estos errores (los errores con `CAP_XX`) no se muestren al usuario final. Asigne esos errores a un error específico de la aplicación. Para los errores de supervisión, puede utilizarse la instrucción `CapGetFailSigs` para ver qué señal específica ha fallado.

### Errores de supervisión

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

|                               |   |
|-------------------------------|---|
| <code>CAP_PRE_ERR</code>      | Este error se produce cuando hay un error en la lista de supervisión <code>PRE</code> , es decir, cuando no se cumplen las condiciones de la lista dentro del periodo de tiempo especificado (especificado en tiempo límite de <code>pre_cond</code> ).           |
| <code>CAP_PRESTART_ERR</code> | Este error se produce cuando hay un error durante la supervisión de la fase <code>PRE</code> .  |
| <code>CAP_END_PRE_ERR</code>  | Este evento se produce cuando hay un error en la lista de supervisión <code>END_PRE</code> , es decir, cuando no se cumplen las condiciones de la lista dentro del periodo de tiempo especificado (especificado en tiempo límite de <code>start_cond</code> ).    |
| <code>CAP_START_ERR</code>    | Este evento se produce cuando hay un error en la lista de supervisión <code>START</code> , es decir, cuando no se cumplen las condiciones de la lista dentro del periodo de tiempo especificado (especificado en tiempo límite de <code>start_cond</code> ).      |
| <code>CAP_MAIN_ERR</code>     | Este error se produce cuando hay un error durante la supervisión de la fase principal.  |
| <code>CAP_ENDMAIN_ERR</code>  | Este error se produce cuando hay un error en la lista de supervisión <code>END_MAIN</code> , es decir, cuando no se cumplen las condiciones de la lista dentro del periodo de tiempo especificado (especificado en tiempo límite de <code>end_main_cond</code> ). |

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.32 CapC - Instrucción de movimiento CAP circular

### Continuous Application Platform (CAP)

#### Continuación

|                     |  |
|---------------------|--|
| CAP_START_POST1_ERR | Este evento se produce cuando hay un error en la lista de supervisión START_POST1, es decir, cuando no se cumplen las condiciones de la lista dentro del periodo de tiempo especificado (especificado en tiempo límite de end_main_cond).  |
| CAP_POST1_ERR       | Este error se produce cuando hay un error durante la supervisión de la fase POST1.   |
| CAP_POST1END_ERR    | Este error se produce cuando hay un error en la lista de supervisión END_POST1, es decir, cuando no se cumplen las condiciones de la lista dentro del periodo de tiempo especificado (especificado en tiempo límite de end_main_cond).   |
| CAP_START_POST2_ERR | Este evento se produce cuando hay un error en la lista de supervisión START_POST1, es decir, cuando no se cumplen las condiciones de la lista dentro del periodo de tiempo especificado (especificado en tiempo límite de end_main_cond).  |
| CAP_POST2_ERR       | Este error se produce cuando hay un error durante la supervisión de la fase POST2.   |
| CAP_POST2END_ERR    | Este error se produce cuando hay un error en la lista de supervisión END_POST2, es decir, cuando no se cumplen las condiciones de la lista dentro del periodo de tiempo especificado (especificado en tiempo límite de end_main_cond).<br><br>Si la supervisión se realiza en dos señales diferentes de la misma fase y ambas fallan, la primera que esté configurada con SetupSuperv es la que genera el error. |

#### Errores relacionados con el sensor

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema ERRNO cambiará a:

|                  |   |
|------------------|---|
| CAP_TRACK_ERR    | Se produce un error de track cuando se leen los datos del sensor y transcurrido un tiempo no se reciben datos válidos. Una razón podría ser que el sensor no puede indicar el cordón. |
| CAP_TRACKSTA_ERR | Se produce un error de inicio de track cuando no se han leído datos válidos del sensor de track láser.  |
| CAP_TRACKCOR_ERR | Se produce un error de corrección de track cuando algo va mal en el cálculo del offset.   |
| CAP_TRACKCOM_ERR | La comunicación entre el controlador del robot y el equipo del sensor no funciona.  |
| CAP_TRACKPFR_ERR | No es posible continuar con el seguimiento si se ha producido un fallo de alimentación eléctrica durante el seguimiento.  |
| CAP_SEN_NO_MEAS  | El controlador no obtuvo una medición válida del sensor.  |
| CAP_SEN_NOREADY  | El sensor no está preparado todavía.  |
| CAP_SEN_GENERRO  | Se produjo un error general del sensor.   |
| CAP_SEN_BUSY     | El sensor está ocupado y no puede responder a la solicitud.   |
| CAP_SEN_UNKNOWN  | El comando enviado al sensor es desconocido para el sensor.   |
| CAP_SEN_ILLEGAL  | El número de variable o el número de bloqueo enviado al sensor es ilegal.   |

Continúa en la página siguiente



## 1.32 CapC - Instrucción de movimiento CAP circular Continuous Application Platform (CAP) Continuación

|                  |  |
|------------------|--|
| CAP_SEN_EXALARM  | Se ha producido una alarma externa en el sensor.   |
| CAP_SEN_CAALARM  | Se ha producido una alarma de cámara en el sensor. |
| CAP_SEN_TEMP     | La temperatura del sensor está fuera de rango.     |
| CAP_SEN_VALUE    | El valor enviado al sensor está fuera de rango.    |
| CAP_SEN_CAMCHECK | Fallo al comprobar la cámara.                      |
| CAP_SEN_TIMEOUT  | El sensor no respondió dentro del tiempo límite.   |

### Errores posibles en sistemas MultiMove

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

|               |   |
|---------------|---|
| ERR_PATH_STOP | Cuando se utiliza el movimiento sincronizado se informa de este error cuando una aplicación que controla una unidad mecánica detecta un error recuperable y notifica a otras aplicaciones que algo ha ido mal. Si este código de error se recibe de una instrucción <code>CapC</code> , el error es una reacción a otro error. Todas las tareas que utilizan instrucciones de movimiento en el modo sincronizado en un sistema <code>MultiMove</code> deben tener este valor <code>ERRNO</code> definido en el gestor de errores. |
|---------------|---|

### Errores heredados de `TriggX`

La instrucción `CapC` se basa en la instrucción `TriggC`. Como consecuencia puede obtener y gestionar los errores `ERR_AO_LIM` y `ERR_DIPLAG_LIM`, como en `TriggC`.

El valor de la variable de sistema `ERRNO` cambia a:

|                |  |
|----------------|--|
| ERR_AO_LIM     | Si los resultados del argumento <code>ScaleValue/SetValue</code> programado para la señal analógica de salida especificada <code>AOp/AOutput</code> en alguna de las instrucciones <code>TriggSpeed/TriggRampAO</code> están fuera del límite de la señal analógica que corresponde a la velocidad <code>Speed</code> programada en esta instrucción, la variable de sistema <code>ERRNO</code> cambia a <code>ERR_AO_LIM</code> . |
| ERR_DIPLAG_LIM | Si el argumento <code>DipLag</code> programado en alguna de las instrucciones <code>TriggSpeed</code> conectadas es demasiado grande respecto al parámetro de sistema utilizado <i>Event Preset Time</i> , la variable del sistema <code>ERRNO</code> cambia a <code>ERR_DIPLAG_LIM</code> .   |

### Otros errores CAP

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

|                  |   |
|------------------|---|
| CAP_ATPROC_START | Este error recuperable se genera al final de la primera instrucción <code>CapC/L</code> de la secuencia si se utiliza el argumento opcional <code>\PreProcessTracking</code> . Puede ser gestionado en el gestor de errores de la rutina al inicio del proceso.<br>Para obtener más información, consulte <i>Operating manual - Tracking and searching with optical sensors</i> . |
| CAP_NOPROC_END   | Este error se produce cuando se utiliza la instrucción <code>CapNoProcess</code> para ejecutar una distancia concreta sin proceso de aplicación y se alcanza el final de esta distancia. No es realmente un error, pero utiliza los mecanismos de recuperación de errores.  |

Continúa en la página siguiente

# 1 Instrucciones

## 1.32 CapC - Instrucción de movimiento CAP circular

### Continuous Application Platform (CAP)

#### Continuación

|                  |   |
|------------------|---|
| CAP_MOV_WATCHDOG | Este error se produce cuando se especifica el modificador <code>\Movestart_timer</code> y el tiempo entre el inicio del proceso (MAIN_STARTED) y el inicio del movimiento del robot supera el tiempo especificado con el modificador. |
|------------------|---|

#### Ejecución de programas

Consulte [MoveL - Mueve el robot siguiendo una trayectoria lineal en la página 447](#) para obtener más información acerca del movimiento lineal.

Consulte [TriggL - Movimiento lineal del robot con eventos en la página 940](#) para obtener más información acerca del movimiento lineal con eventos de disparo.

#### Proceso CAP

Durante la ejecución continua tanto en Modo automático como en Modo manual, el proceso CAP se está ejecutando a no ser que se bloquee. Esto quiere decir que se utilizan todos los datos que controlan el proceso CAP (es decir, `Cdata`, `Weavestart`, `Weave` y `Movestart_timer`). En estos modos se realizan todas las actividades de disparo CAP; consulte [ICap - Conectar eventos CAP a rutinas TRAP en la página 248](#).

En todos los otros modos de ejecución no se ejecuta el proceso CAP y la instrucción `CapC` se comporta como una instrucción `MoveC`.

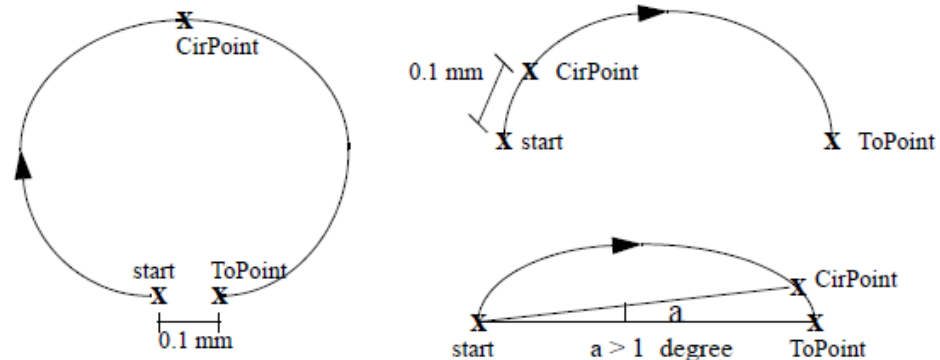
#### Condiciones de disparador [T1] a [T8]

A medida que se cumplen las condiciones de disparo cuando el robot se sitúa más y más cerca del punto final, se realizan las actividades de disparo definidas. Las condiciones de disparo se cumplen a una distancia determinada del punto final de la instrucción o a una distancia determinada del punto de inicio de la instrucción, o bien en un momento determinado (limitado a un tiempo breve) antes del punto final de la instrucción.

Durante la ejecución paso a paso hacia delante, las actividades de E/S se realizan, pero las rutinas de interrupción no se ejecutan. Durante la ejecución paso a paso hacia atrás, no se realiza ninguna actividad de disparo.

#### Limitaciones

Existen algunas limitaciones en cuanto a las posibilidades de posicionamiento de `CirPoint` y `ToPoint`, como se muestra en la figura siguiente.



xx120000175

Continúa en la página siguiente

- La distancia mínima entre el punto de inicio y ToPoint es de 0,1 mm.
- La distancia mínima entre el punto de inicio y CirPoint es de 0,1 mm.
- El ángulo mínimo entre CirPoint y ToPoint a partir del punto de inicio es de 1 grado.

La exactitud puede ser baja cerca de los límites, por ejemplo, si el punto de inicio y el punto ToPoint del círculo están muy cerca el uno del otro, el error causado por la inclinación del círculo puede ser mucho mayor que la exactitud con la que se programaron los puntos.

Los cambios del modo de ejecución de la ejecución hacia adelante a la ejecución hacia atrás, o viceversa, mientras el robot se detiene en una trayectoria circular no se permiten y generan un mensaje de error.

La instrucción CapC (o cualquier otra instrucción que incluya un movimiento circular) no debe empezarse en ningún caso desde el principio, con un TCP entre el punto de círculo y el punto final. De lo contrario, el robot no toma la trayectoria programada (posicionamiento alrededor de la trayectoria circular en otra dirección, en comparación con la programada).

Asegúrese de que el robot pueda alcanzar el punto de círculo durante la ejecución del programa y dividir el segmento del círculo si es necesario.

Si el punto de inicio actual se desvía de lo habitual, de forma que la longitud de posicionamiento total de la instrucción CapC es más corta de lo habitual, es posible que todas las condiciones de disparo o algunas de ellas se satisfagan inmediatamente en la misma posición. En estos casos, la secuencia en la que se realizan las actividades de disparo no estará definida. La lógica de programa del programa del usuario no puede basarse en una secuencia normal de actividades de disparo para un “movimiento incompleto”.

CapC no puede ejecutarse en rutinas de RAPID que estén conectadas a los siguientes eventos especiales del sistema: PowerOn, Stop, QStop, Restart, Reset o Step.

### Sintaxis

```
CapC
[ CirPoint ::= ] < expression (IN) of robtarg >
[ ToPoint ::= ] < expression (IN) of robtarg >
[ '\ 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]
|[ '\ PreProcessTracking]
[ '\ Time ::= ' < expression (IN) of num > ]
```

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.32 CapC - Instrucción de movimiento CAP circular

### Continuous Application Platform (CAP)

#### Continuación

```
[ '\ ' 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 > ] ';' 
```

#### Información relacionada

| Para obtener más información sobre                           | Consulte  |
|--|---|
| <i>Continuous Application Platform</i>                       | <i>Application manual - Continuous Application Platform</i>                         |
| Movimiento circular  | <a href="#">MoveC - Mueve el robot en círculo en la página 389</a>                  |
| Movimiento circular con disparadores                         | <a href="#">TriggC - Movimiento circular del robot con eventos en la página 895</a> |
| Definición de datos CAP                                      | <a href="#">capdata - Datos CAP en la página 1624</a>                               |
| Definición de datos de inicio de oscilación                  | <a href="#">weavestartdata - Datos de inicio de oscilación en la página 1821</a>    |
| Definición de datos de oscilación                            | <a href="#">capweavedata - Datos de oscilación para CAP en la página 1638</a>       |
| Definición de datos de track                                 | <a href="#">captrackdata - Datos de seguimiento de CAP en la página 1635</a>        |
| <i>Path Offset</i>   | <i>Application manual - Controller software IRC5</i>                                |
| Utilizar sensores ópticos para el seguimiento o la búsqueda. | <i>Operating manual - Tracking and searching with optical sensors</i>               |

### 1.33 CapCondSetDO - Activar una señal digital de salida en la parada de TCP Continuous Application Platform (CAP)

## 1.33 CapCondSetDO - Activar una señal digital de salida en la parada de TCP

### Utilización

CapCondSetDO se utiliza para definir una señal digital de salida y su valor, que cambiará cuando el TCP del robot que ejecuta CAP detenga su movimiento durante una instrucción CAP (CapL o CapC) antes de que finalice la secuencia de CAP. Una definición existente de este tipo de señales se borra con la instrucción CAP InitSuperv.

### Ejemplo básico

```
CapCondSetDO do15, 1;
```

La señal do15 cambia a 1 cuando el TCP se detiene.

```
CapCondSetDO weld, off;
```

La señal de soldadura cambia a off cuando se detiene el TCP.

### Argumentos

```
CapCondSetDO Signal Value
```

#### Signal

Tipo de dato: signaldo

El nombre de la señal que debe cambiar.

#### Value

Tipo de dato: dionum

El valor deseado para la señal, 0 ó 1.

| Value especificado        | Cambiar la salida digital a |
|---------------------------|-----------------------------|
| 0                         | 0                           |
| Cualquier valor excepto 0 | 1                           |

### Limitaciones

El valor final de la señal depende de la configuración de la señal. Si la señal está invertida en los parámetros del sistema, el valor del canal físico es el opuesto.

Puede configurarse un máximo de 10 señales por tarea RAPID.

### Sintaxis

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

### Información relacionada

| Para obtener más información sobre | Consulte   |
|------------------------------------|--|
| Continuous Application Platform    | Application manual - Continuous Application Platform                                   |
| Instrucción InitSuperv             | <a href="#">InitSuperv - Restablecer toda la supervisión para CAP en la página 286</a> |

Continúa en la página siguiente

# 1 Instrucciones

---

## 1.33 CapCondSetDO - Activar una señal digital de salida en la parada de TCP

*Continuous Application Platform (CAP)*

*Continuación*

| Para obtener más información sobre    | Consulte   |
|---------------------------------------|--|
| Instrucción <code>SetupSuperv</code>  | <a href="#"><i>SetupSuperv - Configurar las condiciones para la supervisión de señales en CAP en la página 719</i></a> |
| Instrucción <code>RemoveSuperv</code> | <a href="#"><i>RemoveSuperv - Eliminar la condición de una señal en la página 602</i></a>                              |

## 1.34 CapEquiDist - Generar evento equidistante

### Utilización

CapEquiDist se utiliza para indicar a CAP que genere un evento de RAPID equidistante (EQUIDIST) en la trayectoria CAP. El primer evento se genera en el punto de inicio de la primera instrucción CAP en una secuencia de instrucciones CAP. Desde RAPID es posible suscribirse a este evento mediante ICap.

### Ejemplo básico

```
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
```

En este ejemplo, el evento EQUIDIST se generará en la primera trayectoria de CAP. Se enviará cada 5 mm en la trayectoria a varias instrucciones CAP con zonas.

### Argumentos

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

#### [Distance]

*Distancia en mm*

Tipo de dato: num

Los datos proporcionados con este argumento opcional definen la distancia en mm entre dos eventos equidistantes consecutivos.

#### [Reset]

*Restablecer generación de eventos*

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.34 CapEquiDist - Generar evento equidistante

### *Continuous Application Platform (CAP)*

#### *Continuación*

Tipo de dato: `switch`

Si se utiliza este modificador, se restablece la generación de eventos, es decir, el evento equidistante no se generará más en una trayectoria `CapL/CapC`. Este modificador tiene preferencia antes del modificador `\Distance`.

---

#### Limitaciones

Si la trayectoria de CAP es larga comparada con la distancia del evento, el sistema puede agotar los recursos de eventos y aparece el mensaje de error **50368 Distancia demasiado corta entre eventos equidistantes**.

---

#### Sintaxis

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

---

#### Información relacionada

| Para obtener más información sobre     | Consulte  |
|--|---|
| <i>Continuous Application Platform</i> | <i>Application manual - Continuous Application Platform</i> |



## 1.35 CapL - Instrucción de movimiento CAP lineal

## Utilización

CapL se utiliza para mover el punto central de la herramienta (TCP) linealmente a un destino determinado y a la vez controlar un proceso continuo. Además, es posible conectar hasta ocho eventos a CapL. Los eventos se definen utilizando las instrucciones TriggRampAO, TriggIO, TriggEquip, TriggInt, TriggCheckIO o TriggSpeed.

## Ejemplos básicos

## Ejemplo 1

Movimiento lineal con CapL.

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

El TCP de la herramienta, gun1, se mueve a lo largo de una línea hacia la posición p1 con la velocidad definida en cdata y los datos de zona z50.

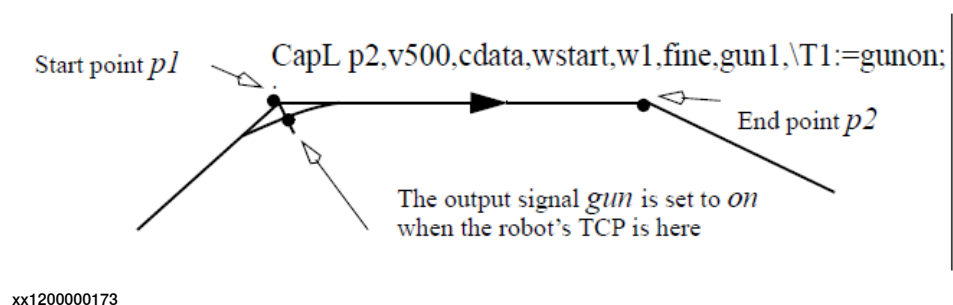
## Ejemplo 2

Movimiento circular con evento de usuario y evento CAP.

```
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
```

La señal digital de salida gun se activa cuando el TCP del robot atraviesa el punto central de la trayectoria de esquina del punto p1. La rutina TRAP start\_trap se ejecuta cuando se inicia el proceso CAP.



xx120000173

Continúa en la página siguiente

# 1 Instrucciones

---

## 1.35 CapL - Instrucción de movimiento CAP lineal

### *Continuous Application Platform (CAP)*

#### *Continuación*

---

#### Argumentos

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

ToPoint

**Tipo de dato:** robtarget

El punto de destino de los ejes del robot y de los ejes adicionales. Se define como una posición con nombre o se almacena directamente en la instrucción (marcada con un asterisco \* en la instrucción).

[ \Id ]

*(Identidad de sincronización)*

**Tipo de dato:** identno

La identidad de sincronización para las instrucciones de RAPID de movimiento en un sistema MultiMove en modo sincronizado.

Speed

**Tipo de dato:** speeddata

Los datos de velocidad que se aplican a los movimientos sin el proceso CAP activo. Los datos de velocidad definen la velocidad del punto central de la herramienta, los ejes adicionales y la reorientación de la herramienta. Si hay un proceso CAP activo (no bloqueado), el argumento Cdata define la velocidad del TCP.

Cdata

*(Datos de proceso CAP)*

**Tipo de dato:** capdata

Datos de proceso CAP; consulte [capdata - Datos CAP en la página 1624](#) para obtener una descripción detallada.

[ \Movestart\_timer ]

*(Tiempo en s)*

**Tipo de dato:** num

Límite superior para la diferencia de tiempo entre la orden de inicio del proceso y el inicio real del movimiento del TCP de los robots en un sistema MultiMove en modo sincronizado.

Weavestart

*(Datos de inicio de oscilación)*

**Tipo de dato:** weavestartdata

Datos de inicio de oscilación para el proceso CAP; consulte [weavestartdata - Datos de inicio de oscilación en la página 1821](#) para obtener una descripción detallada.

Weave

*(Datos de oscilación)*

**Tipo de dato:** capweavedata

*Continúa en la página siguiente*

## 1.35 CapL - Instrucción de movimiento CAP lineal *Continuous Application Platform (CAP)* Continuación

Datos de inicio de oscilación para el proceso CAP; consulte [capweavedata - Datos de oscilación para CAP en la página 1638](#) para obtener una descripción detallada.

Zone

Tipo de dato: zonedata

Los datos de zona del movimiento. Los datos de zona describen el tamaño de la trayectoria de esquina generada.

[ \Inpos ]

*(En posición)*

Tipo de dato: stoppointdata

Este argumento se utiliza para especificar los criterios de convergencia para la posición del TCP del robot en el punto de paro. Los datos de puntos de paro sustituyen a la zona especificada en el parámetro Zone.

Tool

Tipo de dato: tooldata

La herramienta en uso durante el movimiento del robot. El punto central de la herramienta (TCP) es el punto que se mueve hacia la posición de destino especificada.

[ \Wobj ]

Tipo de dato: wobjdata

El objeto de trabajo (sistema de coordenadas) con el que está relacionada la posición de robot indicada en la instrucción.

Es posible omitir este argumento. Si se omite, la posición depende del sistema de coordenadas mundo. Si, por otro lado, se usa un TCP estacionario o ejes externos adicionales, es necesario especificar este argumento para ejecutar un movimiento lineal respecto del objeto de trabajo.

[ \Track ]

*(Datos de sensor de track)*

Tipo de dato: captrackdata

Esta estructura de datos contiene los datos necesarios para el uso de la corrección de trayectoria que generan los sensores junto con CapL; consulte [captrackdata - Datos de seguimiento de CAP en la página 1635](#). Este argumento no se permite junto con el argumento \Corr.

[ \Corr ]

*(Utilizar generador de correcciones)*

Tipo de dato: switch

Este argumento dice a CapL que lea las correcciones de trayectoria de un generador de correcciones, CorrCon. Este argumento no se admite junto con el argumento \Track.

Se requiere RobotWare, opción *Path Offset*, cuando se utiliza este argumento.

[ \PreProcessTracking ]

Tipo de dato: switch

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.35 CapL - Instrucción de movimiento CAP lineal

### Continuous Application Platform (CAP)

#### Continuación

Este argumento solo es eficaz si `first_instruction` cambia a `TRUE` y se ha incluido el argumento `\Track`.

Este argumento activa el *seguimiento del proceso previo*, lo que significa que durante la instrucción `CapX` el robot solo hará el seguimiento, sin proceso. Por lo tanto, los datos del sensor están disponibles para el seguimiento desde el inicio de la trayectoria con proceso, p. ej. de soldadura.

Para obtener más información, consulte *Operating manual - Tracking and searching with optical sensors*.

[ `\Time` ]

Tipo de dato: num

Este argumento se utiliza para especificar el tiempo total en segundos que dura el movimiento del robot y de los ejes adicionales. A continuación, se sustituye por los datos de velocidad correspondientes.

[ `\T1` ] a [ `\T8` ]

(*Trigg x*)

Tipo de dato: trigndata

Las variables que hacen referencia a las condiciones de disparo y la actividad de disparo, definidas anteriormente en el programa usando las instrucciones

`TriggRampAO`, `TriggIO`, `TriggEquip` o `TriggInt`.

[ `\TLoad` ]

Tipo de dato: loaddata

El argumento `\TLoad` describe la carga total usada durante el movimiento. La carga total es la carga de la herramienta más la carga útil transportada por la herramienta. Si se utiliza el argumento `\TLoad`, no se tiene en cuenta el valor de `loaddata` en los `tooldata` actuales.

Si el argumento `\TLoad` se configura como `load0`, entonces no se toma en consideración el argumento y en su lugar se utiliza el `loaddata` en el `tooldata` actual. Para una descripción completa del argumento `TLoad`, consulte `MoveL`.

---

## Gestión de errores

Existen varios tipos diferentes de errores que pueden gestionarse en el gestor de errores para las instrucciones `CapC/CapL`:

- errores de supervisión
- errores específicos de sensores
- error específico para un sistema `MultiMove`
- errores heredados de la funcionalidad `TriggX`
- otros errores `CAP`

Si una de las señales que se supone que se va supervisar no tiene el valor correcto o si cambia el valor durante la supervisión, se establece la variable del sistema `ERRNO`.

Si no se pueden leer los valores del sensor de track, se establece la variable de sistema `ERRNO`.

*Continúa en la página siguiente*

Para un sistema MultiMove que se ejecuta en modo sincronizado, el gestor de errores debe ocuparse de otros dos errores. Uno se utiliza para informar de que alguna otra aplicación ha detectado un error recuperable. Esto activa la gestión de errores recuperables en las tareas RAPID sincronizadas. Se informa del otro error, `CAP_MOV_WATCHDOG`, si vence el tiempo entre la petición de inicio de proceso y el inicio real del movimiento del TCP de los robots en un sistema MultiMove en modo sincronizado. El tiempo utilizado se especifica en el parámetro opcional `Movestart_timer` de la instrucción `CapL`.

Si se detecta cualquier situación anormal, la ejecución del programa se detendrá. Sin embargo, si se ha programado un gestor de errores, los errores definidos a continuación pueden solucionarse sin detener la producción. Aunque se recomienda que algunos de estos errores (los errores con `CAP_XX`) no se muestren al usuario final. Asigne esos errores a un error específico de la aplicación. Para los errores de supervisión, puede utilizarse la instrucción `CapGetFailSigs` para ver qué señal específica ha fallado.

### Errores de supervisión

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

|                                  |   |
|----------------------------------|---|
| <code>CAP_PRE_ERR</code>         | Este error se produce cuando hay un error en la lista de supervisión <code>PRE</code> , es decir, cuando no se cumplen las condiciones de la lista dentro del periodo de tiempo especificado (especificado en tiempo límite de <code>pre_cond</code> ).               |
| <code>CAP_PRESTART_ERR</code>    | Este error se produce cuando hay un error durante la supervisión de la fase <code>PRE</code> .  |
| <code>CAP_END_PRE_ERR</code>     | Este evento se produce cuando hay un error en la lista de supervisión <code>END_PRE</code> , es decir, cuando no se cumplen las condiciones de la lista dentro del periodo de tiempo especificado (especificado en tiempo límite de <code>start_cond</code> ).        |
| <code>CAP_START_ERR</code>       | Este evento se produce cuando hay un error en la lista de supervisión <code>START</code> , es decir, cuando no se cumplen las condiciones de la lista dentro del periodo de tiempo especificado (especificado en tiempo límite de <code>start_cond</code> ).          |
| <code>CAP_MAIN_ERR</code>        | Este error se produce cuando hay un error durante la supervisión de la fase principal.  |
| <code>CAP_ENDMAIN_ERR</code>     | Este error se produce cuando hay un error en la lista de supervisión <code>END_MAIN</code> , es decir, cuando no se cumplen las condiciones de la lista dentro del periodo de tiempo especificado (especificado en tiempo límite de <code>end_main_cond</code> ).     |
| <code>CAP_START_POST1_ERR</code> | Este evento se produce cuando hay un error en la lista de supervisión <code>START_POST1</code> , es decir, cuando no se cumplen las condiciones de la lista dentro del periodo de tiempo especificado (especificado en tiempo límite de <code>end_main_cond</code> ). |
| <code>CAP_POST1_ERR</code>       | Este error se produce cuando hay un error durante la supervisión de la fase <code>POST1</code> .  |
| <code>CAP_POST1END_ERR</code>    | Este error se produce cuando hay un error en la lista de supervisión <code>END_POST1</code> , es decir, cuando no se cumplen las condiciones de la lista dentro del periodo de tiempo especificado (especificado en tiempo límite de <code>end_main_cond</code> ).    |

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.35 CapL - Instrucción de movimiento CAP lineal

### Continuous Application Platform (CAP)

#### Continuación

|                     |  |
|---------------------|--|
| CAP_START_POST2_ERR | Este evento se produce cuando hay un error en la lista de supervisión START_POST1, es decir, cuando no se cumplen las condiciones de la lista dentro del periodo de tiempo especificado (especificado en tiempo límite de end_main_cond).  |
| CAP_POST2_ERR       | Este error se produce cuando hay un error durante la supervisión de la fase POST2.   |
| CAP_POST2END_ERR    | Este error se produce cuando hay un error en la lista de supervisión END_POST2, es decir, cuando no se cumplen las condiciones de la lista dentro del periodo de tiempo especificado (especificado en tiempo límite de end_main_cond).<br><br>Si la supervisión se realiza en dos señales diferentes de la misma fase y ambas fallan, la primera que esté configurada con SetupSuperv es la que genera el error. |

#### Errores relacionados con el sensor

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema ERRNO cambiará a:

|                  |   |
|------------------|---|
| CAP_TRACK_ERR    | Se produce un error de track cuando se leen los datos del sensor y transcurrido un tiempo no se reciben datos válidos. Una razón podría ser que el sensor no puede indicar el cordón. |
| CAP_TRACKSTA_ERR | Se produce un error de inicio de track cuando no se han leído datos válidos del sensor de track láser.  |
| CAP_TRACKCOR_ERR | Se produce un error de corrección de track cuando algo va mal en el cálculo del offset.   |
| CAP_TRACKCOM_ERR | La comunicación entre el controlador del robot y el equipo del sensor no funciona.  |
| CAP_TRACKPFR_ERR | No es posible continuar con el seguimiento si se ha producido un fallo de alimentación eléctrica durante el seguimiento.  |
| CAP_SEN_NO_MEAS  | El controlador no obtuvo una medición válida del sensor.  |
| CAP_SEN_NOREADY  | El sensor no está preparado todavía.  |
| CAP_SEN_GENERRO  | Se produjo un error general del sensor.   |
| CAP_SEN_BUSY     | El sensor está ocupado y no puede responder a la solicitud.   |
| CAP_SEN_UNKNOWN  | El comando enviado al sensor es desconocido para el sensor.   |
| CAP_SEN_ILLEGAL  | El número de variable o el número de bloqueo enviado al sensor es ilegal.   |
| CAP_SEN_EXALARM  | Se ha producido una alarma externa en el sensor.  |
| CAP_SEN_CAALARM  | Se ha producido una alarma de cámara en el sensor.  |
| CAP_SEN_TEMP     | La temperatura del sensor está fuera de rango.  |
| CAP_SEN_VALUE    | El valor enviado al sensor está fuera de rango.   |
| CAP_SEN_CAMCHECK | Fallo al comprobar la cámara.   |
| CAP_SEN_TIMEOUT  | El sensor no respondió dentro del tiempo límite.  |

Continúa en la página siguiente

#### Errores posibles en sistemas MultiMove

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

|                            |   |
|----------------------------|---|
| <code>ERR_PATH_STOP</code> | Cuando se utiliza el movimiento sincronizado se informa de este error cuando una aplicación que controla una unidad mecánica detecta un error recuperable y notifica a otras aplicaciones que algo ha ido mal. Si este código de error se recibe de una instrucción <code>CapL</code> , el error es una reacción a otro error. Todas las tareas que utilizan instrucciones de movimiento en el modo sincronizado en un sistema <code>MultiMove</code> deben tener este valor <code>ERRNO</code> definido en el gestor de errores. |
|----------------------------|---|

#### Errores heredados de `TriggX`

La instrucción `CapL` se basa en la instrucción `TriggL`. Como consecuencia puede obtener y gestionar los errores `ERR_AO_LIM` y `ERR_DIPLAG_LIM`, como en `TriggL`.

El valor de la variable de sistema `ERRNO` cambia a:

|                             |  |
|-----------------------------|--|
| <code>ERR_AO_LIM</code>     | Si los resultados del argumento <code>ScaleValue/SetValue</code> programado para la señal analógica de salida especificada <code>AOp/AOutput</code> en alguna de las instrucciones <code>TriggSpeed/TriggRampAO</code> están fuera del límite de la señal analógica que corresponde a la velocidad <code>Speed</code> programada en esta instrucción, la variable de sistema <code>ERRNO</code> cambia a <code>ERR_AO_LIM</code> . |
| <code>ERR_DIPLAG_LIM</code> | Si el argumento <code>DipLag</code> programado en alguna de las instrucciones <code>TriggSpeed</code> conectadas es demasiado grande respecto al parámetro de sistema utilizado <i>Event Preset Time</i> , la variable del sistema <code>ERRNO</code> cambia a <code>ERR_DIPLAG_LIM</code> .   |

#### Otros errores CAP

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

|                               |  |
|-------------------------------|--|
| <code>CAP_ATPROC_START</code> | Este error recuperable se genera al final de la primera instrucción <code>CapC/L</code> de la secuencia si se utiliza el argumento opcional <code>\PreProcessTracking</code> . Puede ser gestionado en el gestor de errores de la rutina al inicio del proceso. Para obtener más información, consulte <i>Operating manual - Tracking and searching with optical sensors</i> . |
| <code>CAP_NOPROC_END</code>   | Este error se produce cuando se utiliza la instrucción <code>CapNoProcess</code> para ejecutar una distancia concreta sin proceso de aplicación y se alcanza el final de esta distancia. No es realmente un error, pero utiliza los mecanismos de recuperación de errores.   |
| <code>CAP_MOV_WATCHDOG</code> | Este error se produce cuando se especifica el modificador <code>\Movestart_timer</code> y el tiempo entre el inicio del proceso ( <code>MAIN_STARTED</code> ) y el inicio del movimiento del robot supera el tiempo especificado con el modificador.   |

#### Ejecución de programas

Consulte [MoveL - Mueve el robot siguiendo una trayectoria lineal en la página 447](#) para obtener más información acerca del movimiento lineal.

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.35 CapL - Instrucción de movimiento CAP lineal

*Continuous Application Platform (CAP)*

*Continuación*

Consulte [TriggL - Movimiento lineal del robot con eventos en la página 940](#) para obtener más información acerca del movimiento lineal con eventos de disparo.

---

### Proceso CAP

Durante la ejecución continua tanto en Modo automático como en Modo manual, el proceso CAP se está ejecutando a no ser que se bloquee. Esto quiere decir que se utilizan todos los datos que controlan el proceso CAP (es decir, `Cdata`, `Weavestart`, `Weave` y `Movestart_timer`). En estos modos se realizan todas las actividades de disparador CAP; consulte [ICap - Conectar eventos CAP a rutinas TRAP en la página 248](#).

En todos los otros modos de ejecución no se ejecuta el proceso CAP y la instrucción `CapL` se comporta como una instrucción `MoveL`.

---

### Condiciones de disparador [T1] a [T8]

A medida que se cumplen las condiciones de disparo cuando el robot se sitúa más y más cerca del punto final, se realizan las actividades de disparo definidas. Las condiciones de disparo se cumplen a una distancia determinada del punto final de la instrucción o a una distancia determinada del punto de inicio de la instrucción, o bien en un momento determinado (limitado a un tiempo breve) antes del punto final de la instrucción.

Durante la ejecución paso a paso hacia delante, las actividades de E/S se realizan, pero las rutinas de interrupción no se ejecutan. Durante la ejecución paso a paso hacia atrás, no se realiza ninguna actividad de disparo.

---

### Limitaciones

Si el punto de inicio actual se desvía del habitual, de forma que la longitud de posicionamiento total de la instrucción `CapL` es más corta de lo habitual (por ejemplo, al principio de `CapL` con la posición del robot en el punto final), puede ocurrir que se cumplan varias de las condiciones de disparo, o incluso todas ellas, inmediatamente y en una misma posición. En estos casos, la secuencia en la que se realizan las actividades de disparo no estará definida. La lógica de programa del programa del usuario no puede basarse en una secuencia normal de actividades de disparo para un “movimiento incompleto”.

El comportamiento del proceso CAP puede ser indefinido si se produce un error durante las instrucciones `CapL` o `CapC` con movimientos TCP extremadamente cortos (< 1 mm).

`CapL` no puede ejecutarse en rutinas de RAPID que estén conectadas a los siguientes eventos especiales del sistema: `PowerOn`, `Stop`, `QStop`, `Restart`, `Reset` o `Step`.

---

### Sintaxis

```
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 >] ', '
```

*Continúa en la página siguiente*

---



## 1.35 CapL - Instrucción de movimiento CAP lineal

### *Continuous Application Platform (CAP)*

#### *Continuación*

```
[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]
|[ '\ ' PreProcessTracking]
['\ ' 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 >] ';'

```

#### Información relacionada

| Para obtener más información sobre                           | Consulte   |
|--|--|
| <i>Continuous Application Platform</i>                       | <i>Application manual - Continuous Application Platform</i>                              |
| Movimiento lineal  | <a href="#">MoveL - Mueve el robot siguiendo una trayectoria lineal en la página 447</a> |
| Movimiento lineal con disparadores                           | <a href="#">TriggL - Movimiento lineal del robot con eventos en la página 940</a>        |
| Definición de datos CAP                                      | <a href="#">capdata - Datos CAP en la página 1624</a>                                    |
| Definición de datos de inicio de oscilación                  | <a href="#">weavestartdata - Datos de inicio de oscilación en la página 1821</a>         |
| Definición de datos de oscilación                            | <a href="#">capweavedata - Datos de oscilación para CAP en la página 1638</a>            |
| Definición de datos de track                                 | <a href="#">captrackdata - Datos de seguimiento de CAP en la página 1635</a>             |
| <i>Path Offset</i>   | <i>Application manual - Controller software IRC5</i>                                     |
| Utilizar sensores ópticos para el seguimiento o la búsqueda. | <i>Operating manual - Tracking and searching with optical sensors</i>                    |

# 1 Instrucciones

---

## 1.36 CapLATrSetup - Configurar un sensor de seguimiento anticipatorio *Continuous Application Platform (CAP)*

### 1.36 CapLATrSetup - Configurar un sensor de seguimiento anticipatorio

---

#### Utilización

CapLATrSetup (*Configurar un sensor de seguimiento anticipatorio*) se utiliza para configurar un tipo de sensor de seguimiento anticipatorio, por ejemplo, seguimiento láser.

La interfaz de sensores se comunica con un máximo de dos sensores a través de canales serie, utilizando el protocolo de transporte RTP1. Los dos canales deben tener los nombres *laser1:* y *swg:*.

---

#### Ejemplo básico

##### SIO.cfg:

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

##### Código RAPID:

```
! 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;
```

#### Argumentos

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

device

Tipo de dato: string

*Continúa en la página siguiente*

---

## 1.36 CapLATrSetup - Configurar un sensor de seguimiento anticipatorio Continuous Application Platform (CAP) Continuación

El nombre del dispositivo como se definió en sio.cfg.

calibframe

Tipo de dato: `pose`

Base de coordenadas de calibración LATR (posición y orientación en relación a la herramienta predefinida tool0).

CalibPos

Tipo de dato: `pose`

Offset de calibración LATR. Ajuste de la base de coordenadas del sensor que coloca el origen del sistema de coordenadas de corrección de la trayectoria cerca del nivel de la base de coordenadas utilizada durante la calibración.

[WarnMaxCorr]

Tipo de dato: `switch`

Si se utiliza este modificador, la ejecución del programa no se interrumpe cuando se supera el límite de corrección máxima especificado en trackdata. Solo se enviará un aviso.

[Logfile]

Tipo de dato: `string`

Nombre del archivo de registro de Tracklog.

[LogSize]

Tipo de dato: `num`

Tamaño del búfer de anillo de Tracklog, que es el número de mediciones de sensor que pueden almacenarse en el búfer durante el seguimiento.

Predeterminado: 1000.

[SensorFreq]

Tipo de dato: `num`

Define la frecuencia de muestreo del sensor utilizado (por ejemplo, M-Spot-90 tiene una frecuencia de muestreo de 5 Hz).

El valor más alto disponible depende del enlace de comunicaciones y su velocidad. Se recomienda no utilizar valores superiores a los 20 Hz.

Predeterminado: 5 Hz.

[IpolServoDelay]

Tipo de dato: `num`

Define un retardo de tiempo interno del controlador de robot entre la tarea ipol y la tarea servo.

Predeterminado: 74 ms.



### Nota

¡No cambie el valor predeterminado!

Continúa en la página siguiente

## 1 Instrucciones

---

### 1.36 CapLATrSetup - Configurar un sensor de seguimiento anticipatorio

#### Continuous Application Platform (CAP)

##### Continuación

[IpolCorrGain]

Tipo de dato: num

Define el factor de ganancia para la corrección impuesta en ipol.

Predeterminado: 0.0.



#### Nota

¡No cambie el valor predeterminado!

[ServoSensFactor]

Tipo de dato: num

Define el número de correcciones de servo por lectura de sensor.

Predeterminado: 0.



#### Nota

¡No cambie el valor predeterminado!

[CorrFilter]

Tipo de dato: num

Define el filtrado de la corrección calculada, utilizando el valor medio de los valores de filtro de corr.

Predeterminado: 1.



#### Nota

¡No cambie el valor predeterminado!

[IpolCorrFilter]

Tipo de dato: num

Define el filtrado de la corrección ipol , utilizando el valor medio de los valores de filtro de trayectoria.

Predeterminado: 1.



#### Nota

¡No cambie el valor predeterminado!

[ServoCorrFilter]

Tipo de dato: num

Define el filtrado de la corrección de servo, utilizando el valor medio de los valores de filtro servo de trayectoria.

Predeterminado: 1.



#### Nota

¡No cambie el valor predeterminado!

*Continúa en la página siguiente*

## 1.36 CapLATrSetup - Configurar un sensor de seguimiento anticipatorio Continuous Application Platform (CAP) Continuación

### [ErrRampIn]

Tipo de dato: num

Define durante cuántas lecturas de sensor se realiza la activación en rampa después de un error causado por la lectura del sensor.

Predeterminado: 1.

### [ErrorRampOut]

Tipo de dato: num

Define durante cuántas lecturas de sensor se realiza la desactivación en rampa cuando se produce un error causado por la lectura del sensor.

Predeterminado: 1.

### [CBAngle]

Tipo de dato: num

Define el ángulo entre un haz de sensor 3D y el eje Z del sensor

Predeterminado: 0.0.

### [MaxBlind]

Tipo de dato: num

Distancia máxima que el TCP puede moverse suponiendo que la última corrección sigue siendo válida.

Al inicio del seguimiento, la distancia de `MaxBlind` se incrementa automáticamente por el adelanto del sensor.

Predeterminado: sin límite.

### [MaxIncCorr]

Tipo de dato: num

Corrección incremental máxima permitida.

Si la corrección incremental del TCP es superior a `\MaxIncCorr` y se especificó `\WarnMaxCorr`, el robot continúa su trayectoria, pero la corrección incremental aplicada no supera `\MaxIncCorr`. Si no se especificó `\WarnMaxCorr`, se informa de un error de trayectoria y se detiene la ejecución del programa.

Predeterminado: 5 mm.

### [CalibFrame2]

Tipo de dato: pose

El número de bases de coordenadas de calibración LATR 2 (posición y orientación relativas a la herramienta predefinida `tool0`).

### [CalibFrame3]

Tipo de dato: pose

El número de bases de coordenadas de calibración LATR 3 (posición y orientación relativas a la herramienta predefinida `tool0`).

---

### Sintaxis

```
CapLATrSetup  
[device ':='] < expression (IN) of string> ','
```

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.36 CapLATrSetup - Configurar un sensor de seguimiento anticipatorio

### Continuous Application Platform (CAP)

#### Continuación

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

#### Información relacionada

| Para obtener más información sobre     | Consulte  |
|--|---|
| <i>Sensor Interface</i>                | <i>Application manual - Controller software IRC5</i>        |
| <i>Continuous Application Platform</i> | <i>Application manual - Continuous Application Platform</i> |

## 1.37 CapNoProcess - Ejecutar CAP sin proceso

### Utilización

CapNoProcess se utiliza para ejecutar CAP una cierta distancia sin proceso.

Con CapNoProcess, es posible indicar a CAP que ejecute una distancia concreta (en mm) sin proceso. Resulta útil si se produjo un error de proceso recuperable que hace imposible de alguna forma reiniciar el proceso en la ubicación del error.

Al comienzo y al final de la distancia de salto, se suprime el retroceso en la trayectoria (componente `restart_dist` de `capdata`).

Al final de la distancia del salto se genera un error con `errno CAP_NOPROC_END`.

### Ejemplo básico

```
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
```

En este ejemplo, al error recuperable `CAP_MAIN_ERR` le sigue un movimiento de 25 mm (a 10 mm/s) sin proceso para la primera instrucción `CapL` y de 15 mm para

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.37 CapNoProcess - Ejecutar CAP sin proceso

### *Continuous Application Platform (CAP)*

#### *Continuación*

el segundo. Al final de esa distancia, se genera CAP\_NOPROC\_END y se reinicia el proceso.

#### Argumentos

CapNoProcess skip\_distance

#### skip\_distance

*Distancia en mm*

Tipo de dato: num

CapNoProcess tiene una variable num como parámetro de entrada, que define la distancia del salto en mm.

#### Limitaciones

La velocidad del TCP durante el salto está predefinida con un valor de 10 mm/s. La distancia de salto más corta predefinida es de 10 mm.

En los sistemas MultiMove sincronizados, la distancia más corta de todas las distancias de salto definidas para los diferentes robots será la actual.

Si la distancia de salto es mayor que la distancia dese la posición actual del TCP al final de la secuencia actual de instrucciones CAP, no sucederá nada especial: la ejecución de RAPID continúa de la forma habitual, sin detener el robot.

#### Sintaxis

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

#### Información relacionada

| Para obtener más información sobre     | Consulte  |
|--|---|
| <i>Continuous Application Platform</i> | <i>Application manual - Continuous Application Platform</i>   |
| Instrucción InitSuperv                 | <a href="#">InitSuperv - Restablecer toda la supervisión para CAP en la página 286</a>                          |
| Instrucción SetupSuperv                | <a href="#">SetupSuperv - Configurar las condiciones para la supervisión de señales en CAP en la página 719</a> |
| Instrucción RemoveSuperv               | <a href="#">RemoveSuperv - Eliminar la condición de una señal en la página 602</a>                              |



## 1.38 CapRefresh - Actualizar datos de CAP

### Utilización

CapRefresh se utiliza para indicar al proceso CAP que actualice sus datos de proceso. Puede utilizarse, por ejemplo, para ajustar los parámetros del proceso CAP durante la ejecución del programa.

### Ejemplo básico

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

En este ejemplo la velocidad cambiará entre 10 y 15 mm/s a una frecuencia de 1 Hz.

### Argumentos

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

**Sin argumento opcional, los datos de CAP** capdata, capweavedata, weavestartdata, captrackdata y movestarttimer **–si los hay– se vuelven a leer de la variable de RAPID PERSISTENT especificada en la instrucción CAP activa en ese momento.**

#### [MainSpeed]

Tipo de dato: switch

Si se utiliza este modificador, CAP volverá a leer el componente capdata.speed\_data.main de la instrucción CAP activa en ese momento.

#### [MainWeave]

Tipo de dato: switch

Si se utiliza este modificador, CAP volverá a leer los componentes capweavedata.width, capweavedata.length, capweavedata.bias y capweavedata.active de la instrucción CAP activa en ese momento.

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.38 CapRefresh - Actualizar datos de CAP

### *Continuous Application Platform (CAP)*

#### *Continuación*

#### `[\StartWeave]`

Tipo de dato: bool

Si se utiliza este modificador, CAP utiliza su valor en lugar de `weavestartdata.active` de la instrucción CAP activa en ese momento. Los datos de la instrucción CAP activa en ese momento no se alteran.

#### `[\RestartDist]`

Tipo de dato: num

Si se utiliza este modificador, CAP utiliza su valor en lugar de `capdata.restart_dist` de la instrucción CAP activa en ese momento. Los datos de la instrucción CAP activa en ese momento no se alteran.

---

### Sintaxis

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

```

---

### Información relacionada

| Para obtener más información sobre     | Consulte  |
|--|---|
| <i>Continuous Application Platform</i> | <i>Application manual - Continuous Application Platform</i> |

1.39 CAPSetStopMode - Establecer el modo de paro para errores de ejecución  
*Continuous Application Platform (CAP)*

**1.39 CAPSetStopMode - Establecer el modo de paro para errores de ejecución**

**Utilización**

CAPSetStopMode establece el modo de paro que debe utilizarse cuando el movimiento del robot se detenga debido a un error de proceso. El modo de paro predeterminado es SMOOTH\_STOP\_ON\_PATH.

**Ejemplo básico**

```
CAPSetStopMode QUICK_STOP_ON_PATH;
```

**Argumentos**

```
CAPSetStopMode StopMode;
```

**StopMode**

Tipo de dato: capstopmode

Modo de paro, consulte [capstopmode - Define modos de paro para CAP en la página 1634](#).

**Sintaxis**

```
CAPSetStopMode  

  [StopMode ':='] <variable (VAR) of capstopmode> ';' 
```

**Información relacionada**

| Para obtener más información sobre     | Consulte  |
|--|---|
| Tipo de dato capstopmode               | <a href="#">capstopmode - Define modos de paro para CAP en la página 1634</a> |
| <i>Continuous Application Platform</i> | <i>Application manual - Continuous Application Platform</i>                   |

# 1 Instrucciones

---

## 1.40 CapWeaveSync - Configurar señales y niveles para la sincronización de oscilación Continuous Application Platform (CAP)

### 1.40 CapWeaveSync - Configurar señales y niveles para la sincronización de oscilación

---

#### Utilización

CapWeaveSync se utiliza para configurar señales de sincronización de oscilación sin sensores. Las señales de E/S deben estar definidas en EIO.cfg.

---

#### Ejemplo básico

##### Programa RAPID:

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

En este ejemplo, las señales `do_sync_left` and `do_sync_right` están configuradas con un nivel de oscilación del 80%.

La instrucción `CapWeaveSync` solo debe ejecutarse una vez, por ejemplo, en el shelf de puesta en marcha.

---

#### Argumentos

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

##### [\Reset]

Tipo de dato: switch

Borrar los datos de sincronización de oscilación.

##### [\DoLeft]

Tipo de dato: signaldo

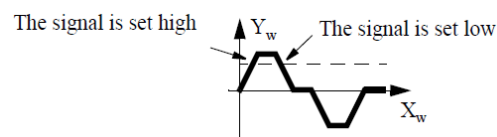
La señal digital de salida para la sincronización de la oscilación en el ciclo de oscilación izquierda.

##### [\LevelLeft]

Tipo de dato: num

La posición de coordinación en el lado izquierdo del patrón de oscilación. El valor especificado es un porcentaje de la anchura a la izquierda del centro de oscilación. Si la oscilación va más allá de este punto, se cambia automáticamente al nivel alto una señal de salida digital (siempre y cuando la señal esté definida).

El tipo de coordinación puede utilizarse para el seguimiento de cordones mediante el seguimiento de a través del arco.



xx120000176

Continúa en la página siguiente

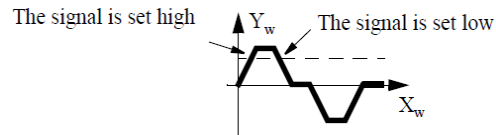
## 1.40 CapWeaveSync - Configurar señales y niveles para la sincronización de oscilación Continuous Application Platform (CAP) Continuación

[LevelLeft]

Tipo de dato: num

La posición de coordinación en el lado izquierdo del patrón de oscilación. El valor especificado es un porcentaje de la anchura a la izquierda del centro de oscilación. Si la oscilación va más allá de este punto, se cambia automáticamente al nivel alto una señal de salida digital (siempre y cuando la señal esté definida).

El tipo de coordinación puede utilizarse para el seguimiento de cordones mediante el seguimiento de a través del arco.



xx1200000176

[DoRight]

Tipo de dato: signaldo

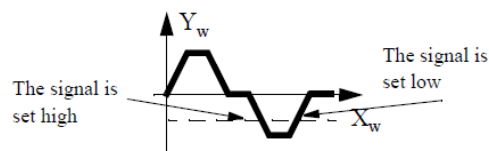
La señal digital de salida para la sincronización en el ciclo de oscilación derecha.

[LevelRight]

Tipo de dato: num

La posición de coordinación en el lado derecho del patrón de oscilación. El valor especificado es un porcentaje de la anchura a la derecha del centro de oscilación. Si la oscilación va más allá de este punto, se cambia automáticamente al nivel alto una señal de salida digital (siempre y cuando la señal esté definida).

El tipo de coordinación puede utilizarse para el seguimiento de cordones mediante el seguimiento de a través del arco.



xx1200000177

### Ejecución de programas

Las señales definidas se comprueban y definen durante la ejecución sin un sensor.

### Limitaciones

Las señales deben estar definidas en EIO.cfg.

No es posible utilizar solo o un nivel o la señal correspondiente. No se generarán errores al cargar el archivo RAPID, pero se producirán errores de tiempo de ejecución de RAPID para la instrucción CapWeaveSync.

### Sintaxis

```
CapWeaveSync
  ['\ ' Reset]
  [DoLeft ':=' < expression (IN) of signaldo >]
```

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.40 CapWeaveSync - Configurar señales y niveles para la sincronización de oscilación

### *Continuous Application Platform (CAP)*

#### *Continuación*

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

---

#### Información relacionada

| Para obtener más información sobre     | Consulte  |
|--|---|
| <i>Continuous Application Platform</i> | <i>Application manual - Continuous Application Platform</i>                   |
| Tipo de dato <code>capweavedata</code> | <a href="#">capweavedata - Datos de oscilación para CAP en la página 1638</a> |

## 1.41 CheckProgRef - Comprobar referencias de programa

### Utilización

CheckProgRef se usa para comprobar la existencia de referencias no resueltas en cualquier momento durante la ejecución.

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción CheckProgRef:

#### Ejemplo 1

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

En este caso, el programa contiene un módulo con el nombre PART\_A.MOD. Se carga un nuevo módulo PART\_B.MOD, que comprueba si todas las referencias son correctas. A continuación se descarga PART\_A.MOD. Para buscar referencias no resueltas tras la descarga, se realiza una llamada a CheckProgRef.

### Ejecución de programas

La ejecución del programa fuerza un nuevo enlace de la tarea de programa y comprueba la existencia de referencias no resueltas.

Si se produce un error durante CheckProgRef, el programa no se ve afectado. Sólo indica que existe una referencia sin resolver en la tarea de programa. Por tanto, utilice CheckProgRef inmediatamente después de cambiar el número de módulos de la tarea de programa (cargando o descargando), con el fin de saber qué módulo provocó el error de enlace.

Esta instrucción también puede usarse como sustituta del uso del argumento opcional \CheckRef en la instrucción Load o WaitLoad.

### Gestión de errores

Pueden generarse los errores recuperables enumerados a continuación. Estos errores pueden ser gestionados en un gestor de errores. El valor de la variable de sistema ERRNO cambia a:

| Nombre      | Causa del error   |
|-------------|---|
| ERR_LINKREF | La tarea de programa contiene referencias no resueltas. |

### Sintaxis

```
CheckProgRef';'
```

### Información relacionada

| Para obtener más información sobre | Consulte  |
|------------------------------------|---|
| Carga de un módulo de programa     | <a href="#">Load - Carga un módulo de programa durante la ejecución en la página 347</a>        |
| Descarga de un módulo de programa  | <a href="#">UnLoad - Descargar un módulo de programa durante la ejecución en la página 1021</a> |

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.41 CheckProgRef - Comprobar referencias de programa

RobotWare - OS

Continuación

| Para obtener más información sobre                | Consulte  |
|---|---|
| Inicio de la carga de un módulo de programa       | <a href="#">StartLoad - Carga de programa durante la ejecución en la página 792</a> |
| Finalización de la carga de un módulo de programa | <a href="#">WaitLoad - Conectar un módulo cargado a una tarea en la página 1064</a> |



### 1.42 CirPathMode - Reorientación de la herramienta durante trayectorias circulares

#### Utilización

`CirPathMode` (*Circle Path Mode*) permite seleccionar formas distintas de reorientar la herramienta durante los movimientos circulares.

Esta instrucción sólo puede usarse en la tarea principal `T_ROB1` o, si se cuenta con un sistema `MultiMove`, en las tareas de movimiento.

#### Ejemplos básicos

Los ejemplos siguientes ilustran la instrucción `CirPathMode`.

##### Ejemplo 1

```
CirPathMode \PathFrame;
```

El modo estándar de reorientación de la herramienta en la base de coordenadas de la trayectoria actual, desde el punto de inicio hasta el `ToPoint` durante todos los movimientos circulares posteriores. Éste es el modo predeterminado del sistema.

##### Ejemplo 2

```
CirPathMode \ObjectFrame;
```

El modo modificado de reorientación de la herramienta en la base de coordenadas del objeto actual, desde el punto de inicio hasta el `ToPoint` durante todos los movimientos circulares posteriores.

##### Ejemplo 3

```
CirPathMode \CirPointOri;
```

El modo modificado de reorientación de la herramienta desde el punto de inicio a través de la orientación `CirPoint` programada hasta el `ToPoint` durante todos los movimientos circulares posteriores.

##### Ejemplo 4

```
CirPathMode \Wrist45;
```

Un modo modificado, de forma que la proyección del eje Z de la herramienta sobre el plano de corte seguirá la orden de movimiento circular programada. Sólo se utilizan los ejes de muñeca 4 y 5. Este modo sólo debe usarse al cortar objetos delgados.

##### Ejemplo 5

```
CirPathMode \Wrist46;
```

Un modo modificado, de forma que la proyección del eje Z de la herramienta sobre el plano de corte seguirá la orden de movimiento de círculo programada. Sólo se utilizan los ejes de muñeca 4 y 6. Este modo sólo debe usarse con objetos delgados.

##### Ejemplo 6

```
CirPathMode \Wrist56;
```

Un modo modificado, de forma que la proyección del eje Z de la herramienta sobre el plano de corte seguirá la orden de movimiento de círculo programada. Sólo se

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.42 CirPathMode - Reorientación de la herramienta durante trayectorias circulares

RobotWare - OS

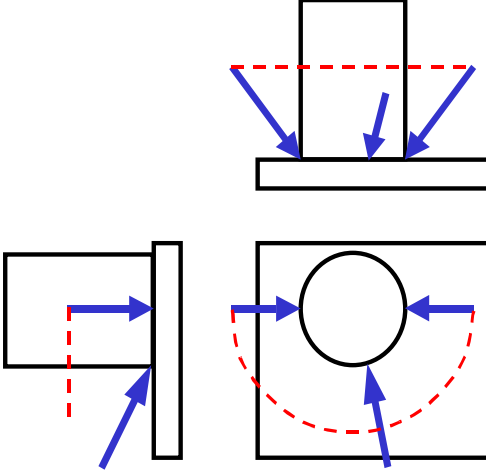
Continuación

utilizan los ejes de muñeca 5 y 6. Este modo sólo debe usarse con objetos delgados.

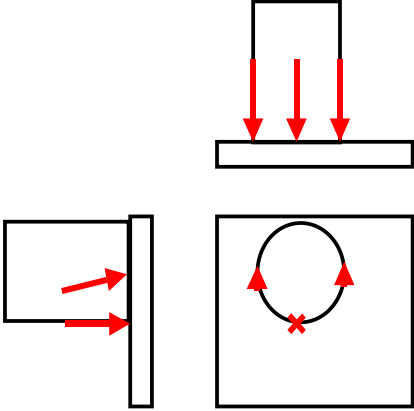
### Descripción

#### PathFrame

La figura de la tabla muestra la reorientación de la herramienta del modo estándar \PathFrame.

| Figura   | Descripción  |
|--|--|
|  <p data-bbox="424 1077 528 1093">xx0500002152</p> | <p data-bbox="927 595 1398 786">Las flechas representan la posición de la herramienta respecto del punto central de la muñeca hasta el punto central de la herramienta, para los puntos programados. La trayectoria del punto central de la muñeca se representa con una línea de puntos en la figura.</p> <p data-bbox="927 790 1398 925">El modo \PathFrame permite conseguir fácilmente el mismo ángulo de la herramienta alrededor del cilindro. La muñeca del robot no pasa a través de la orientación programada en CirPoint</p> |

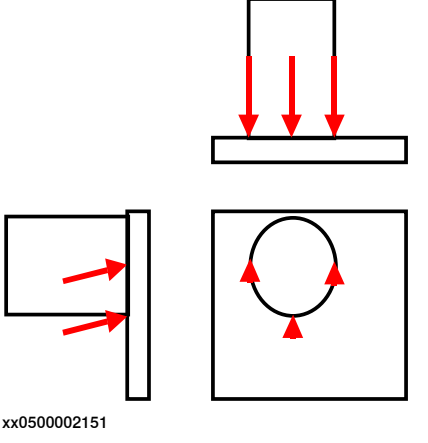
La figura de la tabla muestra el uso del modo estándar \PathFrame con una orientación de herramienta fija.

| Figura  | Descripción  |
|---|--|
|  <p data-bbox="424 1693 528 1709">xx0500002153</p> | <p data-bbox="866 1272 1398 1384">En la figura siguiente se representa la orientación obtenida para la herramienta en la mitad de la trayectoria circular, utilizando una herramienta de programación y el modo \PathFrame.</p> <p data-bbox="866 1388 1398 1444">Compárela con la figura que aparece más abajo para el modo \ObjectFrame.</p> |

Continúa en la página siguiente

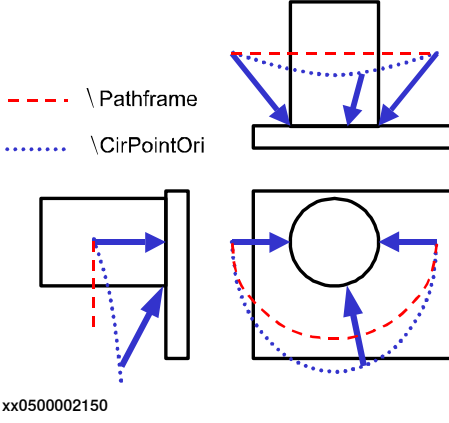
Base de coordenadas del objeto

La figura de la tabla muestra el uso del modo modificado `\ObjectFrame` con una orientación de herramienta fija.

| Figura  | Descripción   |
|---|---|
|  | <p>En esta figura se representa la orientación obtenida para la herramienta en la mitad de la trayectoria circular, utilizando una herramienta de programación y el modo <code>\ObjectFrame</code>.</p> <p>Este modo realizará una reorientación lineal de la herramienta de la misma forma que con <code>MoveL</code>. La muñeca del robot no pasa a través de la orientación programada en <code>CirPoint</code>.</p> <p>Compárela con la figura anterior para el modo <code>\PathFrame</code>.</p> |

CirPointOri

La figura de la tabla representa las diferencias de reorientación de la herramienta existentes entre el modo estándar `\PathFrame` y el modo modificado `\CirPointOri`.

| Figura  | Descripción   |
|---|---|
|  | <p>Las flechas representan la posición de la herramienta respecto del punto central de la muñeca hasta el punto central de la herramienta, para los puntos programados. Las distintas trayectorias del punto central de la muñeca se representan con líneas discontinuas en la figura.</p> <p>El modo <code>\CirPointOri</code> hace que la muñeca del robot atraviese la orientación programada en <code>CirPoint</code>.</p> <p>La trayectoria siempre es la misma en xyz pero la orientación es diferente.</p> |

Continúa en la página siguiente

# 1 Instrucciones

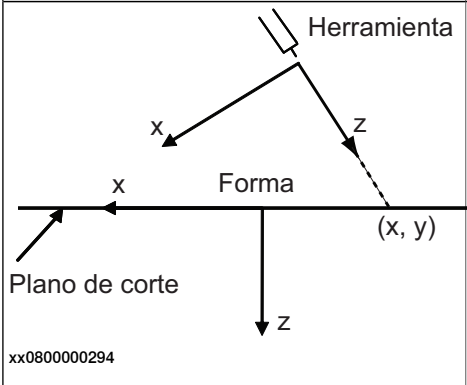
## 1.42 CirPathMode - Reorientación de la herramienta durante trayectorias circulares

RobotWare - OS

Continuación

Wrist45 / Wrist46 / Wrist56

La figura de la tabla muestra los sistemas de coordenadas que intervienen al cortar una figura utilizando los ejes 4 y 5.

| Figura  | Descripción   |
|---|---|
|  | Se supone que el haz de corte está alineado con el eje z de la herramienta. El sistema de coordenadas del plano de corte queda definido por la posición inicial del robot al ejecutar la instrucción <code>MoveC</code> . |

### Argumentos

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

`[\PathFrame]`

Tipo de dato: `switch`

Durante el movimiento circular, la reorientación de la herramienta se realiza de forma continua desde la orientación en el punto de inicio hasta la orientación en `ToPoint`, respecto de la base de coordenadas de la trayectoria actual. Éste es el modo estándar del sistema.



#### Nota

Utilizar el `CirPathMode` sin ningún interruptor, da el mismo resultado que `CirPathMode \PathFrame`.

`[\ObjectFrame]`

Tipo de dato: `switch`

Durante el movimiento circular, la reorientación de la herramienta se realiza de forma continua desde la orientación en el punto de inicio hasta la orientación en `ToPoint`, respecto de la base de coordenadas del objeto actual.

`[\CirPointOri]`

Tipo de dato: `switch`

Durante el movimiento circular, la reorientación de la herramienta se realiza de forma continua desde la orientación en el punto de inicio hasta la orientación programada en `CirPoint` y más adelante hasta la orientación de `ToPoint`.

`[\Wrist45]`

Tipo de dato: `switch`

El robot moverá los ejes 4 y 5 de forma que la proyección del eje Z de la herramienta sobre el plano de corte seguirá la orden de movimiento de círculo programada.

Continúa en la página siguiente

## 1.42 CirPathMode - Reorientación de la herramienta durante trayectorias circulares RobotWare - OS Continuación

Este modo sólo debe usarse con objetos delgados y sólo se utilizan 2 ejes de muñeca, lo que proporciona una mayor exactitud pero a la vez un menor control.



### Nota

Este interruptor requiere la opción *WristMove* de RobotWare.

[\Wrist46]

Tipo de dato: *switch*

El robot moverá los ejes 4 y 6 de forma que la proyección del eje Z de la herramienta sobre el plano de corte seguirá la orden de movimiento de círculo programada. Este modo sólo debe usarse con objetos delgados y sólo se utilizan 2 ejes de muñeca, lo que proporciona una mayor exactitud pero a la vez un menor control.



### Nota

Este interruptor requiere la opción *WristMove* de RobotWare.

[\Wrist56]

Tipo de dato: *switch*

El robot moverá los ejes 5 y 6 de forma que la proyección del eje Z de la herramienta sobre el plano de corte seguirá la orden de movimiento de círculo programada. Este modo sólo debe usarse con objetos delgados y sólo se utilizan 2 ejes de muñeca, lo que proporciona una mayor exactitud pero a la vez un menor control.



### Nota

Este interruptor requiere la opción *WristMove* de RobotWare.

## Ejecución de programas

El modo de reorientación de la herramienta circular especificado se aplica a las siguientes instrucciones de movimiento circular que se ejecuten, sean del tipo que sean (*MoveC*, *SearchC*, *TriggC*, *MoveCDO*, *MoveCSync*, *ArcC*, *PaintC*, etc.), y continúa vigente hasta que se ejecute una nueva instrucción *CirPathMode*.

El modo estándar de reorientación en trayectorias circulares (*CirPathMode* \PathFrame) se establece automáticamente en los casos siguientes:

- cuando se utiliza el modo de reinicio **Restablecer RAPID**
- al cargar un nuevo programa o un nuevo módulo
- al iniciar la ejecución del programa desde el principio
- al mover el puntero del programa a *main*
- al mover el puntero del programa a una rutina
- al mover el puntero de programa de una forma que se pierde el orden de la ejecución.

Continúa en la página siguiente

# 1 Instrucciones

## 1.42 CirPathMode - Reorientación de la herramienta durante trayectorias circulares

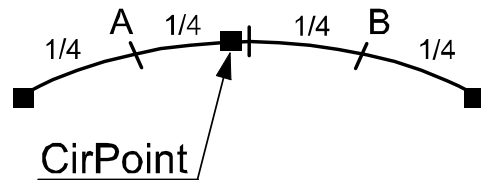
RobotWare - OS

Continuación

### Limitaciones

Esta instrucción sólo afecta a los movimientos circulares.

Cuando se utiliza el modo `\CirPointOri`, el valor de `CirPoint` debe encontrarse entre los puntos A y B que se representan en la figura siguiente, para que el movimiento circular atraviese la orientación programada en `CirPoint`.



xx0500002149

`\Wrist45`, Los modos `\Wrist46` y `\Wrist56` sólo deben usarse para cortar objetos delgados, dado que la posibilidad de controlar el ángulo de la herramienta se pierde al utilizar sólo dos ejes de muñeca. No es posible utilizar movimientos coordinados dado que el eje principal está bloqueado.

Si se trabaja en un área de singularidad de muñeca y se ha ejecutado la instrucción `SingArea\Wrist`, la instrucción `CirPathMode` no tiene ningún efecto porque el sistema selecciona en este caso otro modo de reorientación de la herramienta para los movimientos circulares (la interpolación de ejes).

### Sintaxis

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

### Información relacionada

| Para obtener más información sobre                                    | Consulte   |
|---|--|
| Datos de parámetros de movimiento                                     | <a href="#">motsetdata - Datos de parámetros de movimiento en la página 1705</a>                             |
| Instrucción para movimiento circular                                  | <a href="#">MoveC - Mueve el robot en círculo en la página 389</a>   |
| Definición del método de interpolación alrededor de puntos singulares | <a href="#">SingArea - Define el método de interpolación alrededor de puntos singulares en la página 730</a> |
| Interpolación   | <a href="#">Manual de referencia técnica - RAPID Overview</a>  |
| Movimientos de la muñeca  | <a href="#">Application manual - Controller software IRC5, sección WristMove</a>                             |

## 1.43 Clear - Borra el valor

### Utilización

`Clear` se utiliza para borrar el contenido de una variable numérica o una variable persistente, es decir, asignarle el valor 0.

### Ejemplos básicos

Los ejemplos siguientes ilustran la instrucción `Clear`.

#### Ejemplo 1

```
Clear reg1;
```

Se borra el valor de `Reg1`, con lo que `reg1:=0`.

#### Ejemplo 2

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

```
Clear mydnum;
```

Se borra el valor de `mydnum`, con lo que `mydnum:=0`.

### Argumentos

```
Clear Name | Dname
```

Name

Tipo de dato: num

El nombre de la variable o de la variable persistente cuyo contenido se desea borrar.

Dname

Tipo de dato: dnum

El nombre de la variable o de la variable persistente cuyo contenido se desea borrar.

### Sintaxis

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

```

### Información relacionada

| Para obtener más información sobre                  | Consulte  |
|---|---|
| Incremento de una variable en 1                     | <a href="#">Incr - Aumenta en 1 un valor en la página 262</a> |
| Decremento de una variable en 1                     | <a href="#">Decr - Reduce un número en 1 en la página 189</a> |
| Suma de cualquier valor a una variable              | <a href="#">Add - Suma un valor numérico en la página 32</a>  |
| Cambio de un dato mediante una expresión arbitraria | <a href="#">":=" - Asigna un valor en la página 42</a>        |

# 1 Instrucciones

---

## 1.44 ClearIOBuff - Vacía el búfer de entrada de un canal serie

RobotWare - OS

## 1.44 ClearIOBuff - Vacía el búfer de entrada de un canal serie

---

### Utilización

ClearIOBuff (*Clear I/O Buffer*) se utiliza para vaciar el búfer de entrada de un canal serie. Se desechan todos los caracteres almacenados en el búfer del canal serie de entrada.

---

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción ClearIOBuff:

#### Ejemplo 1

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

Se vacía el búfer de entrada del canal serie al que se hace referencia con channel1. El tiempo de espera garantiza a la operación de borrado un tiempo suficiente para su finalización.

---

### Argumentos

ClearIOBuff IODevice

IODevice

Tipo de dato: iodev

El nombre (la referencia) del canal serie cuyo búfer de entrada se desea vaciar.

---

### Ejecución de programas

Se desechan todos los caracteres almacenados en el búfer del canal serie de entrada. Las instrucciones de lectura siguientes esperarán nuevos datos recibidos a través del canal.

En caso de un reinicio tras una caída de alimentación, todos los archivos o canales serie abiertos del sistema se cierran y el descriptor de E/S de la variable del tipo iodev se restablece.

---

### Limitaciones

Esta instrucción sólo puede usarse con los canales serie. No se espera la confirmación de la operación para finalizar. Deje un tiempo de espera de 0,1 segundos tras la instrucción para proporcionar a la operación un tiempo suficiente en cada aplicación.

---

### Gestión de errores

Pueden generarse los errores recuperables enumerados a continuación. Estos errores pueden ser gestionados en un gestor de errores. El valor de la variable de sistema ERRNO cambia a:

| Nombre      | Causa del error                          |
|-------------|--|
| ERR_FILEACC | La instrucción se utiliza en un archivo. |

Continúa en la página siguiente



## 1.44 ClearIOBuff - Vacía el búfer de entrada de un canal serie

*RobotWare - OS*

*Continuación*

### Sintaxis

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

### Información relacionada

| Para obtener más información sobre | Consulte   |
|------------------------------------|--|
| Apertura de un canal serie         | <i>Manual de referencia técnica - RAPID Overview</i> |
| File and serial channel handling   | <i>Application manual - Controller software IRC5</i> |

# 1 Instrucciones

## 1.45 ClearPath - Elimina la trayectoria actual

Robot Ware - OS

## 1.45 ClearPath - Elimina la trayectoria actual

### Utilización

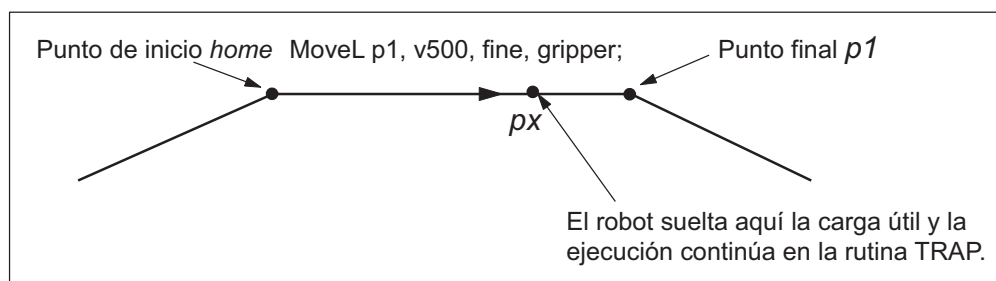
`ClearPath` (*Clear Path*) elimina toda la trayectoria de movimientos del nivel de trayectoria de movimiento actual (nivel base o nivel `StorePath`).

La expresión “trayectoria de movimientos” se refiere a todas las órdenes de movimiento de cualquier instrucción de movimiento que se ejecute en **RAPID** pero que no ha sido realizada aún en el momento de la ejecución de `ClearPath`.

El robot debe encontrarse en un punto de paro o debe ser parado por una instrucción `StopMove` para poder ejecutar la instrucción `ClearPath`.

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción `ClearPath`:



xx0500002154

En el ejemplo de programa siguiente, el robot se mueve desde la posición `home` hasta la posición `p1`. En el punto `px`, la señal `dil` indicará que se ha soltado la carga útil. La ejecución continúa en la rutina `TRAP gohome`. El robot detiene el movimiento (comienza el frenado) en el punto `px`, se borra la trayectoria y el robot se traslada a la posición `home`. El error se eleva hacia la rutina desde la que se hace la llamada, `minicycle`, y todo el ciclo de programa definido por el usuario `procl ... proc2` se ejecuta desde el principio una vez más.

### Ejemplo 1

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

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

PROC procl()
  ...
  proc2;
```

Continúa en la página siguiente

```
...
ENDPROC

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

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

Si se ejecuta el mismo programa pero sin `StopMove` ni `ClearPath` en la rutina `TRAP gohome`, el robot continúa hasta la posición `p1` antes de volver a la posición `home`.

### Limitaciones

A continuación aparecen algunos ejemplos de limitación de la instrucción `ClearPath`.

#### Ejemplo 1 - Limitación

```
VAR intnum int_move_stop;
...
PROC test_move_stop()
  CONNECT int_move_stop WITH trap_move_stop;
  ISignalDI dil, 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
```

Éste es un ejemplo de la limitación de `ClearPath`. Durante el movimiento del robot hacia `p10` y `p20`, el movimiento en curso se detiene y la trayectoria de

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.45 ClearPath - Elimina la trayectoria actual

Robot Ware - OS

Continuación

movimiento se borra, pero no se realiza ninguna acción para interrumpir la instrucción activa MoveJ p10 o MoveL p20 en PROC test\_move\_stop. Por tanto, el movimiento en curso se interrumpirá y el robot pasará a p10 en TRAP trap\_move\_stop, pero no se realizará ningún movimiento adicional hacia p10 o p20 en PROC test\_move\_stop. La ejecución del programa se quedará en suspenso.

Este problema puede resolverse con la recuperación de errores con un salto largo, de la forma descrita en el ejemplo 2 que aparece a continuación, o con un error elevado asíncronamente con la instrucción ProcerrRecovery.

### Ejemplo 2 - Sin limitaciones

```
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 dil, 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 interrupted movement on motion base path level
    StartMove;
    RETRY;
ENDPROC

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

Éste es un ejemplo de cómo usar la recuperación de errores con un salto largo junto con ClearPath y sin limitaciones. Durante el movimiento del robot a p10 y p20, se detiene el movimiento en curso. La trayectoria de movimiento se borra y, debido a la recuperación de errores más allá de los límites de nivel de ejecución, se realiza una interrupción en la instrucción activa MoveJ p10 o MoveL p20. Por tanto, el movimiento en curso se interrumpe y el robot pasa a p10 en ERROR handler, y se ejecuta una vez más la instrucción interrumpida MoveJ p10 o MoveL p20 en PROC test\_move\_stop.

---

### Sintaxis

```
ClearPath ' ; '
```

Continúa en la página siguiente

### Información relacionada

| Para obtener más información sobre | Consulte  |
|------------------------------------|---|
| Paro de los movimientos del robot  | <a href="#">StopMove - Detiene el movimiento del robot en la página 828</a>   |
| Recuperación en caso de error      | <i>Manual de referencia técnica - RAPID Overview</i><br><i>Technical reference manual - RAPID kernel</i>                        |
| Error elevado asincrónamente       | <a href="#">ProcerrRecovery - Genera errores de movimiento de proceso y permite la recuperación tras ellos en la página 543</a> |

# 1 Instrucciones

---

## 1.46 ClearRawBytes - Borra el contenido de un dato de tipo rawbytes

RobotWare - OS

## 1.46 ClearRawBytes - Borra el contenido de un dato de tipo rawbytes

---

### Utilización

`ClearRawBytes` se utiliza para cambiar a 0 todo el contenido de una variable de tipo `rawbytes`.

---

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción `ClearRawBytes`:

#### Ejemplo 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;
```

Se guarda en los primeros 4 bytes el valor de `integer` (a partir del número de índice 1) y, en los 4 bytes siguientes, a partir del número de índice 5, el valor de `float`.

La última ilustración del ejemplo borra el contenido de `raw_data`, a partir del número de índice 5. Es decir, se borra el contenido de `float`, pero el valor de `integer` se conserva en `raw_data`.

La longitud actual de los bytes válidos de `raw_data` cambia a 4.

---

### Argumentos

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

RawData

**Tipo de dato:** `rawbytes`

RawData es el contenedor de datos cuyo contenido se borrará.

[ \FromIndex ]

**Tipo de dato:** `num`

Con `\FromIndex` se especifica a partir de qué punto se empieza a borrar el contenido de RawData. Se borra todo el contenido hasta el final.

Si no se especifica `\FromIndex`, se borran todos los datos que existan a partir del número de índice 1.

---

### Ejecución de programas

Se restablecen a 0 los datos a partir del número de índice 1 (predeterminado) o a partir de `\FromIndex` en la variable especificada.

La longitud actual de los bytes válidos de la variable especificada cambia a 0 (predeterminado) o a `(FromIndex- 1)` si se programa `\FromIndex`.

---

*Continúa en la página siguiente*

## 1.46 ClearRawBytes - Borra el contenido de un dato de tipo rawbytes

RobotWare - OS

Continuación

## Sintaxis

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

## Información relacionada

| Para obtener más información sobre                              | Consulte   |
|---|--|
| rawbytes datos  | <a href="#">rawbytes - Datos sin formato en la página 1736</a>   |
| Obtención de la longitud de un dato rawbytes                    | <a href="#">RawBytesLen - Obtiene la longitud de un dato de tipo rawbytes en la página 1432</a>        |
| Copiado del contenido de un dato de tipo rawbytes               | <a href="#">CopyRawBytes - Copia el contenido de un dato de tipo rawbytes en la página 171</a>         |
| Empaquetamiento de un encabezado de DeviceNet en datos rawbytes | <a href="#">PackDNHeader - Empaqueta un encabezado de DeviceNet en datos rawbytes en la página 496</a> |
| Empaquetamiento de datos en datos rawbytes                      | <a href="#">PackRawBytes - Empaqueta datos en un dato de tipo rawbytes en la página 499</a>            |
| Escritura de un dato rawbytes                                   | <a href="#">WriteRawBytes - Escribe un dato de tipo rawbytes en la página 1111</a>                     |
| Lectura de un dato rawbytes                                     | <a href="#">ReadRawBytes - Lee datos de tipo rawbytes en la página 589</a>                             |
| Desempaquetamiento de datos de un dato rawbytes                 | <a href="#">UnpackRawBytes - Desempaqueta datos de un dato de tipo rawbytes en la página 1024</a>      |
| Gestión de archivos y dispositivos de E/S                       | <a href="#">Application manual - Controller software IRC5</a>  |

# 1 Instrucciones

---

1.47 ClkReset - Pone a cero un reloj utilizado como temporizador  
RobotWare - OS

## 1.47 ClkReset - Pone a cero un reloj utilizado como temporizador

---

### Utilización

ClkReset se utiliza para poner a cero un reloj que funciona como un cronómetro para funciones de temporización.

Esta instrucción puede usarse antes de usar un reloj, para garantizar que está puesto a cero.

---

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción ClkReset:

#### Ejemplo 1

```
ClkReset clock1;
```

Se pone a cero el reloj clock1.

---

### Argumentos

```
ClkReset Clock
```

Reloj

Tipo de dato: clock

El nombre del reloj que se desea poner a cero.

---

### Ejecución de programas

Tras poner a cero un reloj, su valor es 0.

Si el reloj está en marcha, se detendrá y se pondrá a cero.

---

### Sintaxis

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

---

### Información relacionada

| Para obtener más información sobre | Consulte   |
|------------------------------------|--|
| Otras instrucciones de reloj       | <i>Manual de referencia técnica - RAPID Overview</i> |



### 1.48 ClkStart - Pone en marcha un reloj utilizado para la temporización

---

#### Utilización

`ClkStart` se utiliza para poner en marcha un reloj que funciona como un cronómetro para funciones de temporización.

---

#### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción `ClkStart`:

##### Ejemplo 1

```
ClkStart clock1;
```

Se pone en marcha el reloj `clock1`.

---

#### Argumentos

```
ClkStart Clock
```

##### Reloj

Tipo de dato: `clock`

El nombre del reloj que se desea poner en marcha.

---

#### Ejecución de programas

Cuando se pone en marcha el reloj, éste sigue contando segundos hasta que se decide detenerlo.

El reloj sigue en marcha incluso cuando se detiene el programa que lo puso en marcha. Sin embargo, es posible que el evento que se intenta temporizar ya no sea válido. Por ejemplo, si el programa estaba midiendo el tiempo de espera de una entrada, es posible que la entrada se haya recibido mientras el programa estaba parado. En este caso, el programa no podrá ser consciente de que el evento se ha producido mientras la ejecución estuvo parada.

Los relojes siguen funcionando cuando se apaga el robot, siempre y cuando la energía de respaldo de la batería proteja al programa que contiene la variable de reloj.

Mientras está en marcha, cualquier reloj puede ser leído, parado o puesto a cero.

---

#### Más ejemplos

A continuación aparecen más ejemplos de la instrucción `ClkStart`.

##### Ejemplo 1

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

Se mide el tiempo que se espera a que `di1` tenga el valor 1.

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.48 ClkStart - Pone en marcha un reloj utilizado para la temporización

RobotWare - OS

Continuación

---

### Gestión de errores

Pueden generarse los errores recuperables enumerados a continuación. Estos errores pueden ser gestionados en un gestor de errores. El valor de la variable de sistema `ERRNO` cambia a:

| Nombre                    | Causa del error  |
|---------------------------|--|
| <code>ERR_OVERFLOW</code> | El reloj funciona de forma continuada durante 4.294.967 segundos (49 días, 17 horas, 2 minutos y 47 segundos), tras lo cual se desborda. |

---

### Sintaxis

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

---

### Información relacionada

| Para obtener más información sobre | Consulte   |
|------------------------------------|--|
| Otras instrucciones de reloj       | <i>Manual de referencia técnica - RAPID Overview</i> |

## 1.49 ClkStop - Detiene un reloj utilizado para la temporización

### Utilización

`ClkStop` se utiliza para parar un reloj que funciona como un cronómetro para funciones de temporización.

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción `ClkStop`:

```
ClkStop clock1;
```

Se detiene el reloj `clock1`.

### Argumentos

```
ClkStop Clock
```

Reloj

Tipo de dato: `clock`

El nombre del reloj a detener.

### Ejecución de programas

Quando se detiene un reloj, éste deja de funcionar.

Quando un reloj está parado, puede ser leído, puesto en marcha de nuevo o puesto a cero.

### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                    | Causa del error  |
|---------------------------|--|
| <code>ERR_OVERFLOW</code> | El reloj funciona durante 4.294.967 segundos (49 días, 17 horas, 2 minutos y 47 segundos), tras lo cual se desborda. |

### Sintaxis

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

### Información relacionada

| Para obtener más información sobre | Consulte  |
|------------------------------------|---|
| Otras instrucciones de reloj       | <i>Manual de referencia técnica - RAPID Overview</i>  |
| Más ejemplos                       | <a href="#">ClkStart - Pone en marcha un reloj utilizado para la temporización en la página 149</a> |

# 1 Instrucciones

---

1.50 Cerrar - Cierra un archivo o dispositivo de E/S  
*RobotWare - OS*

## 1.50 Cerrar - Cierra un archivo o dispositivo de E/S

---

### Utilización

`Close` se utiliza para cerrar un archivo o un dispositivo de E/S.

---

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción `Close`:

#### Ejemplo 1

```
Close channel2;
```

El dispositivo de E/S al que hace referencia `channel2` se cierra.

---

### Argumentos

```
Close IODevice
```

IODevice

Tipo de dato: `iodev`

El nombre (referencia) del archivo o dispositivo de E/S que debe cerrarse.

---

### Ejecución de programas

El archivo o dispositivo de E/S especificado se cierra y es necesario abrirlo de nuevo antes de leer o escribir en él. Si ya está cerrado, la instrucción no se tiene en cuenta.

---

### Sintaxis

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

---

### Información relacionada

| Para obtener más información sobre        | Consulte   |
|---|--|
| Abrir un archivo o dispositivo de E/S     | <i>Manual de referencia técnica - RAPID Overview</i> |
| Gestión de archivos y dispositivos de E/S | <i>Application manual - Controller software IRC5</i> |

## 1.51 CloseDir - Cierra un directorio

### Utilización

CloseDir se utiliza para cerrar un directorio, como contraposición de OpenDir.

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción CloseDir:

#### Ejemplo 1

```

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

```

Este ejemplo imprime los nombres de todos los archivos o subdirectorios que se encuentran dentro del directorio especificado.

### Argumentos

CloseDir Dev

Dev

**Tipo de dato:** dir

Una variable que hace referencia a un directorio y capturada con la instrucción OpenDir.

### Sintaxis

```

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

```

### Información relacionada

| Para obtener más información sobre        | Consulte  |
|---|---|
| Directorio                                | <a href="#">dir - Estructura de directorio de archivos en la página 1659</a>          |
| Creación de un directorio                 | <a href="#">MakeDir - Crea un nuevo directorio en la página 358</a>                   |
| Apertura de un directorio                 | <a href="#">OpenDir - Abre un directorio en la página 494</a>                         |
| Lectura de un directorio                  | <a href="#">ReadDir - Lee la siguiente entrada de un directorio en la página 1437</a> |
| Eliminación de un directorio              | <a href="#">RemoveDir - Elimina un directorio en la página 598</a>                    |
| Eliminación de un archivo                 | <a href="#">RemoveFile - Elimina un archivo en la página 600</a>                      |
| Cambio del nombre de un archivo           | <a href="#">RenameFile - Cambia el nombre de un archivo en la página 604</a>          |
| Gestión de archivos y dispositivos de E/S | <a href="#">Application manual - Controller software IRC5</a>                         |

# 1 Instrucciones

---

1.52 Comentario - Comentario  
RobotWare - OS

## 1.52 Comentario - Comentario

---

### Utilización

`Comment` sólo se utiliza para facilitar la comprensión del programa. No tiene ningún efecto sobre la ejecución del programa.

---

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción `Comment`:

#### Ejemplo 1

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

Se inserta un comentario en el programa para facilitar su comprensión.

---

### Argumentos

```
! Comment
```

`Comment`

Cadena de texto  
Cualquier texto.

---

### Ejecución de programas

Cuando se ejecuta esta instrucción, no ocurre nada.

---

### Limitaciones

#### Comentarios en un registro

En un registro común, no se permite tener un comentario en una línea separada a menos que sea la última línea.

```
RECORD my_rec
  ! DISALLOWED COMMENT
  num mynum; ! allowed comment (not separate line)
  string mystring;
  ! allowed comment on last line
ENDRECORD
```

---

### Sintaxis

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

---

### Información relacionada

| Para obtener más información sobre                     | Consulte   |
|--|--|
| Caracteres permitidos en los comentarios               | <i>Manual de referencia técnica - RAPID Overview</i> |
| Comentarios dentro de declaraciones de datos y rutinas | <i>Manual de referencia técnica - RAPID Overview</i> |

---

## 1.53 Compact IF (IF compacto) - Si se cumple una condición, entonces... (una instrucción)

RobotWare - OS

### 1.53 Compact IF (IF compacto) - Si se cumple una condición, entonces... (una instrucción)

#### Utilización

`Compact IF` se utiliza cuando sólo es necesario ejecutar una instrucción si se cumple una condición determinada.

Si es necesario ejecutar distintas instrucciones en función de si se cumple o no una condición especificada, se utiliza la instrucción `IF`.

#### Ejemplos básicos

Los ejemplos siguientes ilustran la instrucción `CompactIF`.

##### Ejemplo 1

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

Si `reg1` es mayor que 5, la ejecución del programa continúa a partir de la etiqueta `next`.

##### Ejemplo 2

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

La señal `do1` se activa si `counter > 10`.

#### Argumentos

```
IF Condition ...
```

#### Condition

Tipo de dato: `bool`

La condición que debe cumplirse para que se ejecute la instrucción.

#### Sintaxis

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

#### Información relacionada

| Para obtener más información sobre | Consulte   |
|------------------------------------|--|
| Condiciones (expresiones lógicas)  | <i>Manual de referencia técnica - RAPID Overview</i>                               |
| IF con varias instrucciones        | <i>IF - Si se cumple una condición, ...; de lo contrario, ... en la página 260</i> |

# 1 Instrucciones

---

## 1.54 ConfJ - Controla la configuración durante el movimiento de los ejes RobotWare - OS

### 1.54 ConfJ - Controla la configuración durante el movimiento de los ejes

---

#### Utilización

`ConfJ` (*Configuration Joint*) se utiliza para especificar si es necesario controlar la configuración del robot durante el movimiento de los ejes. Si no se controla, es posible que el robot utilice en ocasiones una configuración distinta de la programada.

Con `ConfJ \Off`, el robot no puede cambiar de configuración de ejes principales, sino que busca una solución con la misma configuración de ejes principales que la actual, moviéndose sin embargo hasta la configuración de muñeca más cercana para los ejes 4 y 6.

Esta instrucción sólo puede usarse en la tarea principal `T_ROB1` o, si se cuenta con un sistema MultiMove, en las tareas de movimiento.



#### Nota

En el caso del IRB5400 y IRB5500, la monitorización del robot está siempre desactivada, independientemente de lo que se especifique en `ConfJ`.

---

#### Ejemplos básicos

Los ejemplos siguientes ilustran la instrucción `ConfJ`.

##### Ejemplo 1

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

El robot se mueve hacia la posición y la orientación programadas. Si es posible alcanzar esta posición de varias formas con distintas configuraciones de ejes, se elige la posición más cercana posible.

##### Ejemplo 2

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

El robot se mueve a la posición, orientación y configuración de eje programadas.

---

#### Argumentos

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

`[\On]`

Tipo de dato: `switch`

El robot se mueve a la posición programada con parámetros de configuración iguales o cercanos a los parámetros de configuración proporcionados en `confdata`. Si se activa un desplazamiento del programa o corrección de la ruta, el riesgo de movimientos grandes aumenta ya que los datos de la configuración programada se basan en la configuración original.

El robot IRB 5400 se moverá hasta la configuración programada del eje o a una configuración de eje cercana a la programada.

*Continúa en la página siguiente*



[ \Off ]

Tipo de dato: `switch`

El robot se mueve hasta la posición programada utilizando la configuración de eje más cercana.

### Ejecución de programas

La configuración se aplica a la siguiente instrucción de movimiento del robot ejecutada y es válida hasta que se ejecute una nueva instrucción `ConfJ`.

Si se elige el argumento `\On` (o no se elige ningún argumento), el robot se mueve hasta la posición programada con parámetros de configuración iguales o cercanos a los parámetros de configuración proporcionados.

Si se activa un desplazamiento del programa o corrección de la ruta, el riesgo de movimientos grandes aumenta ya que los datos de la configuración programada se basan en la configuración original.

Si se elige el argumento `\Off`, el robot se mueve siempre hacia la configuración de ejes más cercana. Esta configuración puede ser distinta de la programada si la configuración se ha especificado incorrectamente de forma manual o si se ha realizado un desplazamiento de programa.

El control de la configuración (`ConfJ \On`) está activo de forma predeterminada. Esto se establece automáticamente

- cuando se utiliza el modo de reinicio **Restablecer RAPID**
- al cargar un nuevo programa o un nuevo módulo
- al iniciar la ejecución del programa desde el principio
- al mover el puntero del programa a `main`
- al mover el puntero del programa a una rutina
- al mover el puntero de programa de una forma que se pierde el orden de la ejecución.

### Sintaxis

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

### Información relacionada

| Para obtener más información sobre                   | Consulte  |
|--|---|
| Manejo de configuraciones diferentes                 | <i>Manual de referencia técnica - RAPID Overview</i>  |
| Configuración del robot durante el movimiento lineal | <a href="#">ConfL - Monitoriza la configuración durante el movimiento lineal en la página 158</a> |
| Datos de parámetros de movimiento                    | <a href="#">motsetdata - Datos de parámetros de movimiento en la página 1705</a>                  |
| Datos de configuración el robot                      | <a href="#">confdata - Datos de configuración del robot en la página 1648</a>                     |

# 1 Instrucciones

---

## 1.55 ConfL - Monitoriza la configuración durante el movimiento lineal RobotWare - OS

### 1.55 ConfL - Monitoriza la configuración durante el movimiento lineal

---

#### Utilización

ConfL (*Configuration Linear*) se utiliza para especificar si es necesario controlar la configuración del robot durante los movimientos lineales o circulares. Si no se controla, la configuración utilizada en el momento de la ejecución puede ser distinta de la configuración programada. Esto también puede dar lugar a movimientos de barrido inesperados en el robot cuando se cambia el modo al movimiento de ejes. Esta instrucción sólo puede usarse en la tarea principal T\_ROB1 o, si se cuenta con un sistema MultiMove, en las tareas de movimiento.



#### Nota

En el caso del IRB5400 y IRB5500, la monitorización del robot está siempre desactivada, independientemente de lo que se especifique en ConfL.

#### Ejemplos básicos

Los ejemplos siguientes ilustran la instrucción ConfL.

##### Ejemplo 1

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

La ejecución del programa se detiene si no es posible alcanzar la configuración programada desde la posición actual.

##### Ejemplo 2

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

El robot se mueve hacia la posición, la orientación y la configuración de ejes de muñeca programadas. Si no es posible, se detiene la ejecución del programa.

##### Ejemplo 3

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

El robot se mueve hacia la posición y la orientación programadas, pero usando la configuración de ejes más cercana posible, que puede ser distinta de la programada.

#### Argumentos

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

[ \On ]

Tipo de dato: switch

Se monitoriza la configuración del robot.

[ \Off ]

Tipo de dato: switch

No se monitoriza la configuración del robot.

*Continúa en la página siguiente*

### Ejecución de programas

La configuración se aplica a la siguiente instrucción de movimiento ejecutada y es válida hasta que se ejecute una nueva instrucción `ConfL`.

Durante un movimiento lineal o circular, el robot se mueve siempre hacia la posición y la orientación programadas que presenten la configuración de ejes más cercana posible. Si se elige el argumento `\On` (o no se utiliza ningún argumento), la ejecución del programa se detiene tan pronto como aparezca el riesgo de que la configuración de la posición programada no pueda alcanzarse desde la posición actual. La forma con la que se decide varía entre los tipos de robots, consulte [confdata - Datos de configuración del robot en la página 1648](#).

Antes de que se inicie un movimiento ordenado, se realiza una verificación para ver si es posible conseguir la configuración programada. Si no es posible, el programa se detiene. Cuando finaliza el movimiento (en una zona o un punto fino), también se verifica si el robot ha alcanzado la configuración programada.

Si se utiliza `SingArea \Wrist`, el robot se mueve siempre hacia la configuración de ejes de muñeca programada.

Si se utiliza el argumento `\Off`, no se realiza ninguna monitorización.

Tras una detención causada por un error de configuración es posible reiniciar el programa de RAPID en el modo manual. Recuerde que en este caso, debido al error indicado, es muy probable que el robot no se mueva hacia la configuración correcta.

Si se utiliza `ConfL \Off` con un movimiento grande, este puede causar paros en ese mismo momento o más tarde en el programa, generando el error 50050 `Position outside reach` o 50080 `Position not compatible`.

La monitorización de la configuración (`ConfL \On`) está activa de forma predeterminada. Esto se establece automáticamente

- cuando se utiliza el modo de reinicio **Restablecer RAPID**
- al cargar un nuevo programa o un nuevo módulo
- al iniciar la ejecución del programa desde el principio
- al mover el puntero del programa a `main`
- al mover el puntero del programa a una rutina
- al mover el puntero de programa de una forma que se pierde el orden de la ejecución.



#### Recomendación

Una regla sencilla para evitar problemas, tanto con `ConfL\On` como con `\Off`, es insertar puntos intermedios para hacer que el movimiento de cada eje sea inferior a 180 grados entre dos puntos.

Continúa en la página siguiente

# 1 Instrucciones

## 1.55 ConfL - Monitoriza la configuración durante el movimiento lineal

RobotWare - OS

Continuación



### Recomendación

En un programa con `ConfL \Off` se recomienda tener puntos de partida con puntos de configuración conocidos con “`ConfJ \On` y `MoveJ`” o “`ConfL \On` y `SingArea \Wrist` y `MoveL`” antes de movimientos en diferentes partes del programa.

### Sintaxis

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

### Información relacionada

| Para obtener más información sobre                                    | Consulte   |
|---|--|
| Manejo de configuraciones diferentes                                  | <a href="#">Manual de referencia técnica - RAPID Overview</a>  |
| Configuración del robot durante el movimiento de ejes                 | <a href="#">ConfJ - Controla la configuración durante el movimiento de los ejes en la página 156</a>         |
| Definición del método de interpolación alrededor de puntos singulares | <a href="#">SingArea - Define el método de interpolación alrededor de puntos singulares en la página 730</a> |
| Datos de parámetros de movimiento                                     | <a href="#">motsetdata - Datos de parámetros de movimiento en la página 1705</a>                             |
| Datos de configuración el robot                                       | <a href="#">confdata - Datos de configuración del robot en la página 1648</a>                                |

## 1.56 CONNECT - Conecta una interrupción a una rutina TRAP

### Utilización

`CONNECT` se utiliza para determinar la identidad de una interrupción y conectarla a una rutina TRAP.

La interrupción se define mediante la petición de un evento de interrupción y la especificación de su identidad. Por tanto, cuando se produce un evento, la rutina TRAP se ejecuta automáticamente.

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción `CONNECT`:

#### Ejemplo 1

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

Se crea una identidad de interrupción `feeder_low` que se conecta a la rutina TRAP `feeder_empty`. Se producirá una interrupción en el momento en el que la entrada `di1` pasa al nivel alto. En otras palabras, cuando la señal pasa al modo activo, se ejecuta la rutina TRAP `feeder_empty`.

### Argumentos

```
CONNECT Interrupt WITH Trap routine
```

#### Interrupt

Tipo de dato: `intnum`

La variable a la que se desea asignar la identidad de la interrupción. La declaración NO DEBE hacerse dentro de una rutina (como dato de la rutina).

#### Trap routine

*Identifier*

El nombre de la rutina TRAP.

### Ejecución de programas

Se asigna a la variable una identidad de interrupción que se usará a partir de ese momento para pedir o desactivar interrupciones. Esta identidad también se conecta a la rutina TRAP especificada.



#### Nota

Todas las interrupciones de una tarea se cancelan cuando el puntero de programa se sitúa en Main para esa tarea y deben ser reconectadas. Las interrupciones no se ven afectadas por caídas de alimentación ni un Reinicio.

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.56 CONNECT - Conecta una interrupción a una rutina TRAP

RobotWare - OS

Continuación

### Limitaciones

No es posible conectar una interrupción (una identidad de interrupción) a más de una rutina TRAP a la vez. Sin embargo, es posible conectar varias interrupciones a una misma rutina TRAP.

Cuando se conecta una interrupción a una rutina TRAP, no es posible conectarla de nuevo ni transferirla a otra rutina. Es necesario eliminarla en primer lugar mediante la instrucción `IDelete`.

Las interrupciones que lleguen o que no hayan sido gestionadas antes de la detención de la ejecución del programa no se procesan. Las interrupciones no se tienen en cuenta al detener el programa. Las interrupciones definidas como seguras no serán descuidadas al realizar el paro de programa. Se gestionarán al iniciar de nuevo el programa.

### Gestión de errores

Pueden generarse los errores recuperables enumerados a continuación. Estos errores pueden ser gestionados en un gestor de errores. El valor de la variable de sistema `ERRNO` cambia a:

| Nombre                     | Causa del error  |
|----------------------------|--|
| <code>ERR_ALRDYCNT</code>  | La variable de interrupción ya está vinculada a una rutina TRAP. |
| <code>ERR_CNTNOTVAR</code> | La variable de interrupción no es una referencia a una variable. |
| <code>ERR_INOMAX</code>    | No hay más números de interrupción disponibles.                  |

### Sintaxis

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

### Información relacionada

| Para obtener más información sobre                 | Consulte   |
|--|--|
| Resumen de interrupciones                          | <i>Manual de referencia técnica - RAPID Overview</i>                 |
| Más información sobre la gestión de interrupciones | <i>Manual de referencia técnica - RAPID Overview</i>                 |
| Tipos de datos de las interrupciones               | <a href="#">intnum - Identidad de interrupción en la página 1687</a> |
| Cancelación de una interrupción                    | <a href="#">IDelete - Cancela una interrupción en la página 254</a>  |

## 1.57 ContactL - Movimiento de contacto lineal

### Utilización

ContactL (*Contact Linear*) se utiliza para que el robot YuMi obtenga contacto con un objeto en una posición deseada mientras se mueve linealmente el punto central de la herramienta (TCP).

El nivel de detección de colisiones aumenta hasta su valor máximo y durante el movimiento el robot supervisa el par interno y lo compara con un nivel de par indicado por el usuario. Cuando se alcanza el nivel de par solicitado por el usuario, el robot realiza un paro rígido y continúa con el resto del programa.

Esta instrucción puede usarse típicamente cuando la herramienta sostenida por el robot debe presionar un objeto para introducirlo en su posición.

Esta instrucción solo puede usarse en la tarea principal T\_ROB1 o en las tareas de movimiento en los sistemas *MultiMove*.

La velocidad máxima para una instrucción ContactL es 1000 mm/s.

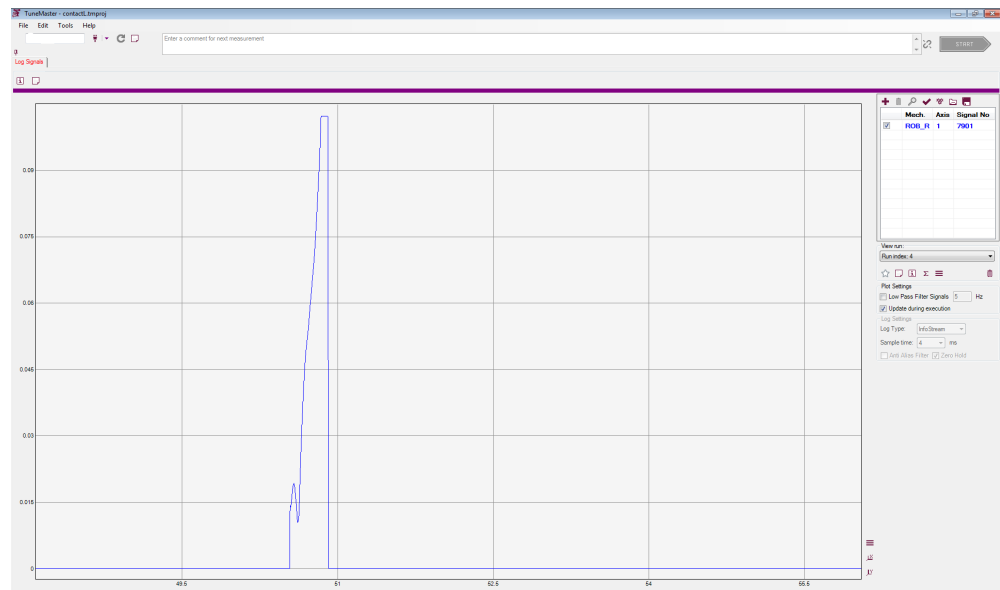


#### Nota

Durante la programación, se recomienda probar primero con baja velocidad, <100 mm/s, y a continuación aumentar gradualmente la velocidad hasta el valor deseado.

### Descripción

Para determinar el valor del nivel de par `desiredTorque` es necesario probar la aplicación y visualizar una señal de prueba interna, la señal 7901, mediante TuneMaster.



xx150000649

Para obtener más información acerca de TuneMaster, consulte la sección de ayuda incluida en la aplicación.

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.57 ContactL - Movimiento de contacto lineal

YuMi

Continuación

---

### Ejemplos básicos

Los siguientes ejemplos ilustran la función `ContactL`:

Consulte también [Más ejemplos en la página 167](#).

#### Ejemplo 1

```
desiredTorque := 0.1;
ContactL \DesiredTorque:=desiredTorque, p10, v100, tool1;
```

El TCP de `tool1` se mueve linealmente hacia la posición `p10` a una velocidad de `v100`. Cuando el valor del nivel de par interno rebasa el nivel `desiredTorque` especificado por el usuario, el robot realiza un paro rígido y, a continuación, el programa continúa en la posición en la que se paró el robot.

El argumento `DesiredTorque` es opcional. Si se omite `DesiredTorque`, la instrucción `ContactL` solo eleva el nivel de detección de colisiones a su valor máximo, lo que supone que da la oportunidad de mantener la presión en un objeto mientras se mueve el TCP.

Si no se alcanza el `desiredTorque` cuando el robot alcanza la posición deseada, se producirá un error de ejecución y el sistema se para con una entrada en el registro de eventos. Por tanto, se recomienda implementar un gestor de errores para estos casos; consulte [Gestión de errores en la página 166](#).

#### Ejemplo 2

```
ContactL RelTool (CRobT(),5,5,0), v100, \Zone:=z10, tool1;
```

El robot se mueve hasta una posición situada a 5 mm de su posición actual en la dirección X y a 5 mm de su posición actual en la dirección Y de la herramienta. Si se omite el argumento `Zone`, la instrucción `ContactL` utiliza un punto fino de forma predeterminada.

En el ejemplo se omite el argumento `DesiredTorque`. La instrucción solo eleva el nivel de detección de colisiones hasta su valor máximo y la instrucción `ContactL` funcionará de forma similar a una instrucción `MoveL`.

#### Ejemplo 3

```
desiredTorque := 0.9;
ContactL \DesiredTorque:=desiredTorque, p10, v100, tool1;
ContactL RelTool (CRobT(),5,5,0), v100, \Zone:=z10, tool1;
ContactL RelTool (CRobT(),5,5,-10), v100, \Zone:=z10, tool1;
MoveL ...
```

Es importante recordar que se requiere la instrucción `ContactL` mientras existe contacto, pero también al eliminar el contacto. Una instrucción de movimiento normal disparará muy probablemente la supervisión del movimiento.

---

### Argumentos

```
ContactL [\DesiredTorque] ToPoint [\ID] Speed [\Zone] Tool [\WObj]
```

```
[ \DesiredTorque ]
```

Tipo de dato: num

Nivel de par deseado definido por el usuario.

*Continúa en la página siguiente*



ContactL utiliza siempre un punto fino como dato de zona para el destino si se define `DesiredTorque`. Si se omite `DesiredTorque`, la instrucción ContactL solo eleva el nivel de detección de colisiones y no supervisa el nivel de par interno.

ToPoint

Tipo de dato: `robtarget`

El punto de destino de los ejes del robot y de los ejes externos. Se define como una posición con nombre o se almacena directamente en la instrucción (marcada con un asterisco \* en la instrucción).

[ \ID ]

*Synchronization id*

Tipo de dato: `identno`

El argumento [ \ID ] es obligatorio en los sistemas MultiMove, si el movimiento es sincronizado o sincronizado coordinado. Este argumento no está permitido en ningún otro caso. El número de ID especificado debe ser el mismo en todas las tareas de programa que cooperan entre sí. Al usar el número de ID los movimientos no se mezclan en tiempo de ejecución.

Speed

Tipo de dato: `speeddata`

Los datos de velocidad que se aplican a los movimientos. Los datos de velocidad definen la velocidad del punto central de la herramienta, la reorientación de la misma y los ejes externos.

[ \Zone ]

Tipo de dato: `zonedata`

Los datos de zona del movimiento. Los datos de zona describen el tamaño de la trayectoria de esquina generada y solo se usan si se omite `DesiredTorque`.

Si se omite el argumento [ \Zone ] , la instrucción ContactL utilizará un punto fino de forma predeterminada.

Tool

Tipo de dato: `tooldata`

La herramienta en uso durante el movimiento del robot. El punto central de la herramienta es el punto que se mueve hacia la posición de destino especificada.

[ \Wobj ]

*Work Object*

Tipo de dato: `wobjdata`

El objeto de trabajo (sistema de coordenadas) con el que está relacionada la posición de robot indicada en la instrucción.

Es posible omitir este argumento. Si se omite, la posición depende del sistema de coordenadas mundo. Si por otro lado se usa un TCP estacionario o ejes externos coordinados, es necesario especificar este argumento para ejecutar un movimiento lineal respecto del objeto de trabajo.

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.57 ContactL - Movimiento de contacto lineal

YuMi

Continuación

---

### Ejecución de programas

Consulte la instrucción `MoveL` para obtener más información acerca del movimiento lineal.

El movimiento del robot se detiene cuando el nivel de par interno rebasa el nivel de par definido por el usuario, siempre y cuando se haya definido el argumento `DesiredTorque`. De lo contrario, el movimiento del robot siempre continúa hacia el punto de destino programado.

Si se omite el argumento `DesiredTorque`, el nivel de detección de colisiones se eleva a su valor máximo y no se realiza ninguna supervisión del nivel de par interno, lo que supone que da la oportunidad de mantener la presión en un objeto mientras se mueve el TCP.

---

### Gestión de errores

Durante una operación `ContactL`, se generan errores en los siguientes casos:

- `ContactL` alcanza el punto especificado en el argumento `ToPoint` sin alcanzar el `DesiredTorque` especificado por el usuario. Con ello se genera el error `ERR_CONTACTL`.

Los errores pueden gestionarse de formas distintas en función del modo de ejecución seleccionado:

- **Ejecución continua hacia adelante/Instrucciones hacia adelante:**  
No se devuelve ninguna posición y el movimiento continúa siempre hacia el punto de destino programado. La variable de sistema `ERRNO` cambia a `ERR_CONTACTL` y es posible gestionar el error en el gestor de errores de la rutina.
- **Instrucción hacia atrás:**  
Durante la ejecución hacia atrás, la instrucción realiza el movimiento pero no realiza la supervisión de par.

### Ejemplo

```
VAR num desiredTorque;
...
desiredTorque := 0.1;
MoveL p10, v100, fine, tool1;
ContactL \DesiredTorque:=desiredTorque, p20, v100, tool1;
...
ERROR
  IF ERRNO=ERR_CONTACTL THEN
    StorePath;
    MoveL p10, v100, fine, tool1;
    RestoPath;
    ClearPath;
    StartMove;
    RETRY;
  ELSE
    Stop;
  ENDIF
ENDPROC
```

*Continúa en la página siguiente*

---

El robot se mueve de la posición `p10` a la `p20`. Si el robot alcanza `p20` sin llegar al `DesiredTorque` especificado por el usuario, el robot vuelve a la posición `p10` y lo intenta una vez más.

### Más ejemplos

A continuación aparecen más ejemplos de la instrucción `ContactL`.

#### Ejemplo 1

```
ContactL p10, v100, \Zone:=z10, tool1;
```

El TCP de `tool1` se mueve linealmente hacia la posición `p10` a una velocidad de `v100` y un tamaño de zona de 10 mm.

Dado que se omite el argumento `DesiredTorque`, la instrucción `ContactL` solo eleva el nivel de detección de colisiones hasta su valor máximo y no supervisa el nivel de par interno.

### Limitaciones

La instrucción `ContactL` solo puede usarse con los robots YuMi.

### Sintaxis

```
ContactL
  ['\' DesiredTorque ',']
  [ToPoint ':='] <expression (IN) of robtarg>
  ['\' ID ':='] <expression (IN) of identno>'],'
  [Speed ':='] <expression (IN) of speeddata>
  ['\' Zone ':='] <expression (IN) of zonedata>'],'
  [Tool ':='] <persistent (PERS) of tooldata>
  ['\' WObj ':='] <persistent (PERS) of wobjdata>]';'
```

### Información relacionada

| Para obtener más información sobre                          | Consulte   |
|---|--|
| Utilización del IRB 14000                                   | <a href="#">Manual del operador - IRB 14000</a>  |
| Escritura en una entrada de corrección                      | <a href="#">CorrWrite - Escribe en un generador de correcciones en la página 183</a>     |
| Movimiento lineal del robot                                 | <a href="#">MoveL - Mueve el robot siguiendo una trayectoria lineal en la página 447</a> |
| Definición de carga   | <a href="#">loaddata - Datos de carga en la página 1694</a>                              |
| Definición de la carga útil de un robot                     | <a href="#">GripLoad - Define la carga útil de un robot en la página 244</a>             |
| Utilización de gestores de errores                          | <a href="#">Manual de referencia técnica - RAPID Overview</a>                            |
| Movimiento lineal   |  |
| Movimiento en general                                       |  |
| LoadIdentify, rutina de servicio de identificación de carga | <a href="#">Manual del operador - IRC5 con FlexPendant</a>                               |

Continúa en la página siguiente

# 1 Instrucciones

---

## 1.57 ContactL - Movimiento de contacto lineal

YuMi

Continuación

| Para obtener más información sobre   | Consulte   |
|--|--|
| Señal de entrada de sistema <i>Sim-Mode</i> para mover el robot en el modo simulado sin carga útil.  | <i>Manual de referencia técnica - Parámetros del sistema</i> |
| Parámetro de sistema <i>ModalPayloadMode</i> para la activación y la desactivación de la carga útil. |  |

## 1.58 CopyFile - Copia un archivo

### Utilización

CopyFile se utiliza para hacer una copia de un archivo existente.

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción CopyFile:

#### Ejemplo 1

```
CopyFile "HOME:/miarchivo", "HOME:/suarchivo";
```

El archivo myfile se copia a yourfile. En este momento, los dos archivos son idénticos.

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

El archivo myfile se copia a yourfile en el directorio mydir.

### Argumentos

```
CopyFile OldPath NewPath
```

OldPath

Tipo de dato: string

La ruta completa del archivo del que se desea copiar.

NewPath

Tipo de dato: string

La ruta completa a la que se desea copiar el archivo.

### Ejecución de programas

El archivo especificado en OldPath se copia al archivo especificado en NewPath.

### Gestión de errores

Pueden generarse los errores recuperables enumerados a continuación. Estos errores pueden ser gestionados en un gestor de errores. El valor de la variable de sistema ERRNO cambia a:

| Nombre        | Causa del error                               |
|---------------|---|
| ERR_FILEACC   | El archivo especificado en OldPath no existe. |
| ERR_FILEEXIST | El archivo especificado en NewPath ya existe. |

### Sintaxis

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

### Información relacionada

| Para obtener más información sobre | Consulte  |
|------------------------------------|---|
| Creación de un directorio          | <a href="#">MakeDir - Crea un nuevo directorio en la página 358</a> |

Continúa en la página siguiente

# 1 Instrucciones

---

## 1.58 CopyFile - Copia un archivo

RobotWare - OS

Continuación

| Para obtener más información sobre              | Consulte   |
|---|--|
| Eliminación de un directorio                    | <a href="#">RemoveDir - Elimina un directorio en la página 598</a>                     |
| Cambio del nombre de un archivo                 | <a href="#">RenameFile - Cambia el nombre de un archivo en la página 604</a>           |
| Eliminación de un archivo                       | <a href="#">RemoveFile - Elimina un archivo en la página 600</a>                       |
| Comprobación del tipo del archivo               | <a href="#">IsFile - Comprobar el tipo de un archivo en la página 1351</a>             |
| Comprobación del tamaño del archivo             | <a href="#">FileSize - Obtiene el tamaño de un archivo en la página 1285</a>           |
| Comprobación del tamaño del sistema de archivos | <a href="#">FSSize - Obtiene el tamaño de un sistema de archivos en la página 1291</a> |
| Gestión de archivos y dispositivos de E/S       | <a href="#">Application manual - Controller software IRC5</a>                          |

## 1.59 CopyRawBytes - Copia el contenido de un dato de tipo rawbytes

### Utilización

CopyRawBytes se utiliza para copiar todo el contenido o una parte del mismo de una variable de tipo rawbytes a otra.

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción CopyRawBytes:

#### Ejemplo 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);
```

En este ejemplo, la variable from\_raw\_data de tipo rawbytes se borra en primer lugar, es decir, se cambian a 0 todos sus bytes. A continuación, se almacena en los 4 primeros bytes el valor de integer y en los siguientes 4 bytes el valor de float.

Después de guardar un dato en from\_raw\_data, el contenido (8 bytes) se copia a to\_raw\_data, empezando por la posición 3.

### Argumentos

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

FromRawData

**Tipo de dato:** rawbytes

FromRawData es el contenedor de datos del cual se copiarán los datos de tipo rawbytes.

FromIndex

**Tipo de dato:** num

FromIndex es la posición de FromRawData a partir de la cual comienza el copiado de los datos. El primer número de índice es 1.

ToRawData

**Tipo de dato:** rawbytes

ToRawData es el contenedor de datos al cual se copiarán los datos de tipo rawbytes.

ToIndex

**Tipo de dato:** num

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.59 CopyRawBytes - Copia el contenido de un dato de tipo rawbytes

RobotWare - OS

Continuación

ToIndex es la posición de ToRawData a partir de la cual se guardarán los datos copiados. Se copia todo el contenido hasta el final. El primer número de índice es 1.

[ \NoOfBytes ]

Tipo de dato: num

El valor especificado con \NoOfBytes es el número de bytes que se desea copiar de FromRawData a ToRawData.

Si no se especifica \NoOfBytes, se copian todos los bytes que van desde FromIndex hasta el final de la longitud actual de bytes válidos de FromRawData.

### Ejecución de programas

Durante la ejecución del programa, se copian los datos de una variable rawbytes a otra.

La longitud actual de los bytes válidos de la variable ToRawData cambia a:

- (ToIndex + número de bytes copiados - 1)
- La longitud actual de los bytes válidos de la variable ToRawData no cambia si la totalidad de la operación de copia se realiza dentro de la longitud anterior de bytes válidos de la variable ToRawData.

### Limitaciones

CopyRawBytes no puede usarse para copiar parte de los datos de una variable de tipo rawbytes a otra parte de la misma variable rawbytes.

### Sintaxis

```
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> ]';'
```

### Información relacionada

| Para obtener más información sobre                              | Consulte   |
|---|--|
| rawbytes datos  | <a href="#">rawbytes - Datos sin formato en la página 1736</a>   |
| Obtención de la longitud de un dato rawbytes                    | <a href="#">RawBytesLen - Obtiene la longitud de un dato de tipo rawbytes en la página 1432</a>        |
| Borrado del contenido de un dato de tipo rawbytes               | <a href="#">ClearRawBytes - Borra el contenido de un dato de tipo rawbytes en la página 146</a>        |
| Empaquetamiento de un encabezado de DeviceNet en datos rawbytes | <a href="#">PackDNHeader - Empaqueta un encabezado de DeviceNet en datos rawbytes en la página 496</a> |
| Empaquetamiento de datos en datos rawbytes                      | <a href="#">PackRawBytes - Empaqueta datos en un dato de tipo rawbytes en la página 499</a>            |
| Escritura de un dato rawbytes                                   | <a href="#">WriteRawBytes - Escribe un dato de tipo rawbytes en la página 1111</a>                     |

Continúa en la página siguiente



## 1.59 CopyRawBytes - Copia el contenido de un dato de tipo rawbytes

RobotWare - OS

Continuación

| Para obtener más información sobre                           | Consulte  |
|--|---|
| Lectura de un dato <code>rawbytes</code>                     | <a href="#">ReadRawBytes - Lee datos de tipo rawbytes en la página 589</a>                        |
| Desempaquetamiento de datos de un dato <code>rawbytes</code> | <a href="#">UnpackRawBytes - Desempaqueta datos de un dato de tipo rawbytes en la página 1024</a> |
| Gestión de archivos y dispositivos de E/S                    | <i>Application manual - Controller software IRC5</i>  |

# 1 Instrucciones

---

## 1.60 CornerPathWarning - Mostrar u ocultar avisos de trayectoria de esquina *RobotWare - OS*

### 1.60 CornerPathWarning - Mostrar u ocultar avisos de trayectoria de esquina

---

#### Utilización

`CornerPathWarning` se utiliza para activar/desactivar avisos de fallo de trayectoria de esquina (registro de evento 50024) todas las instrucciones de movimiento posteriores.

Esta instrucción sólo puede usarse en la tarea principal `T_ROB1` o, si se cuenta con un sistema `MultiMove`, en las tareas de movimiento.

---

#### Ejemplos básicos

Los siguientes ejemplos ilustran la instrucción `CornerPathWarning`.

##### Ejemplo 1

```
CornerPathWarning TRUE;
```

Activar los avisos de trayectoria de esquina.

##### Ejemplo 2

```
PROC main()  
  ! Deactivate corner path warning on all  
  ! subsequent movement instructions  
  CornerPathWarning FALSE;  
  ...  
  ! Check if warning is suppressed  
  IF C_MOTSET.corner_path_warn_suppress=TRUE THEN  
    CornerPathWarning TRUE;  
  ENDIF  
  MyProcess;
```

Desactivar los avisos de trayectoria de esquina en el arranque del programa. Posteriormente, comprobar si el aviso de trayectoria de esquina se ha eliminado. En caso afirmativo, activar el aviso de trayectoria de esquina antes de llamar a `MyProcess`.

---

#### Argumentos

```
CornerPathWarning Active
```

Active

Tipo de dato: `bool`

Especifica si los avisos de trayectoria de esquina deben estar activados.

---

#### Ejecución de programas

El ajuste se aplica a la siguiente instrucción de movimiento ejecutada, tanto en el robot como en los ejes externos hasta que se ejecute una nueva instrucción `CornerPathWarning`.

Se produce un fallo de trayectoria de esquina cuando el robot ejecuta una instrucción de movimiento de zona de esquina y la ejecución del programa `RAPID` no proporciona una nueva instrucción de movimiento a tiempo. Esto fuerza al sistema a convertir el punto de paso programado en un punto fino.

---

*Continúa en la página siguiente*

## 1.60 CornerPathWarning - Mostrar u ocultar avisos de trayectoria de esquina RobotWare - OS Continuación

Si se establece como verdadero, todas las instrucciones de movimiento exitosas con un fallo de trayectoria de esquina mostrarán un aviso en el registro de eventos.

Si cambian a falso, los fallos de trayectoria de esquina se siguen ejecutando como puntos finos, pero el aviso no se mostrará en el registro de eventos.

Para conseguir el mismo comportamiento que en versiones anteriores de Robotware, en las que se podía eliminar el aviso en la configuración, se recomienda poner `CornerPathWarning FALSE;` en la rutina del evento que se ejecuta al inicio del programa (evento `START`).

El valor predeterminado (informe de error de trayectoria de esquina) se establece de forma automática

- cuando se utiliza el modo de reinicio **Restablecer RAPID**
- al cargar un nuevo programa o un nuevo módulo
- al iniciar la ejecución del programa desde el principio
- al mover el puntero del programa a `main`
- al mover el puntero del programa a una rutina
- al mover el puntero de programa de una forma que se pierde el orden de la ejecución.



### Nota

Se recomienda tener activado el aviso de trayectoria de esquina. Apague el aviso de trayectoria de esquina solo cuando suceda un aviso. Estas situaciones incluyen esperar una entrada (de E/S, cámaras, sensores u otros equipos externos). Un aviso de trayectoria de esquina puede conllevar un mal rendimiento del proceso y peores tiempos de ciclo. Deben analizarse todas estas situaciones antes de eliminar el aviso de trayectoria de esquina.

### Sintaxis

```
CornerPathWarning
[Active ':='] <expression (IN) of bool>';'
```

### Información relacionada

| Para obtener más información sobre       | Consulte  |
|--|---|
| Instrucción de movimiento                | <a href="#">MoveL - Mueve el robot siguiendo una trayectoria lineal en la página 447</a>                        |
| Comportamiento de trayectoria de esquina | Sección <i>Interpolación de trayectorias de esquina</i> en <i>Manual de referencia técnica - RAPID Overview</i> |
| Parámetros de movimiento                 | <a href="#">motsetdata - Datos de parámetros de movimiento en la página 1705</a>                                |
| Definición de rutinas de evento          | <i>Manual de referencia técnica - Parámetros del sistema</i>  |

# 1 Instrucciones

## 1.61 CorrClear - Elimina todos los generadores de correcciones

*Path Offset*

## 1.61 CorrClear - Elimina todos los generadores de correcciones

### Descripciones

`CorrClear` se utiliza para eliminar todos los generadores de correcciones conectados. La instrucción puede usarse para eliminar todos los offsets proporcionados anteriormente por todos los generadores de correcciones.

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción `CorrClear`:

#### Ejemplo 1

```
CorrClear ;
```

Esta instrucción elimina todos los generadores de corrección conectados.



#### Nota

Una forma sencilla de garantizar que todos los generadores de correcciones (con sus correcciones) se eliminen al iniciarse el programa, es ejecutar `CorrClear` en una rutina de evento `START`.

Consulte *Manual de referencia técnica - Parámetros del sistema*, tema *Controller*.

### Sintaxis

```
CorrClear ';' ;
```

### Información relacionada

| Para obtener más información sobre          | Consulte   |
|---|--|
| Conexión con un generador de correcciones   | <a href="#">CorrCon - Establece una conexión con un generador de correcciones en la página 177</a> |
| Desconexión de un generador de correcciones | <a href="#">CorrDiscon - Cierra la conexión con un generador de correcciones en la página 182</a>  |
| Escritura en un generador de correcciones   | <a href="#">CorrWrite - Escribe en un generador de correcciones en la página 183</a>               |
| Lectura de los offsets totales actuales     | <a href="#">CorrRead - Lee los offsets totales actuales en la página 1235</a>                      |
| Descriptor de corrección                    | <a href="#">corrdescr - Descriptor de generador de correcciones en la página 1655</a>              |

## 1.62 CorrCon - Establece una conexión con un generador de correcciones

### Utilización

CorrCon se utiliza para conectarse a un generador de correcciones.

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción CorrCon:  
Consulte también [Más ejemplos en la página 177](#).

#### Ejemplo 1

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

La variable `id` se conecta a una referencia del generador de correcciones.

### Argumentos

CorrCon Descr

Descr

Tipo de dato: corrdescr

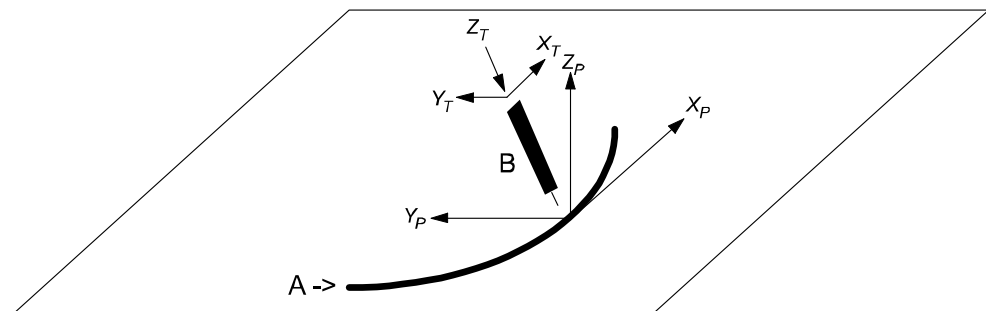
Descriptor del generador de correcciones.

### Más ejemplos

A continuación aparecen más ejemplos de la instrucción CorrCon.

#### Sistema de coordenadas de la trayectoria

Todas las correcciones de la trayectoria (offsets aplicados a la trayectoria) se añaden en el sistema de coordenadas de la trayectoria. El sistema de coordenadas de la trayectoria se define de la forma mostrada a continuación:



xx0500002156

|   |  |
|---|--|
| A | Dirección de la trayectoria              |
| B | Herramienta                              |
| P | Sistema de coordenadas de la trayectoria |
| T | Sistema de coordenadas de la herramienta |

- El eje de coordenadas X de la trayectoria se indica como la tangente de la trayectoria.

Continúa en la página siguiente

# 1 Instrucciones

## 1.62 CorrCon - Establece una conexión con un generador de correcciones

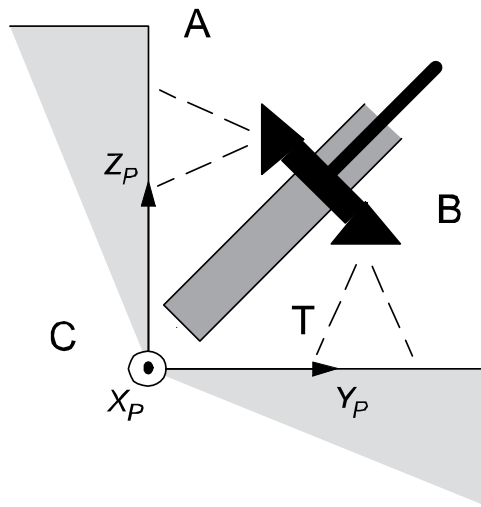
### Path Offset

#### Continuación

- El eje de coordenadas Y se deriva como el producto cruzado del eje de coordenadas X de la trayectoria y el eje de coordenadas Z de la herramienta.
- El eje de coordenadas Z se deriva como el producto cruzado del eje de coordenadas Z de la trayectoria y el eje de coordenadas Y de la trayectoria.

#### Ejemplo de aplicación

Un ejemplo de una aplicación en la que se utilizan correcciones de trayectoria es un robot que sostiene una herramienta con dos sensores montados en él para detectar las distancias vertical y horizontal respecto de un objeto de trabajo. En la figura siguiente se muestra un dispositivo de corrección de trayectoria.



xx0500002155

|   |  |
|---|--|
| A | Sensor para corrección horizontal        |
| B | Sensor para corrección vertical          |
| C | Sistema de coordenadas de la trayectoria |
| T | Herramienta                              |

#### Ejemplo de programa



#### Nota

hori\_sig y vert\_sig son señales analógicas definidas en los parámetros del sistema.

```
CONST num TARGET_DIST := 5;  
CONST num SCALE_FACTOR := 0.5;  
VAR intnum intnol;  
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.
```

Continúa en la página siguiente

## 1.62 CorrCon - Establece una conexión con un generador de correcciones

*Path Offset*

*Continuación*

```
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 intn01 WITH ReadSensors;
ITimer\Single, 0.2, intn01;

! 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 intn01;
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;
```

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.62 CorrCon - Establece una conexión con un generador de correcciones

### Path Offset

#### Continuación

```
CorrWrite vert_id, write_offset;  
ENDIF  
!Setup interrupt again  
IDelete intnol;  
CONNECT intnol WITH ReadSensors;  
ITimer\single, 0.2, intnol;  
ENDTRAP
```

#### Explicación del programa

Se establece la conexión a dos generadores de correcciones con la instrucción `CorrCon`. Se hace referencia a cada generador de correcciones con un descriptor exclusivo (`hori_id` y `vert_id`) del tipo `corrdescr`. Cada uno de los dos sensores utilizará un generador de correcciones.

Se configura una interrupción de temporización para llamar a la rutina TRAP `ReadSensors` con una frecuencia de 5 Hz. Los offsets necesarios para la corrección de la trayectoria se calculan en la rutina TRAP y se escriben en el generador de correcciones correspondiente (al que se hace referencia con los descriptors `hori_id` y `vert_id`) con la instrucción `CorrWrite`. Todas las correcciones tienen un efecto inmediato sobre la trayectoria.

La instrucción `MoveL` debe ser programada con el argumento modificador `Corr` si se utilizan correcciones de trayectoria. De lo contrario, no se ejecuta ninguna corrección.

Cuando la primera instrucción `MoveL` está preparada, se utiliza la función `CorrRead` para leer la suma de todas las correcciones (la corrección total de la trayectoria) indicada por todos los generadores de correcciones conectados. El resultado de la corrección vertical total de la trayectoria se escribe en el FlexPendant con la instrucción `TPWrite`.

A continuación, `CorrDiscon` desconecta el generador de correcciones de la corrección vertical (al que se hace referencia con el descriptor `vert_id`). Todas las correcciones añadidas por este generador de correcciones se eliminarán de la corrección total de la trayectoria. Las correcciones añadidas por el generador de correcciones para la corrección horizontal se siguen conservando.

Por último, la función `CorrClear` elimina todos los generadores de correcciones restantes, junto con sus correcciones añadidas previamente. En este caso, el único que se elimina es el generador de correcciones para la corrección horizontal. La interrupción de temporización también se elimina con la instrucción `IDelete`.

#### Los generadores de correcciones

En la tabla siguiente se muestran los generadores de correcciones.

| X | Y | Z | Eje de coordenadas de la trayectoria   |
|---|---|---|--|
| 0 | 0 | 3 | Generador de correcciones vertical, con la suma de todas sus correcciones de trayectoria   |
| 0 | 1 | 0 | Generador de correcciones horizontal, con la suma de todas sus correcciones de trayectoria |
| - | - | - | Generador de correcciones no conectado   |
| - | - | - | Generador de correcciones no conectado   |
| - | - | - | Generador de correcciones no conectado   |

Continúa en la página siguiente



## 1.62 CorrCon - Establece una conexión con un generador de correcciones

*Path Offset*

*Continuación*

| X | Y | Z | Eje de coordenadas de la trayectoria  |
|---|---|---|---|
| 0 | 1 | 3 | La suma de todas las correcciones realizadas por todos los generadores de corrección conectados |

### Limitaciones

- Pueden conectarse un máximo de 5 generadores de correcciones a la vez.
- Los generadores de correcciones no se mantienen al reiniciar el controlador.
- Las esquinas afiladas y ejecución hacia atrás deben evitarse cuando se utilice un generador de correcciones ya que la corrección se añade en el sistema de coordenadas de la trayectoria.

### Sintaxis

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

```

### Información relacionada

| Para obtener más información sobre                   | Consulte  |
|--|---|
| Desconexión de un generador de correcciones          | <a href="#">CorrDiscon - Cierra la conexión con un generador de correcciones en la página 182</a> |
| Escritura en un generador de correcciones            | <a href="#">CorrWrite - Escribe en un generador de correcciones en la página 183</a>              |
| Lectura de los offsets totales actuales              | <a href="#">CorrRead - Lee los offsets totales actuales en la página 1235</a>                     |
| Eliminación de todos los generadores de correcciones | <a href="#">CorrClear - Elimina todos los generadores de correcciones en la página 176</a>        |
| Descriptor de generador de correcciones              | <a href="#">corrdescr - Descriptor de generador de correcciones en la página 1655</a>             |

# 1 Instrucciones

## 1.63 CorrDiscon - Cierra la conexión con un generador de correcciones

*Path Offset*

## 1.63 CorrDiscon - Cierra la conexión con un generador de correcciones

### Descripción

`CorrDiscon` se utiliza para desconectarse de un generador de correcciones con el que se ha conectado anteriormente. Esta instrucción puede utilizarse para eliminar las correcciones obtenidas anteriormente.

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción `CorrDiscon`:  
Consulte también [Más ejemplos en la página 182](#).

#### Ejemplo 1

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

`CorrDiscon` cierra la conexión con el generador de correcciones con el que se había conectado anteriormente y al que se hace referencia con el descriptor `id`.

### Argumentos

`CorrDiscon` `Descr`

`Descr`

Tipo de dato: `corrdescr`

Descriptor del generador de correcciones.

### Más ejemplos

Para obtener más ejemplos de la instrucción `CorrDiscon`, consulte [CorrCon - Establece una conexión con un generador de correcciones en la página 177](#).

### Sintaxis

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

### Información relacionada

| Para obtener más información sobre                   | Consulte   |
|--|--|
| Conexión con un generador de correcciones            | <a href="#">CorrCon - Establece una conexión con un generador de correcciones en la página 177</a> |
| Escritura en un generador de correcciones            | <a href="#">CorrWrite - Escribe en un generador de correcciones en la página 183</a>               |
| Lectura de los offsets totales actuales              | <a href="#">CorrRead - Lee los offsets totales actuales en la página 1235</a>                      |
| Eliminación de todos los generadores de correcciones | <a href="#">CorrClear - Elimina todos los generadores de correcciones en la página 176</a>         |
| Descriptor de corrección                             | <a href="#">corrdescr - Descriptor de generador de correcciones en la página 1655</a>              |

## 1.64 CorrWrite - Escribe en un generador de correcciones

### Descripción

`CorrWrite` se utiliza para escribir offsets en el sistema de coordenadas de trayectoria de un generador de correcciones.

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción `CorrWrite`:

#### Ejemplo 1

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

Los offsets actuales, almacenados en la variable `offset`, se escriben en un generador de correcciones al que se hace referencia con el descriptor `id`.

### Argumentos

`CorrWrite Descr Data`

#### Descr

Tipo de dato: `corrdescr`  
Descriptor del generador de correcciones.

#### Datos

Tipo de dato: `pos`  
El offset a escribir.

### Más ejemplos

Para obtener más ejemplos de la instrucción `CorrWrite`, consulte [CorrCon - Establece una conexión con un generador de correcciones en la página 177](#).

### Limitaciones

El mayor rendimiento se consigue en las trayectorias rectas. A medida que aumentan la velocidad y los ángulos entre dos trayectorias lineales consecutivas, la desviación respecto de la trayectoria esperada aumenta consiguientemente. Esto mismo se aplica a los círculos a medida que se reduce el radio del círculo.

### Sintaxis

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

### Información relacionada

| Para obtener más información sobre        | Consulte   |
|---|--|
| Conexión con un generador de correcciones | <a href="#">CorrCon - Establece una conexión con un generador de correcciones en la página 177</a> |

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.64 CorrWrite - Escribe en un generador de correcciones

*Path Offset*

*Continuación*

| <b>Para obtener más información sobre</b>            | <b>Consulte</b>  |
|--|--|
| Desconexión de un generador de correcciones          | <a href="#"><i>CorrDiscon - Cierra la conexión con un generador de correcciones en la página 182</i></a> |
| Lectura de los offsets totales actuales              | <a href="#"><i>CorrRead - Lee los offsets totales actuales en la página 1235</i></a>                     |
| Eliminación de todos los generadores de correcciones | <a href="#"><i>CorrClear - Elimina todos los generadores de correcciones en la página 176</i></a>        |
| Descriptor de generador de correcciones              | <a href="#"><i>corrdescr - Descriptor de generador de correcciones en la página 1655</i></a>             |

## 1.65 DeactEventBuffer - Desactivación de búfer de eventos

### Descripción

`DeactEventBuffer` no se utiliza para desactivar el uso del búfer de eventos en la tarea actual de programa de movimiento.

Las instrucciones `DeactEventBuffer` y `ActEventBuffer` deben usarse al combinar una aplicación con puntos finos y una aplicación continua en la que las señales deben activarse de antemano debido al uso de equipos de proceso lentos.

Esta instrucción sólo puede usarse en la tarea principal `T_ROB1` o, si se cuenta con un sistema `MultiMove`, en las tareas de movimiento.

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción `DeactEventBuffer`:

#### Ejemplo 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` desactiva el búfer de eventos configurado. Al utilizar una aplicación con puntos finos, el arranque del robot desde el punto fino será más rápido. Al activar el búfer de eventos con `ActEventBuffer`, es posible activar las señales de antemano para una aplicación que usa equipos de proceso lentos.

### Ejecución de programas

La desactivación de un búfer de eventos sólo se aplica al siguiente movimiento de robot ejecutado, de cualquier tipo, y se mantiene vigente hasta la ejecución de una instrucción `ActEventBuffer`.

La instrucción esperará hasta que el robot y los ejes externos hayan alcanzado el punto de paro (`ToPoint` de la instrucción de movimiento actual) antes de la desactivación del búfer de eventos. Por tanto, se recomienda programar con un punto fino la instrucción de movimiento precedente a `DeactEventBuffer`.

El valor predeterminado (`ActEventBuffer`) se establece automáticamente

- cuando se utiliza el modo de reinicio **Restablecer RAPID**
- al cargar un nuevo programa o un nuevo módulo
- al iniciar la ejecución del programa desde el principio
- al mover el puntero del programa a `main`
- al mover el puntero del programa a una rutina
- al mover el puntero de programa de una forma que se pierde el orden de la ejecución.

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.65 DeactEventBuffer - Desactivación de búfer de eventos

RobotWare - OS

Continuación

---

### Limitaciones

DeactEventBuffer no puede ejecutarse en rutinas de RAPID que estén conectadas a los siguientes eventos especiales del sistema: PowerOn, Stop, QStop, Restart o Step.

---

### Sintaxis

```
DeactEventBuffer ';' ;'
```

---

### Información relacionada

| Para obtener más información sobre | Consulte   |
|------------------------------------|--|
| Activación de un búfer de eventos  | <a href="#">ActEventBuffer - Activación de búfer de eventos en la página 28</a>  |
| Configuración de Event preset time | <i>Manual de referencia técnica - Parámetros del sistema</i>                     |
| Datos de parámetros de movimiento  | <a href="#">motsetdata - Datos de parámetros de movimiento en la página 1705</a> |

## 1.66 DeactUnit - Desactiva una unidad mecánica

### Utilización

DeactUnit se utiliza para desactivar una unidad mecánica.

Puede usarse para determinar qué unidad debe estar activa, por ejemplo cuando se utilizan unidades de accionamiento comunes.

Esta instrucción sólo puede usarse en la tarea principal T\_ROB1 o, si se cuenta con un sistema MultiMove, en las tareas de movimiento.

### Ejemplos

Los ejemplos siguientes ilustran la instrucción DeactUnit.

#### Ejemplo 1

```
DeactUnit orbit_a;
```

Desactivación de la unidad mecánica orbit\_a.

#### Ejemplo 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;
```

La unidad track\_motion quedará estacionaria cuando el robot se mueve hacia p20 y p30. A continuación, tanto el robot como track\_motion se mueven hacia p40.

#### Ejemplo 3

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

Se desactiva la unidad orbit1 y se activa orbit2.

### Argumentos

```
DeactUnit MechUnit
```

#### MechUnit

*Mechanical Unit*

Tipo de dato: mecunit

El nombre de la unidad mecánica que debe desactivarse.

### Ejecución de programas

Cuando la trayectoria del robot y de los ejes externos ha sido completada, la trayectoria del nivel de trayectoria actual se borra y se desactiva la unidad mecánica especificada. Esto significa que no se controlará ni monitorizará hasta que se reactive.

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.66 DeactUnit - Desactiva una unidad mecánica

RobotWare - OS

Continuación

Si varias unidades mecánicas comparten una misma unidad de accionamiento, la desactivación de una de estas unidades mecánicas también desconecta la unidad de la unidad de accionamiento común.

---

### Limitaciones

La instrucción `DeactUnit` no puede utilizarse si una de las unidades mecánicas se encuentra en el modo independiente.

Si esta instrucción va precedida de una instrucción de movimiento, ésta última debe programarse con un punto de paro (zonedata `fine`), no un punto de paso. De lo contrario, no será posible reanudar la ejecución tras una caída de suministro eléctrico.

`DeactUnit` no puede ejecutarse en rutinas de RAPID que estén conectadas a los siguientes eventos especiales del sistema: `PowerOn`, `Stop`, `QStop`, `Restart` o `Step`.

Es posible usar `ActUnit - DeactUnit` en el nivel `StorePath`, pero al ejecutar `RestoPath` deben estar activadas las mismas unidades mecánicas al ejecutar `StorePath`. En una operación de este tipo, la grabadora de trayectorias y la trayectoria del nivel básico estarán intactas, pero la trayectoria del nivel `StorePath` se borrará.

---

### Sintaxis

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

---

### Información relacionada

| Para obtener más información sobre                   | Consulte   |
|--|--|
| Activación de unidades mecánicas                     | <a href="#">ActUnit - Activa una unidad mecánica en la página 30</a>   |
| Unidades mecánicas                                   | <a href="#">mecunit - Unidad mecánica en la página 1703</a>  |
| Comprobar si una unidad mecánica está activada o no. | <a href="#">IsMechUnitActive - Indica si una unidad mecánica está activa en la página 1357</a>   |
| Grabadora de trayectorias                            | <a href="#">PathRecMoveBwd - Hace retroceder la grabadora de trayectorias en la página 514</a> <a href="#">mecunit - Unidad mecánica en la página 1703</a> |



## 1.67 Decr - Reduce un número en 1

### Utilización

`Decr` se utiliza para restar 1 a una variable o una variable persistente de tipo numérico.

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción `Decr`:  
Consulte también [Más ejemplos en la página 189](#).

#### Ejemplo 1

```
Decr reg1;
```

Se resta 1 a `reg1`, es decir `reg1:=reg1-1`.

### Argumentos

```
Decr Name | Dname
```

Name

Tipo de dato: num

El nombre de la variable o de la variable persistente cuyo valor se desea reducir.

Dname

Tipo de dato: dnum

El nombre de la variable o de la variable persistente cuyo valor se desea reducir.

### Más ejemplos

A continuación aparecen más ejemplos de la instrucción `Decr`.

#### Ejemplo 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
```

Se pregunta al operador cuántas piezas deben producirse. La variable `no_of_parts` se utiliza para contar el número de piezas que quedan por producir.

#### Ejemplo 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
```

Se pregunta al operador cuántas piezas deben producirse. La variable `no_of_parts` se utiliza para contar el número de piezas que quedan por producir.

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.67 Decr - Reduce un número en 1

RobotWare - OS

Continuación

---

### Sintaxis

Decr

[ Name ':=' ] < var or pers (**INOUT**) of num >

| [ Dname ':=' ] < var or pers (**INOUT**) of dnum > ' ; '

---

### Información relacionada

| Para obtener más información sobre  | Consulte  |
|---|---|
| Incremento de una variable en 1   | <a href="#">Incr - Aumenta en 1 un valor en la página 262</a> |
| Sustracción de cualquier valor a una variable                                       | <a href="#">Add - Suma un valor numérico en la página 32</a>  |
| Cambio de un dato mediante una expresión arbitraria, por ejemplo una multiplicación | <a href="#">":=" - Asigna un valor en la página 42</a>        |

## 1.68 DropSensor - Colocación de un objeto en el sensor

### Utilización

DropSensor se utiliza para desconectarse del objeto actual para dejar el programa preparado para el siguiente objeto.

DropSensor se utiliza para la sincronización de sensores, pero no para la sincronización analógica.

### Ejemplo básico

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

### Argumentos

DropSensor MechUnit

MechUnit

#### *Mechanical Unit*

Tipo de dato: mecunit

La unidad mecánica en movimiento con la que está relacionada la posición de robot de la instrucción.

### Ejecución de programas

Al soltar el objeto, la unidad de codificador deja de seguir al objeto. El objeto se elimina de la cola de objetos y no puede ser recuperado.

### Limitaciones

Si se ejecuta la instrucción mientras el trabajo está utilizando activamente el objeto del sensor, el movimiento se detiene. La instrucción debe ser ejecutada una vez que el robot haya pasado por el último `robtarget` sincronizado.

La instrucción sólo puede ejecutarse después de utilizar un movimiento no sincronizado en las instrucciones de movimiento precedentes, con instrucciones de punto fino o (>1) zonas de esquina.

### Sintaxis

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

### Información relacionada

| Para obtener más información sobre | Consulte   |
|------------------------------------|--|
| Espera a la conexión de un sensor  | <a href="#">WaitSensor - Espera a la conexión de un sensor en la página 1071</a> |
| Sincronización con un sensor       | <a href="#">SyncToSensor - Sincronización con un sensor en la página 866</a>     |

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.68 DropSensor - Colocación de un objeto en el sensor

### *Machine Synchronization*

#### *Continuación*

| Para obtener más información sobre | Consulte   |
|------------------------------------|--|
| <i>Machine Synchronization</i>     | <i>Application manual - Controller software IRC5</i> |

## 1.69 DropWObj - Suelta un objeto de trabajo sobre un transportador

### Utilización

DropWObj (*Drop Work Object*) se utiliza para desconectarse del objeto actual para dejar el programa preparado para el siguiente objeto.

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción DropWObj:

#### Ejemplo 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;
```

### Argumentos

DropWObj WObj

WObj

#### *Work Object*

Tipo de dato: wobjdata

El objeto de trabajo móvil (sistema de coordenadas) con el que está relacionada la posición de robot indicada en la instrucción. El transportador de la unidad mecánica debe especificarse con *ufmec* en el objeto de trabajo.

### Ejecución de programas

Al soltar el objeto de trabajo, la unidad de codificador deja de seguir al objeto. El objeto se elimina de la cola de objetos y no puede ser recuperado.

### Limitaciones

Si se ejecuta la instrucción mientras el trabajo está utilizando activamente el objeto de trabajo coordinado con el transportador, el movimiento se detiene.

La instrucción sólo puede ejecutarse después de utilizar un objeto de trabajo fijo en las instrucciones de movimiento precedentes, con instrucciones de punto fino o (>1) zonas de esquina.

### Sintaxis

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

### Información relacionada

| Para obtener más información sobre | Consulte  |
|------------------------------------|---|
| Espera a objetos de trabajo        | <a href="#">WaitWObj - Esperar a un objeto de trabajo en un transportador en la página 1090</a> |
| Seguimiento de transportadores     | <i>Application manual - Conveyor tracking</i>   |

# 1 Instrucciones

## 1.70 EOffsOff - Desactiva un offset de ejes adicionales

RobotWare - OS

## 1.70 EOffsOff - Desactiva un offset de ejes adicionales

### Utilización

`EOffsOff` (*External Offset Off*) se utiliza para desactivar un offset de ejes adicionales.

El offset de los ejes adicionales se activa con la instrucción `EOffsSet` o `EOffsOn` y se aplica a todos los movimientos hasta que se activa otro offset para los ejes adicionales o hasta que se desactiva el offset de los ejes adicionales.

Esta instrucción sólo puede usarse en la tarea principal `T_ROB1` o, si se cuenta con un sistema MultiMove, en las tareas de movimiento.

### Ejemplos básicos

Los ejemplos siguientes ilustran la instrucción `EOffsOff`.

#### Ejemplo 1

```
EOffsOff;
```

Desactivación del offset de ejes adicionales.

#### Ejemplo 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;
```

Se define un offset como la diferencia entre la posición de cada eje en `p10` y `p11`. El desplazamiento afecta al movimiento hacia `p20` y `p30`, pero no hacia `p40`.

### Ejecución de programas

Se restablecen los offsets activos para los ejes adicionales.

### Sintaxis

```
EOffsOff ' ; '
```

### Información relacionada

| Para obtener más información sobre                     | Consulte   |
|--|--|
| Definición del desplazamiento con dos posiciones       | <a href="#">EOffsOn - Activa un offset de ejes adicionales en la página 195</a>                                |
| Definición de offsets a partir de valores conocidos    | <a href="#">EOffsSet - Activa un offset de ejes adicionales a partir de valores conocidos en la página 197</a> |
| Desactivación del desplazamiento de programa del robot | <a href="#">PDispOff - Desactiva el desplazamiento de programa en la página 532</a>                            |

## 1.71 EOffsOn - Activa un offset de ejes adicionales

### Utilización

EOffsOn (*External Offset On*) se utiliza para definir y activar un offset de ejes adicionales a partir de dos posiciones.

Esta instrucción sólo puede usarse en la tarea principal T\_ROB1 o, si se cuenta con un sistema MultiMove, en las tareas de movimiento.

### Ejemplos básicos

Los ejemplos siguientes ilustran la instrucción EOffsOn.

Consulte también [Más ejemplos en la página 196](#).

#### Ejemplo 1

```
MoveL p10, v500, z10, tool1;
EOffsOn \ExeP:=p10, p20;
```

Activación de un offset de ejes adicionales. El offset se calcula para cada eje, a partir de la diferencia entre las posiciones p10 y p20.

#### Ejemplo 2

```
MoveL p10, v500, fine \Inpos := inpos50, tool1;
EOffsOn *;
```

Activación de un offset de ejes adicionales. Dado que en la instrucción anterior se ha utilizado un punto de paro bien definido, no es necesario utilizar el argumento \ExeP. El desplazamiento se calcula a partir de la diferencia existente entre la posición real de cada eje y el punto programado (\*) almacenado en la instrucción.

### Argumentos

```
EOffsOn [\ExeP] ProgPoint
```

[ \ExeP ]

#### *Executed Point*

Tipo de dato: robtarget

La nueva posición, utilizada para el cálculo del offset. Si se omite el argumento, se utiliza la posición actual de los ejes en el momento de la ejecución del programa.

ProgPoint

#### *Programmed Point*

Tipo de dato: robtarget

La posición original de los ejes en el momento de la programación.

### Ejecución de programas

El offset se calcula como la diferencia entre \ExeP y ProgPoint en cada eje adicional. Si no se ha especificado el valor de \ExeP, se utiliza en su lugar la posición actual de los ejes en el momento de la ejecución del programa. Dado que lo que se utiliza es la posición real de los ejes, los ejes no deben estar en movimiento cuando se ejecuta EOffsOn.

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.71 EOffsOn - Activa un offset de ejes adicionales

RobotWare - OS

Continuación

A continuación, se utiliza este offset para desplazar la posición de los ejes adicionales en las instrucciones de posicionamiento posteriores. El offset permanece activo hasta que se activa un offset diferente (con la instrucción `EOffsSet` o `EOffsOn`) o hasta que se desactiva el offset de los ejes adicionales (con la instrucción `EOffsOff`).

Sólo es posible activar al mismo tiempo un offset para cada eje adicional individual. Por otro lado, es posible programar varios `EOffsOn` uno tras otro y, si se hace, se suman los distintos offsets.

El offset de los ejes adicionales se restablece automáticamente en los siguientes casos:

- cuando se utiliza el modo de reinicio **Restablecer RAPID**
- al cargar un nuevo programa o un nuevo módulo
- al iniciar la ejecución del programa desde el principio
- al mover el puntero del programa a `main`
- al mover el puntero del programa a una rutina
- al mover el puntero de programa de una forma que se pierde el orden de la ejecución.

### Más ejemplos

A continuación aparecen más ejemplos de cómo usar la instrucción `EOffsOn`.

#### Ejemplo 1

```
SearchL sen1, psearch, p10, v100, tool1;  
PDispOn \ExeP:=psearch, *, tool1;  
EOffsOn \ExeP:=psearch, *;
```

Se realiza una búsqueda en la cual la posición buscada tanto de los ejes del robot como de los ejes adicionales se almacena en la posición `psearch`. Cualquier movimiento realizado a continuación parte de esta posición utilizando un desplazamiento de programa tanto de los ejes del robot como de los ejes adicionales. El desplazamiento se calcula a partir de la diferencia existente entre la posición buscada y el punto programado (\*) almacenado en la instrucción.

### Sintaxis

```
EOffsOn  
[ '\ ' ExeP ' := ' < expression (IN) of robtarget> ', ' ]  
[ ProgPoint ' := ' ] < expression (IN) of robtarget> ' ;'
```

### Información relacionada

| Para obtener más información sobre                  | Consulte   |
|---|--|
| Desactivación de offset de ejes adicionales         | <a href="#">EOffsOff - Desactiva un offset de ejes adicionales en la página 194</a>                            |
| Definición de offsets a partir de valores conocidos | <a href="#">EOffsSet - Activa un offset de ejes adicionales a partir de valores conocidos en la página 197</a> |
| Desplazamiento de los movimientos del robot         | <a href="#">PDispOn - Activa el desplazamiento de programa en la página 533</a>                                |
| Sistemas de coordenadas                             | <a href="#">Manual de referencia técnica - RAPID Overview</a>  |



## 1.72 EOffset - Activa un offset de ejes adicionales a partir de valores conocidos

## Utilización

EOffset (*External Offset Set*) se utiliza para definir y activar un offset de ejes adicionales a partir de valores conocidos.

Esta instrucción sólo puede usarse en la tarea principal T\_ROB1 o, si se cuenta con un sistema MultiMove, en las tareas de movimiento.

## Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción EOffset:

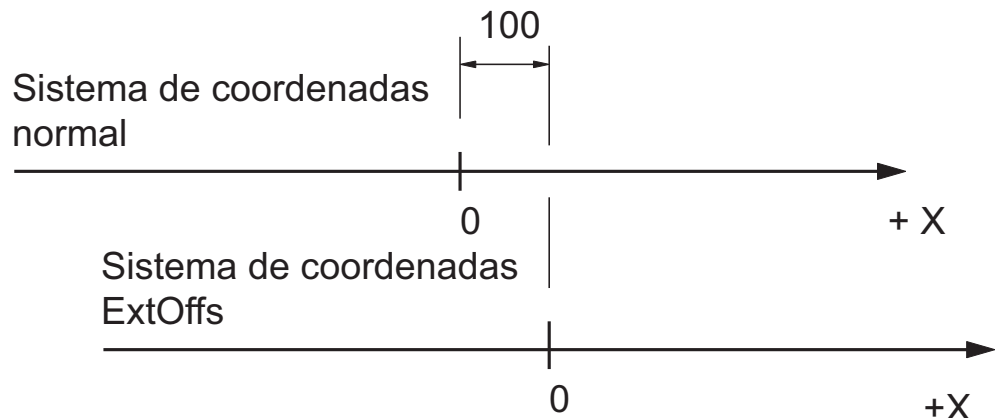
## Ejemplo 1

```
VAR extjoint eax_a_p100 := [100, 0, 0, 0, 0, 0];
...
EOffset eax_a_p100;
```

Activación de un offset `eax_a_p100` para los ejes adicionales, lo que implica que (siempre y cuando el eje adicional lógico "a" sea lineal):

- El sistema de coordenadas `ExtOffs` se desplaza 100 mm para el eje lógico "a" (consulte la figura siguiente).
- Siempre y cuando esté activado este offset, todas las posiciones se desplazan 100 mm en la dirección del eje x.

La figura que aparece a continuación muestra el desplazamiento de un eje adicional.



xx0500002162

## Argumentos

EOffset EAxOffs

## EAxOffs

*External Axes Offset*

Tipo de dato: `extjoint`

El offset de los ejes adicionales se define como un dato del tipo `extjoint`, expresado en:

- mm en el caso de los ejes lineales
- Grados en el caso de los ejes de rotación

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.72 EOffsSet - Activa un offset de ejes adicionales a partir de valores conocidos

RobotWare - OS

Continuación

### Ejecución de programas

El offset de los ejes adicionales se activa cuando se ejecuta la instrucción `EOffsSet` y permanece activo hasta que se activa otro offset (con una instrucción `EOffsSet` o `EOffsOn`) o hasta que se desactiva el offset de los ejes adicionales (con una instrucción `EOffsOff`).

Sólo es posible activar un offset para los ejes adicionales al mismo tiempo. No es posible sumar un offset a otro mediante `EOffsSet`.

El offset de los ejes adicionales se restablece automáticamente en los siguientes casos:

- cuando se utiliza el modo de reinicio **Restablecer RAPID**
- al cargar un nuevo programa o un nuevo módulo
- al iniciar la ejecución del programa desde el principio
- al mover el puntero del programa a `main`
- al mover el puntero del programa a una rutina
- al mover el puntero de programa de una forma que se pierde el orden de la ejecución.

### Sintaxis

```
EOffsSet  
  [ EAxOffs ':= ' ] < expression (IN) of extjoint> ';' 
```

### Información relacionada

| Para obtener más información sobre                 | Consulte  |
|--|---|
| Activación de un offset de ejes adicionales        | <a href="#">EOffsOn - Activa un offset de ejes adicionales en la página 195</a>     |
| Desactivación de offset de ejes adicionales        | <a href="#">EOffsOff - Desactiva un offset de ejes adicionales en la página 194</a> |
| Desplazamiento de los movimientos del robot        | <a href="#">PDispOn - Activa el desplazamiento de programa en la página 533</a>     |
| Definición de datos del tipo <code>extjoint</code> | <a href="#">extjoint - Posición de los ejes externos en la página 1677</a>          |
| Sistemas de coordenadas                            | <a href="#">Manual de referencia técnica - RAPID Overview</a>                       |

### 1.73 EraseModule - Elimina un módulo

#### Utilización

`EraseModule` se utiliza para eliminar un módulo de la memoria de programas durante la ejecución.

No existe ninguna restricción en cuanto a cómo se haya cargado el módulo. Puede haber sido cargado manualmente, desde la configuración o con una combinación de las instrucciones `Load`, `StartLoad` y `WaitLoad`.

El módulo no puede definirse como *Shared* en la configuración.

#### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción `EraseModule`:

##### Ejemplo 1

```
EraseModule "PART_A";
```

Borra el módulo de programa `PART_A` de la memoria de programas.

#### Argumentos

```
EraseModule ModuleName
```

ModuleName

Tipo de dato: `string`

El nombre del módulo que debe eliminarse. Recuerde que se trata del nombre del módulo, no el nombre del archivo.

#### Ejecución de programas

La ejecución del programa espera a que el módulo de programa termine el proceso de eliminación antes de que la ejecución continúe con la instrucción siguiente.

Una vez eliminado el módulo de programa, se vinculan los demás módulos de programa.

#### Limitaciones

No se permite eliminar módulos de programa que se estén ejecutando.

Las rutinas TRAP, los eventos de E/S del sistema y otras tareas de programa no pueden ejecutarse durante el proceso de eliminación.

Evite tener movimientos en curso durante la eliminación.

La detención del programa durante la ejecución de la instrucción `EraseModule` da lugar a un paro protegido con los motores apagados y un mensaje de error "20025 Tiempo excesivo Orden paro" en el FlexPendant.

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.73 EraseModule - Elimina un módulo

RobotWare - OS

Continuación

---

### Gestión de errores

Pueden generarse los errores recuperables enumerados a continuación. Estos errores pueden ser gestionados en un gestor de errores. El valor de la variable de sistema `ERRNO` cambia a:

| Nombre                  | Causa del error  |
|-------------------------|--|
| <code>ERR_MODULE</code> | El archivo de la instrucción <code>EraseModule</code> no puede ser eliminado porque no ha sido encontrado. |

---

### Sintaxis

```
EraseModule  
[ModuleName':=']<expression (IN) of string>;'
```

---

### Información relacionada

| Para obtener más información sobre   | Consulte   |
|--|--|
| Descarga de un módulo de programa  | <a href="#">UnLoad - Descargar un módulo de programa durante la ejecución en la página 1021</a>  |
| Carga de módulos de programa en paralelo con la ejecución de otro programa | <a href="#">StartLoad - Carga de programa durante la ejecución en la página 792</a><br><a href="#">WaitLoad - Conectar un módulo cargado a una tarea en la página 1064</a> |
| Aceptación de una referencia no resuelta                                   | <a href="#">Manual de referencia técnica - Parámetros del sistema, sección Controller</a>  |

---

## 1.74 ErrLog - Escribe un mensaje de error

---

### Utilización

`ErrLog` se utiliza para mostrar un mensaje de error en el FlexPendant y escribirlo en el registro de eventos. Es necesario indicar el número de error y cinco argumentos acerca del mismo. El mensaje se almacena en el dominio de proceso del registro del robot. `ErrLog` también puede usarse para mostrar avisos y mensajes de información.

---

### Ejemplos básicos

Los ejemplos siguientes ilustran la instrucción `ErrLog`.

#### Ejemplo 1

Si desea crear su propio archivo `.xml`, puede usar el valor de `ErrorId` 4800 como en el ejemplo siguiente:

```
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;
```

En el FlexPendant, el mensaje tendría el aspecto siguiente:

```
Mensaje de evento: 4800
myerror
errortext1
errortext2
errortext3
errortext4
```

#### Ejemplo 2

Las ID `ErrorId` deben declararse en un archivo `.xml`. El número debe estar entre 5.000 y 9.999. El mensaje de error se escribe en el archivo `.xml` y los argumentos del mensaje enviados en la instrucción `ErrLog`. El valor de `ErrorId` del archivo `.xml` es el mismo que se indica en la instrucción `ErrLog`.

**NOTA:** Si usa un valor de `ErrorId` de entre 5.000-9.999, debe instalar su propio archivo `xml`.

Ejemplo de mensaje en un archivo `.xml`:

```
<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>
```

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.74 ErrLog - Escribe un mensaje de error

RobotWare - OS

Continuación

### Ejemplo de instrucción:

```
MODULE MyModule
  PROC main()
    VAR num errorid := 5210;
    VAR errstr arg := "P1";
    ErrLog errorid, ERRSTR_TASK, arg, ERRSTR_CONTEXT, ERRSTR_UNUSED,
      ERRSTR_UNUSED;
    ErrLog errorid \W, ERRSTR_TASK, arg,
      ERRSTR_CONTEXT, ERRSTR_UNUSED, ERRSTR_UNUSED;
  ENDPROC
ENDMODULE
```

En el FlexPendant, el mensaje tendría el aspecto siguiente:

**Mensaje de evento: 5210**

**Error de parámetro**

**Tarea: T\_ROB1**

**El símbolo P1 es de sólo lectura.**

**Contexto: MyModule/main/ErrLog**

La primera instrucción `ErrLog` genera un mensaje de error. El mensaje se almacena en el registro del robot, en el dominio de proceso. También se muestra en la pantalla del FlexPendant.

La segunda instrucción es una advertencia. Sólo se almacena un mensaje en el registro del robot.

En ambos casos, el programa sigue ejecutándose una vez terminada la instrucción.

---

### Argumentos

```
ErrLog ErrorID [\W] | [\I] Argument1 Argument2 Argument3 Argument4
      Argument5
```

ErrorId

**Tipo de dato:** num

El número de un error concreto que se desea monitorizar. El número de error debe estar en el intervalo 4.800-4.814 si se utiliza el archivo xml preinstalado y en el intervalo 5.000-9.999 si se usa un archivo xml propio.

[ \W ]

**Warning**

**Tipo de dato:** switch

Genera una advertencia que sólo se almacena en el registro de eventos (no se muestra directamente en la pantalla del FlexPendant).

[ \I ]

**Information**

**Tipo de dato:** switch

Genera un mensaje de información que sólo se almacena en el registro de eventos (no se muestra directamente en la pantalla del FlexPendant).

*Continúa en la página siguiente*

Si no se especifica ninguno de los argumentos `\W` o `\I`, la instrucción genera un mensaje de error directamente en el FlexPendant y lo almacena también en el registro de eventos.

Argument1

Tipo de dato: `errstr`

Primer argumento del mensaje de error. Puede usar cualquier cadena de tipo string o un dato predefinido del tipo `errstr`.

Argument2

Tipo de dato: `errstr`

Segundo argumento del mensaje de error. Puede usar cualquier cadena de tipo string o un dato predefinido del tipo `errstr`.

Argument3

Tipo de dato: `errstr`

Tercer argumento del mensaje de error. Puede usar cualquier cadena de tipo string o un dato predefinido del tipo `errstr`.

Argument4

Tipo de dato: `errstr`

Cuarto argumento del mensaje de error. Puede usar cualquier cadena de tipo string o un dato predefinido del tipo `errstr`.

Argument5

Tipo de dato: `errstr`

Quinto argumento del mensaje de error. Puede usar cualquier cadena de tipo string o un dato predefinido del tipo `errstr`.

---

### Ejecución de programas

Un mensaje de error (máximo 5 líneas) se muestra en el FlexPendant y se escribe en el registro de eventos.

En el caso del argumento `\W` o el argumento `\I` se escribe un aviso o un mensaje de información en el registro de eventos.

`ErrLog` genera errores de programa en el intervalo 4.800-4.814 si se usa el archivo de programa instalado por el sistema y en el intervalo 5.000-9.999 si se instala un archivo xml propio. El error generado depende del valor de `ErrorID` indicado.

El mensaje se almacena en el dominio de proceso del registro de eventos.

La forma de instalar un archivo xml propio se describe *Application manual - RobotWare Add-Ins*; consulte Información relacionada más abajo.

---

### Limitaciones

La longitud total de la cadena (Argument1-Argument5) está limitada a 195 caracteres.

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.74 ErrLog - Escribe un mensaje de error

RobotWare - OS

Continuación

### Sintaxis

```
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> ';'
```

### Información relacionada

| Para obtener más información sobre                            | Consulte   |
|---|--|
| Datos predefinidos de tipo errstr                             | <a href="#">errstr - Cadena de error en la página 1672</a>   |
| Visualización de mensajes en el FlexPendant                   | <a href="#">TPWrite - Escribe en el FlexPendant en la página 892</a><br><a href="#">UIMsgBox - Cuadro de mensaje de usuario de tipo básico en la página 1002</a> |
| Registro de eventos   | <a href="#">Manual del operador - IRC5 con FlexPendant</a>   |
| Mensajes del registro de eventos, explicación del archivo XML | <a href="#">Application manual - RobotWare Add-Ins, sección Mensajes del registro de eventos</a>   |
| Cómo instalar archivos XML al utilizar complementos           | <a href="#">Application manual - RobotWare Add-Ins</a>   |



### 1.75 ErrRaise - Escribe un aviso y llama a un gestor de errores

#### Utilización

`ErrRaise` se utiliza para crear un error en el programa y llamar a continuación al gestor de errores de la rutina. Se almacena un aviso en el registro de eventos. `ErrRaise` también puede usarse en el gestor de errores para propagar el error actual hacia el gestor de errores de la rutina desde la que se llamó a la rutina actual.

Es necesario indicar el nombre del error, el número de error y cinco argumentos acerca del mismo. El mensaje se almacena en el dominio de proceso del registro del robot.

#### Ejemplos básicos

Los ejemplos siguientes ilustran la instrucción `ErrRaise`.

##### Ejemplo 1

Si desea crear su propio archivo .xml, puede usar el valor de `ErrorId` 4800 como en el ejemplo siguiente:

```
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
```

En el FlexPendant, el mensaje tendría el aspecto siguiente (aviso y/o un error):

**Mensaje de evento: 4800**

**Estado de batería de respaldo**

**Tarea: main**

**La batería de respaldo está totalmente cargada.**

**Contexto: MyModule/main/ErrRaise**

Es necesario registrar un número de error con la instrucción `BookErrNo`. La cadena correspondiente se indica como primer argumento, `ErrorName` en la instrucción `ErrRaise`.

`ErrRaise` crear un error y llama al gestor de errores. Si el error se ha resuelto, se genera un aviso en el registro de eventos, dentro del dominio de proceso. De lo contrario, se genera un error no recuperable y el programa se detiene.

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.75 ErrRaise - Escribe un aviso y llama a un gestor de errores

RobotWare - OS

Continuación

`ErrRaise` también puede usarse en un gestor de errores en una subrutina. En este caso, la ejecución continúa en el gestor de errores de la rutina desde la que se realiza la llamada.

### Ejemplo 2

Las ID `ErrorId` deben declararse en un archivo `.xml`. El número debe estar entre 5.000 y 9.999. El mensaje de error se escribe en el archivo `.xml` y los argumentos del mensaje enviados en la instrucción `ErrRaise`. El valor de `ErrorId` del archivo `.xml` es el mismo que se indica en la instrucción `ErrRaise`.

**NOTA:** Si usa un valor de `ErrorId` de entre 5.000-9.999, debe instalar su propio archivo `xml`.

Ejemplo de mensaje en un archivo `.xml`:

```
<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>
```

Ejemplo de instrucción:

```
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
```

En el FlexPendant, el mensaje tendría el aspecto siguiente (aviso y/o un error):

**Mensaje de evento: 7055**

**Estado de batería de respaldo**

**Tarea: main**

**La batería de respaldo está totalmente cargada.**

**Contexto: MyModule/main/ErrRaise**

Es necesario registrar un número de error con la instrucción `BookErrNo`. La cadena correspondiente se indica como primer argumento, `ErrorMessage` en la instrucción `ErrRaise`.

`ErrRaise` crear un error y llama al gestor de errores. Si el error se ha resuelto, se genera un aviso en el registro de eventos, dentro del dominio de proceso. De lo contrario, se genera un error no recuperable y el programa se detiene.

*Continúa en la página siguiente*

## 1.75 ErrRaise - Escribe un aviso y llama a un gestor de errores

*RobotWare - OS*

*Continuación*

`ErrRaise` también puede usarse en un gestor de errores en una subrutina. En este caso, la ejecución continúa en el gestor de errores de la rutina desde la que se realiza la llamada.

### Argumentos

```
ErrRaise ErrorName ErrorId Argument1 Argument2 Argument3 Argument4  
Argument5
```

ErrorName

Tipo de dato: `string`

Es necesario registrar un número de error con la instrucción `BookErrNo`. La variable correspondiente se indica como `ErrorName`.

ErrorId

Tipo de dato: `num`

El número de un error concreto que se desea monitorizar. El número de error debe estar en el intervalo 4.800-4.814 si se utiliza el archivo xml preinstalado y en el intervalo 5.000-9.999 si se usa un archivo xml propio.

Argument1

Tipo de dato: `errstr`

Primer argumento del mensaje de error. Puede usar cualquier cadena de tipo `string` o un dato predefinido del tipo `errstr`.

Argument2

Tipo de dato: `errstr`

Segundo argumento del mensaje de error. Puede usar cualquier cadena de tipo `string` o un dato predefinido del tipo `errstr`.

Argument3

Tipo de dato: `errstr`

Tercer argumento del mensaje de error. Puede usar cualquier cadena de tipo `string` o un dato predefinido del tipo `errstr`.

Argument4

Tipo de dato: `errstr`

Cuarto argumento del mensaje de error. Puede usar cualquier cadena de tipo `string` o un dato predefinido del tipo `errstr`.

Argument5

Tipo de dato: `errstr`

Quinto argumento del mensaje de error. Puede usar cualquier cadena de tipo `string` o un dato predefinido del tipo `errstr`.

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.75 ErrRaise - Escribe un aviso y llama a un gestor de errores

RobotWare - OS

Continuación

---

### Ejecución de programas

ErrRaise genera avisos de programa en el intervalo 4.800-4.814 si se usa el archivo de programa instalado por el sistema y en el intervalo 5.000-9.999 si se instala un archivo xml propio. El error generado depende del valor de ErrorID indicado. Se almacena un aviso en el registro de mensajes del robot, en el dominio de proceso.

Cuando se ejecuta ErrRaise, el comportamiento depende de cuándo se ejecute:

- Si se ejecuta la instrucción en el cuerpo de la rutina, se genera un aviso y la ejecución continúa en el gestor de errores.
- Si se ejecuta la instrucción en un gestor de errores, el aviso anterior se omite, se genera uno nuevo y el control es elevado a la instrucción desde la que se hizo la llamada.

---

### Limitaciones

La longitud total de la cadena (Argument1-Argument5) está limitada a 195 caracteres.

---

### Más ejemplos

A continuación aparecen más ejemplos de cómo usar la instrucción ErrRaise.

#### Ejemplo 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;
  ERROR
  IF ERRNO = ERR_BATT THEN
    ErrRaise "ERR_NEW_ERR",7156,ERRSTR_TASK,ERRSTR_CONTEXT,
      ERRSTR_UNUSED,ERRSTR_UNUSED, ERRSTR_UNUSED;
  ENDIF
ENDPROC
```

Generar un nuevo aviso 7156 desde un gestor de errores. Elevar el control a la rutina desde la que se hizo la llamada y detener la ejecución.

---

### Sintaxis

```
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> ','
```

Continúa en la página siguiente

---

## 1.75 ErrRaise - Escribe un aviso y llama a un gestor de errores

*RobotWare - OS*

*Continuación*

```
[Argument4 ':='] < expression (IN) of errstr> ','  
[Argument5 ':='] < expression (IN) of errstr> ';' 
```

### Información relacionada

| Para obtener más información sobre | Consulte  |
|------------------------------------|---|
| Datos predefinidos de tipo errstr  | <a href="#">errstr - Cadena de error en la página 1672</a>                                  |
| Registro de números de error       | <a href="#">BookErrNo - Registra un número de error de sistema de RAPID en la página 50</a> |
| Gestión de errores                 | <a href="#">Manual de referencia técnica - RAPID Overview</a>                               |
| <i>Advanced RAPID</i>              | <a href="#">Application manual - Controller software IRC5</a>                               |

# 1 Instrucciones

---

## 1.76 ErrWrite - Escribe un mensaje de error

RobotWare - OS

## 1.76 ErrWrite - Escribe un mensaje de error

---

### Utilización

`ErrWrite` (*Error Write*) se utiliza para mostrar un mensaje de error en el FlexPendant y escribirlo en el registro de eventos. También puede usarse para mostrar avisos y mensajes de información.

---

### Ejemplos básicos

Los ejemplos siguientes ilustran la instrucción `ErrWrite`.

#### Ejemplo 1

```
ErrWrite "PLC error", "Fatal error in PLC" \RL2:="Call service";  
Stop;
```

Se almacena un mensaje en el registro del robot. También se muestra en la pantalla del FlexPendant.

#### Ejemplo 2

```
ErrWrite \W, "Search error", "No hit for the first search";  
RAISE try_search_again;
```

Sólo se almacena un mensaje en el registro del robot. Después la ejecución del programa continúa.

---

### Argumentos

```
ErrWrite [ \W ] | [ \I ] Header Reason [ \RL2 ] [ \RL3 ] [ \RL4 ]
```

[ \W ]

#### *Warning*

Tipo de dato: `switch`

Genera una advertencia que sólo se almacena en el registro de mensajes del robot (no se muestra directamente en la pantalla del FlexPendant).

[ \I ]

#### *Information*

Tipo de dato: `switch`

Genera un mensaje de información que sólo se almacena en el registro de eventos (no se muestra directamente en la pantalla del FlexPendant).

Si no se especifica ninguno de los argumentos `\W` o `\I`, la instrucción genera un mensaje de error directamente en el FlexPendant y lo almacena también en el registro de eventos.

Header

Tipo de dato: `string`

El título del mensaje de error (máximo 46 caracteres).

Reason

Tipo de dato: `string`

Motivo del error.

*Continúa en la página siguiente*

[ \RL2]

*Reason Line 2*

Tipo de dato: string

Motivo del error.

[ \RL3]

*Reason Line 3*

Tipo de dato: string

Motivo del error.

[ \RL4]

*Reason Line 4*

Tipo de dato: string

Motivo del error.

### Ejecución de programas

Un mensaje de error (máximo 5 líneas) se muestra en el FlexPendant y se escribe en el registro de mensajes del robot.

En el caso del argumento \W o el argumento \I se escribe un aviso o un mensaje de información en el registro de eventos.

ErrWrite genera el error de programa nº 80001 para un error, nº 80002 para un aviso (\W) y nº 80003 para un mensaje de información (\I).

### Limitaciones

La longitud total de la cadena (título+motivo+\RL2+\RL3+\RL4) está limitada a 195 caracteres.

### Sintaxis

```
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> ] ';'

```

### Información relacionada

| Para obtener más información sobre          | Consulte   |
|---|--|
| Datos predefinidos de tipo errstr           | <a href="#">errstr - Cadena de error en la página 1672</a>   |
| Visualización de mensajes en el FlexPendant | <a href="#">TPWrite - Escribe en el FlexPendant en la página 892</a><br><a href="#">UIMsgBox - Cuadro de mensaje de usuario de tipo básico en la página 1002</a> |
| Registro de eventos                         | <a href="#">Manual del operador - IRC5 con FlexPendant</a>   |
| Escritura de un mensaje de error - ErrLog   | <a href="#">ErrLog - Escribe un mensaje de error en la página 201</a>  |

# 1 Instrucciones

---

## 1.77 EXIT - Finaliza la ejecución del programa

RobotWare - OS

## 1.77 EXIT - Finaliza la ejecución del programa

---

### Utilización

`EXIT` se utiliza para finalizar la ejecución de un programa. Se impide la reanudación del programa, es decir, éste sólo puede ejecutarse empezando de nuevo desde la primera instrucción de la rutina `main`.

La instrucción `EXIT` debe usarse cuando se producen errores no recuperables o cuando se debe detener definitivamente la ejecución del programa. Para detener temporalmente la ejecución del programa, se utiliza la instrucción `Stop`. Tras la ejecución de la instrucción `EXIT`, el puntero de programa se pierde. Para continuar con la ejecución del programa, es necesario establecer el puntero de programa.

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción `EXIT`:

#### Ejemplo 1

```
ErrWrite "Fatal error","Illegal state";  
EXIT;
```

La ejecución del programa se detiene y no puede reanudarse en la posición en la que se detiene el programa.

### Sintaxis

```
EXIT ';' ;'
```

### Información relacionada

| Para obtener más información sobre              | Consulte   |
|---|--|
| Detención temporal de la ejecución del programa | <a href="#">Stop - Detención de la ejecución del programa en la página 822</a> |



## 1.78 ExitCycle - Interrumpe el ciclo actual y pasa al siguiente

### Utilización

`ExitCycle` se utiliza para interrumpir el ciclo actual y trasladar el puntero de programa (PP) a la primera instrucción de la rutina principal.

Si el programa se ejecuta en modo continuo, se empezará a ejecutar el ciclo siguiente.

Si la ejecución se encuentra en el modo cíclico, la ejecución se detiene en la primera instrucción de la rutina principal.

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción `ExitCycle`:

#### Ejemplo 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
```

En este programa, se ejecuta el ciclo siguiente si se activa la señal `di_error`.

### Ejecución de programas

La ejecución de `ExitCycle` en la tarea de programa que controla las unidades mecánicas da lugar a lo siguiente en la tarea actual:

- Se detienen los movimientos en curso del robot.
- Se eliminan todas las trayectorias del robot que no se hayan realizado aún en los distintos niveles de trayectoria (tanto en el nivel normal como en el nivel `StorePath` ).
- Se interrumpen todas las instrucciones iniciadas pero no terminadas en todos los niveles de ejecución (tanto en la ejecución normal como en las rutinas `TRAP`).
- El puntero de programa se traslada a la primera instrucción de la rutina principal.
- La ejecución del programa continúa y pasa a ejecutar el ciclo siguiente.

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.78 ExitCycle - Interrumpe el ciclo actual y pasa al siguiente

RobotWare - OS

Continuación

La ejecución de `ExitCycle` en cualquier otra tarea de programa que no controle las unidades mecánicas da lugar a lo siguiente en la tarea actual:

- Se interrumpen todas las instrucciones iniciadas pero no terminadas en todos los niveles de ejecución (tanto en la ejecución normal como en las rutinas `TRAP`).
- El puntero de programa se traslada a la primera instrucción de la rutina principal.
- La ejecución del programa continúa y pasa a ejecutar el ciclo siguiente.

Los demás elementos modales del programa y del sistema no se ven afectados por la instrucción `ExitCycle`. Entre ellos se encuentran los siguientes:

- El valor actual de las variables o variables persistentes.
- Cualquier parámetro de movimiento, como secuencias `StorePath-RestoPath` , zonas mundo, etc.
- Archivos abiertos, directorios, etc.
- Interrupciones definidas, etc.

Cuando se utiliza `ExitCycle` en llamadas a rutinas y la rutina de entrada se define con "Mover el puntero de programa a la rutina..." o "Llamar a la rutina...", `ExitCycle` interrumpe el ciclo actual y devuelve el puntero de programa a la primera instrucción de la rutina de entrada (en lugar de la rutina principal como se especificaba anteriormente).

### Sintaxis

```
ExitCycle';'
```

### Información relacionada

| Para obtener más información sobre         | Consulte   |
|--|--|
| Paro después de un error no recuperable    | <a href="#">EXIT - Finaliza la ejecución del programa en la página 212</a>     |
| Finalización de la ejecución del programa  | <a href="#">EXIT - Finaliza la ejecución del programa en la página 212</a>     |
| Paro para acciones de programa             | <a href="#">Stop - Detención de la ejecución del programa en la página 822</a> |
| Finalización de la ejecución de una rutina | <a href="#">RETURN - Finaliza la ejecución de una rutina en la página 619</a>  |

## 1.79 FitCircle: Se ajusta a un círculo con puntos 3D

### Utilización

`FitCircle` se utiliza para ajustar un círculo a un conjunto de puntos 3D.

### Ejemplos básicos

Los siguientes ejemplos ilustran la instrucción `FitCircle`.

#### Ejemplo 1

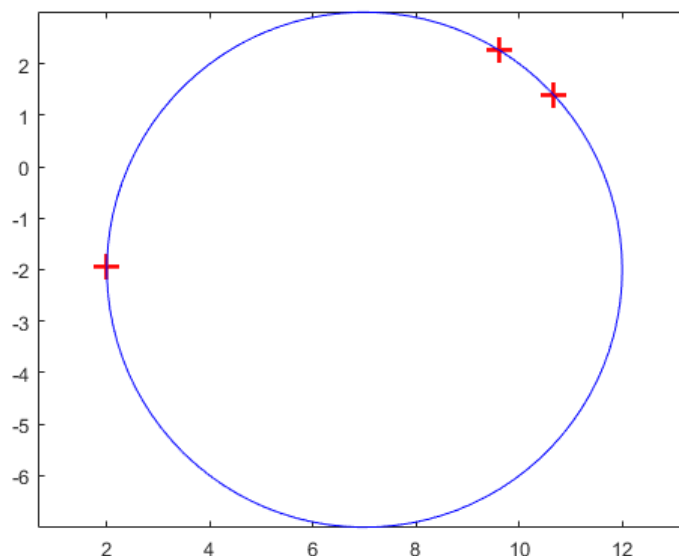
```
VAR pos points{3}:=[ [2.000264140454799, -1.948606082287765, 3],
                    [10.666326255802462, 1.399713485871053, 3],
                    [9.609499187363362, 2.265033879249959, 3]];
VAR num radius;
VAR pos center;
VAR pos normal;
FitCircle points, center, radius, normal;
```

Con solo tres puntos especificados, `FitCircle` calcula un círculo que pasa exactamente por todos los puntos. En este ejemplo, el círculo resultante es:

```
center = [7,-2,3]
radius = 5
normal = [0,0,1]
```

En este ejemplo, todos los puntos 3D tienen la misma coordenada z y, por tanto, el círculo identificado debe estar en el plano xy. En un caso general, el método identificará el plano que contiene el círculo. El plano se describe por el retorno normal, que es un vector de unidad perpendicular al círculo.

El círculo y los puntos de entrada se muestran en la siguiente figura.



xx1700000731

#### Ejemplo 2

```
VAR pos points{10}:=[
```

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.79 FitCircle: Se ajusta a un círculo con puntos 3D

RobotWare - OS

Continuación

```
[-7.2, -5.254055558738700, 3.307061712883007],
[-7.2, -3.211963764808295, 3.090792209685409],
[-7.2, -5.650784519354138, 4.074184087507510],
[-7.2, -4.261241341238363, 3.549808031134106],
[-7.2, -3.780658372123544, 12.570750298513245],
[-7.2, 1.309442476421255, 11.013856026376601],
[-7.2, -4.649041803426594, 3.435039251520052],
[-7.2, 1.403058916454365, 6.576147932013719],
[-7.2, -7.546395053424201, 9.841667138860654],
[-7.2, -2.490309697793828, 2.788705869634919]];
VAR pos center;
VAR num radius;
VAR pos normal;
VAR num rms;
VAR num maxErr;
VAR num indexWorst;

FitCircle points, center, radius, normal \RMS:=rms \MaxErr:=maxErr
\IndexWorst:=indexWorst;
```

En este caso, el círculo está ajustado a diez puntos que no están dispuestos en círculo. El resultado es un círculo que se ajusta a los puntos en sentido de mínimos cuadrados. Para simplificar el ejemplo, los puntos están todos en un plano paralelo al plano yz.

El círculo resultante es:

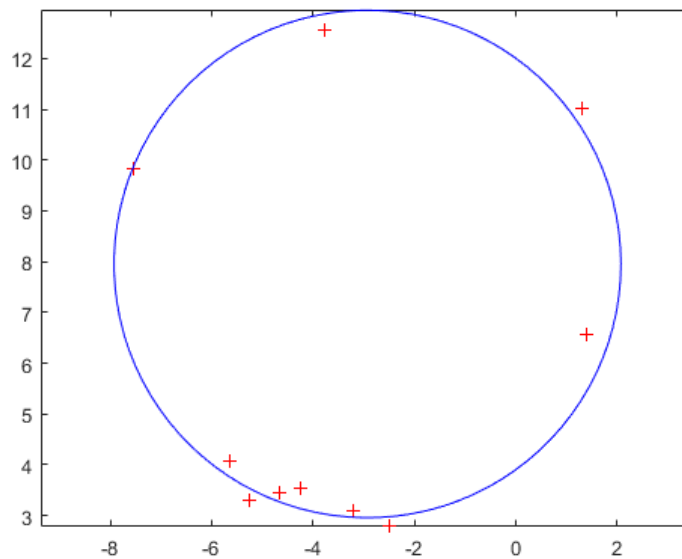
```
center = [-7.2,-2.92489,7.96317]
radius = 4.88656
normal = [1,0,0]
```

Los parámetros de error del ajuste son:

```
rms = 0.2387
maxErr = 0.3418
indexWorst = 8
```

Continúa en la página siguiente

El círculo y los puntos de entrada se muestran en la siguiente figura.



xx170000732

### Argumentos

```
FitCircle Points [\NumPoints] Center Radius Normal [\RMS] [\MaxErr]
[\IndexWorst]
```

Points

**Tipo de dato:** array of pos

Points es una matriz que contiene los puntos 3D para el ajuste del círculo.

[\NumPoints]

**Tipo de dato:** num

Con el argumento opcional NumPoints es posible especificar cuántos de los puntos se deben usar. Si se omiten, se utilizan todos los puntos de la matriz Points.

Center

**Tipo de dato:** pos

El centro del círculo resultante.

Radius

**Tipo de dato:** num

El radio del círculo resultante.

Normal

**Tipo de dato:** pos

Un vector de longitud de unidad que es perpendicular al plano del círculo identificado.

[\RMS]

**Tipo de dato:** num

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.79 FitCircle: Se ajusta a un círculo con puntos 3D

RobotWare - OS

Continuación

Argumento opcional que contiene el error del valor cuadrático medio del ajuste del círculo.

[ \MaxErr ]

Tipo de dato: num

Argumento opcional que contiene la máxima distancia entre el círculo resultante y los puntos de entrada.

[ \IndexWorst ]

Tipo de dato: num

Argumento opcional que contiene el índice del punto más distante respecto al círculo.

---

### Ejecución de programas

CircleFit ajusta un círculo a un conjunto de puntos 3D.

---

### Gestión de errores

Se generan los siguientes errores recuperables que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` se establecerá en:

|                |   |
|----------------|---|
| ERR_ARRAY_SIZE | Se utilizan dimensiones incorrectas o valores incorrectos en los argumentos opcionales. |
|----------------|---|

---

### Limitaciones

El número de puntos 3D debe estar entre 3 y 100.

---

### Sintaxis

```
CircleFit
[ Points ':='] <array {*} expression (IN) of dnum> ', '
[ '\ NumPoints ':= ' < expression (IN) of num> ] ', '
[ Center ':='] <variable (VAR) of pos> ', '
[ Radius ':='] <variable (VAR) of num> ', '
[ Normal ':='] <variable (VAR) of pos>
[ '\ RMS ':= ' <variable (VAR) of num> ]
[ '\ MaxErr ':= ' <variable (VAR) of num> ]
[ '\ IndexWorst ':= ' <variable (VAR) of num> ] ';'

```

---

### Información relacionada

| Para obtener más información sobre     | Consulte                                      |
|--|---|
| Instrucciones y funciones matemáticas. | Manual de referencia técnica - RAPID Overview |

### 1.80 FOR - Repite un número determinado de veces

#### Utilización

FOR se utiliza cuando es necesario repetir una o varias instrucciones un número determinado de veces.

#### Ejemplos básicos

Los ejemplos siguientes ilustran la instrucción FOR.  
Consulte también [Más ejemplos en la página 220](#).

#### Ejemplo 1

```
FOR i FROM 1 TO 10 DO
  routine1;
ENDFOR
```

Repite el procedimiento routine1 10 veces.

#### Argumentos

```
FOR Loop counter FROM Start value TO End value [STEP Step value]
DO ... ENDFOR
```

Loop counter

##### *Identifier*

El nombre del dato que contendrá el valor del contador del bucle actual. Este dato se declara automáticamente.

Si el nombre del contador del bucle es el mismo que cualquier otro dato que exista dentro del ámbito actual, el dato existente se oculta dentro del bucle FOR y no se ve afectado de ningún modo.

Start value

Tipo de dato: Num

El valor inicial deseado para el contador del bucle. (normalmente valores enteros)

End value

Tipo de dato: Num

El valor final deseado para el contador del bucle. (normalmente valores enteros)

Step value

Tipo de dato: Num

El valor en el que debe incrementarse (o reducirse) el contador del bucle con cada bucle. (normalmente valores enteros)

Si no se especifica este valor, el valor de paso es automáticamente 1 (o -1 si el valor inicial es mayor que el valor final).

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.80 FOR - Repite un número determinado de veces

RobotWare - OS

Continuación

---

### Más ejemplos

A continuación aparecen más ejemplos de cómo usar la instrucción FOR.

#### Ejemplo 1

```
FOR i FROM 10 TO 2 STEP -2 DO
  a{i} := a{i-1};
ENDFOR
```

Los valores de una matriz se ajustan hacia arriba de modo que  $a\{10\} := a\{9\}$ ,  $a\{8\} := a\{7\}$ , etc.

---

### Ejecución de programas

- 1 Se evalúan las expresiones usadas para los valores inicial, final y de paso.
  - 2 Se asigna al contador del bucle el valor inicial.
  - 3 El valor del contador del bucle se comprueba para comprobar si se encuentra entre el valor inicial y el final o si es igual al valor inicial o final. Si el valor de un contador de bucle queda fuera de este rango, el bucle FOR se detiene y la ejecución del programa continúa en la instrucción que aparece a continuación de ENDFOR.
  - 4 Se ejecutan las instrucciones incluidas dentro del bucle FOR.
  - 5 Se incrementa (o reduce) el contador del bucle acorde con el valor de paso.
  - 6 El bucle FOR se repite, empezando por el punto 3.
- 

### Limitaciones

El contador del bucle (del tipo de dato num) sólo está disponible desde dentro del bucle FOR y por tanto supone la ocultación de otros datos y rutinas que tengan el mismo nombre. Sólo pueden leerlo (pero no actualizarlo) las instrucciones del bucle FOR.

No es posible utilizar valores decimales para los valores de inicio, final o paso en combinación con condiciones de finalización exactas para el bucle FOR (sin una definición de si se está utilizando o no la última pasada del bucle).

---

### Comentarios

Si el número de repeticiones de las operaciones depende de que una expresión determinada dé como resultado el valor TRUE, se deben utilizar instrucciones WHILE en lugar de FOR.

---

### Sintaxis

```
FOR <loop variable> FROM <expression> TO <expression>
  [ STEP <expression> ] DO
  <statement list>
ENDFOR
```

---

### Información relacionada

| Para obtener más información sobre | Consulte                                      |
|------------------------------------|---|
| Expresiones                        | Manual de referencia técnica - RAPID Overview |

---

Continúa en la página siguiente



1.80 FOR - Repite un número determinado de veces

*RobotWare - OS*

*Continuación*

| Para obtener más información sobre                  | Consulte  |
|---|---|
| Repetición siempre y cuando se cumpla una condición | <i>WHILE - Repetir siempre y cuando se cumpla una condición en la página 1094</i> |
| Identificadores                                     | <i>Manual de referencia técnica - RAPID Overview</i>                              |

# 1 Instrucciones

---

## 1.81 FricIdInit - Iniciar identificación de fricción

### *Advanced Shape Tuning*

## 1.81 FricIdInit - Iniciar identificación de fricción

---

### Utilización

`FricIdInit` marca el punto inicial de una secuencia de instrucciones de movimiento que se repetirán para calcular la fricción interna del robot.

---

### Ejemplo

Un ejemplo básico en el cual se utiliza un movimiento circular para calcular la fricción interna del robot para este movimiento:

```
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;
```

---

### Requisitos previos

El parámetro de sistema *Friction FFW On* debe tener el valor TRUE. De lo contrario, la instrucción `FricIdInit` no hará nada.

---

### Limitaciones

- `FricIdInit` sólo funciona con los robots de TCP.
  - Sólo puede ejecutarse desde tareas de movimiento.
  - El robot debe moverse en el nivel básico de trayectoria.
  - El ajuste de fricción no puede combinarse con el movimiento sincronizado. Es decir, no se permite `SyncMoveOn` entre `FricIdInit` y `FricIdEvaluate`.
  - La secuencia de movimientos para la cual se realiza el ajuste de la fricción debe comenzar y terminar en un punto fino. En caso contrario, se insertarán automáticamente puntos finos durante el proceso de ajuste.
- 

### Sintaxis

```
FricIdInit ';' ;
```

---

### Información relacionada

| Para obtener más información sobre | Consulte   |
|------------------------------------|--|
| <i>Advanced robot motion</i>       | <i>Application manual - Controller software IRC5</i> |

---

## 1.82 FricIdEvaluate - Evaluar identificación de fricción

### Utilización

`FricIdEvaluate` hace que el robot repita el movimiento entre las instrucciones `FricIdInit` y `FricIdEvaluate` a la vez que calcula la fricción para cada eje del robot.

### Ejemplo

Un ejemplo básico en el cual se utiliza un movimiento circular para calcular la fricción interna del robot para este movimiento:

```

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;

```

### Argumentos

```

FricIdEvaluate FricLevels [\MechUnit] [\BwdSpeed] [\NoPrint]
[\FricLevelMax] [\FricLevelMin] [\OptTolerance]

```

`FricLevels`

#### *Friction levels*

Tipo de dato: array of num

Cuando finaliza `FricIdEvaluate`, la matriz `FricLevels` contendrá los niveles de fricción ajustados para todos los ejes del robot. Esta matriz debe estar declarada de forma que contenga tantos elementos como ejes tenga el robot. Recuerde que es necesario llamar a la instrucción `FricIdSetFricLevels` para que estos valores tengan efecto.

`[\MechUnit]`

#### *Mechanical unit*

Tipo de dato: mecunit

El argumento `MechUnit` es opcional. Si se omite, el ajuste de fricción se realizará para la unidad mecánica representada por la variable predefinida `RAPID ROB_ID`, que es una referencia al robot con TCP de la tarea de programa actual. La compensación de la fricción sólo es posible para los robots con TCP.

`[\BwdSpeed]`

#### *Backward speed*

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.82 FricIdEvaluate - Evaluar identificación de fricción

### *Advanced Shape Tuning*

#### *Continuación*

Tipo de dato: speeddata

Después de cada iteración del proceso de ajuste, el robot retrocede siguiendo la trayectoria programada. De forma predeterminada, el movimiento de retroceso se realiza a la velocidad programada. Para agilizar el proceso, es posible utilizar el argumento opcional `BwdSpeed` para especificar una velocidad mayor durante el movimiento de retroceso. Este hecho *no* influye en el resultado de ajuste.

[\NoPrint]

Tipo de dato: switch

Si se utiliza el argumento `NoPrint`, no se escribe ningún texto en el FlexPendant acerca del progreso de las iteraciones de identificación de la fricción.

[\FricLevelMax]

*Friction level max*

Tipo de dato: num

Normalmente, el valor de fricción óptimo se determina probando con valores de entre el 1% y el 500% del valor de fricción configurado. En casos extremos esto puede generar un mensaje de error (error de velocidad de ejes). Para evitarlo, utilice el argumento `FricLevelMax` y ajústelo a un valor inferior a 500. Por ejemplo, si se cambia el valor de `FricLevelMax` a 400, se probarán valores de entre el 1% y el 400%.

Los valores permitidos son 101-500.

[\FricLevelMin]

*Friction level min*

Tipo de dato: num

Normalmente, el valor de fricción óptimo se determina probando con valores de entre el 1% y el 500% del valor de fricción configurado. Para establecer un valor inicial superior al 1%, utilice el argumento `FricLevelMin`. Por ejemplo, si `FricLevelMin` tiene el valor 80, se prueba con valores de entre el 80% y el 500%.

Los valores permitidos son 1-99.

[\OptTolerance]

*Optimization tolerance*

Tipo de dato: num

Normalmente, el valor de fricción óptimo se determina probando los valores hasta que se alcanza una pequeña tolerancia. Para agilizar el proceso, es posible aumentar este valor, si bien al incrementarlo es posible que se alcance un resultado menos exacto.

Los valores permitidos son 1-10. El valor predeterminado es 1.

*Continúa en la página siguiente*

### Gestión de errores

Pueden generarse los errores recuperables enumerados a continuación. Estos errores pueden ser gestionados en un gestor de errores. El valor de la variable de sistema `ERRNO` cambia a:

| Nombre                          | Causa del error                                    |
|---------------------------------|--|
| <code>ERR_FRICTUNE_FATAL</code> | Se produce un error durante el ajuste de fricción. |



#### Nota

Si el ajuste de fricción no tiene ningún efecto, asegúrese de que el parámetro de sistema *Friction FFW On* tenga el valor `TRUE`.

### Requisitos previos

El parámetro de sistema *Friction FFW On* debe tener el valor `TRUE`. De lo contrario, la instrucción `FricIdEvaluate` no hará nada.

### Limitaciones

- `FricIdEvaluate` sólo funciona con los robots de TCP.
- `FricIdEvaluate` sólo puede ejecutarse desde tareas de movimiento.
- El robot debe moverse en el nivel básico de trayectoria.
- En un sistema `MultiMove`, el ajuste de la fricción sólo puede hacerse en un robot cada vez. Varios robots pueden ejecutar `FricIdEvaluate` simultáneamente, pero se detendrán automáticamente y esperarán su turno mientras haya otro robot ocupado en realizar el ajuste de fricción.
- El ajuste de fricción no puede combinarse con el movimiento sincronizado. Es decir, no se permite `SyncMoveOn` entre `FricIdInit` y `FricIdEvaluate`.
- La secuencia de movimientos para la cual se realiza el ajuste de la fricción debe comenzar y terminar en un punto fino. En caso contrario, se insertarán automáticamente puntos finos durante el proceso de ajuste.

### Sintaxis

```
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 >] ';'

```

### Información relacionada

| Para obtener más información sobre | Consulte   |
|------------------------------------|--|
| <i>Advanced robot motion</i>       | <i>Application manual - Controller software IRC5</i> |

## 1 Instrucciones

---

### 1.83 FricIdSetFricLevels - Establecimiento de niveles de fricción tras la identificación de fricción *Advanced Shape Tuning*

### 1.83 FricIdSetFricLevels - Establecimiento de niveles de fricción tras la identificación de fricción

---

#### Utilización

`FricIdSetFricLevels` se utiliza para establecer el nivel de fricción para cada eje de una unidad mecánica.

---

#### Ejemplo

Un ejemplo básico en el cual se utiliza un movimiento circular para calcular la fricción interna del robot para este movimiento:

```
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;
```

#### Argumentos

```
FricIdSetFricLevels FricLevels [\MechUnit]
```

`FricLevels`

##### *Friction levels*

Tipo de dato: array of num

La matriz `FricLevels` especifica el nivel de fricción para cada eje como un porcentaje de la fricción predeterminada. Los valores deben estar en el intervalo 0-500.

`[\MechUnit]`

##### *Mechanical unit*

Tipo de dato: mecunit

El argumento `MechUnit` es opcional. Si se omite, los niveles de fricción se establecerán para la unidad mecánica representada por la variable predefinida `RAPID ROB_ID`. La compensación de la fricción sólo es posible para los robots con TCP.

---

#### Ejecución de programas

Los ajustes de los niveles de fricción permanecerán activos hasta que:

- La ejecución de programa se inicia desde el principio (PP a Main)
- Se realiza otra llamada a `FricIdSetFricLevels`

*Continúa en la página siguiente*

---

## 1.83 FricIdSetFricLevels - Establecimiento de niveles de fricción tras la identificación de fricción

*Advanced Shape Tuning*  
Continuación

- Se carga un nuevo programa
- Se reinicia el controlador con el modo de reinicio **Restablecer sistema**.

### Requisitos previos

El parámetro de sistema *Friction FFW On* debe tener el valor TRUE. De lo contrario, la instrucción `FricIdSetFricLevels` no hará nada.

### Limitaciones

- `FricIdSetFricLevels` sólo funciona con los robots de TCP.

### Sintaxis

```
FricIdSetFricLevels  
[ FricLevels ':=' ] < array {*} (IN) of num >  
['\ ' MechUnit ':=' < variable (VAR) of mecunit > ] ';' 
```

### Información relacionada

| Para obtener más información sobre | Consulte   |
|------------------------------------|--|
| <i>Advanced robot motion</i>       | <i>Application manual - Controller software IRC5</i> |

## 1 Instrucciones

---

### 1.84 GetDataVal - Obtiene el valor de un objeto de datos

RobotWare - OS

### 1.84 GetDataVal - Obtiene el valor de un objeto de datos

---

#### Utilización

GetDataVal (*Get Data Value*) permite obtener un valor de un objeto de datos que se especifica mediante una variable de cadena de caracteres.

---

#### Ejemplos básicos

Los ejemplos siguientes ilustran la instrucción GetDataVal.

##### Ejemplo 1

```
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
```

Esta sesión imprime en el FlexPendant todas las variables de tipo num cuyo nombre comience con my en el módulo mymod, junto con sus valores respectivos.

##### Ejemplo 2

```
VAR num NumArrConst_copy{2};
...
GetDataVal "NumArrConst", NumArrConst_copy;
TPWrite "Pos1 = " \Num:=NumArrConst_copy{1};
TPWrite "Pos2 = " \Num:=NumArrConst_copy{2};
```

Esta sesión imprimirá las variables num de la matriz NumArrConst.

---

#### Argumentos

```
GetDataVal Object [\Block]|[\TaskRef]|[\TaskName] Value
```

Object

Tipo de dato: string

El nombre del objeto de datos.

[ \Block ]

Tipo de dato: datapos

El bloque que contiene el objeto de datos. Sólo es posible realizar la obtención con la función GetNextSym.

Si se omite el argumento, se captura el valor del objeto de datos visible en el ámbito de ejecución actual del programa.

[ \TaskRef ]

*Task Reference*

Tipo de dato: taskid

*Continúa en la página siguiente*



La identidad de tarea de programa en la que se buscará el objeto de datos especificado. Con ayuda de este argumento, puede buscar declaraciones PERS o TASKPERS en otras tareas. Cualquier otra declaración dará lugar a un error.

Existen variables predefinidas con el tipo de dato taskId para todas las tareas de programa del sistema. La identificación de la variable será "nombre\_tarea"+"ID". Por ejemplo, para la tarea T\_ROB1 la identificación de la variable es T\_ROB1Id.

[ \TaskName ]

Tipo de dato: string

El nombre de la tarea de programa en la que se buscará el objeto de datos especificado. Con ayuda de este argumento, puede buscar declaraciones PERS o TASKPERS en otras tareas. Cualquier otra declaración dará lugar a un error.

Value

Tipo de dato: anytype

La variable en la que se almacena el valor obtenido. Su tipo de dato debe ser el mismo que el del objeto de datos a buscar. El valor obtenido puede capturarse de una constante, una variable o una variable persistente, pero debe almacenarse en una variable.

### Gestión de errores

Pueden generarse los errores recuperables enumerados a continuación. Estos errores pueden ser gestionados en un gestor de errores. El valor de la variable de sistema ERRNO cambia a:

| Nombre          | Causa del error   |
|-----------------|---|
| ERR_SYM_ACCESS  | <ul style="list-style-type: none"> <li>El objeto de datos no existe.</li> <li>El objeto de datos es un dato o un parámetro de rutina y no está situado en la rutina activa actualmente.</li> <li>Buscar en otras tareas declaraciones distintas de PERS o TASK PERS.</li> </ul> |
| ERR_INVDIM      | El objeto de datos y la variable utilizados en el argumento Value tienen dimensiones diferentes.  |
| ERR_SYMBOL_TYPE | El objeto de datos y la variable utilizados en el argumento Value tienen tipos diferentes. Si se utilizan tipos de datos ALIAS, también se produce este ERROR, aunque los tipos tengan el mismo tipo de dato básico.  |
| ERR_TASKNAME    | Si el nombre de tarea de programa contenido en \TaskName no se encuentra en el sistema, la variable de sistema ERRNO cambia a ERR_TASKNAME.   |

Con ayuda de los argumentos TaskRef o TaskName, puede buscar declaraciones PERS o TASK PERS en otras tareas. Cualquier otra declaración da lugar a un error y la variable de sistema ERRNO cambia a ERR\_SYM\_ACCESS. La búsqueda de una PERS declarada como LOCAL en otras tareas también da lugar a un error y al cambio de la variable ERRNO al valor ERR\_SYM\_ACCESS.

Continúa en la página siguiente

# 1 Instrucciones

## 1.84 GetDataVal - Obtiene el valor de un objeto de datos

RobotWare - OS

Continuación

### Limitaciones

Para un tipo de datos de semivalor, no es posible buscar el tipo de datos de valor asociado. Por ejemplo, si busca `dionum`, no se obtendrá ninguna coincidencia para señales `signaldi` y si busca `num`, no se obtendrá ninguna coincidencia para las señales `signalgi` ni `signalai`.

No es posible obtener el valor de una variable declarada como `LOCAL` en un módulo de `RAPID` incorporado.

### Sintaxis

```
GetDataVal
[ 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>';'
```

### Información relacionada

| Para obtener más información sobre                              | Consulte  |
|---|---|
| Definición de un conjunto de símbolos en una sesión de búsqueda | <a href="#">SetDataSearch - Definir el conjunto de símbolos de una secuencia de búsqueda en la página 696</a>             |
| Obtención del siguiente símbolo coincidente                     | <a href="#">GetNextSym - Obtiene el siguiente símbolo coincidente en la página 1309</a>                                   |
| Asignación del valor de un objeto de datos                      | <a href="#">SetDataVal - Establece el valor de un objeto de datos en la página 701</a>                                    |
| Asignación del valor de varios objetos de datos                 | <a href="#">SetAllDataVal - Establece un valor en todos los objetos de datos de un conjunto definido en la página 691</a> |
| El tipo de datos relacionado <code>datapos</code>               | <a href="#">datapos - Inclusión de un bloque para un objeto de datos en la página 1657</a>                                |
| <i>Advanced RAPID</i>   | <i>Application manual - Controller software IRC5</i>  |

## 1.85 GetGroupSignalInfo - Leer información sobre una señal digital de grupo *RobotWare - OS*

### 1.85 GetGroupSignalInfo - Leer información sobre una señal digital de grupo

#### Utilización

`GetGroupSignalInfo` se utiliza para leer información sobre una señal digital de grupo desde la E/S.

#### Ejemplos básicos

El ejemplo siguiente ilustra la instrucción `GetGroupSignalInfo`.

#### Ejemplo 1

```
VAR dnum DValue;  
VAR string label;  
VAR string devicename;  
  
GetGroupSignalInfo go1 \MaxDValue:=DValue \Label:=label  
                    \DeviceName:=devicename;
```

Lea el valor máximo de la señal que puede establecerse en la señal `go1`, la etiqueta de identificación de señal y el nombre de dispositivo al que se asigna la señal.

#### Argumentos

```
GetGroupSignalInfo Signal [\MaxDValue] [\Label] [\DeviceName]
```

Signal

Tipo de dato: `signalxx`

El identificador de señal de acuerdo con el programa (tipo de datos `signalgo` o `signalgi`) sobre el que obtener información.

\MaxDValue

Tipo de dato: `dnum`

`MaxDValue` es el valor máximo que puede establecerse a la señal. Se basa en el mapa de dispositivo de la señal.

\Label

Tipo de dato: `string`

Label es la etiqueta de identificación de señal especificada.

\DeviceName

Tipo de dato: `string`

`DeviceName` es el dispositivo al que se asigna la señal.

#### Ejecución de programas

La instrucción lee información sobre una señal digital de grupo desde la E/S.

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.85 GetGroupSignalInfo - Leer información sobre una señal digital de grupo

RobotWare - OS

Continuación

### Gestión de errores

Se generan los siguientes errores recuperables que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` se establecerá en:

| Nombre                          | Causa del error   |
|---------------------------------|---|
| <code>ERR_NO_ALIASIO_DEF</code> | La variable de señal es una variable declarada en RAPID. No se ha conectado a una señal E/S definida en la configuración de E/S con la instrucción <code>AliasIO</code> . |

### Sintaxis

```
GetGroupSignalInfo
[ Signal ':= ' ] < variable (VAR) of anytype >
[ '\ ' MaxDvalue ':= ' < variable (VAR) of dnum >]
[ '\ ' Label ':= ' < variable (VAR) of string >]
[ '\ ' DeviceName ':= ' < variable (VAR) of string >] ';'

```

### Información relacionada

| Para obtener más información sobre                 | Consulte   |
|--|--|
| Lectura de un atributo de un parámetro del sistema | <a href="#">ReadCfgData - Lee un atributo de un parámetro del sistema en la página 582</a> |
| Configuración de E/S                               | <i>Manual de referencia técnica - Parámetros del sistema</i>                               |

## 1.86 GetJointData: Obtener datos conjuntos específicos

### Utilización

GetJointData se utiliza para leer datos conjuntos específicos de una unidad mecánica especificada. La información que se puede leer respecto al eje especificado es la posición, la velocidad, el par y el par externo estimado.

### Ejemplos básicos

El ejemplo siguiente ilustra la instrucción GetJointData.

#### Ejemplo 1

```
VAR num position;
VAR num speed;
VAR num torque;
VAR num exttorque;
...
GetJointData \MechUnit:=ROB_1, 1 \Position:=position \Speed:=speed
\Torque:=torque \ExtTorque:=exttorque;
```

Se lee la posición actual, la velocidad, el par y el par externo estimado del primer eje de ROB\_1.

### Argumentos

```
GetJointData [\MechUnit] Axis [\Position] [\Speed] [\Torque]
[\ExtTorque]
```

[\MechUnit]

#### *Mechanical Unit*

Tipo de dato: mecunit

El nombre de la unidad mecánica cuyos valores de eje se desea comprobar. Si se omite este argumento, se obtiene el valor de un eje del robot conectado.

Axis

Tipo de dato: num

El número del eje cuyo valor se desea obtener (de 1 a 6).

[\Position]

Tipo de dato: num

La posición actual del eje especificado del robot o el eje externo del lado del brazo. El valor de los ejes rotatorios se expresa en grados y el de los ejes lineales, en milímetros.

Se debe utilizar al menos uno de los parámetros opcionales \Position, \Speed, \Torque o \ExtTorque.

[\Speed]

Tipo de dato: num

La velocidad actual del eje especificado del robot o el eje externo en el lado del brazo. El valor de los ejes rotatorios se expresa en grados/segundo y el de los ejes lineales, en milímetros/segundo.

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.86 GetJointData: Obtener datos conjuntos específicos

RobotWare - OS

Continuación

Se debe utilizar al menos uno de los parámetros opcionales `\Position`, `\Speed`, `\Torque` o `\ExtTorque`.

[`\Torque`]

Tipo de dato: num

El par actual en Nm del eje especificado del robot o del eje externo en el lado del brazo.

Se debe utilizar al menos uno de los parámetros opcionales `\Position`, `\Speed`, `\Torque` o `\ExtTorque`.

[`\ExtTorque`]

Tipo de dato: num

El par externo estimado actual en Nm del eje especificado del robot o del eje externo en el lado del brazo.

Se debe utilizar al menos uno de los parámetros opcionales `\Position`, `\Speed`, `\Torque` o `\ExtTorque`.

---

### Ejecución de programas

La instrucción lee la posición, la velocidad el par y el par externo estimado del robot y de los ejes externos.

Los valores de lectura también se pueden ver cuando se utiliza *TuneMaster* con los números de señal de prueba 4000, 4001, 4002 y 4003.

---

### Gestión de errores

Se generan los siguientes errores recuperables que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` se establecerá en:

|                           |  |
|---------------------------|--|
| <code>ERR_AXIS_PAR</code> | Parámetro de eje incorrecto en la instrucción. |
| <code>ERR_AXIS_ACT</code> | El eje no está activado.                       |

---

### Sintaxis

```
GetJointData
  [ '\ MechUnit :=' < variable (VAR) of mecunit> ',' ]
  [ Axis :=' ] < expression (IN) of num>
  [ '\ Position :=' < variable (VAR) of num> ]
  [ '\ Speed :=' < variable (VAR) of num> ]
  [ '\ Torque :=' < variable (VAR) of num> ]
  [ '\ ExtTorque :=' < variable (VAR) of num> ] ';' ;
```

## 1.87 GetSysData - Obtiene datos del sistema

### Utilización

GetSysData captura el valor y, opcionalmente, el nombre de símbolo de un dato de sistema actual del tipo de dato especificado.

Esta instrucción permite capturar datos sobre la herramienta, el objeto de trabajo, la carga útil o la carga total del robot, además de su nombre, en la tarea de movimiento actual o la conectada, o bien en cualquier tarea de movimiento que tenga un nombre.

### Ejemplos básicos

Los ejemplos siguientes ilustran la instrucción GetSysData.

#### Ejemplo 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;
```

Copia el valor actual del dato de la herramienta a la variable persistente curtoolvalue.

#### Ejemplo 2

```
GetSysData curtoolvalue \ObjectName := curtoolname;
```

También copia el nombre de la herramienta activa a la curtoolname.

#### Ejemplo 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
```

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.87 GetSysData - Obtiene datos del sistema

RobotWare - OS

Continuación

Si `ModalPayLoadMode` es 1, copia la carga útil activa actualmente y su nombre a la variable `name`.

Si `ModalPayLoadMode` es 0, copia la carga total actual y su nombre a la variable `name`.

---

### Argumentos

`GetSysData [ \TaskRef ] [ \TaskName ] DestObject [ \ObjectName ]`

[ \TaskRef ]

#### *Task Reference*

Tipo de dato: `taskid`

La identidad de tarea de programa desde la cual deben leerse los datos del sistema activo actual.

Existen variables predefinidas con el tipo de dato `taskid` para todas las tareas de programa del sistema. La identificación de la variable será "nombre\_tarea"+"ID". Por ejemplo, para la tarea `T_ROB1` la identificación de la tarea es `T_ROB1Id`.

[ \TaskName ]

Tipo de dato: `string`

El nombre de tarea de programa desde la cual deben leerse los datos del sistema activo actual.

Si no se especifica ninguno de los argumentos, ni `\TaskRef` ni `\TaskName`, se usa la tarea actual.

DestObject

Tipo de dato: `anytype`

La variable persistente para el almacenamiento del valor actual del dato de sistema.

El tipo de dato de este argumento también especifica el tipo de dato de sistema (herramienta, objeto de trabajo, carga útil o carga total) que se desea capturar. Si se utiliza el argumento opcional `TLoad` en las instrucciones de movimiento, se captura la carga total en lugar de la carga útil si se utiliza el tipo de datos `loaddata`.

| Tipo de dato          | Tipo de dato de sistema |
|-----------------------|-------------------------|
| <code>tooldata</code> | Herramienta             |
| <code>wobjdata</code> | Objeto de trabajo       |
| <code>loaddata</code> | Carga útil/carga total  |

No es posible usar una matriz ni un componente de registro.

[ \ObjectName ]

Tipo de dato: `string`

Argumento opcional (variable o persistente) para capturar también el nombre del dato de sistema activo.

---

### Ejecución de programas

Cuando se ejecuta la instrucción `GetSysData`, el valor actual del dato se almacena en la variable persistente especificada en el argumento `DestObject`.

*Continúa en la página siguiente*




Si se utiliza el argumento `\ObjectName`, el nombre del dato actual se almacena en la variable o la variable persistente especificada en el argumento `ObjectName`.

El dato de sistema actual para la herramienta, el objeto de trabajo o la carga total se activa mediante la ejecución de cualquier instrucción de movimiento. La carga útil se activa mediante la ejecución de la instrucción `GripLoad`.

### Gestión de errores

Pueden generarse los errores recuperables enumerados a continuación. Estos errores pueden ser gestionados en un gestor de errores. El valor de la variable de sistema `ERRNO` cambia a:

| Nombre           | Causa del error  |
|------------------|--|
| ERR_NOT_MOVETASK | <p>Los argumentos <code>\TaskRef</code> o <code>\TaskName</code> especifican una tarea sin movimiento.</p> <p> <b>Nota</b></p> <p>No se genera ningún error si los argumentos <code>\TaskRef</code> o <code>\TaskName</code> especifican la tarea sin movimiento que ejecuta esta función <code>GetSysData</code> (hace referencia a la propia tarea sin movimiento). Los datos del sistema actual se capturan de la tarea de movimiento conectada.</p> |
| ERR_TASKNAME     | Si el nombre de tarea de programa contenido en <code>\TaskName</code> no se encuentra en el sistema, la variable de sistema <code>ERRNO</code> cambia a <code>ERR_TASKNAME</code> .  |

### Sintaxis

```
GetSysData
['\' TaskRef' :=] <variable (VAR) of taskid>]
[['\' TaskName' :=] <expression (IN) of string>]
[ DestObject' :=] <persistent(PERS) of anytype>
['\'ObjectName' :=] <variable or persistent (INOUT) of string>];'
```

### Información relacionada

| Para obtener más información sobre   | Consulte   |
|--|--|
| Definición de herramientas   | <a href="#">tooldata - Datos de herramienta en la página 1795</a>                        |
| Definición de objetos de trabajo   | <a href="#">wobjdata - Datos del objeto de trabajo en la página 1823</a>                 |
| Definición de una carga útil   | <a href="#">loaddata - Datos de carga en la página 1694</a>                              |
| Establecimiento de datos del sistema   | <a href="#">SetSysData - Establece datos del sistema en la página 714</a>                |
| Parámetro de sistema <code>ModalPayloadMode</code> para la activación y la desactivación de la carga útil. | <a href="#">Manual de referencia técnica - Parámetros del sistema</a>                    |
| Ejemplo de cómo usar <code>TLoad</code> , carga total.   | <a href="#">MoveL - Mueve el robot siguiendo una trayectoria lineal en la página 447</a> |

# 1 Instrucciones

---

## 1.88 GetTorqueMargin - Lee el menor margen de par RobotWare - OS

## 1.88 GetTorqueMargin - Lee el menor margen de par

---

### Utilización

GetTorqueMargin se utiliza para leer el menor margen de par desde que se ejecutó ResetTorqueMargin y puede usarse en el robot y los ejes externos.

---

### Ejemplos básicos

Los siguientes ejemplos ilustran la instrucción GetTorqueMargin.

#### Ejemplo 1

```
VAR num torque_margin;  
ResetTorqueMargin \AxisNo:=5;  
! Insert Program Here  
! ...  
! ...  
torque_margin := GetTorqueMargin (5);
```

El menor margen de par, desde la última ejecución de ResetTorqueMargin 5, del eje 5 del robot está almacenado en torque\_margin.

#### Ejemplo 2

```
VAR num torque_margin1;  
VAR num torque_margin2;  
VAR num torque_margin3;  
  
ResetTorqueMargin \AxisNo:=5;  
! Insert Program Here  
! ...  
! ...  
torque_margin1 := GetTorqueMargin (5);  
ResetTorqueMargin 5;  
! Change arm config  
! Insert Program Here  
! ...  
! ...  
torque_margin2 := GetTorqueMargin (5);  
ResetTorqueMargin 5;  
! Change arm config  
! Insert Program Here  
! ...  
! ...  
torque_margin3 := GetTorqueMargin (5);  
  
! compare torque_margin1, torque_margin2, torque_margin3 etc
```

---

### Valor de retorno

**Tipo de dato:** num

El menor margen de par en % de par disponible desde que se ejecutó ResetTorqueMargin.

*Continúa en la página siguiente*

### Argumentos

GetTorqueMargin [ \MecUnit ] AxisNo

[ \MecUnit ]

Tipo de dato: mecunit

El nombre de la unidad mecánica cuyos valores de eje se desea comprobar. Si se omite este argumento, se obtiene el valor de un eje del robot conectado.

AxisNo

Tipo de dato: num

El número del eje cuyo valor se desea obtener (de 1 a 6).

### Ejecución de programas

La función leer el menor par motor desde el último `ResetTorqueMargin`.

El margen de par motor también puede leerse a través del número de señal de prueba 4040. (Esta señal siempre muestra el margen de par actual).

### Limitaciones

Cuando fuerzas externas afectan al robot, debe prestarse atención para no forzar la estructura excesivamente. Este método solo lee el valor de par en los motores.

### Gestión de errores

Se generan los siguientes errores recuperables que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` se establecerá en:

|              |  |
|--------------|--|
| ERR_AXIS_PAR | Parámetro de eje incorrecto en la función. |
|--------------|--|

### Sintaxis

```
GetTorqueMargin '('
  [ '\ ' MechUnit ' := ' ] < variable (VAR) of mecunit > ', '
  [ '\ ' AxisNo ' := ' < expression (IN) of num > ] ')'
```

Una función con un valor de retorno del tipo de dato num.

### Información relacionada

| Para obtener más información sobre | Consulte  |
|------------------------------------|---|
| Restablecer margen de par          | <a href="#">ResetTorqueMargin - Restablecer el menor margen de par en la página 614</a> |

# 1 Instrucciones

---

## 1.89 GetTrapData - Obtiene datos de interrupción para la rutina TRAP actual RobotWare - OS

### 1.89 GetTrapData - Obtiene datos de interrupción para la rutina TRAP actual

---

#### Utilización

`GetTrapData` se utiliza en rutinas TRAP para obtener todos los datos sobre la interrupción que causó la ejecución de la rutina TRAP.

Debe utilizarse en las rutinas TRAP generadas por la instrucción `IError`, antes del uso de la instrucción `ReadErrData`.

---

#### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción `GetTrapData`:

Consulte también [Más ejemplos en la página 240](#).

#### Ejemplo 1

```
VAR trapdata err_data;  
GetTrapData err_data;
```

La información de las interrupciones se almacenan en la variable sin valor `err_data`.

---

#### Argumentos

```
GetTrapData TrapEvent
```

#### TrapEvent

Tipo de dato: `trapdata`

La variable en la que se desea almacenar la información de qué hecho provocó la ejecución de la rutina TRAP.

---

#### Limitación

Esta instrucción sólo puede usarse desde dentro de una rutina TRAP.

---

#### Más ejemplos

A continuación aparecen más ejemplos de la instrucción `GetTrapData`.

#### Ejemplo 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
```

Cuando se detecta un error con la rutina TRAP `trap_err`, el dominio, el número y el tipo de error se almacenan en las variables adecuadas sin valor, del tipo `trapdata`.

---

*Continúa en la página siguiente*

## 1.89 GetTrapData - Obtiene datos de interrupción para la rutina TRAP actual

RobotWare - OS

Continuación

### Sintaxis

```
GetTrapData
  [TrapEvent ':=' ] <variable (VAR) of trapdata>;'
```

### Información relacionada

| Para obtener más información sobre                 | Consulte  |
|--|---|
| Resumen de interrupciones                          | <i>Manual de referencia técnica - RAPID Overview</i>  |
| Más información sobre la gestión de interrupciones | <i>Manual de referencia técnica - RAPID Overview</i>  |
| Datos de interrupción para la rutina TRAP actual   | <a href="#">trapdata - Datos de interrupción para la rutina TRAP actual en la página 1803</a> |
| Solicitud de una interrupción para errores         | <a href="#">IError - Solicita una interrupción para errores en la página 257</a>              |
| Obtención de información sobre un error            | <a href="#">ReadErrData - Obtiene información sobre un error en la página 586</a>             |
| <i>Advanced RAPID</i>                              | <i>Application manual - Controller software IRC5</i>  |

# 1 Instrucciones

---

## 1.90 GOTO - Salta a otra instrucción

RobotWare - OS

## 1.90 GOTO - Salta a otra instrucción

---

### Utilización

GOTO se utiliza para transferir la ejecución del programa a otra línea (una etiqueta) creada dentro de la misma rutina.

---

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción GOTO:

#### Ejemplo 1

```
GOTO next;  
...  
next:
```

La ejecución del programa continúa en la instrucción que sigue a la etiqueta "next".

#### Ejemplo 2

```
reg1 := 1;  
next:  
...  
reg1 := reg1 + 1;  
IF reg1<=5 GOTO next;
```

La ejecución se transfiere cuatro veces a next (for reg1= 2, 3, 4, 5).

#### Ejemplo 3

```
IF reg1>100 THEN  
  GOTO highvalue  
ELSE  
  GOTO lowvalue  
ENDIF  
lowvalue:  
...  
GOTO ready;  
highvalue:  
...  
ready:
```

Si reg1 es mayor que 100, la ejecución se transfiere a la etiqueta highvalue. De lo contrario, la ejecución se transfiere a la etiqueta lowvalue.

---

### Argumentos

GOTO Label

Label

#### *Identifier*

La etiqueta que indica dónde debe continuar la ejecución del programa.

---

### Limitaciones

Sólo es posible transferir la ejecución del programa a una etiqueta que se encuentra dentro de la misma rutina.

---

*Continúa en la página siguiente*

Sólo es posible transferir la ejecución del programa a una etiqueta que se encuentre dentro de una instrucción `IF` o `TEST` si la instrucción `GOTO` se encuentra también dentro de la misma bifurcación de la instrucción.

Sólo es posible transferir la ejecución del programa a una etiqueta que se encuentre dentro de una instrucción `FOR` o `WHILE` si la instrucción `GOTO` se encuentra también dentro de la instrucción.

## Sintaxis

```
GOTO <identifier>' ;'
```

## Información relacionada

| Para obtener más información sobre                      | Consulte  |
|---|---|
| Adhesivo  | <a href="#">Etiqueta - Nombre de línea en la página 346</a> |
| Otras instrucciones que modifican el flujo del programa | <i>Manual de referencia técnica - RAPID Overview</i>        |

# 1 Instrucciones

---

## 1.91 GripLoad - Define la carga útil de un robot RobotWare - OS

### 1.91 GripLoad - Define la carga útil de un robot

---

#### Utilización

GripLoad se utiliza para definir la carga útil que se sostiene con la pinza del robot.

---

#### Descripción

GripLoad especifica la carga que lleva el robot. La carga especificada la utiliza el sistema de control de forma que permita controlar de la mejor manera posible los movimientos del robot.

La carga útil se conecta y desconecta mediante la instrucción GripLoad, lo que suma o resta el peso de la carga útil al peso de la pinza.



#### ¡AVISO!

Es importante definir siempre la carga real de la herramienta y, si se usa, la carga útil del robot (por ejemplo, una pieza sujeta por una pinza). Una definición incorrecta de los datos de carga puede dar lugar a la sobrecarga de la estructura mecánica del robot. Existe también el riesgo de que pueda superarse la velocidad en el modo manual a velocidad reducida.

Cuando se especifican datos de carga incorrectos, este hecho suele tener las consecuencias siguientes:

- El robot no puede funcionar a su capacidad máxima.
- Peor exactitud de la trayectoria, con riesgo de sobrepasar posiciones.
- Riesgo de sobrecarga de la estructura mecánica.

El controlador monitoriza continuamente la carga y escribe un registro de eventos si la carga es más elevada que la prevista. Este registro de eventos se guarda y registra en la memoria del controlador.

---

#### Ejemplos básicos

Los siguientes ejemplos ilustran la instrucción GripLoad.

##### Ejemplo 1

```
Set doGripper;  
!wait to grip  
WaitTime 0.3;  
GripLoad piece1;
```

Conexión de la carga útil, piece1, especificada en el momento en que el robot sujeta la carga.

##### Ejemplo 2

```
Reset doGripper;  
!wait to release  
WaitTime 0.3;  
GripLoad load0;
```

Desconexión de la carga útil, especificada en el momento en que el robot suelta una carga útil.

---

*Continúa en la página siguiente*



### Argumentos

GripLoad Load

Load

Tipo de dato: loaddata

El dato de carga que describe la carga útil actual.

Es posible realizar una ejecución de prueba del programa sin ninguna carga útil utilizando una señal de entrada digital conectada a la entrada de sistema *SimMode* (modo simulado). Si la señal digital tiene el valor 1, el loaddata de la instrucción GripLoad no se considera y sólo se utiliza el loaddata del tooldata actual.

### Ejecución de programas

La carga especificada se aplica a la siguiente instrucción de movimiento ejecutada y es válida hasta que se ejecute una nueva instrucción GripLoad.

La carga especificada afecta al rendimiento del robot.

La carga predeterminada, (load0), 0 kg, se establece automáticamente.

- cuando se utiliza el modo de reinicio **Restablecer RAPID**
- al cargar un nuevo programa o un nuevo módulo
- al iniciar la ejecución del programa desde el principio
- al mover el puntero del programa a *main*
- al mover el puntero del programa a una rutina
- al mover el puntero de programa de una forma que se pierde el orden de la ejecución.

Se actualiza la carga útil para la unidad mecánica controlada desde la tarea de programa actual. Si GripLoad se utiliza desde una tarea sin movimiento, la carga útil se actualiza para la unidad mecánica controlada por la tarea de movimiento conectada.

### Sintaxis

GripLoad

```
[Load ':=' ] <persistent (PERS) of loaddata>';'
```

### Información relacionada

| Para obtener más información sobre   | Consulte  |
|--|---|
| Identificación de la carga de la herramienta, carga útil o carga de brazo                          | <i>Manual del operador - IRC5 con FlexPendant</i> , sección <i>Programación y testing - Rutinas de servicio</i> |
| Definición de una carga útil para unidades mecánicas   | <a href="#">MechUnitLoad - Define una carga útil para una unidad mecánica en la página 372</a>                  |
| Definición de datos de carga   | <a href="#">loaddata - Datos de carga en la página 1694</a>   |
| Señal de entrada de sistema <i>SimMode</i> para mover el robot en el modo simulado sin carga útil. | <i>Manual de referencia técnica - Parámetros del sistema</i>  |

# 1 Instrucciones

---

## 1.92 HollowWristReset - Restablecer la muñeca hueca

RobotWare - OS

## 1.92 HollowWristReset - Restablecer la muñeca hueca

---

### Utilización

`HollowWristReset` (*Reset hollow wrist*) restablece la posición de las articulaciones de muñeca de los manipuladores de muñeca hueca, como los modelos IRB 5402 y IRB 5403.

Esta instrucción hace posible evitar el rebobinado de los ejes 4 y 5 de la muñeca después de que hayan descrito una o varias revoluciones en un sentido. Después de ejecutar una instrucción `HollowWristReset`, los ejes de la muñeca pueden continuar el giro en el mismo sentido.

### Descripción

`HollowWristReset` facilita la creación de programas de aplicación. Usted no tiene por qué asegurarse de que la posición de la muñeca esté dentro de  $\pm 2$  revoluciones en el momento de la programación. Además, puede ahorrar tiempo de ciclo porque el robot no tiene que perder tiempo rebobinando la muñeca. Existe una limitación de  $\pm 144$  revoluciones para el avance de los ejes 4 y 5 antes de que la posición de la muñeca se restablezca con `HollowWristReset`. El programador del robot debe ser consciente de esta limitación y tenerla en cuenta al planificar los programas del robot. Para garantizar que no se supere el límite de 144 revoluciones después de ejecutar varias veces un programa de *giro de muñeca*, debe esperar siempre a que el robot se haya detenido completamente y restablecer la posición absoluta en cada programa (o cada ciclo/rutina/módulo, etc., según sus necesidades). Recuerde que todos los ejes deben permanecer parados durante la ejecución de la instrucción `HollowWristReset`. Siempre que se tengan en cuenta estas limitaciones, los ejes 4 y 5 pueden girar indefinidamente y de forma independiente del eje 6 durante la ejecución del programa.

Utilice `HollowWristReset` en lugar de `IndReset` para restablecer la muñeca hueca, dado que esta instrucción conserva los límites del eje 6, impidiendo así que se produzca una torsión excesiva de los tubos o cables de las aplicaciones de pintura.

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción `HollowWristReset`:

#### Ejemplo 1

```
MoveL p10,v800,fine,paintgun1\WObj:=workobject1;  
HollowWristReset;
```

Todos los ejes activos se detienen con un punto de parada y la muñeca se restablece.

### Limitaciones

Todos los ejes activos deben permanecer parados mientras se ejecuta la instrucción `HollowWristReset`.

Los ejes de la muñeca deben ser restablecidos antes de que cualquiera de ellos alcance el límite de  $\pm 144$  revoluciones (es decir, 51 840 grados/904 rad).

*Continúa en la página siguiente*

Siempre que tiene lugar un paro de programa, un paro de emergencia o un paro de caída de alimentación, el controlador conserva el contexto de la trayectoria para poder volver a ella y permitir que el robot prosiga con la ejecución del programa en el punto en el que se interrumpió la trayectoria. En el modo manual, si se ha sacado al manipulador de la trayectoria entre un paro y el reinicio, se informa de este hecho al operador con el mensaje siguiente del FlexPendant: **¡Fuera de trayectoria! El robot ha sido movido tras el paro de programa. ¿Desea que el robot vuelva a la trayectoria en el momento del inicio? Sí/No/Cancelar.** De esta forma, podrá volver a la trayectoria antes del reinicio. En el modo automático, el robot vuelve automáticamente a la trayectoria.

HollowWristReset elimina el contexto de la trayectoria. Esto significa que no es posible volver a la trayectoria en caso de un reinicio de programa, si en el intervalo se ha ejecutado la instrucción HollowWristReset. Si se ejecuta esta instrucción manualmente, sólo debe ejecutarse en situaciones en las que no sea necesario volver a la trayectoria. Es decir, debe usarse una vez que un programa se haya ejecutado completamente o que una instrucción haya finalizado completamente la ejecución paso a paso sin que el manipulador haya sido sacado de su trayectoria con movimientos manuales, etc.

### Sintaxis

```
HollowWristReset ' ; '
```

### Información relacionada

| Para obtener más información sobre  | Consulte   |
|-------------------------------------|--|
| Parámetros del sistema relacionados | <i>Manual de referencia técnica - Parámetros del sistema</i> |
| Regreso a la trayectoria            | <i>Manual de referencia técnica - RAPID Overview</i>         |

# 1 Instrucciones

---

## 1.93 ICap - Conectar eventos CAP a rutinas TRAP *Continuous Application Platform (CAP)*

### 1.93 ICap - Conectar eventos CAP a rutinas TRAP

---

#### Utilización

ICap se utiliza para conectar un número de interrupción (que ya está conectado a una rutina TRAP) con un evento CAP específico, consulte los Argumentos que aparecen a continuación para ver una lista de los Eventos disponibles. Al utilizar ICap, se crea una asociación entre un evento de proceso específico y una rutina TRAP definida por el usuario. En otras palabras, la rutina TRAP en cuestión se ejecuta cuando se produce el evento CAP asociado.

Se recomienda colocar las rutinas TRAP en una tarea en segundo plano.

---

#### Ejemplo básico

A continuación se muestra un ejemplo en el que el evento CAP\_START de CAP está asociado con la rutina TRAP `start_trap`.

```
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
```

---

#### Argumentos

ICap Interrupt Event

#### Interrupt

**Tipo de dato:** intnum

La identidad de la interrupción. La interrupción debe estar ya conectada a una rutina TRAP mediante la instrucción `CONNECT`.

#### Event

**Tipo de dato:** num

El número de evento CAP que se debe asociar con la interrupción. Estos eventos son constantes predefinidas.

---

*Continúa en la página siguiente*

## Eventos CAP disponibles

Consulte la sección *Acoplamiento entre fases y eventos en Application manual - Continuous Application Platform*.

| Eventos           | Fase  | Número de evento | Descripción  |
|-------------------|-------|------------------|--|
| CAP_START         |       | 0                | Este evento se produce en cuanto se inicia el proceso CAP.   |
| START_PRE         | PRE   | 1                | Este evento se produce cuando está activada la supervisión de la fase PRE, si existe. Si se utiliza un <i>flying start</i> , no se distribuye ningún evento porque ya hay un movimiento de TCP. En el inicio se distribuye este evento.  |
| PRE_STARTED       | PRE   | 2                | Este evento se produce cuando todos los requisitos de la lista de supervisión PRE se cumplen, es decir cuando se inicia la fase PRE_START. Si se utiliza un <i>flying start</i> , no se distribuye ningún evento porque ya hay un movimiento de TCP. En el inicio se distribuye este evento. |
| START_MAIN        | START | 3                | Este evento se produce cuando finaliza la fase PRE_START y se inicia la fase MAIN.   |
| MAIN_STARTED      | START | 4                | Este evento se produce cuando se cumplen todas las condiciones de la lista de supervisión START, es decir, cuando se inicia la fase MAIN.  |
| STOP_WEAVESTART   | MAIN  | 5                | Este evento se produce antes del inicio de la oscilación, pero solo si se ha ordenado el inicio de la oscilación. Si se utiliza un <i>flying start</i> , no se distribuye ningún evento porque ya hay un movimiento de TCP. En el reinicio se distribuye este evento.                        |
| WEAVESTART_REGAIN | MAIN  | 6                | Este evento se produce cuando el robot ha recuperado la trayectoria después de un inicio de oscilación. Si se utiliza un <i>flying start</i> , no se distribuye ningún evento porque ya hay un movimiento de TCP. En el inicio se distribuye este evento.                                    |
| MOTION_DELAY      | MAIN  | 7                | Este evento se produce tras el retardo, si existe, de un inicio de movimiento. Si se utiliza un <i>flying start</i> , no se distribuye ningún evento porque ya hay un movimiento de TCP. En el reinicio se distribuye este evento.   |
| STARTSPEED_TIME   | MAIN  | 8                | Este evento se produce cuando el tiempo de utilizar <i>Start Speed</i> se agota y llega el momento de cambiar a los datos de movimiento principal.   |
| MAIN_MOTION       | MAIN  | 9                | Este evento se produce cuando se activa el movimiento principal con la ejecución del proceso.  |

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.93 ICap - Conectar eventos CAP a rutinas TRAP

### Continuous Application Platform (CAP)

#### Continuación

| Eventos           | Fase      | Número de evento | Descripción  |
|-------------------|-----------|------------------|--|
| MOVE_STARTED      | MAIN      | 10               | Este evento se produce en cuanto el robot inicia el movimiento a lo largo de la trayectoria del proceso. Si se utiliza un <i>flying start</i> , no se distribuye ningún evento porque ya hay un movimiento de TCP. En el reinicio se distribuye este evento. |
| RESTART           | MAIN      | 11               | Este evento se produce cuando se ordena el reinicio.   |
| NEW_INSTR         | MAIN      | 12               | Este evento se produce cuando se captura una nueva instrucción CapL o CapC del programa RAPID.   |
| AT_POINT          | MAIN      | 13               | Este evento se produce en todos los robtarget de la trayectoria del proceso excepto en el punto de inicio y el punto de final.   |
| AT_RESTARTPOINT   | MAIN      | 14               | Este evento se produce cuando el robot vuelve a moverse la distancia de reinicio en la trayectoria de proceso tras una parada.   |
| LAST_SEGMENT      | MAIN      | 15               | Este evento se produce en el punto de inicio del último segmento.  |
| PROCESS_END_POINT | MAIN      | 16               | Este evento se produce cuando el robot alcanza el punto final del proceso, es decir, donde se supone que termina el proceso. Si se utiliza un <i>flying end</i> , no se distribuye ningún evento.  |
| END_MAIN          | END_MAIN  | 17               | Este evento se produce en el punto de la trayectoria de proceso en el que se inicia la supervisión de la secuencia final, es decir, donde el robot alcanza el punto final del proceso.   |
| MAIN_ENDED        | END_MAIN  | 18               | Este evento se produce cuando se cumplen todas las condiciones de la lista de supervisión END_MAIN, es decir, cuando el proceso principal se considera finalizado.   |
| PATH_END_POINT    |           | 19               | Este evento se produce cuando el robot alcanza el punto final de la trayectoria, es decir, el punto fino o la mitad de la zona (para <i>flying end</i> ) en la última instrucción CAP.   |
| PROCESS_ENDED     |           | 20               | Este evento solo se produce cuando el proceso finaliza en el punto fino o en la mitad de la zona (para <i>flying end</i> ) en la última instrucción CAP.   |
| END_POST1         | END_POST1 | 21               | Este evento se produce cuando es el momento de finalizar la fase POST1, es decir, cuando es el momento de cambiar de la fase POST1 a la fase POST2. Si se utiliza un <i>flying end</i> , no se distribuye ningún evento.                                     |

Continúa en la página siguiente

## 1.93 ICap - Conectar eventos CAP a rutinas TRAP Continuous Application Platform (CAP)

Continuación

| Eventos          | Fase      | Número de evento | Descripción  |
|------------------|-----------|------------------|--|
| POST1_ENDED      | END_POST1 | 22               | Este evento se produce cuando se cumplen todas las condiciones de la lista de supervisión END_POST1, es decir, cuando la fase POST1 finaliza correctamente y se inicia la fase POST2. Si se utiliza un <i>flying end</i> , no se distribuye ningún evento.   |
| END_POST2        | END_POST2 | 23               | Este evento se produce cuando la fase POST2 está en el final, es decir, cuando es el momento de finalizar por fin el proceso. Si se utiliza un <i>flying end</i> , no se distribuye ningún evento.   |
| POST2_ENDED      | END_POST2 | 24               | Este evento se produce cuando se cumplen todas las condiciones de la lista de supervisión END_POST2, es decir, cuando la fase POST2 y por tanto el proceso completo, finaliza correctamente. Si se utiliza un <i>flying end</i> , no se distribuye ningún evento.  |
| CAP_STOP         |           | 25               | Este evento es un evento obligatorio. Si se utiliza algún otro evento, este evento también debe definirse. El evento/TRAP se ejecuta en cuanto es posible una vez que el controlador se detiene debido un error o un paro de programa. Un error puede ser un error recuperable detectado en CAP, un error fatal detectado en CAP o un error interno que detiene el controlador. El código ejecutado en este TRAP debe poner todos los equipos externos en un estado seguro, por ejemplo, restablecer todas las señales E/S externas. |
| CAP_PF_RESTART   | MAIN      | 26               | Este evento se produce cuando se ordena el reinicio.   |
| EQUIDIST         | MAIN      | 27               | Este evento se envía, si se ordena con la instrucción <code>CapEquiDist</code> .   |
| AT_ERRORPOINT    | MAIN      | 28               | Este evento se produce después del reinicio, cuando el TCP alcanza la posición del error de supervisión.   |
| FLY_START        | MAIN      | 29               | Este evento se produce cuando se utiliza <i>flying start</i> . Este evento solo está disponible con <i>flying start</i> .  |
| FLY_END          | MAIN      | 30               | Este evento se produce cuando se utiliza <i>flying end</i> . Este evento solo está disponible con <i>flying end</i> .  |
| LAST_INSTR_ENDED | MAIN      | 31               | Este evento se produce cuando finaliza la ejecución de RAPID de la última instrucción de CAP durante <i>flying end</i> . Este evento solo está disponible con <i>flying end</i> .  |

Continúa en la página siguiente

# 1 Instrucciones

## 1.93 ICap - Conectar eventos CAP a rutinas TRAP

### Continuous Application Platform (CAP)

#### Continuación

| Eventos       | Fase  | Número de evento | Descripción   |
|---------------|-------|------------------|---|
| END_PRE       | PRE   | 32               | Este evento se produce cuando está activada la supervisión de la fase PRE, si existe. Si se utiliza un <i>flying start</i> , no se distribuye ningún evento porque ya hay un movimiento de TCP. En el inicio se distribuye este evento.   |
| PRE_ENDED     | PRE   | 33               | Este evento se produce cuando está activada la supervisión de la fase PRE, si existe. Si se utiliza un <i>flying start</i> , no se distribuye ningún evento porque ya hay un movimiento de TCP. En el inicio se distribuye este evento.   |
| START_POST1   | POST1 | 34               | Este evento se produce cuando está activada la supervisión de la fase POST1, si existe. Si se utiliza un <i>flying start</i> , no se distribuye ningún evento porque ya hay un movimiento de TCP. En el inicio se distribuye este evento. |
| POST1_STARTED | POST1 | 35               | Este evento se produce cuando está activada la supervisión de la fase POST1, si existe. Si se utiliza un <i>flying start</i> , no se distribuye ningún evento porque ya hay un movimiento de TCP. En el inicio se distribuye este evento. |
| START_POST2   | POST2 | 36               | Este evento se produce cuando está activada la supervisión de la fase POST1, si existe. Si se utiliza un <i>flying start</i> , no se distribuye ningún evento porque ya hay un movimiento de TCP. En el inicio se distribuye este evento. |
| POST2_STARTED | POST2 | 37               | Este evento se produce cuando está activada la supervisión de la fase POST1, si existe. Si se utiliza un <i>flying start</i> , no se distribuye ningún evento porque ya hay un movimiento de TCP. En el inicio se distribuye este evento. |

#### Limitaciones

No es posible utilizar más de una vez la identidad de la interrupción sin eliminarla previamente. Por tanto, las interrupciones deben tratarse de la forma mostrada, con una de las alternativas siguientes.

```
PROC setup_events ()  
  VAR intnum start_intno;  
  IDelete start_intno;  
  CONNECT start_intno WITH start_trap;  
  ICap start_intno, CAP_START;  
ENDPROC
```

Al principio del programa se produce la activación de todas las interrupciones. En este caso, las instrucciones iniciales se mantienen fuera del flujo principal del programa. La instrucción ICap debe ejecutarse solo una vez, por ejemplo, en la

Continúa en la página siguiente



## 1.93 ICap - Conectar eventos CAP a rutinas TRAP Continuous Application Platform (CAP) Continuación

rutina de evento del sistema de puesta en marcha. Se recomienda colocar todas las rutinas TRAP en una tarea en segundo plano.

### Sintaxis

```
ICap  
  [Interrupt ':=' ] < variable (IN) of intnum > ','  
  [Event ':=' ] < variable (IN) of num > ';' 
```

### Información relacionada

| Para obtener más información sobre                    | Consulte  |
|---|---|
| <i>Continuous Application Platform</i>                | <i>Application manual - Continuous Application Platform</i>                           |
| Conectar una interrupción con una rutina TRAP         | <a href="#">CONNECT - Conecta una interrupción a una rutina TRAP en la página 161</a> |
| Cancelar una interrupción conectada a una rutina TRAP | <a href="#">IDelete - Cancela una interrupción en la página 254</a>                   |
| Tipo de dato <code>intnum</code>                      | <a href="#">intnum - Identidad de interrupción en la página 1687</a>                  |

# 1 Instrucciones

## 1.94 IDelete - Cancela una interrupción

*IDelete*

### 1.94 IDelete - Cancela una interrupción

#### Utilización

*IDelete* (*Interrupt Delete*) se utiliza para cancelar (eliminar) una interrupción. Si sólo se desea desactivar temporalmente la interrupción, deben utilizarse las instrucciones *ISleep* o *IDisable*.

#### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción *IDelete*:

##### Ejemplo 1

```
IDelete feeder_low;
```

Se cancela la interrupción *feeder\_low*.

#### Argumentos

```
IDelete Interrupt
```

Interrupt

Tipo de dato: *intnum*

La identidad de la interrupción.

#### Ejecución de programas

La definición de la interrupción se elimina completamente. Para definirla de nuevo, es necesario reconectarla primero a la rutina TRAP.

Se recomienda colocar un punto de paro antes de *IDelete*. De lo contrario, la interrupción se desactivará antes de alcanzar el punto final de la trayectoria del movimiento.

No es imprescindible eliminar las interrupciones, ya que esto se produce automáticamente en los casos siguientes:

- Cuando se carga un nuevo programa
- Cuando se reinicia el programa desde el principio
- Cuando se traslada el puntero de programa al principio de una rutina

#### Sintaxis

```
IDelete [ Interrupt ':' ] < variable (VAR) of intnum > ';' 
```

#### Información relacionada

| Para obtener más información sobre                 | Consulte   |
|--|--|
| Resumen de interrupciones                          | <a href="#">Manual de referencia técnica - RAPID Overview</a>                  |
| Más información sobre la gestión de interrupciones | <a href="#">Manual de referencia técnica - RAPID Overview</a>                  |
| Desactivación temporal de una interrupción         | <a href="#">ISleep - Desactiva una interrupción en la página 335</a>           |
| Desactivación temporal de todas las interrupciones | <a href="#">IDisable - Desactiva todas las interrupciones en la página 255</a> |

## 1.95 IDisable - Desactiva todas las interrupciones

### Utilización

`IDisable` (*Interrupt Disable*) se utiliza para desactivar temporalmente todas las interrupciones. Por ejemplo, puede usarse en una parte especialmente delicada del programa en la que no debe permitirse que se produzcan interrupciones, si éstas impiden la ejecución normal del programa.

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción `IDisable`:

#### Ejemplo 1

```
IDisable;
FOR i FROM 1 TO 100 DO
  character[i]:=ReadBin(sensor);
ENDFOR
IEnable;
```

No se permite ninguna interrupción durante la lectura del sensor. Cuando se completa la lectura, las interrupciones vuelven a permitirse.

### Ejecución de programas

Las interrupciones que se produzcan durante el periodo en el que esté vigente la instrucción `IDisable` se almacenan en una cola. Cuando vuelven a permitirse las interrupciones, se empiezan a generar inmediatamente las interrupciones de la cola, que se ejecutan en un orden FIFO.

`IEnable` está activo de forma predeterminado. `IEnable` se define automáticamente en los casos siguientes:

- cuando se utiliza el modo de reinicio **Restablecer RAPID**
- al cargar un nuevo programa o un nuevo módulo
- al iniciar la ejecución del programa desde el principio
- al mover el puntero del programa a `main`
- al mover el puntero del programa a una rutina
- al mover el puntero de programa de una forma que se pierda el orden de la ejecución.
- tras ejecutar un ciclo (más allá de `main`) o tras ejecutar `ExitCycle`.

### Sintaxis

```
IDisable';'
```

### Información relacionada

| Para obtener más información sobre                 | Consulte   |
|--|--|
| Resumen de interrupciones                          | <i>Manual de referencia técnica - RAPID Overview</i>                         |
| Más información sobre la gestión de interrupciones | <i>Manual de referencia técnica - RAPID Overview</i>                         |
| Activación de interrupciones                       | <a href="#">IEnable - Habilita el uso de interrupciones en la página 256</a> |

# 1 Instrucciones

---

## 1.96 IEnable - Habilita el uso de interrupciones

RobotWare - OS

## 1.96 IEnable - Habilita el uso de interrupciones

---

### Utilización

`IEnable` (*Interrupt Enable*) se utiliza para permitir el uso de interrupciones durante la ejecución del programa.

---

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción `IEnable`:

#### Ejemplo 1

```
IDisable;
FOR i FROM 1 TO 100 DO
  character[i]:=ReadBin(sensor);
ENDFOR
IEnable;
```

No se permite ninguna interrupción durante la lectura del sensor. Cuando se completa la lectura, las interrupciones vuelven a permitirse.

---

### Ejecución de programas

Las interrupciones que se produzcan durante el periodo en el que esté vigente la instrucción `IDisable` se almacenan en una cola. Cuando vuelven a permitirse las interrupciones (`IEnable`), se empiezan a generar inmediatamente las interrupciones, que se ejecutan en un orden FIFO. A partir de ese momento, la ejecución del programa continúa del modo normal y las interrupciones que se produzcan a continuación se procesan tan pronto como se producen.

De forma predeterminada, se permite el uso de interrupciones siempre que se empieza a ejecutar un programa. Las interrupciones desactivadas por la instrucción `ISleep` no se ven afectadas por la instrucción `IEnable`.

---

### Sintaxis

```
IEnable';'
```

---

### Información relacionada

| Para obtener más información sobre                 | Consulte  |
|--|---|
| Resumen de interrupciones                          | <i>Manual de referencia técnica - RAPID Overview</i>                                  |
| Más información sobre la gestión de interrupciones | <i>Manual de referencia técnica - RAPID Overview</i>                                  |
| Desactivación de interrupciones                    | <a href="#"><i>IDisable - Desactiva todas las interrupciones en la página 255</i></a> |

---

## 1.97 IError - Solicita una interrupción para errores

---

### Utilización

`IError` (*Interrupt Errors*) se utiliza para solicitar y activar una interrupción cuando se produce un error.

`IError` permite registrar los errores, advertencias o cambios de estado.

---

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción `IError`:

Consulte también [Más ejemplos en la página 258](#).

#### Ejemplo 1

```
VAR intnum err_int;
...
PROC main()
  CONNECT err_int WITH err_trap;
  IError COMMON_ERR, TYPE_ALL, err_int;
```

Solicita una interrupción de RAPID y la ejecución de la rutina `TRAPerr_trap` cada vez que el sistema genera un error, una advertencia o un cambio de estado.

---

### Argumentos

```
IError ErrorDomain [ \ErrorId ] ErrorType Interrupt
```

ErrorDomain

Tipo de dato: `errdomain`

El dominio de error que debe monitorizarse. Consulte los datos predefinidos del tipo `errdomain`. Para especificar cualquier dominio, utilice `COMMON_ERR`.

[ \ErrorId ]

Tipo de dato: `num`

Opcionalmente, el número de un error concreto que se desea monitorizar. El número de error debe especificarse sin el primer dígito (el que corresponde al dominio del error) del número de error completo.

Por ejemplo, el error 10008 Program restarted debe especificarse como 0008 o simplemente 8.

ErrorType

Tipo de dato: `errtype`

El tipo de evento, por ejemplo un error, un aviso o un cambio de estado, que se desea monitorizar. Consulte los datos predefinidos del tipo `errtype`. Para especificar cualquier tipo, utilice `TYPE_ALL`.

Interrupt

Tipo de dato: `intnum`

La identidad de la interrupción. La interrupción debe estar ya conectada a una rutina TRAP mediante la instrucción `CONNECT`.

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.97 IError - Solicita una interrupción para errores

RobotWare - OS

Continuación

---

### Ejecución de programas

La llamada a la rutina TRAP correspondiente se realiza automáticamente cuando se produce un error, en el dominio especificado, del tipo especificado y opcionalmente con el número de error especificado. Una vez ejecutada la rutina, la ejecución del programa continúa a partir del punto en el que se produjo la interrupción.

---

### Más ejemplos

A continuación aparecen más ejemplos de la instrucción IError.

```
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
```

Quando se produce un error (sólo en el caso de los errores, no las advertencias ni los cambios de estado), el número de error se obtiene en la rutina TRAP y su valor se utiliza para activar dos grupos de señales digitales de salida.

---

### Limitación

No es posible solicitar interrupciones para los errores internos.

En una tarea de tipo NORMAL, el evento es descartado al pararse el programa, de forma que no todos los eventos pueden ser capturados en una tarea NORMAL.

Para capturar todos los eventos, la tarea debe ser de tipo estático o semiestático.

No es posible utilizar más de una vez la identidad de la interrupción sin eliminarla previamente. Por tanto, las interrupciones deben tratarse de la forma mostrada, con una de las alternativas siguientes.

```
VAR intnum err_interrupt;  
PROC main ( )  
    CONNECT err_interrupt WITH err_trap;  
    IError COMMON_ERR, TYPE_ERR, err_interupt;  
    WHILE TRUE DO  
    :  
    :
```

Continúa en la página siguiente

---

```
ENDWHILE
ENDPROC
```

Las interrupciones están activadas cuando se empieza a ejecutar el programa. En este caso, las instrucciones se mantienen inicialmente fuera del flujo principal del programa.

```
VAR intnum err_interrupt;
PROC main ( )
CONNECT err_interrupt WITH err_trap;
IError COMMON_ERR, TYPE_ERR, err_interupt;
:
:
IDelete err_interrupt;
ENDPROC
```

La interrupción se elimina al final del programa y se activa de nuevo. En este caso, recuerde que la interrupción permanece inactiva durante un periodo breve.

### Sintaxis

```
IError
[ErrorDomain ':='] <expression (IN) of errdomain>
['\ErrorId' :=] <expression (IN) of num>\ ' , '
[ErrorType' :=] <expression (IN) of errtype> ' , '
[Interrupt' :=] <variable (VAR) of intnum>;'
```

### Información relacionada

| Para obtener más información sobre                            | Consulte  |
|---|---|
| Resumen de interrupciones                                     | <i>Manual de referencia técnica - RAPID Overview</i>  |
| Más información sobre la gestión de interrupciones            | <i>Manual de referencia técnica - RAPID Overview</i>  |
| Dominios de error, constantes predefinidas                    | <a href="#">errdomain - Dominio del error en la página 1662</a>   |
| Tipos de errores, constantes predefinidas                     | <a href="#">errtype - Tipo de error en la página 1673</a>   |
| Obtención de datos de interrupción para la rutina TRAP actual | <a href="#">GetTrapData - Obtiene datos de interrupción para la rutina TRAP actual en la página 240</a> |
| Obtención de información sobre un error                       | <a href="#">ReadErrData - Obtiene información sobre un error en la página 586</a>                       |
| <i>Advanced RAPID</i>   | <i>Application manual - Controller software IRC5</i>  |

# 1 Instrucciones

---

1.98 IF - Si se cumple una condición, ...; de lo contrario, ...  
RobotWare - OS

## 1.98 IF - Si se cumple una condición, ...; de lo contrario, ...

---

### Utilización

IF se utiliza cuando es necesario ejecutar instrucciones diferentes en función de si se cumple una condición.

---

### Ejemplos básicos

A continuación aparecen algunos ejemplos básicos de la instrucción IF.  
Consulte también [Más ejemplos en la página 260](#).

#### Ejemplo 1

```
IF reg1 > 5 THEN
  Set do1;
  Set do2;
ENDIF
```

Las señales do1 y do2 sólo se activan si reg1 es mayor que 5.

#### Ejemplo 2

```
IF reg1 > 5 THEN
  Set do1;
  Set do2;
ELSE
  Reset do1;
  Reset do2;
ENDIF
```

Las señales do1 y do2 se activan o restablecen en función de si reg1 es mayor o no de 5.

---

### Argumentos

```
IF Condition THEN ...
  {ELSEIF Condition THEN ...}
  [ELSE ...]
ENDIF
```

Condition

**Tipo de dato:** bool

La condición que debe cumplirse para que se ejecuten las instrucciones que se encuentran entre THEN y ELSE/ELSEIF.

---

### Más ejemplos

A continuación aparecen más ejemplos de cómo usar la instrucción IF.

#### Ejemplo 1

```
IF counter > 100 THEN
  counter := 100;
ELSEIF counter < 0 THEN
  counter := 0;
ELSE
  counter := counter + 1;
```

*Continúa en la página siguiente*

---



ENDIF

Se incrementa el valor de `counter` en 1. Sin embargo, si el valor de `counter` se encuentra fuera de los límites 0-100, se asigna a `counter` el valor de límite correspondiente.

### Ejecución de programas

Las condiciones se comprueban una tras otra hasta que una de ellas se cumple. La ejecución del programa continúa con las instrucciones asociadas con la condición. Si no se cumple ninguna de las condiciones, la ejecución del programa continúa con las instrucciones que aparecen a continuación de `ELSE`. Si se cumple más de una condición, sólo se ejecutan las instrucciones asociadas con la primera de las condiciones.

### Sintaxis

```
IF <conditional expression> THEN
  <statement list>
{ ELSEIF <conditional expression> THEN
  <statement list> | <EIT> }
[ ELSE
  <statement list> ]
ENDIF
```

### Información relacionada

| Para obtener más información sobre | Consulte   |
|------------------------------------|--|
| Condiciones (expresiones lógicas)  | <i>Manual de referencia técnica - RAPID Overview</i> |

# 1 Instrucciones

---

## 1.99 Incr - Aumenta en 1 un valor

RobotWare - OS

## 1.99 Incr - Aumenta en 1 un valor

---

### Utilización

Incr se utiliza para sumar 1 a una variable o una variable persistente de tipo numérico.

---

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción Incr:

Consulte también [Más ejemplos en la página 262](#).

#### Ejemplo 1

```
Incr reg1;
```

Se suma 1 a reg1, es decir, reg1:=reg1+1.

---

### Argumentos

```
Incr Name | Dname
```

Name

**Tipo de dato:** num

El nombre de la variable o de la variable persistente que se desea cambiar.

Dname

**Tipo de dato:** dnum

El nombre de la variable o de la variable persistente que se desea cambiar.

---

### Más ejemplos

A continuación aparecen más ejemplos de la instrucción Incr.

#### Ejemplo 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
```

Con cada ciclo, se actualiza en el FlexPendant el número de piezas. La producción sigue en marcha siempre y cuando no se active la señal de entrada stop\_production.

#### Ejemplo 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
```

*Continúa en la página siguiente*

Con cada ciclo, se actualiza en el FlexPendant el número de piezas. La producción sigue en marcha siempre y cuando no se active la señal de entrada `stop_production`.

## Sintaxis

Incr

[ Name ' := ' ] < var or pers ( **INOUT** ) of num >

| [ Dname ' := ' ] < var or pers ( **INOUT** ) of dnum > ' ; '

## Información relacionada

| Para obtener más información sobre   | Consulte  |
|--|---|
| Decremento de una variable en 1  | <a href="#">Decr - Reduce un número en 1 en la página 189</a> |
| Suma de cualquier valor a una variable   | <a href="#">Add - Suma un valor numérico en la página 32</a>  |
| Cambio de un dato mediante una expresión arbitraria, por ejemplo, una multiplicación | <a href="#">":=" - Asigna un valor en la página 42</a>        |

# 1 Instrucciones

---

## 1.100 IndAMove - Movimiento independiente de posición absoluta *Independent Axis*

### 1.100 IndAMove - Movimiento independiente de posición absoluta

---

#### Utilización

`IndAMove` (*Independent Absolute Movement*) se utiliza para cambiar un eje al modo independiente y mover el eje a una posición determinada.

Los ejes independientes son ejes que se mueven independientemente de los demás ejes del sistema de robot. Dado que la ejecución del programa prosigue inmediatamente, es posible ejecutar otras instrucciones (incluidas las instrucciones de posicionamiento) durante el tiempo del movimiento del eje independiente.

Si el eje debe moverse dentro de una revolución, se debe utilizar en su lugar la instrucción `IndRMove`. Si el movimiento debe producirse a una corta distancia de la posición actual, se debe utilizar la instrucción `IndDMove`.

Esta instrucción sólo puede usarse en la tarea principal `T_ROB1` o, si se cuenta con un sistema *MultiMove*, en las tareas de movimiento.

#### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción `IndAMove`:  
Consulte también [Más ejemplos en la página 266](#).

#### Ejemplo 1

```
IndAMove Station_A,2\ToAbsPos:=p4,20;
```

El eje 2 de `Station_A` se mueve hasta la posición `p4` a una velocidad de 20 grados/s.

#### Argumentos

```
IndAMove MecUnit Axis [\ToAbsPos] | [\ToAbsNum] Speed [\Ramp]
```

#### MecUnit

***Mechanical Unit***

Tipo de dato: `mecunit`

El nombre de la unidad mecánica.

#### Axis

Tipo de dato: `num`

El número del eje actual de la unidad mecánica (del 1 al 6)

#### [\ToAbsPos]

***To Absolute Position***

Tipo de dato: `robtarget`

La posición del eje se especifica como un `robtarget`. Sólo se utiliza el componente de este eje `Axis` en concreto. El valor se utiliza como un valor de posición absoluta en grados (mm en el caso de los ejes lineales).

La posición del eje se verá afectada si el eje se desplaza utilizando la instrucción `EOffsSet` o `EOffsOn`.

En el caso de los ejes del robot, se debe utilizar en su lugar el argumento `\ToAbsNum`.

*Continúa en la página siguiente*

## 1.100 IndAMove - Movimiento independiente de posición absoluta *Independent Axis* Continuación

[ \ToAbsNum ]

*To Absolute Numeric value*

Tipo de dato: num

La posición del eje se define en grados (mm si es un eje lineal).

Mediante este argumento, la posición NO se verá afectada por ningún desplazamiento, por ejemplo EOffsSet o PDispOn.

Tiene la misma función que \ToAbsPos, pero la posición se define como un valor numérico para facilitar el cambio manual de la posición.

Speed

Tipo de dato: num

Velocidad del eje en grados/s (mm/s si es un eje lineal).

[ \Ramp ]

Tipo de dato: num

Reducción de la aceleración y deceleración respecto del rendimiento máximo (1 - 100%, 100% = rendimiento máximo).

### Ejecución de programas

Cuando se ejecuta IndAMove, el eje especificado se mueve a la velocidad programada hasta la posición de eje especificada. Si se programa \Ramp, se producirá una reducción de la aceleración o deceleración.

Para devolver el eje al modo normal, se utiliza la instrucción IndReset. En conexión con este cambio, es posible cambiar la posición lógica del eje de forma que se eliminen varias revoluciones completas de la posición para evitar que se produzca el retroceso en el giro para el movimiento siguiente.

La velocidad puede alterarse mediante la ejecución de otra instrucción IndAMove (u otra instrucción IndXMove). Si se selecciona una velocidad en el sentido opuesto, el eje se detiene y acelera hasta la nueva velocidad y en el nuevo sentido.

Durante la ejecución paso a paso de la instrucción, el eje se ajusta sólo en el modo independiente. El eje empieza a moverse cuando se ejecuta la instrucción siguiente y continúa siempre y cuando tenga lugar la ejecución del programa. Para obtener más información, consulte el *Manual de referencia de RAPID - Descripción general de RAPID, sección Principios de movimiento y E/S - Posicionamiento durante la ejecución del programa - Ejes independientes*.

Cuando se sitúa un puntero de programa al principio del programa o en una nueva rutina, todos los ejes cambian automáticamente al modo normal, sin cambiar el sistema de medición (lo que equivale a la ejecución de la instrucción IndReset\Old).



#### Nota

Una instrucción IndAMove a continuación de una operación IndCMove puede dar lugar a que el eje deshaga el movimiento realizado en la instrucción IndCMove. Para evitarlo, utilice una instrucción IndReset antes de la instrucción IndAMove o bien utilice una instrucción IndRMove.

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.100 IndAMove - Movimiento independiente de posición absoluta

### *Independent Axis*

#### *Continuación*

---

#### Limitaciones

Los ejes en el modo independiente no pueden tener movimientos asignados. Si se intenta ejecutar el eje manualmente, éste no se mueve y se muestra un mensaje de error. Ejecute una instrucción `IndReset` o mueva el puntero de programa a `main` para poder salir del modo independiente.

Si se produce una caída de alimentación mientras hay un eje en modo independiente, no es posible reanudar el programa. En este caso aparece un mensaje de error y es necesario reiniciar un programa desde el principio.

Esta instrucción no es recomendable en el caso de los ejes de muñeca de robot acoplados (consulte *Manual de referencia de RAPID - Descripción general de RAPID*, sección *Principios de movimiento y E/S - Posicionamiento durante la ejecución del programa - Ejes independientes*).

---

#### Más ejemplos

A continuación aparecen más ejemplos de la instrucción `IndAMove`.

#### Ejemplo 1

```
ActUnit Station_A;  
weld_stationA;  
IndAMove Station_A,1\ToAbsNum:=90,20\Ramp:=50;  
ActUnit Station_B;  
weld_stationB_1;  
WaitUntil IndInpos(Station_A,1) = TRUE;  
WaitTime 0.2;  
DeactUnit Station_A;  
weld_stationB_2;
```

Se activa `Station_A` y se inicia la soldadura en la estación A.

A continuación, se mueve `Station_A` (el eje 1) hasta la posición de 90 grados mientras el robot realiza la soldadura en la estación B. La velocidad del eje es de 20 grados/s. La velocidad cambia con la aceleración/deceleración reducida al 50% del rendimiento máximo.

Cuando la estación A alcanza esta posición, es desactivada y es posible realizar la recarga en la estación al mismo tiempo que el robot continúa soldando en la estación B.

---

#### Gestión de errores

Pueden generarse los errores recuperables enumerados a continuación. Estos errores pueden ser gestionados en un gestor de errores. El valor de la variable de sistema `ERRNO` cambia a:

| Nombre                    | Causa del error          |
|---------------------------|--------------------------|
| <code>ERR_AXIS_ACT</code> | El eje no está activado. |

---

#### Sintaxis

```
IndAMove  
[MecUnit '[:='] <variable (VAR) of mecunit>','  
[Axis '[:='] <expression (IN) of num>  
['\ ' ToAbsPos '[:='] <expression (IN) of rotarget>]
```

---

*Continúa en la página siguiente*

## 1.100 IndAMove - Movimiento independiente de posición absoluta

### *Independent Axis*

#### *Continuación*

```
[['\ ' ToAbsNum ':=' <expression (IN) of num>'],''
[Speed ':=' ] <expression (IN) of num>
[['\ ' Ramp ':=' <expression (IN) of num>'],';'
```

#### Información relacionada

| Para obtener más información sobre                              | Consulte   |
|---|--|
| Ejes independientes en general                                  | <i>Manual de referencia técnica - RAPID Overview</i>   |
| <i>Independent Axis</i>   | <i>Application manual - Controller software IRC5</i>   |
| Cambio de nuevo al modo manual                                  | <a href="#">IndReset - Restablecimiento independiente en la página 276</a>   |
| Restablecimiento del sistema de medición                        | <a href="#">IndReset - Restablecimiento independiente en la página 276</a>   |
| Otros movimientos de ejes independientes                        | <a href="#">IndRMove - Movimiento independiente de posición relativa en la página 281</a><br><a href="#">IndDMove - Movimiento independiente de posición delta en la página 272</a><br><a href="#">IndCMove - Movimiento independiente continuo en la página 268</a> |
| Comprobación del estado de velocidad de los ejes independientes | <a href="#">IndInpos - Estado de velocidad de un eje independiente en la página 1340</a>   |
| Comprobación del estado de posición de los ejes independientes  | <a href="#">IndInpos - Estado de posición de un eje independiente en la página 1338</a>  |
| Activación de ejes independientes                               | <i>Manual de referencia técnica - Parámetros del sistema, tema Motion, tipo Arm</i>  |

# 1 Instrucciones

---

## 1.101 IndCMove - Movimiento independiente continuo *Independent Axis*

### 1.101 IndCMove - Movimiento independiente continuo

---

#### Utilización

IndCMove (*Independent Continuous Movement*) se utiliza para cambiar un eje al modo independiente y empezar a moverlo continuamente a una velocidad determinada.

Los ejes independientes son ejes que se mueven independientemente de los demás ejes del sistema de robot. Dado que la ejecución del programa prosigue inmediatamente, es posible ejecutar otras instrucciones (incluidas las instrucciones de posicionamiento) durante el tiempo del movimiento del eje independiente.

Esta instrucción sólo puede usarse en la tarea principal T\_ROB1 o, si se cuenta con un sistema *MultiMove*, en las tareas de movimiento.

#### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción IndCMove:  
Consulte también [Más ejemplos en la página 270](#).

#### Ejemplo 1

```
IndCMove Station_A,2,-30.5;
```

El eje 2 de Station\_A empieza a moverse en el sentido negativo a una velocidad de 30,5 grados/s.

#### Argumentos

```
IndCMove MecUnit Axis Speed [\Ramp]
```

MecUnit

*Mechanical Unit*

Tipo de dato: mecunit

El nombre de la unidad mecánica.

Axis

Tipo de dato: num

El número del eje actual de la unidad mecánica (del 1 al 6).

Speed

Tipo de dato: num

Velocidad del eje en grados/s (mm/s si es un eje lineal).

El sentido del movimiento se especifica con el signo del argumento de velocidad.

[\Ramp]

Tipo de dato: num

Reducción de la aceleración y deceleración respecto del rendimiento máximo (1 - 100%, 100% = rendimiento máximo).

*Continúa en la página siguiente*



#### **Ejecución de programas**

Cuando se ejecuta `IndCMove`, el eje especificado empieza a moverse a la velocidad programada. El sentido del movimiento se especifica con el signo del argumento de velocidad. Si se programa `\Ramp`, se producirá una reducción de la aceleración o deceleración.

Para devolver el eje al modo normal, se utiliza la instrucción `IndReset`. La posición lógica del eje puede cambiarse en conexión con este cambio. Por ejemplo, es posible eliminar un número de revoluciones completas para evitar que se produzca el retroceso en el giro para el movimiento siguiente.

Es posible cambiar la velocidad ejecutando una instrucción `IndCMove` posterior. Si se solicita una velocidad en el sentido opuesto, el eje se detiene y acelera hasta la nueva velocidad y en el nuevo sentido. Para detener el eje, puede usarse el argumento de velocidad 0. En este caso, sigue teniendo el modo independiente.

Durante la ejecución paso a paso de la instrucción, el eje se ajusta sólo en el modo independiente. El eje empieza a moverse cuando se ejecuta la instrucción siguiente y continúa siempre y cuando continúe también la ejecución del programa. Para obtener más información, consulte el *Manual de referencia de RAPID - Descripción general de RAPID*, sección *Principios de movimiento y E/S - Posicionamiento durante la ejecución del programa - Ejes independientes*.

Cuando se sitúa un puntero de programa al principio del programa o en una nueva rutina, todos los ejes cambian automáticamente al modo normal, sin cambiar el sistema de medición (lo que equivale a la ejecución de la instrucción `IndReset\Old`).

#### **Limitaciones**

La resolución de la posición del eje empeora a medida que se mueve hasta su posición cero lógica (normalmente el centro del área de trabajo). Para volver a disponer de una resolución elevada, es posible cambiar a cero el área de trabajo lógica con la instrucción `IndReset`. Para obtener más información, consulte el *Manual de referencia de RAPID - Descripción general de RAPID*, sección *Principios de movimiento y E/S - Posicionamiento durante la ejecución del programa - Ejes independientes*.

Los ejes en el modo independiente no pueden tener movimientos asignados. Si se intenta ejecutar el eje manualmente, éste no se mueve y se muestra un mensaje de error. Ejecute una instrucción `IndReset` o mueva el puntero de programa a `main` para poder salir del modo independiente.

Si se produce una caída de alimentación cuando el eje se encuentra en el modo independiente, no es posible reanudar el programa. En este caso aparece un mensaje de error y es necesario reiniciar un programa desde el principio.

Esta instrucción no es recomendable en el caso de los ejes de muñeca de robot acoplados (consulte *Manual de referencia de RAPID - Descripción general de RAPID*, sección *Principios de movimiento y E/S - Posicionamiento durante la ejecución del programa - Ejes independientes*).

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.101 IndCMove - Movimiento independiente continuo

### Independent Axis

#### Continuación

#### Más ejemplos

A continuación aparecen más ejemplos de la instrucción IndCMove.

```
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;
```

El eje 2 de Station\_A empieza a moverse en el sentido positivo a una velocidad de 20 grados/s. Cuando el eje ha alcanzado la velocidad seleccionada, los ejes del robot empiezan a moverse.

Cuando el robot alcanza la posición p10, el eje externo cambia de sentido y gira a una velocidad de 10 grados/s. El cambio de velocidad se realiza con una aceleración/deceleración reducida al 50% del rendimiento máximo. Al mismo tiempo, el robot ejecuta el movimiento hacia p20.

A continuación, el eje 2 de Station\_A se detiene lo más rápidamente posible en la posición p1 dentro de la revolución actual.

Una vez que el eje 2 ha alcanzado esta posición y el robot se ha detenido en la posición p30, el eje 2 vuelve de nuevo al modo normal. El offset del sistema de medición para este eje se cambia a un número entero de revoluciones de eje, de forma que la posición actual esté lo más cerca posible de p40.

A continuación, cuando el robot se mueve hasta la posición p40, el eje 2 de Station\_A es movido por la instrucción MoveL p40 por la vía más corta hasta la posición p40 (máx. ±180 grados).

#### Gestión de errores

Pueden generarse los errores recuperables enumerados a continuación. Estos errores pueden ser gestionados en un gestor de errores. El valor de la variable de sistema ERRNO cambia a:

| Nombre       | Causa del error          |
|--------------|--------------------------|
| ERR_AXIS_ACT | El eje no está activado. |

#### Sintaxis

```
IndCMove  
[MecUnit ':='] <variable (VAR) of mecunit>', '  
[Axis ':='] <expression (IN) of num>', '  
[Speed ':='] <expression (IN) of num>  
['\' Ramp ':='] <expression (IN) of num>]';'
```

Continúa en la página siguiente

#### Información relacionada

| Para obtener más información sobre                              | Consulte   |
|---|--|
| Ejes independientes en general                                  | <i>Manual de referencia técnica - RAPID Overview</i>   |
| <i>Independent Axis</i>   | <i>Application manual - Controller software IRC5</i>   |
| Cambio de nuevo al modo manual                                  | <a href="#">IndReset - Restablecimiento independiente en la página 276</a>   |
| Restablecimiento del sistema de medición                        | <a href="#">IndReset - Restablecimiento independiente en la página 276</a>   |
| Otros movimientos de ejes independientes                        | <a href="#">IndAMove - Movimiento independiente de posición absoluta en la página 264</a><br><a href="#">IndRMove - Movimiento independiente de posición relativa en la página 281</a><br><a href="#">IndDMove - Movimiento independiente de posición delta en la página 272</a> |
| Comprobación del estado de velocidad de los ejes independientes | <a href="#">IndInpos - Estado de velocidad de un eje independiente en la página 1340</a>   |
| Comprobación del estado de posición de los ejes independientes  | <a href="#">IndInpos - Estado de posición de un eje independiente en la página 1338</a>  |
| Activación de ejes independientes                               | <i>Manual de referencia técnica - Parámetros del sistema, tema Motion, tipo Arm</i>  |

# 1 Instrucciones

---

## 1.102 IndDMove - Movimiento independiente de posición delta *Independent Axis*

### 1.102 IndDMove - Movimiento independiente de posición delta

---

#### Utilización

IndDMove(*Independent Delta Movement* se utiliza para cambiar un eje al modo independiente y mover el eje a una distancia determinada.

Los ejes independientes son ejes que se mueven independientemente de los demás ejes del sistema de robot. Dado que la ejecución del programa prosigue inmediatamente, es posible ejecutar otras instrucciones (incluidas las instrucciones de posicionamiento) durante el tiempo del movimiento del eje independiente.

Si se desea mover el eje hasta una posición determinada, debe utilizar en su lugar una instrucción IndAMove o IndRMove.

Esta instrucción sólo puede usarse en la tarea principal T\_ROB1 o, si se cuenta con un sistema *MultiMove*, en las tareas de movimiento.

---

#### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción IndDMove:

Consulte también [Más ejemplos en la página 274](#).

#### Ejemplo 1

```
IndDMove Station_A,2,-30,20;
```

Se mueve el eje 2 de Station\_A30 grados en el sentido negativo a una velocidad de 20 grados/s.

---

#### Argumentos

```
IndDMove MecUnit Axis Delta Speed [\Ramp]
```

MecUnit

***Mechanical Unit***

Tipo de dato: mecunit

El nombre de la unidad mecánica.

Axis

Tipo de dato: num

El número del eje actual de la unidad mecánica (del 1 al 6).

Delta

Tipo de dato: num

La distancia que se desea mover el eje actual, expresada en grados (mm en el caso de los ejes lineales). El signo especifica el sentido del movimiento.

Speed

Tipo de dato: num

Velocidad del eje en grados/s (mm/s si es un eje lineal).

[ \Ramp ]

Tipo de dato: num

*Continúa en la página siguiente*

Reducción de la aceleración y deceleración respecto del rendimiento máximo (1 - 100%, 100% = rendimiento máximo).

#### **Ejecución de programas**

Cuando se ejecuta `IndDMove`, el eje especificado se mueve a la velocidad programada hasta la distancia especificada. El sentido del movimiento se especifica con el signo del argumento `Delta`. Si se programa `\Ramp`, se producirá una reducción de la aceleración o deceleración.

Si el eje se está moviendo, la nueva posición se calcula a partir de la posición momentánea que tiene el eje en el momento de ejecutar la instrucción `IndDMove`. Si se ejecuta una instrucción `IndDMove` con una distancia 0 y el eje ya está cambiando de posición, el eje se detiene y retrocede hasta la posición ocupada por el eje en el momento de la ejecución de la instrucción.

Para devolver el eje al modo normal, se utiliza la instrucción `IndReset`. La posición lógica del eje puede cambiarse en conexión con este cambio. Por ejemplo, es posible eliminar un número de revoluciones completas de la posición para evitar que se produzca el retroceso en el giro para el movimiento siguiente.

La velocidad puede cambiarse ejecutando una instrucción `IndDMove` adicional (u otra instrucción `IndXMove`). Si se selecciona una velocidad en el sentido opuesto, el eje se detiene y acelera hasta la nueva velocidad y en el nuevo sentido.

Durante la ejecución paso a paso de la instrucción, el eje se ajusta sólo en el modo independiente. El eje empieza a moverse cuando se ejecuta la instrucción siguiente y continúa siempre y cuando continúe también la ejecución del programa. Para obtener más información, consulte el *Manual de referencia de RAPID - Descripción general de RAPID*, sección *Principios de movimiento y E/S - Posicionamiento durante la ejecución del programa - Ejes independientes*.

Cuando se sitúa un puntero de programa al principio del programa o en una nueva rutina, todos los ejes cambian automáticamente al modo normal, sin cambiar el sistema de medición (lo que equivale a la ejecución de la instrucción `IndReset \old`).

#### **Limitaciones**

Los ejes en el modo independiente no pueden tener movimientos asignados. Si se intenta ejecutar el eje manualmente, éste no se mueve y se muestra un mensaje de error. Ejecute una instrucción `IndReset` o mueva el puntero de programa a `main` para poder salir del modo independiente.

Si se produce un fallo de caída de alimentación cuando el eje se encuentra en el modo independiente, no es posible reanudar el programa. En este caso aparece un mensaje de error y es necesario reiniciar un programa desde el principio.

Esta instrucción no es recomendable en el caso de los ejes de muñeca de robot acoplados (consulte *Manual de referencia de RAPID - Descripción general de RAPID*, sección *Principios de movimiento y E/S - Posicionamiento durante la ejecución del programa - Ejes independientes*).

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.102 IndDMove - Movimiento independiente de posición delta

### *Independent Axis*

#### *Continuación*

#### Más ejemplos

A continuación aparecen más ejemplos de la instrucción `IndDMove`.

#### Ejemplo 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;
```

Se mueve el eje 6 del robot hasta las posiciones siguientes:

- 90 grados
- 60 grados
- 460 grados (1 revolución + 100 grados)

#### Gestión de errores

Pueden generarse los errores recuperables enumerados a continuación. Estos errores pueden ser gestionados en un gestor de errores. El valor de la variable de sistema `ERRNO` cambia a:

| Nombre                    | Causa del error          |
|---------------------------|--------------------------|
| <code>ERR_AXIS_ACT</code> | El eje no está activado. |

#### Sintaxis

```
IndDMove  
[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>]';'
```

#### Información relacionada

| Para obtener más información sobre                              | Consulte  |
|---|---|
| Ejes independientes en general                                  | <i>Manual de referencia técnica - RAPID Overview</i>  |
| <i>Independent Axis</i>   | <i>Application manual - Controller software IRC5</i>  |
| Cambio de nuevo al modo manual                                  | <a href="#">IndReset - Restablecimiento independiente en la página 276</a>  |
| Restablecimiento del sistema de medición                        | <a href="#">IndReset - Restablecimiento independiente en la página 276</a>  |
| Otros movimientos de ejes independientes                        | <a href="#">IndAMove - Movimiento independiente de posición absoluta en la página 264</a><br><a href="#">IndRMove - Movimiento independiente de posición relativa en la página 281</a><br><a href="#">IndCMove - Movimiento independiente continuo en la página 268</a> |
| Comprobación del estado de velocidad de los ejes independientes | <a href="#">IndInpos - Estado de velocidad de un eje independiente en la página 1340</a>  |

*Continúa en la página siguiente*

## 1.102 IndDMove - Movimiento independiente de posición delta

*Independent Axis*

*Continuación*

| Para obtener más información sobre                             | Consulte  |
|--|---|
| Comprobación del estado de posición de los ejes independientes | <a href="#">IndInpos - Estado de posición de un eje independiente en la página 1338</a> |
| Activación de ejes independientes                              | <i>Manual de referencia técnica - Parámetros del sistema, tema Motion, tipo Arm</i>     |

# 1 Instrucciones

---

## 1.103 IndReset - Restablecimiento independiente *Independent Axis*

### 1.103 IndReset - Restablecimiento independiente

---

#### Utilización

`IndReset` (*Independent Reset*) se utiliza para devolver un eje independiente al modo normal. Al mismo tiempo, el sistema de medición de los ejes de rotación puede moverse un número de revoluciones de eje.

Esta instrucción sólo puede usarse en la tarea principal `T_ROB1` o, si se cuenta con un sistema *MultiMove*, en las tareas de movimiento.

---

#### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción `IndReset`:

Consulte también [Más ejemplos en la página 279](#).

```
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;
```

Se mueve en primer lugar el eje 2 de `Station_A` en el modo independiente, devolviéndolo a continuación al modo normal. El eje se mantendrá en su posición.



#### Nota

Ni el eje independiente actual ni los ejes normales deben moverse cuando se ejecuta la instrucción `IndReset`. Por eso la posición anterior es un punto de paro y la instrucción `IndCMove` se ejecuta con la velocidad cero. Además, se utiliza una pausa de 0,2 segundos para garantizar que se ha conseguido el estado correcto.

---

#### Argumentos

```
IndReset MecUnit Axis [\RefPos] | [\RefNum] [\Short] | [\Fwd]  
| [\Bwd] | \Old]
```

`MecUnit`

#### *Mechanical Unit*

Tipo de dato: `mecunit`

El nombre de la unidad mecánica.

`Axis`

Tipo de dato: `num`

El número del eje actual de la unidad mecánica (del 1 al 6).

[ `\RefPos` ]

#### *Reference Position*

Tipo de dato: `robtarg`

*Continúa en la página siguiente*



La posición del eje de referencia se especifica como un `robtarget`. Sólo se utiliza el componente de este eje `Axis` en concreto. La posición debe estar dentro del área de trabajo normal.

En el caso de los ejes del robot, se debe utilizar en su lugar el argumento `\RefNum`. Este argumento sólo debe ser definido junto con el argumento `\Short`, `\Fwd` o `\Bwd`. No se permite junto con el argumento `\Old`.

[ `\RefNum` ]

#### *Reference Numeric value*

Tipo de dato: `num`

La posición del eje de referencia se define en grados (mm si es un eje lineal). La posición debe estar dentro del área de trabajo normal.

Este argumento sólo debe ser definido junto con el argumento `\Short`, `\Fwd` o `\Bwd`. No se permite junto con el argumento `\Old`.

Tiene la misma función que `\RefPos`, pero la posición se define como un valor numérico para facilitar el cambio manual de la posición.

[ `\Short` ]

Tipo de dato: `switch`

El sistema de medición cambiará un número entero de revoluciones en el lado del eje, de forma que el eje quede lo más cerca posible de la posición especificada con `\RefPos` o `\RefNum`. Si se ejecuta una instrucción de posicionamiento con la misma posición después de `IndReset`, el eje se desplaza por la ruta más corta, inferior a  $\pm 180$  grados, para alcanzar la posición.

[ `\Fwd` ]

#### *Forward*

Tipo de dato: `switch`

El sistema de medición cambiará un número entero de revoluciones en el lado del eje, de forma que la posición de referencia quede en el lado positivo de la posición especificada con `\RefPos` o `\RefNum`. Si se ejecuta una instrucción de posicionamiento con la misma posición después de `IndReset`, el eje gira en sentido positivo menos de 360 grados para alcanzar la posición.

[ `\Bwd` ]

#### *Backward*

Tipo de dato: `switch`

El sistema de medición cambiará un número entero de revoluciones en el lado del eje, de forma que la posición de referencia quede en el lado negativo de la posición especificada con `\RefPos` o `\RefNum`. Si se ejecuta una instrucción de posicionamiento con la misma posición después de `IndReset`, el eje gira en sentido negativo menos de 360 grados para alcanzar la posición.

[ `\Old` ]

Tipo de dato: `switch`

Mantiene la posición anterior.

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.103 IndReset - Restablecimiento independiente

### *Independent Axis*

#### *Continuación*



#### Nota

La resolución se reduce en las posiciones más alejadas de la posición cero. Si no se especifica ningún argumento `\Short`, `\Fwd`, `\Bwd` u `\Old`, se utiliza `\Old` como valor predeterminado.

---

### Ejecución de programas

Si se ejecuta `IndReset`, el eje independiente vuelve al modo normal. Al mismo tiempo, el sistema de medición del eje puede moverse un número entero de revoluciones de eje.

Esta instrucción también puede utilizarse en el modo normal para cambiar de sistema de medición.



#### Nota

La posición sólo se utiliza para ajustar el sistema de medición. El eje en sí no se mueve hasta la posición.

---

### Limitaciones

Esta instrucción sólo puede ejecutarse cuando todos los ejes activos que se encuentren en el modo normal estén parados. Todos los ejes activos de todas las unidades mecánicas conectadas al mismo planificador de movimientos deben permanecer parados. El eje independiente que se desea cambiar al modo normal también debe estar en reposo. En el caso de los ejes que se encuentran en el modo normal, esto se consigue ejecutando una instrucción de movimiento con el argumento `fine`. El eje independiente se detiene con una instrucción `IndCMove` con `Speed:=0` (seguida de un periodo de espera de 0,2 segundos), `IndRMoveIndAMove`, o `IndDMove`.

La resolución de las posiciones se reduce al alejarse de la posición lógica 0. Por tanto, un eje que gira progresivamente más y más desde la posición 0 debe ser puesto a cero con la instrucción `IndReset`, utilizando un argumento distinto de `\Old`.

No es posible cambiar el sistema de medición de los ejes lineales.

Para garantizar una puesta en marcha adecuada tras ejecutar `IndReset` con un eje y utilizando un sistema de medición relativo (modificadores de sincronización), es necesario añadir un retardo adicional de 0,12 segundos tras la instrucción `IndReset`.

El único eje del robot que puede usarse como eje independiente es el eje 6. Sin embargo, también puede usarse la instrucción `IndReset` con el eje 4 de los modelos IRB 1600, 2600 y 4600 (excepto para la versión ID). Si se utiliza `IndReset` con el eje 4 del robot, el eje 6 no debe estar en el modo independiente.

Si esta instrucción va precedida de una instrucción de movimiento, ésta última debe programarse con un punto de paro (`zonedata fine`), no un punto de paso. De lo contrario, no será posible reanudar la ejecución tras una caída de suministro eléctrico.

*Continúa en la página siguiente*

IndReset no puede ejecutarse en rutinas de RAPID que estén conectadas a los siguientes eventos especiales del sistema: PowerOn, Stop, QStop, Restart o Step.

IndReset sólo conmuta el estado independiente de un solo eje. No puede usarse para detener un movimiento independiente. Para detener un movimiento independiente, debe alcanzarse una condición de paro o el usuario debe, por ejemplo, mover el PP a main.

### Más ejemplos

A continuación aparecen más ejemplos de la instrucción IndReset.

#### Ejemplo 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;
```

Se mueve en primer lugar el eje 1 de Station\_A de forma independiente hasta la posición de 750 grados (2 revoluciones más 30 grados). En el mismo momento en que cambia al modo normal, se cambia la posición lógica a 30 grados.

A continuación, se mueve el eje 1 de Station\_A de forma independiente hasta la posición de 750 grados (2 revoluciones más 30 grados). En el mismo momento en que cambia al modo normal, se cambia la posición lógica a 390 grados (1 revolución más 30 grados).

### Gestión de errores

Pueden generarse los errores recuperables enumerados a continuación. Estos errores pueden ser gestionados en un gestor de errores. El valor de la variable de sistema ERRNO cambia a:

| Nombre          | Causa del error            |
|-----------------|----------------------------|
| ERR_AXIS_ACT    | El eje no está activado.   |
| ERR_AXIS_MOVING | El eje está en movimiento. |

### Sintaxis

```
IndReset
[MecUnit ':='] <variable (VAR) of mecunit>','
[Axis ':='] <expression (IN) of num>
['\' RefPos ':=' <expression (IN) of robtarg>]|
['\' RefNum ':=' <expression (IN) of num>]
['\' Short] | ['\' Fwd] | ['\' Bwd] | ['\' Old]';'
```

### Información relacionada

| Para obtener más información sobre | Consulte                                      |
|------------------------------------|---|
| Ejes independientes en general     | Manual de referencia técnica - RAPID Overview |

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.103 IndReset - Restablecimiento independiente

### *Independent Axis*

#### *Continuación*

| Para obtener más información sobre                              | Consulte  |
|---|---|
| <i>Independent Axis</i>   | <i>Application manual - Controller software IRC5</i>  |
| Cambio de un eje al modo independiente                          | <a href="#"><i>IndAMove - Movimiento independiente de posición absoluta en la página 264</i></a><br><a href="#"><i>IndCMove - Movimiento independiente continuo en la página 268</i></a><br><a href="#"><i>IndDMove - Movimiento independiente de posición delta en la página 272</i></a><br><a href="#"><i>IndRMove - Movimiento independiente de posición relativa en la página 281</i></a> |
| Comprobación del estado de velocidad de los ejes independientes | <a href="#"><i>IndInpos - Estado de velocidad de un eje independiente en la página 1340</i></a>   |
| Comprobación del estado de posición de los ejes independientes  | <a href="#"><i>IndInpos - Estado de posición de un eje independiente en la página 1338</i></a>  |
| Activación de ejes independientes                               | <i>Manual de referencia técnica - Parámetros del sistema, tema Motion, tipo Arm</i>   |

## 1.104 IndRMove - Movimiento independiente de posición relativa

### Utilización

IndRMove (*Independent Relative Movement*) se utiliza para cambiar un eje de rotación al modo independiente y mover el eje a una posición determinada dentro de una revolución.

Los ejes independientes son ejes que se mueven independientemente de los demás ejes del sistema de robot. Dado que la ejecución del programa prosigue inmediatamente, es posible ejecutar otras instrucciones (incluidas las instrucciones de posicionamiento) durante el tiempo del movimiento del eje independiente.

Si se desea mover el eje hasta una posición absoluta (varias revoluciones) o se trata de un eje lineal, se debe utilizar en su lugar la instrucción IndAMove. Si el movimiento debe producirse a una distancia determinada de la posición actual, se debe utilizar la instrucción IndDMove.

Esta instrucción sólo puede usarse en la tarea principal T\_ROB1 o, si se cuenta con un sistema *MultiMove*, en las tareas de movimiento.

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción IndRMove:

Consulte también [Más ejemplos en la página 283](#).

#### Ejemplo 1

```
IndRMove Station_A,2\ToRelPos:=p5 \Short,20;
```

Se mueve el eje 2 de Station\_A por la ruta más corta hasta la posición p5 dentro de una revolución (rotación máxima  $\pm 180$  grados) a una velocidad de 20 grados/s.

### Argumentos

```
IndRMove MecUnit Axis [\ToRelPos] | [\ToRelNum] [\Short] | [\Fwd]
| [\Bwd] Speed [\Ramp]
```

MecUnit

**Mechanical Unit**

Tipo de dato: mecunit

El nombre de la unidad mecánica.

Axis

Tipo de dato: num

El número del eje actual de la unidad mecánica (del 1 al 6).

[ \ToRelPos ]

**To Relative Position**

Tipo de dato: robtarget

La posición del eje se especifica como un robtarget. Sólo se utiliza el componente de este eje Axis en concreto. El valor se utiliza como un valor de posición en grados dentro de una revolución de eje. Esto significa que el eje se mueve menos de una revolución.

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.104 IndRMove - Movimiento independiente de posición relativa

### *Independent Axis*

#### *Continuación*

La posición del eje se verá afectada si el eje se desplaza utilizando la instrucción `EOffsSet` o `EOffsOn`.

En el caso de los ejes del robot, se debe utilizar en su lugar el argumento `\ToRelNum`.

[ `\ToRelNum` ]

***To Relative Numeric value***

Tipo de dato: `num`

La posición del eje, definida en grados.

Mediante este argumento, la posición NO se verá afectada por ningún desplazamiento, por ejemplo `EOffsSet` o `PDispOn`.

Tiene la misma función que `\ToRelPos`, pero la posición se define como un valor numérico para facilitar el cambio manual de la posición.

[ `\Short` ]

Tipo de dato: `switch`

El eje se mueve por la ruta más corta hasta la nueva posición. Esto significa que la rotación máxima será de 180 grados en cualquier sentido. Por tanto, el sentido del movimiento depende de la ubicación actual del eje.

[ `\Fwd` ]

***Forward***

Tipo de dato: `switch`

El eje se mueve en sentido positivo hasta la nueva posición. Esto significa que la rotación máxima será de 360 grados y siempre en sentido positivo (con aumento del valor de posición).

[ `\Bwd` ]

***Backward***

Tipo de dato: `switch`

El eje se mueve en sentido negativo hasta la nueva posición. Esto significa que la rotación máxima será de 360 grados y siempre en sentido negativo (con reducción del valor de posición).

Si se omite el argumento `\Short`, `\Fwd` o `\Bwd`, se utiliza `\Short` como valor predeterminado.

Speed

Tipo de dato: `num`

La velocidad del eje en grados/s.

[ `\Ramp` ]

Tipo de dato: `num`

Reducción de la aceleración y deceleración respecto del rendimiento máximo (1 - 100%, 100% = rendimiento máximo).

*Continúa en la página siguiente*

## 1.104 IndRMove - Movimiento independiente de posición relativa *Independent Axis* *Continuación*

### Ejecución de programas

Cuando se ejecuta `IndRMove`, el eje especificado se mueve a la velocidad programada hasta la posición de eje especificada, pero sólo una revolución como máximo. Si se programa `\Ramp`, se producirá una reducción de la aceleración o deceleración.

Para devolver el eje al modo normal, se utiliza la instrucción `IndReset`. La posición lógica del eje puede cambiarse en conexión con este cambio. Por ejemplo, es posible eliminar un número de revoluciones completas de la posición para evitar que se produzca el retroceso en el giro para el movimiento siguiente.

La velocidad puede cambiarse ejecutando una instrucción `IndRMove` adicional (u otra instrucción `IndXMove`). Si se selecciona una velocidad en el sentido opuesto, el eje se detiene y acelera hasta la nueva velocidad y en el nuevo sentido.

Durante la ejecución paso a paso de la instrucción, el eje se ajusta sólo en el modo independiente. El eje empieza a moverse cuando se ejecuta la instrucción siguiente y continúa siempre y cuando continúe también la ejecución del programa. Para obtener más información, consulte el *Manual de referencia de RAPID - Descripción general de RAPID*, sección *Principios de movimiento y E/S - Posicionamiento durante la ejecución del programa - Ejes independientes*.

Cuando se sitúa un puntero de programa al principio del programa o en una nueva rutina, todos los ejes cambian automáticamente al modo normal, sin cambiar el sistema de medición (lo que equivale a la ejecución de la instrucción `IndReset \Old`).

### Limitaciones

Los ejes en el modo independiente no pueden tener movimientos asignados. Si se intenta ejecutar el eje manualmente, éste no se mueve y se muestra un mensaje de error. Ejecute una instrucción `IndReset` o mueva el puntero de programa a `main` para poder salir del modo independiente.

Si se produce una caída de alimentación cuando el eje se encuentra en el modo independiente, no es posible reanudar el programa. En este caso aparece un mensaje de error y es necesario reiniciar un programa desde el principio.

Esta instrucción no es recomendable en el caso de los ejes de muñeca de robot acoplados (consulte el *Manual de referencia de RAPID - Descripción general de RAPID*, sección *Principios de movimiento y E/S - Posicionamiento durante la ejecución del programa - Ejes independientes*).

### Más ejemplos

A continuación aparecen más ejemplos de la instrucción `IndRMove`.

#### Ejemplo 1

```
IndRMove Station_A,1\ToRelPos:=p5 \Fwd,20\Ramp:=50;
```

El eje 1 de `Station_A` empieza a moverse en sentido positivo hacia la posición `p5` dentro de una revolución (rotación máxima 360 grados) a una velocidad de 20 grados/s. La velocidad cambia con la aceleración/deceleración reducida al 50% del rendimiento máximo.

```
IndAMove Station_A,1\ToAbsNum:=90,20;
```

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.104 IndRMove - Movimiento independiente de posición relativa

### *Independent Axis*

#### *Continuación*

```
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;
```

Se mueve el eje 1 de Station\_A hasta las posiciones siguientes:

- 90 grados
- 440 grados (1 revolución + 80 grados)
- 410 grados (1 revolución + 50 grados)
- 510 grados (1 revolución + 150 grados)
- 10 grados

### Gestión de errores

Pueden generarse los errores recuperables enumerados a continuación. Estos errores pueden ser gestionados en un gestor de errores. El valor de la variable de sistema ERRNO cambia a:

| Nombre       | Causa del error          |
|--------------|--------------------------|
| ERR_AXIS_ACT | El eje no está activado. |

### Sintaxis

```
IndRMove
[MecUnit ':=' ] <variable (VAR) of mecunit>', '
[Axis ':=' ] <expression (IN) of num>
['\ ' ToRelPos ':=' <expression (IN) of robtargets>]
| ['\ ' ToRelNum ':=' <expression (IN) of num>]
['\ ' Short] | ['\ ' Fwd] | ['\ ' Bwd]', '
[Speed ':=' ] <expression (IN) of num>
['\ ' Ramp ':=' <expression (IN) of num>]';'
```

### Información relacionada

| Para obtener más información sobre       | Consulte   |
|--|--|
| Ejes independientes en general           | <i>Manual de referencia técnica - RAPID Overview</i>                       |
| <i>Independent Axis</i>                  | <i>Application manual - Controller software IRC5</i>                       |
| Cambio de nuevo al modo manual           | <a href="#">IndReset - Restablecimiento independiente en la página 276</a> |
| Restablecimiento del sistema de medición | <a href="#">IndReset - Restablecimiento independiente en la página 276</a> |

*Continúa en la página siguiente*



## 1.104 IndRMove - Movimiento independiente de posición relativa

*Independent Axis*

*Continuación*

| Para obtener más información sobre                              | Consulte   |
|---|--|
| Otros movimientos de ejes independientes                        | <a href="#">IndAMove - Movimiento independiente de posición absoluta en la página 264</a><br><a href="#">IndDMove - Movimiento independiente de posición delta en la página 272</a><br><a href="#">IndCMove - Movimiento independiente continuo en la página 268</a> |
| Comprobación del estado de velocidad de los ejes independientes | <a href="#">IndInpos - Estado de velocidad de un eje independiente en la página 1340</a>   |
| Comprobación del estado de posición de los ejes independientes  | <a href="#">IndInpos - Estado de posición de un eje independiente en la página 1338</a>  |
| Activación de ejes independientes                               | <i>Manual de referencia técnica - Parámetros del sistema, tema Motion, tipo Arm</i>  |

# 1 Instrucciones

---

## 1.105 InitSuperv - Restablecer toda la supervisión para CAP *Continuous Application Platform (CAP)*

### 1.105 InitSuperv - Restablecer toda la supervisión para CAP

---

#### Utilización

`InitSuperv` se utiliza para iniciar la supervisión de CAP. Significa que se borran todas las listas de supervisión y que se eliminan todas las suscripciones de E/S.

---

#### Ejemplo

```
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` se utiliza para borrar todas las listas de supervisión antes de configurar la nueva supervisión.

---

#### Limitaciones

La instrucción `InitSuperv` solo debe ejecutarse una vez, por ejemplo, en el shelf de puesta en marcha.

---

#### Sintaxis

```
InitSuperv ';' 
```

---

#### Información relacionada

| Para obtener más información sobre     | Consulte  |
|--|---|
| <i>Continuous Application Platform</i> | <i>Application manual - Continuous Application Platform</i>   |
| Instrucción <code>SetupSuperv</code>   | <a href="#">SetupSuperv - Configurar las condiciones para la supervisión de señales en CAP en la página 719</a> |
| Instrucción <code>RemoveSuperv</code>  | <a href="#">RemoveSuperv - Eliminar la condición de una señal en la página 602</a>                              |

## 1.106 InvertDO - Invierte el valor de una señal de salida digital

### Utilización

`InvertDO` (*Invert Digital Output*) invierte el valor de una señal digital de salida (0 -> 1 y 1 -> 0).

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción `InvertDO`:

#### Ejemplo 1

```
InvertDO do15;
```

Se invierte el valor actual de la señal `do15`.

### Argumentos

```
InvertDO Signal
```

Signal

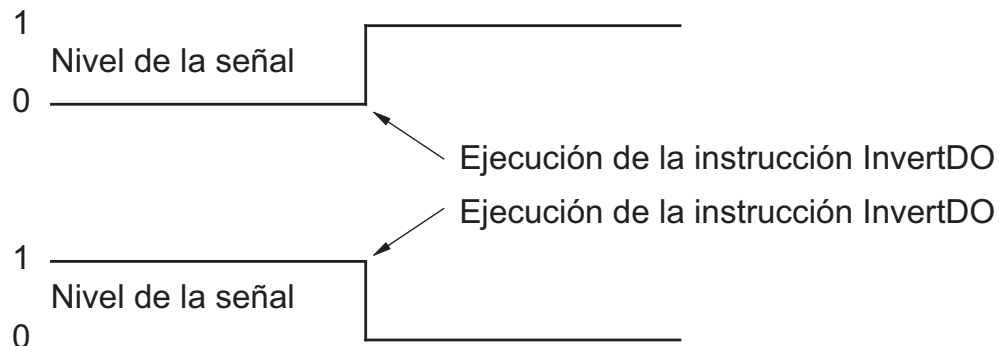
Tipo de dato: `signaldo`

El nombre de la señal a invertir.

### Ejecución de programas

Se invierte el valor actual de la señal (consulte la figura siguiente).

En la figura siguiente se muestra la inversión de una señal digital de salida.



xx0500002164

### Gestión de errores

Pueden generarse los errores recuperables enumerados a continuación. Estos errores pueden ser gestionados en un gestor de errores. El valor de la variable de sistema `ERRNO` cambia a:

| Nombre                          | Causa del error   |
|---------------------------------|---|
| <code>ERR_NO_ALIASIO_DEF</code> | La variable de señal es una variable declarada en RAPID. No se ha conectado a una señal E/S definida en la configuración de E/S con la instrucción <code>AliasIO</code> . |
| <code>ERR_NORUNUNIT</code>      | Se ha perdido el contacto con el dispositivo de E/S.  |
| <code>ERR_SIG_NOT_VALID</code>  | La señal de E/S no está disponible (solo válido para el bus de campo ICI).  |

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.106 InvertDO - Invierte el valor de una señal de salida digital

RobotWare - OS

Continuación

---

### Sintaxis

```
InvertDO  
  [Signal '[:='] <variable (VAR) of signaldo>'];'
```

---

### Información relacionada

| Para obtener más información sobre         | Consulte   |
|--|--|
| Instrucciones de entrada/salida            | <i>Manual de referencia técnica - RAPID Overview</i>         |
| Funcionalidad de entrada/salida en general | <i>Manual de referencia técnica - RAPID Overview</i>         |
| Configuración de E/S                       | <i>Manual de referencia técnica - Parámetros del sistema</i> |

## 1.107 IOBusStart - Start of I/O network

### Utilización

`IOBusStart` se utiliza para poner en marcha una determinada red de E/S.

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción `IOBusStart`:

#### Ejemplo 1

```
IOBusStart "IBS";
```

La instrucción pone en marcha la red de E/S con el nombre `IBS`.

### Argumentos

```
IOBusStart BusName
```

BusName

Tipo de dato: `string`

El nombre de la red de E/S que se desea poner en marcha.

### Ejecución de programas

Se pone en marcha la red de E/S cuyo nombre se especifica en el parámetro `BusName`.

### Gestión de errores

Pueden generarse los errores recuperables enumerados a continuación. Estos errores pueden ser gestionados en un gestor de errores. El valor de la variable de sistema `ERRNO` cambia a:

| Nombre                        | Causa del error                       |
|-------------------------------|---------------------------------------|
| <code>ERR_NAME_INVALID</code> | El nombre de la red de E/S no existe. |

### Sintaxis

```
IOBusStart  
[ BusName ':= ' ] < expression (IN) of string>;'
```

### Información relacionada

| Para obtener más información sobre      | Consulte   |
|---|--|
| Cómo obtener el estado de la red de E/S | <a href="#">IOBusState - Obtener el estado actual de una red de E/S en la página 290</a> |
| Configuración de E/S                    | <a href="#">Manual de referencia técnica - Parámetros del sistema</a>                    |

# 1 Instrucciones

---

## 1.108 IOBusState - Obtener el estado actual de una red de E/S

RobotWare - OS

## 1.108 IOBusState - Obtener el estado actual de una red de E/S

---

### Utilización

IOBusState se usa para leer el estado de una red de E/S determinada. Su estado físico y estado lógico definen el estado de una red de E/S.

---

### Ejemplos básicos

Los ejemplos siguientes ilustran la instrucción IOBusState.

#### Ejemplo 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
```

Esta instrucción devuelve el estado físico de la red de E/S de IBS en la variable bstate del tipo busstate.

#### Ejemplo 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
```

Esta instrucción devuelve el estado lógico de la red de E/S de IBS en la variable bstate del tipo busstate.

---

### Argumentos

```
IOBusState BusName State [\Phys] | [\Logic]
```

BusName

Tipo de dato: string

El nombre de la red de E/S cuyo estado se desea averiguar.

State

Tipo de dato: busstate

La variable en la que se devuelve el estado de la red de E/S. Consulte los datos predefinidos del tipo busstate que aparecen más abajo, en Ejecución de programas.

*Continúa en la página siguiente*

[ \Phys ]

### Physical

Tipo de dato: switch

Si se utiliza este parámetro, se lee el estado físico de la red de E/S .

[ \Logic ]

### Logical

Tipo de dato: switch

Si se utiliza este parámetro, se lee el estado lógico de la red de E/S.

## Ejecución de programas

Devuelve en el parámetro `State` el estado de la red de E/S especificado en el parámetro `BusName`.

Los estados lógicos de la red de E/S describen el estado cuya activación puede solicitar el usuario. El estado de la red de E/S se define en la tabla siguiente cuando se utiliza el argumento opcional `\Logic`.

| Valor de retorno | Constante simbólica     | Comentario  |
|------------------|-------------------------|---|
| 10               | IOBUS_LOG_STATE_STOPPED | El bus se ha detenido debido a un error <sup>2)</sup> |
| 11               | IOBUS_LOG_STATE_STARTED | El bus se ha iniciado <sup>1)</sup>                   |

El estado físico de la red de E/S describe el estado cuya activación puede ordenar el controlador de bus de campo. El estado de la red de E/S se define en la tabla siguiente cuando se utiliza el argumento opcional `\Phys`.

| Valor de retorno | Constante simbólica      | Comentario  |
|------------------|--------------------------|---|
| 20               | IOBUS_PHYS_STATE_HALTED  | El bus se ha detenido <sup>3)</sup>   |
| 21               | IOBUS_PHYS_STATE_RUNNING | Bus en funcionamiento <sup>1)</sup>   |
| 22               | IOBUS_PHYS_STATE_ERROR   | El bus no está trabajando <sup>2)</sup>   |
| 23               | IOBUS_PHYS_STATE_STARTUP | El bus está en el modo de puesta en marcha. No se comunica con ningún dispositivo de E/S. |
| 24               | IOBUS_PHYS_STATE_INIT    | El bus sólo ha sido creado <sup>3)</sup>  |



### Nota

El estado de la red de E/S se define en la tabla siguiente cuando no se utiliza ninguno de los argumentos opcionales, `\Phys` ni `\Logic`.

| Valor de retorno | Constante simbólica | Comentario                              |
|------------------|---------------------|---|
| 0                | BUSSTATE_HALTED     | El bus se ha detenido <sup>3)</sup>     |
| 1                | BUSSTATE_RUN        | Bus en funcionamiento <sup>1)</sup>     |
| 2                | BUSSTATE_ERROR      | El bus no está trabajando <sup>2)</sup> |

Continúa en la página siguiente

# 1 Instrucciones

## 1.108 IOBusState - Obtener el estado actual de una red de E/S

RobotWare - OS

Continuación

| Valor de retorno | Constante simbólica | Comentario  |
|------------------|---------------------|---|
| 3                | BUSSTATE_STARTUP    | El bus está en el modo de puesta en marcha. No se comunica con ningún dispositivo de E/S. |
| 4                | BUSSTATE_INIT       | El bus sólo ha sido creado <sup>3)</sup>  |

1) Si la red de E/S está funcionando, el estado devuelto en el argumento `State` de la instrucción `IOBusState` puede ser `IOBUS_LOG_STATE_STARTED`, `IOBUS_PHYS_STATE_RUNNING` o `BUSSTATE_RUN` en función de si se utilizan o no parámetros opcionales en `IOBusState`.

2) Si la red de E/S se ha detenido debido a algún error, el estado devuelto en el argumento `State` puede ser `IOBUS_LOG_STATE_STOPPED`, `IOBUS_PHYS_STATE_ERROR` o `BUSSTATE_ERROR` en función de si se usan parámetros opcionales en `IOBusState`.

3) No es posible obtener este estado en el programa de RAPID con la versión actual de Robotware - OS.

### Gestión de errores

Pueden generarse los errores recuperables enumerados a continuación. Estos errores pueden ser gestionados en un gestor de errores. El valor de la variable de sistema `ERRNO` cambia a:

| Nombre                        | Causa del error                       |
|-------------------------------|---------------------------------------|
| <code>ERR_NAME_INVALID</code> | El nombre de la red de E/S no existe. |

### Sintaxis

```
IOBusState
  [ BusName ':= ' ] < expression (IN) of string> ', '
  [ State ':= ' ] < variable (VAR) of busstate>
  [ '\ ' Phys ] | [ '\ ' Logic ] ';' ;
```

### Información relacionada

| Para obtener más información sobre         | Consulte   |
|--|--|
| Definición de estado de la red de E/S      | <a href="#">busstate - Estado de la red de E/S en la página 1613</a> |
| Inicio de la red de E/S                    | <a href="#">IOBusStart - Start of I/O network en la página 289</a>   |
| Funcionalidad de entrada/salida en general | <i>Manual de referencia técnica - RAPID Overview</i>                 |
| Configuración de E/S                       | <i>Manual de referencia técnica - Parámetros del sistema</i>         |



## 1.109 IODisable - Desactivar un dispositivo de E/S

### Utilización

`IODisable` se utiliza para desactivar un dispositivo de E/S durante la ejecución del programa.

Los dispositivos de E/S se activan de forma automática después de la puesta en marcha si están definidos en los parámetros del sistema. En las ocasiones en que sea necesario, es posible desactivar o activar los dispositivos de E/S durante la ejecución del programa.



#### Nota

No se puede desactivar un dispositivo de E/S cuyo `Unit Trustlevel` tenga el valor `Required`.

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción `IODisable`:

Consulte también [Más ejemplos en la página 294](#).

#### Ejemplo 1

```
CONST string board1:="board1";  
IODisable board1, 5;
```

Desactivar un dispositivo de E/S denominado `board1`. Esperar un máximo de 5 segundos.

### Argumentos

```
IODisable UnitName MaxTime
```

UnitName

Tipo de dato: `string`

El nombre de un dispositivo de E/S (el nombre de dispositivo debe estar presente en los parámetros del sistema).

MaxTime

Tipo de dato: `num`

El periodo máximo permitido para el tiempo de espera, expresado en segundos. Si se agota este tiempo antes de que el dispositivo de E/S haya finalizado los pasos de desactivación, se llamará al gestor de errores, si lo hay, con el código de error `ERR_IODISABLE`. Si no hay ningún gestor de errores, se detiene la ejecución del programa. Los pasos de desactivación del dispositivo de E/S siempre continúan con independencia de `MaxTime` o del error.

La desactivación de un dispositivo de E/S requiere de 0 a 5 segundos aproximadamente.

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.109 IODisable - Desactivar un dispositivo de E/S

RobotWare - OS

Continuación

### Ejecución de programas

El dispositivo de E/S especificado inicia los pasos de desactivación. La instrucción está preparada tan pronto como terminen los pasos de la desactivación. Si se agota el tiempo límite `MaxTime` antes de que el dispositivo de E/S haya finalizado los pasos de desactivación, se genera un error recuperable.

Después de la desactivación de un dispositivo de E/S, el establecimiento de salidas de esta unidad dará lugar a un error.

### Gestión de errores

Pueden generarse los errores recuperables enumerados a continuación. Estos errores pueden ser gestionados en un gestor de errores. El valor de la variable de sistema `ERRNO` cambia a:

| Nombre                        | Causa del error  |
|-------------------------------|--|
| <code>ERR_IODISABLE</code>    | El tiempo límite de espera se alcanza antes de que el dispositivo de E/S se desactive.                             |
| <code>ERR_NAME_INVALID</code> | El nombre del dispositivo de E/S no existe.  |
| <code>ERR_TRUSTLEVEL</code>   | El dispositivo de E/S no puede desactivarse si <code>Unit Trustlevel</code> tiene el valor <code>Required</code> . |

### Más ejemplos

A continuación aparecen más ejemplos de la instrucción `IODisable`.

#### Ejemplo 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";
```

Continúa en la página siguiente

```

        Stop;
    ENDIF
ENDPROC

```

Con el fin de ahorrar tiempo de ciclo, se desactiva el dispositivo de E/S `board1` durante el movimiento del robot hasta la posición `home`. Cuando el robot se encuentra en la posición `home`, se realiza una comprobación para determinar si el dispositivo de E/S `board1` se ha desactivado completamente. Después del número máximo de reintentos (4 con un tiempo de espera de 5 s), la ejecución del robot se detiene con un mensaje de error.

El mismo principio puede usarse con `IOEnable` (de esta forma, se ahorra tiempo de ciclo en comparación con `IODisable`).

### Sintaxis

```

IODisable
    [ UnitName ':= ' ] < expression (IN) of string> ', '
    [ MaxTime ':= ' ] < expression (IN) of num> '; '

```

### Información relacionada

| Para obtener más información sobre   | Consulte  |
|--------------------------------------|---|
| Activación de un dispositivo de E/S  | <a href="#">IOEnable - Activar un dispositivo de E/S en la página 296</a> |
| Instrucciones de entrada/salida      | Manual de referencia técnica - <i>RAPID Overview</i>                      |
| Función de entrada/salida en general | Manual de referencia técnica - <i>RAPID Overview</i>                      |
| Configuración de E/S                 | Manual de referencia técnica - <i>Parámetros del sistema</i>              |

# 1 Instrucciones

---

## 1.110 IOEnable - Activar un dispositivo de E/S

RobotWare - OS

## 1.110 IOEnable - Activar un dispositivo de E/S

---

### Utilización

IOEnable se utiliza para activar un dispositivo de E/S durante la ejecución del programa.

Los dispositivos de E/S se activan de forma automática después del inicio, si están definidos en los parámetros del sistema. En las ocasiones en que sea necesario, los dispositivos de E/S pueden desactivarse y activarse durante la ejecución del programa.

La acción del controlador al activar un dispositivo de E/S depende del Unit Trustlevel definido en los parámetros del sistema.

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción IOEnable:

Consulte también [Más ejemplos en la página 297](#).

#### Ejemplo 1

```
CONST string board1:="board1";
IOEnable board1, 5;
```

Activar un dispositivo de E/S denominado board1. El tiempo de espera es de 5 segundos.

### Argumentos

```
IOEnable UnitName MaxTime
```

UnitName

Tipo de dato: string

El nombre de un dispositivo de E/S (el nombre de un dispositivo de E/S debe estar presente en los parámetros del sistema).

MaxTime

Tipo de dato: num

El periodo máximo permitido para el tiempo de espera, expresado en segundos. Si se agota este tiempo antes de que el dispositivo de E/S haya finalizado los pasos de activación, se llamará al gestor de errores, si lo hay, con el código de error ERR\_IOENABLE. Si no hay ningún gestor de errores, se detiene la ejecución. Los pasos de activación del dispositivo de E/S siempre continúan con independencia de MaxTime o del error.

La activación de un dispositivo de E/S requiere de 2 a 5 segundos aproximadamente.

### Ejecución de programas

El dispositivo de E/S especificado inicia los pasos de activación. La instrucción queda preparada tan pronto como terminan los pasos de la activación. Si se agota el tiempo límite MaxTime antes de que el dispositivo de E/S haya finalizado los pasos de activación, se genera un error recuperable.

*Continúa en la página siguiente*

Después de una secuencia de IODisable - IOEnable, todas las salidas del dispositivo de E/S actual vuelven a los valores anteriores (antes de IODisable).

## Gestión de errores

Pueden generarse los errores recuperables enumerados a continuación. Estos errores pueden ser gestionados en un gestor de errores. El valor de la variable de sistema ERRNO cambia a:

| Nombre           | Causa del error  |
|------------------|--|
| ERR_IOENABLE     | El tiempo límite se agota antes de que el dispositivo de E/S se active.  |
| ERR_NAME_INVALID | El nombre del dispositivo de E/S no existe   |
| ERR_BUSSTATE     | La red de E/S se encuentra en el estado de error o entra en el estado de error antes de que el dispositivo de E/S se active. |

## Más ejemplos

IOEnable también puede utilizarse para comprobar si algún dispositivo de E/S está desconectado por algún motivo.

A continuación aparecen más ejemplos de cómo usar la instrucción IOEnable.

### Ejemplo 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

```

Antes de usar señales del dispositivo de E/S board1, se realiza una comprobación mediante un intento de activación del dispositivo de E/S con un tiempo límite de 0 segundos. Si la comprobación falla, se salta al gestor de errores. En el gestor de errores, la ejecución del programa espera durante 1 segundo y se hace un nuevo intento. Después de 4 reintentos, el error ERR\_IOENABLE se propaga hacia la rutina desde la que se llama a esta rutina de prueba.

## Sintaxis

```

IOEnable
  [ UnitName ':' ] < expression (IN) of string>' , '

```

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.110 IOEnable - Activar un dispositivo de E/S

RobotWare - OS

Continuación

```
[ MaxTime' :=' ] < expression (IN) of num > ';' 
```

---

### Información relacionada

| Para obtener más información sobre         | Consulte  |
|--|---|
| Desactivación de un dispositivo de E/S     | <a href="#">IODisable - Desactivar un dispositivo de E/S en la página 293</a> |
| Instrucciones de entrada/salida            | <i>Manual de referencia técnica - RAPID Overview</i>                          |
| Funcionalidad de entrada/salida en general | <i>Manual de referencia técnica - RAPID Overview</i>                          |
| Configuración de E/S                       | <i>Manual de referencia técnica - Parámetros del sistema</i>                  |

## 1.111 IOEventMessage: activar/desactivar mensajes de eventos de E/S desde el dispositivo

### Utilización

IOEventMessage se utiliza para definir si deben enviarse mensajes de eventos de E/S desde el dispositivo de E/S.

Normalmente, puede utilizarse para un ciclo de alimentación planificado o una desconexión del dispositivo de E/S, por ejemplo durante un cambio de herramienta con FastDeviceStartup. Para obtener más información acerca de FastDeviceStartup, consulte *Application manual - PROFINET Controller/Device 3HAC065546--001*.

### Ejemplos básicos

El ejemplo que se muestra a continuación ilustra la instrucción IOEventMessage:

#### Ejemplo

```
PROC IOEventMessage(switch \On | switch \Off, string DeviceName);
```

### Argumentos

```
IOEventMessage [\On] | [ \Off] DeviceName
```

[\On]

**Tipo de dato:** switch

**Activar los mensajes de eventos de E/S desde el dispositivo.**

[\Off]

**Tipo de dato:** switch

**Desactivar los mensajes de eventos de E/S desde el dispositivo.**

### Ejecución de programas

```
PERS string TC_IO := "ToolChanger_IO"
! I/O Device name

PROC setTool(num Tool)
! We will now switch tool and thus shut down the current active
  tool, we stop event logs while performing toolchange
IOEventMessage \Off, TC_IO;
! Stop Event Messages

IF (tool = 1) THEN
  SetDO Tool2_PS, 0;
  ! shut down Power supply Tool 2
  WaitDi Tool_Active, 0\MaxTime:=1;
  ! wait for device to shut down

  SetDO Tool1_PS, 1;
  ! power up Tool 1
  WaitDi Tool_Active, 1\MaxTime:=1;
  ! wait for device to activate
```

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.111 IOEventMessage: activar/desactivar mensajes de eventos de E/S desde el dispositivo

RobotWare - OS

Continuación

```
! Check device is up and running OK
IF (IOUnitState (TC_IO) = IOUNIT_RUNERROR) THEN
  Log "ERROR: Unit is not working because of some runtime
  error.";

ELSEIF (IOUnitState (TC_IO) = IOUNIT_OTHERERR) THEN
  Log "ERROR: Other configuration or startup errors.";
ENDIF

IF (tool = 2) THEN
  !Same as above but for tool 2

! Toolchange finished, now we want event logs active again
IOEventMessage \On, TC_IO;
! Turn Event Messages

ENDPROC
```

---

### Gestión de errores

Pueden generarse los errores recuperables enumerados a continuación. Estos errores pueden ser gestionados en un gestor de errores. El valor de la variable de sistema `ERRNO` cambia a:

| Nombre                        | Causa del error                            |
|-------------------------------|--|
| <code>ERR_NAME_INVALID</code> | El nombre del dispositivo de E/S no existe |

---

### Sintaxis

```
IOEventMessage
[ '\ On ] | [ '\ Off ]
[ DeviceName ' := ' ] < expression (IN) of string > ' ; '
```

---

### Información relacionada

| Para obtener más información sobre      | Consulte   |
|---|--|
| Puesta en marcha rápida del dispositivo | <i>Application manual - PROFINET Controller/Device</i> |



## 1.112 IPathPos - Obtener valor robtarget de la línea central en la oscilación Continuous Application Platform (CAP)

### 1.112 IPathPos - Obtener valor robtarget de la línea central en la oscilación

#### Utilización

IPathPos se utiliza para obtener la posición de la línea central durante la oscilación con CAP.

Esta función se utiliza principalmente junto con la funcionalidad de seguimiento. Es necesario activar la oscilación y las señales de sincronización tanto en el lado izquierdo como el derecho.

#### Ejemplo básico

```
connect intpt, TRP_ipathpos IPathPos p_robt, sen_pos, intpt;
```

Cuando `p_robt` obtiene un nuevo valor calculado, se envía la interrupción `intpt` y se ejecuta la rutina TRAP `TRP_ipathpos`.

#### Argumentos

```
IPathPos p_robt, sen_pos, intpt [\NoDispl] [\EOffs]
```

#### `p_robt`

Tipo de dato: `robtarget`

`p_robt` mantiene el valor más reciente de `robtarget` calculado.

#### `sen_pos`

Tipo de dato: `pos`

`sen_pos` no se utiliza.

#### `intpt`

Tipo de dato: `intno`

`intpt` especifica la interrupción que se recibirá cada vez que se asigne un nuevo valor a `p_robt`.

#### `[\NoDispl]`

Tipo de dato: `switch`

Si se especifica `\NoDispl`, el valor devuelto en PERS `p_robt` no incluirá ningún desplazamiento que pudiera especificarse mediante las instrucciones `PDispSet` y `PDispOn` de RAPID.

#### `[\EOffs]`

Tipo de dato: `switch`

Si se especifica `[\EOffs]`, el valor devuelto en PERS `p_robt` incluirá cualquier desplazamiento que pudiera especificarse mediante la instrucción `EOffsSet` de RAPID.

#### Limitaciones

Es necesario activar la oscilación y la sincronización de oscilación (con o sin seguimiento).

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.112 IPathPos - Obtener valor robtarget de la línea central en la oscilación

### Continuous Application Platform (CAP)

#### Continuación

#### Sintaxis

```
IPathPos
  [p_robt ':='] < persistent (PERS) of robtarget > ','
  [sen_pos ':='] < persistent (PERS) of pos > ','
  [Interrupt ':='] < variable (IN) of intnum >
  ['\ ' EOffs ]
  ['\ ' NoDispl ] ';'

```

#### Información relacionada

| Para obtener más información sobre     | Consulte  |
|--|---|
| <i>Continuous Application Platform</i> | <i>Application manual - Continuous Application Platform</i>   |
| Instrucción CapWeaveSync               | <a href="#">CapWeaveSync - Configurar señales y niveles para la sincronización de oscilación en la página 128</a> |
| Instrucción CapAPTrSetup               | <a href="#">CapAPTrSetup - Configuración de un seguimiento de punto en la página 83</a>                           |
| Instrucción CapLATrSetup               | <a href="#">CapLATrSetup - Configurar un sensor de seguimiento anticipatorio en la página 118</a>                 |

---

## 1.113 IPers - Interrupción en caso de cambio de valor de una variable persistente

---

### Utilización

IPers (*Interrupt Persistent*) se utiliza para solicitar y activar la generación de interrupciones en caso de cambio de valor de una variable persistente.

---

### Ejemplos básicos

Los ejemplos que aparecen a continuación ilustran la instrucción IPers:

#### Ejemplo 1

```
VAR intnum perslint;
PERS num counter := 0;

PROC main()
  CONNECT perslint WITH iroutine1;
  IPers counter, perslint;
  ...
  IDelete perslint;
ENDPROC

TRAP iroutine1
  TPWrite "Current value of counter = " \Num:=counter;
ENDTRAP
```

Solicita una interrupción que debe tener lugar cada vez que cambie el valor de la variable persistente `counter`. En este caso, se realiza una llamada a la rutina `TRAP iroutine1`.

---

### Argumentos

```
IPers [ \Single ] | [ \SingleSafe ] Name Interrupt
```

[ \Single ]

Tipo de dato: switch

Especifica si la interrupción debe producirse una sola vez o de forma cíclica.

Si se utiliza el argumento `Single`, la interrupción se produce como máximo una sola vez. Si se omiten los argumentos `Single` y `SingleSafe`, se genera una interrupción cada vez que se cumpla la condición.

[ \SingleSafe ]

Tipo de dato: switch

Especifica que la interrupción es única y segura. Para la definición de única, consulte la descripción del argumento `Single`. Una interrupción segura no puede ponerse en reposo con la instrucción `ISleep`. El evento de interrupción segura se almacena en la cola en caso de paro del programa y ejecución paso a paso y cuando se inicia de nuevo en el modo continuo, la interrupción se ejecuta. El único momento en el que una interrupción segura se desecha es cuando la cola de interrupciones está llena. En este caso se genera un error. La interrupción no sobrevive al restablecimiento del programa, por ejemplo el traslado del PP a main.

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.113 IPers - Interrupción en caso de cambio de valor de una variable persistente

RobotWare - OS

Continuación

Name

Tipo de dato: anytype

La variable persistente a partir de la cual deben generarse las interrupciones.

Puede usar todo tipo de datos, como atómico, registro, componente de registro, matriz o elemento de matriz.

Interrupt

Tipo de dato: intnum

La identidad de la interrupción. La interrupción debe estar ya conectada a una rutina TRAP mediante la instrucción `CONNECT`.

### Ejecución de programas

En el momento en el que la variable persistente cambia de valor, se realiza una llamada a la rutina TRAP correspondiente. Una vez ejecutada la rutina, la ejecución del programa continúa a partir del punto en el que se produjo la interrupción.

Si la variable persistente cambia de valor durante un paro de programa, no tendrá lugar ninguna interrupción cuando el programa se inicie de nuevo.

### Limitaciones

No es posible utilizar más de una vez la identidad de la interrupción sin eliminarla previamente. Consulte la instrucción `ISignalDI`.

Si se indica una parte de un dato, por ejemplo un componente de registro o un elemento de matriz, en el parámetro `Name`, la interrupción tendrá lugar cada vez que se cambie cualquier parte del dato.

Al ejecutar la rutina TRAP y leer el valor de la variable persistente, no existe ninguna garantía de que el valor leído sea el que disparó la interrupción.

### Sintaxis

```
IPers
[ '\ ' Single ] | [ '\ ' SingleSafe ] ','
[ Name ' := ' ] < persistent ( PERS ) of anytype > ','
[ Interrupt ' := ' ] < variable ( VAR ) of intnum > ';'

```

### Información relacionada

| Para obtener más información sobre                    | Consulte  |
|---|---|
| Resumen de interrupciones y gestión de interrupciones | <i>Manual de referencia técnica - RAPID Overview</i>  |
| Interrupción a partir de una señal de entrada         | <a href="#">ISignalDI - Solicita interrupciones a partir de una señal digital de entrada en la página 323</a> |
| Identidad de interrupción                             | <a href="#">intnum - Identidad de interrupción en la página 1687</a>  |
| <i>Advanced RAPID</i>                                 | <i>Application manual - Controller software IRC5</i>  |

## 1.114 IRMQMessage - Ordenar interrupciones de RMQ para un tipo de dato *FlexPendant Interface, PC Interface, or Multitasking*

### 1.114 IRMQMessage - Ordenar interrupciones de RMQ para un tipo de dato

#### Utilización

`IRMQMessage` (*Interrupt RAPID Message Queue Message*) se usa para solicitar y activar interrupciones para un tipo de dato en concreto al utilizar la función `RMQ`.

#### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción `IRMQMessage`:

Consulte también [IRMQMessage - Ordenar interrupciones de RMQ para un tipo de dato en la página 305](#).

#### Ejemplo 1

```
VAR intnum rmqint;  
VAR string dummy;  
...  
PROC main()  
    CONNECT rmqint WITH iroutinel;  
    IRMQMessage dummy, rmqint;
```

Solicita una interrupción que debe producirse cada vez que se reciba un nuevo `rmqmessage` que contenga el tipo de dato `string`. En este caso, se realiza una llamada a la rutina `TRAPiroutinel`.

#### Argumentos

```
IRMQMessage InterruptDataType Interrupt
```

##### InterruptDataType

**Tipo de dato:** `anytype`

Una referencia a una variable, una variable persistente o una constante de un tipo de dato que generará una interrupción cuando se reciba un `rmqmessage` con el tipo de dato especificado.

##### Interrupt

**Tipo de dato:** `intnum`

La identidad de la interrupción. La interrupción debe estar ya conectada a una rutina `TRAP` mediante la instrucción `CONNECT`.

#### Ejecución de programas

En el momento en el que se recibe el mensaje `RMQ` con el tipo de dato especificado, se realiza una llamada a la rutina `TRAP` correspondiente. Una vez ejecutada la rutina, la ejecución del programa continúa a partir del punto en el que se produjo la interrupción.

Todos los mensajes que contengan datos del mismo tipo de dato, independientemente de su número de dimensiones, serán gestionadas por la misma interrupción. Si se utilizan diferentes dimensiones, utilice `RMQGetMsgHeader` para adaptarse a ello.

Cualquier mensaje que contenga datos de un tipo de dato que no tiene conectada ninguna interrupción generará un aviso.

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.114 IRMQMessage - Ordenar interrupciones de RMQ para un tipo de dato

*FlexPendant Interface, PC Interface, or Multitasking*

*Continuación*

La instrucción `RMQSendWait` tiene la máxima prioridad si se recibe un mensaje y éste se corresponde con la descripción tanto de la respuesta esperada como de un mensaje conectado a una rutina TRAP con la instrucción `IRMQMessage`.

No todos los tipos de datos pueden usarse en el argumento `InterruptDataType` (consulte las limitaciones).

Se considera que la interrupción es una interrupción segura. Una interrupción segura no puede ponerse en reposo con la instrucción `ISleep`. El evento de interrupción segura se coloca en la cola en caso de paro del programa y de ejecución paso a paso. La interrupción se ejecuta al iniciar nuevamente el modo continuo. El único momento en el que una interrupción segura se desecha es cuando la cola de interrupciones está llena. En este caso se genera un error. La interrupción no sobrevive al restablecimiento del programa, por ejemplo, PP a main.

---

### Más ejemplos

A continuación aparecen más ejemplos de cómo usar la instrucción `IRMQMessage`.

#### Ejemplo 1

```
MODULE ReceiverMod
  VAR intnum intnol;
  VAR rmqheader rmqheader1;
  VAR rmqslot rmqslot1;
  VAR rmqmessage rmqmessage1;

  PROC main()
    VAR string interrupt_on_str := stEmpty;
    CONNECT intnol WITH RecMsgs;
    ! Set up interrupts for data type string
    IRMQMessage interrupt_on_str, intnol;

    ! 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);
```

*Continúa en la página siguiente*

## 1.114 IRMQMessage - Ordenar interrupciones de RMQ para un tipo de dato *FlexPendant Interface, PC Interface, or Multitasking* Continuación

```
TPWrite "Rec string: " + receivestr;
TPWrite "User Def: " + ValToStr(userdef);
TPWrite "From: " + client_name;
ELSE
  TPWrite "Faulty data received!"
ENDIF

ENDTRAP
ENDMODULE
```

Este ejemplo muestra cómo configurar interrupciones para un tipo de dato en concreto. Cuando se recibe un mensaje, se ejecuta la rutina `TRAPRecMsgs` y los datos recibidos en el mensaje se muestran en el FlexPendant. Si el tipo de dato recibido o la dimensión de los datos son distintos de lo esperado, esta situación se muestra en el FlexPendant.

### Limitaciones

No se permite ejecutar `IRMQMessage` en el modo sincronizado. Provocaría un error de tiempo de ejecución no recuperable.

No es posible usar en interrupciones, enviar ni recibir instancias de tipos de datos que no tienen valor, son de semivalor o corresponden al tipo de dato `motsetdata`.

No es posible utilizar más de una vez la identidad de la interrupción sin eliminarla previamente. Por tanto, las interrupciones deben tratarse de la forma mostrada, con una de las alternativas siguientes.

```
VAR intnum rmqint;
PROC main ( )
  VAR mytype dummy;
  CONNECT rmqlint WITH iroutinel;
  IRMQMessage dummy, rmqint;
  WHILE TRUE DO
    ...
  ENDWHILE
ENDPROC
```

Al principio del programa se produce la activación de todas las interrupciones. En este caso, las instrucciones iniciales se mantienen fuera del flujo principal del programa.

```
VAR intnum rmqint;
PROC main ( )
  VAR mytype dummy;
  CONNECT rmqint WITH iroutinel;
  IRMQMessage dummy, rmqint;
  ...
  IDelete rmqint;
ENDPROC
```

La interrupción se elimina al final del programa y se activa de nuevo. En este caso, recuerde que la interrupción permanece inactiva durante un periodo breve.

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.114 IRMQMessage - Ordenar interrupciones de RMQ para un tipo de dato

*FlexPendant Interface, PC Interface, or Multitasking*

*Continuación*

### Sintaxis

```
IRMQMessage  
[ InterruptDataType ':' ] < reference (REF) of anytype >  
[ Interrupt ':' ] < variable (VAR) of intnum >';'
```

### Información relacionada

| Para obtener más información sobre  | Consulte  |
|---|---|
| Descripción de la funcionalidad de RAPID Message Queue  | <a href="#">Application manual - Controller software IRC5</a>   |
| Enviar datos a la cola de una tarea de RAPID o de un cliente de Robot Application Builder.                                  | <a href="#">RMQFindSlot - Buscar una identidad de ranura para el nombre de ranura en la página 624</a>    |
| Obtener el primer mensaje de una cola de RAPID Message Queue.   | <a href="#">RMQGetMessage - Obtener un mensaje de RMQ en la página 626</a>                                |
| Enviar datos a la cola de una tarea de RAPID o un cliente de Robot Application Builder y esperar una respuesta del cliente. | <a href="#">RMQSendWait - Enviar un mensaje de datos de RMQ y esperar una respuesta en la página 642</a>  |
| Extraer los datos de encabezado de un <code>rmqmessage</code> .   | <a href="#">RMQGetMsgHeader - Obtener información de encabezado de un mensaje de RMQ en la página 632</a> |
| Enviar datos a la cola de una tarea de RAPID o de un cliente de Robot Application Builder.                                  | <a href="#">RMQSendMessage - Enviar un mensaje de datos de RMQ en la página 638</a>                       |
| Extraer los datos de un <code>rmqmessage</code> .   | <a href="#">RMQGetMsgData - Obtener la parte de datos de un mensaje de RMQ en la página 629</a>           |
| Obtener el nombre de ranura de una identidad de ranura especificada.  | <a href="#">RMQGetSlotName - Obtener el nombre de un cliente de RMQ en la página 1456</a>                 |



---

## 1.115 ISignalAI - Interrupciones a partir de una señal analógica de entrada

---

### Utilización

ISignalAI (*Interrupt Signal Analog Input*) se utiliza para solicitar y activar interrupciones a partir de una señal analógica de entrada.

---

### Ejemplos básicos

Los ejemplos siguientes ilustran la instrucción ISignalAI.

#### Ejemplo 1

```
VAR intnum siglint;
PROC main()
  CONNECT siglint WITH iroutine1;
  ISignalAI \Single, ail, AIO_BETWEEN, 1.5, 0.5, 0, siglint;
```

Solicita una interrupción que debe producirse la primera vez que el valor lógico de la señal analógica de entrada `ail` se encuentre entre 0.5 y 1.5. En este caso, se realiza una llamada a la rutina TRAP `iroutine1`.

#### Ejemplo 2

```
ISignalAI ail, AIO_BETWEEN, 1.5, 0.5, 0.1, siglint;
```

Solicita una interrupción que debe producirse cada vez que el valor lógico de la señal analógica de entrada `ail` se encuentre entre 0.5 y 1.5 y cuando la diferencia absoluta de la señal respecto del valor de referencia almacenado sea superior a 0.1.

#### Ejemplo 3

```
ISignalAI ail, AIO_OUTSIDE, 1.5, 0.5, 0.1, siglint;
```

Se solicita una interrupción que debe producirse cada vez que el valor lógico de la señal analógica de entrada `ail` sea inferior a 0,5 o superior a 1,5 y cuando la diferencia absoluta de la señal respecto del valor de referencia almacenado sea superior a 0,1.

---

### Argumentos

```
ISignalAI [\Single] | [\SingleSafe] Signal Condition HighValue
          LowValue DeltaValue [\DPos] | [\DNeg] Interrupt
```

`[\Single]`

Tipo de dato: `switch`

Especifica si la interrupción debe producirse una sola vez o de forma cíclica. Si se utiliza el argumento `Single`, la interrupción se produce como máximo una sola vez. Si se omiten los argumentos `Single` y `SingleSafe`, se genera una interrupción cada vez que se cumpla la condición.

`[\SingleSafe]`

Tipo de dato: `switch`

Especifica que la interrupción es única y segura. Para la definición de única, consulte la descripción del argumento `Single`. Una interrupción segura no puede ponerse en reposo con la instrucción `ISleep`. El evento de interrupción segura se coloca en la cola en caso de paro del programa y de ejecución paso a paso. La

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.115 ISignalAI - Interrupciones a partir de una señal analógica de entrada

RobotWare - OS

Continuación

interrupción se ejecuta al iniciar nuevamente el modo continuo. El único momento en el que una interrupción segura se desecha es cuando la cola de interrupciones está llena. En este caso se genera un error. La interrupción no sobrevive al restablecimiento del programa, por ejemplo, PP a main.

Signal

Tipo de dato: `signalai`

El nombre de la señal a partir de la cual deben generarse las interrupciones.

Condition

Tipo de dato: `aiotrigg`

Especifica cómo `HighValue` y `LowValue` definen la condición que debe cumplirse:

| Valor | Constante simbólica | Comentario   |
|-------|---------------------|--|
| 1     | AIO_ABOVE_HIGH      | La señal genera interrupciones si se encuentra por encima del valor máximo especificado  |
| 2     | AIO_BELOW_HIGH      | La señal genera interrupciones si se encuentra por debajo del valor máximo especificado  |
| 3     | AIO_ABOVE_LOW       | La señal genera interrupciones si se encuentra por encima del valor mínimo especificado  |
| 4     | AIO_BELOW_LOW       | La señal genera interrupciones si se encuentra por debajo del valor mínimo especificado  |
| 5     | AIO_BETWEEN         | La señal genera interrupciones si se encuentra entre los valores mínimo y máximo especificados                                     |
| 6     | AIO_OUTSIDE         | La señal genera interrupciones si se encuentra por debajo del valor mínimo especificado o por encima del valor máximo especificado |
| 7     | AIO_ALWAYS          | La señal siempre genera interrupciones   |

HighValue

Tipo de dato: `num`

El valor lógico de límite máximo utilizado para definir la condición.

LowValue

Tipo de dato: `num`

El valor lógico de límite mínimo utilizado para definir la condición.

DeltaValue

Tipo de dato: `num`

Define la diferencia mínima entre dos señales lógicas antes de que se genere una nueva interrupción. El valor actual de la señal, comparado con el valor de referencia almacenado, debe ser mayor que el valor especificado en `DeltaValue` para que se genere una nueva interrupción.

[`\DPos`]

Tipo de dato: `switch`

Especifica que sólo las diferencias positivas entre señales lógicas suponen la generación de nuevas interrupciones.

Continúa en la página siguiente

`[\DNeg]`

Tipo de dato: `switch`

Especifica que sólo las diferencias negativas entre señales lógicas suponen la generación de nuevas interrupciones.

Si no se utiliza el argumento `\DPos` ni el argumento `\DNeg` tanto las diferencias positivas como las negativas suponen la generación de nuevas interrupciones.

`Interrupt`

Tipo de dato: `intnum`

La identidad de la interrupción. La interrupción debe estar ya conectada a una rutina TRAP mediante la instrucción `CONNECT`.

---

### Ejecución de programas

En el momento en el que la señal satisface las condiciones especificadas (tanto `Condition` como `DeltaValue`), se realiza una llamada a la rutina TRAP correspondiente. Una vez ejecutada la rutina, la ejecución del programa continúa a partir del punto en el que se produjo la interrupción.

### Condiciones para la generación de interrupciones

Antes de solicitar la suscripción a una interrupción, cada vez que se muestre la señal, el valor de ésta se lee, se guarda y se utiliza posteriormente como valor de referencia para la condición `DeltaValue`.

En el momento de la suscripción de la interrupción si se ha especificado `DeltaValue = 0`, y tras el momento de la suscripción de la interrupción se muestrea la señal. A continuación, se compara el valor de la señal con `HighValue` y `LowValue` según el valor de `Condition` y teniendo en cuenta el valor de `DeltaValue` para decidir si debe generarse o no una interrupción. Si el nuevo valor leído satisface los valores especificados en `HighValue`, `LowValue` y `Condition`, pero su diferencia respecto del último valor de referencia almacenado es menor o igual al valor del argumento `DeltaValue`, no se produce ninguna interrupción. Si la diferencia entre las señales no se produce en el sentido especificado, no se genera ninguna interrupción (argumento `\DPos` o `\DNeg`).

El valor de referencia almacenado para la condición `DeltaValue` se actualiza con un nuevo valor leído para su uso en los muestreos posteriores, siempre y cuando se cumplan las condiciones siguientes:

- Argumento `Condition` con valores especificados en `HighValue` y `LowValue` (dentro de límites)
- Argumento `DeltaValue` (variación suficiente de la señal en cualquier sentido, independientemente del modificador `\DPos` o `\DNeg` especificado)

El valor de referencia sólo se actualiza en el momento del muestreo, no en el momento de la suscripción de la interrupción.

También se genera una interrupción en el momento del muestreo que se hace para actualizar el valor de referencia, siempre y cuando la diferencia entre las señales cumpla el argumento especificado (en cualquier sentido, `\DPos` o `\DNeg`).

*Continúa en la página siguiente*

# 1 Instrucciones

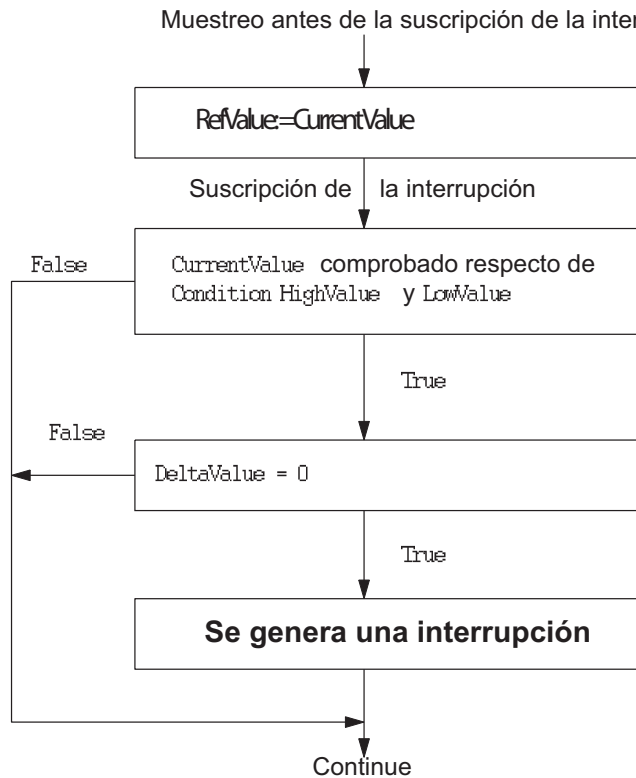
## 1.115 ISignalAI - Interrupciones a partir de una señal analógica de entrada

RobotWare - OS

Continuación

Si se utiliza el modificador `\Single`, sólo se genera una interrupción como máximo. Si el modificador `\Single` (interrupción cíclica) no se utiliza, se realiza una nueva prueba con las condiciones especificadas (tanto `Condition` como `DeltaValue`) con cada muestreo del valor de la señal. Se realiza una comparación entre el valor actual de la señal y el último valor de referencia almacenado para decidir si debe generarse una interrupción o no.

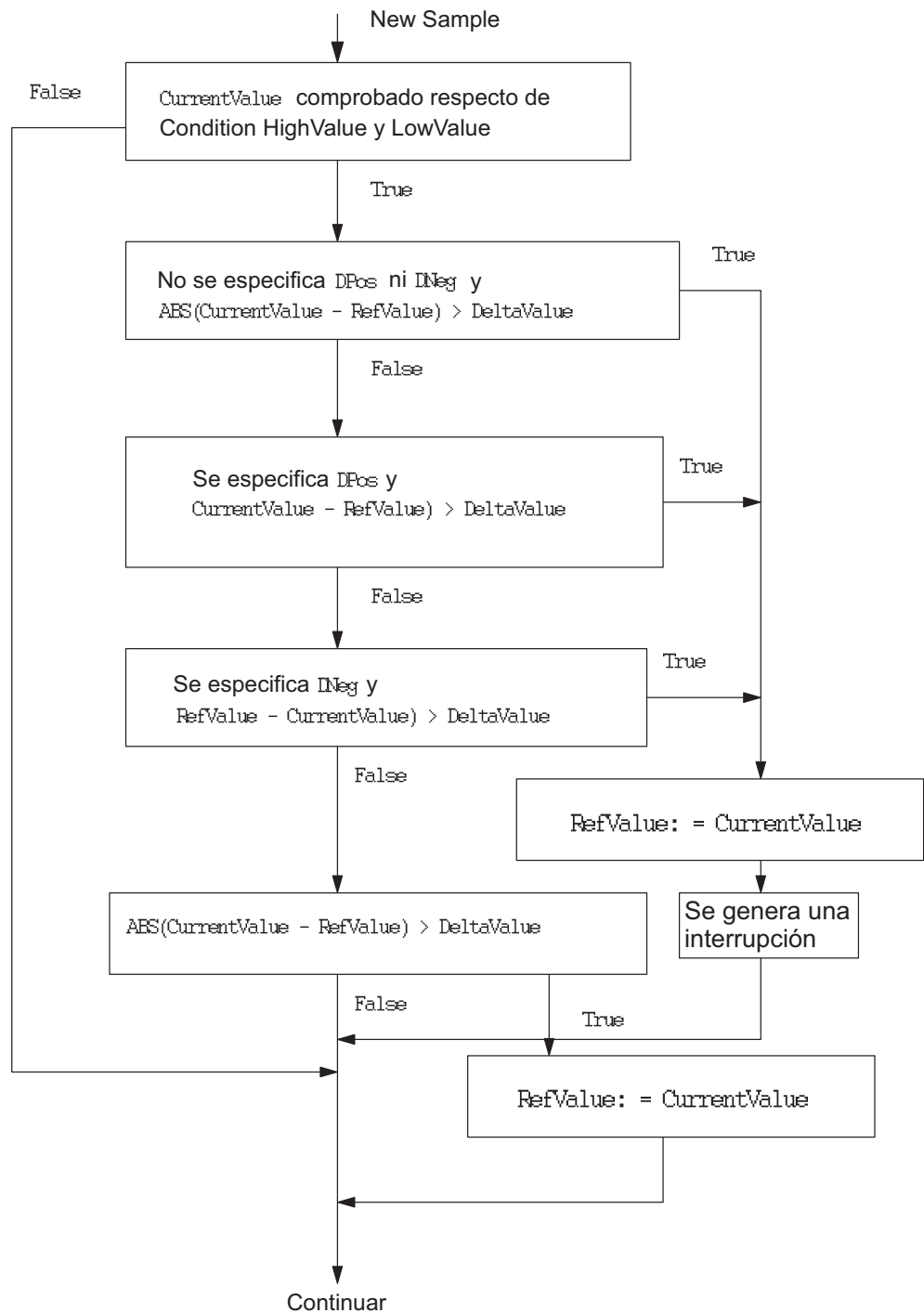
Condición para la generación de una interrupción en el momento de la suscripción



xx0500002165

Continúa en la página siguiente

Condición para la generación de una interrupción con cada muestreo a partir del momento de la suscripción



xx0500002166

Continúa en la página siguiente

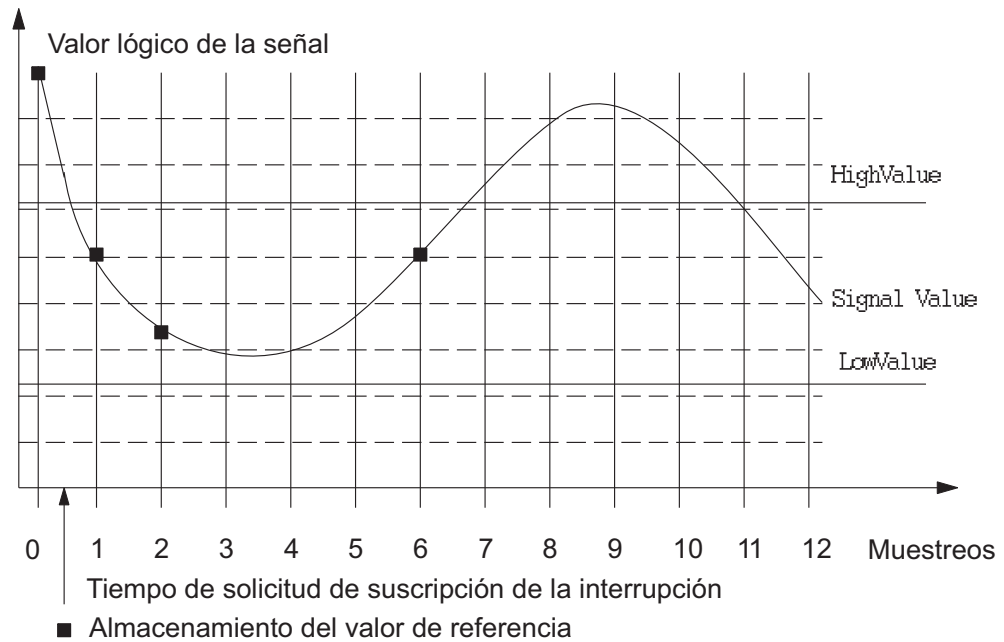
# 1 Instrucciones

## 1.115 ISignalAI - Interrupciones a partir de una señal analógica de entrada

RobotWare - OS

Continuación

### Ejemplo 1 de generación de interrupciones



xx0500002167

Suponiendo que la interrupción se solicita entre el muestreo 0 y el 1, la instrucción siguiente da lugar a los resultados siguientes:

```
ISignalAI ail, AIO_BETWEEN, 6.1, 2.2, 1.0, siglint;
```

El muestreo 1 genera una interrupción, ya que el valor de la señal se encuentra entre `HighValue` y `LowValue` y la diferencia entre las señales respecto del muestreo 0 es superior a `DeltaValue`.

El muestreo 2 genera una interrupción, ya que el valor de la señal se encuentra entre `HighValue` y `LowValue` y la diferencia entre las señales respecto del muestreo 1 es superior a `DeltaValue`.

Ni el muestreo 3, el 4 ni el 5 generan interrupciones, ya que la diferencia entre las señales es inferior a `DeltaValue`.

El muestreo 6 genera una interrupción.

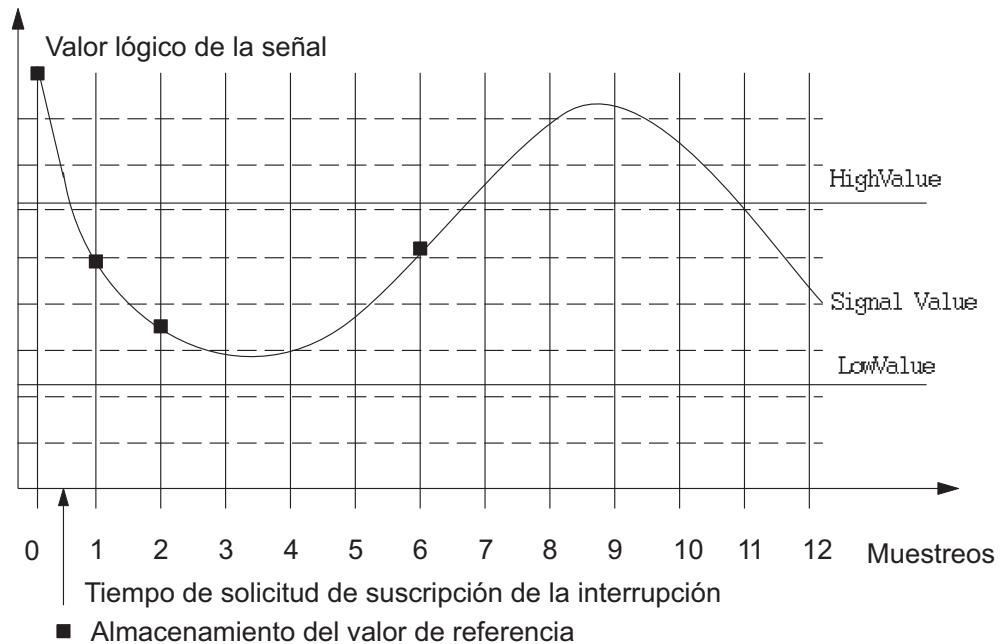
Los muestreos del 7 al 10 no generan ninguna interrupción, porque la señal se encuentra por encima de `HighValue`.

El muestreo 11 no genera ninguna interrupción, porque la diferencia entre las señales respecto del muestreo 6 es igual a `DeltaValue`.

El muestreo 12 no genera ninguna interrupción, porque la diferencia entre las señales respecto del muestreo 6 es inferior a `DeltaValue`.

Continúa en la página siguiente

## Ejemplo 2 de generación de interrupciones



xx0500002168

Suponiendo que la interrupción se solicita entre el muestreo 0 y el 1, la instrucción siguiente da lugar a los resultados siguientes:

```
ISignalAI ai1, AIO_BETWEEN, 6.1, 2.2, 1.0 \DPos, siglint;
```

Se almacena un nuevo valor de referencia en los muestreos 1 y 2, ya que la señal se encuentra dentro de los límites y el valor absoluto de la diferencia entre el valor actual y el último valor de referencia almacenado es superior a 1.0. No se genera ninguna interrupción porque las variaciones de la señal se producen en sentido negativo.

El muestreo 6 genera una interrupción, ya que el valor de la señal se encuentra entre HighValue y LowValue y la diferencia entre las señales en sentido positivo respecto del muestreo 2 es superior a DeltaValue.

Continúa en la página siguiente

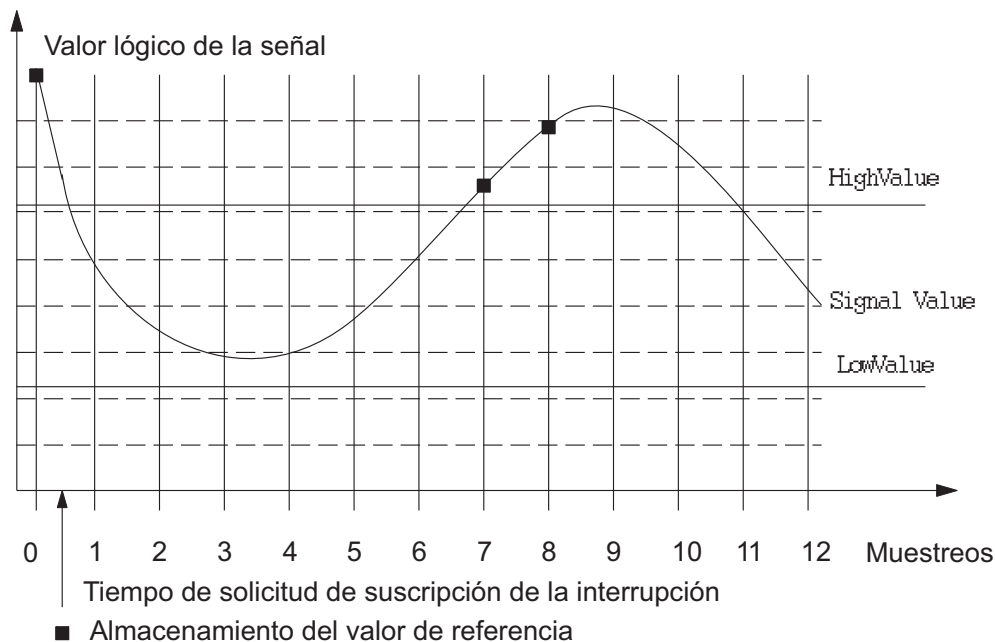
# 1 Instrucciones

## 1.115 ISignalAI - Interrupciones a partir de una señal analógica de entrada

RobotWare - OS

Continuación

### Ejemplo 3 de generación de interrupciones



xx0500002169

Suponiendo que la interrupción se solicita entre el muestreo 0 y el 1, la instrucción siguiente da lugar a los resultados siguientes:

```
ISignalAI \Single, ai1, AIO_OUTSIDE, 6.1, 2.2, 1.0 \DPos, siglint;
```

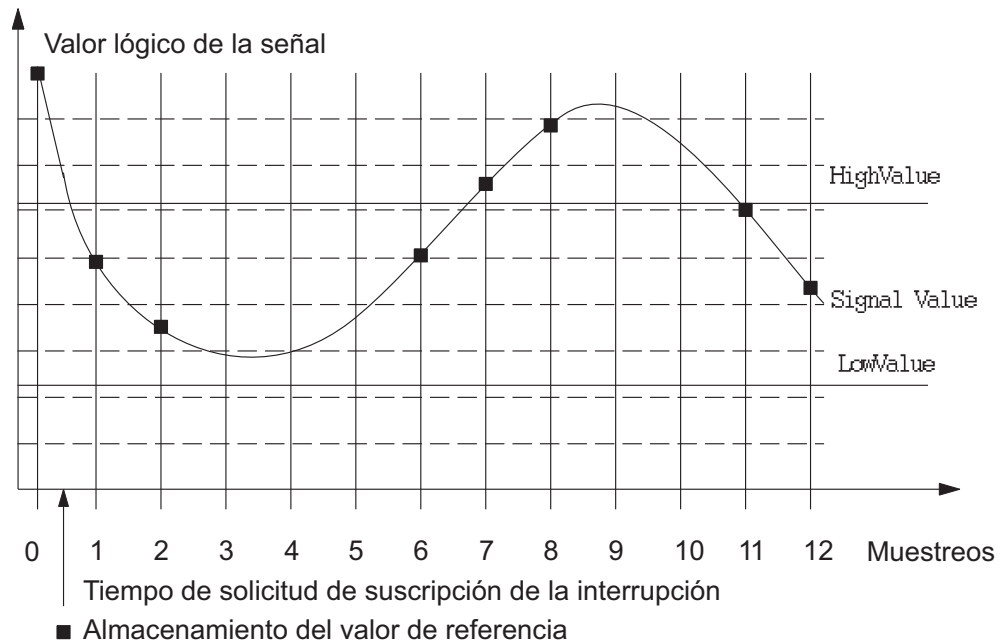
Se almacena un nuevo valor de referencia en el muestreo 7, ya que la señal se encuentra dentro de los límites y el valor absoluto de la diferencia entre el valor actual y el último valor de referencia almacenado es superior a 1,0

El muestreo 8 genera una interrupción, ya que el valor de la señal se encuentra por encima de HighValue y la diferencia en sentido positivo entre las señales respecto del muestreo 7 es superior a DeltaValue.

Continúa en la página siguiente



## Ejemplo 4 de generación de interrupciones



xx0500002170

Suponiendo que la interrupción se solicita entre el muestreo 0 y el 1, la instrucción siguiente da lugar a los resultados siguientes:

```
ISignalAI ai1, AIO_ALWAYS, 6.1, 2.2, 1.0 \DPos, siglint;
```

Se almacena un nuevo valor de referencia en los muestreos 1 y 2, ya que la señal se encuentra dentro de los límites y el valor absoluto de la diferencia entre el valor actual y el último valor de referencia almacenado es superior a 1.0

El muestreo 6 genera una interrupción, ya que la diferencia entre las señales en sentido positivo respecto del muestreo 2 es superior a `DeltaValue`.

Los muestreos 7 y 8 generan una interrupción, ya que la diferencia entre las señales en sentido positivo respecto del muestreo anterior es superior a `DeltaValue`.

Se almacena un nuevo valor de referencia en los muestreos 11 y 12, ya que la señal se encuentra dentro de los límites y el valor absoluto de la diferencia entre el valor actual y el último valor de referencia almacenado es superior a 1.0

## Gestión de errores

Si se produce la suscripción de una interrupción para una señal analógica de entrada, se genera una interrupción para cada variación en el valor analógico que cumpla la condición especificada al solicitar la suscripción de la interrupción. Si el valor analógico presenta ruido, es posible que se generen muchas interrupciones, incluso a pesar de que sólo varíen uno o dos bits del valor analógico.

Para evitar la generación de interrupciones para pequeños cambios del valor de la entrada analógica, utilice en `DeltaValue` un nivel superior a 0. A partir de ese momento, no se genera ninguna interrupción hasta que el cambio del valor analógico sea superior al valor especificado para `DeltaValue`.

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.115 ISignalAI - Interrupciones a partir de una señal analógica de entrada

RobotWare - OS

Continuación

Pueden generarse los errores recuperables enumerados a continuación. Estos errores pueden ser gestionados en un gestor de errores. El valor de la variable de sistema `ERRNO` cambia a:

| Nombre                          | Causa del error  |
|---------------------------------|--|
| <code>ERR_NO_ALIASIO_DEF</code> | La variable de señal es una variable declarada en <code>RAPID</code> y no se ha conectado a una señal de E/S definida en la configuración de E/S con la instrucción <code>AliasIO</code> . |
| <code>ERR_AO_LIM</code>         | El argumento programado <code>HighValue</code> o <code>LowValue</code> para la señal analógica de entrada especificada <code>Signal</code> está fuera de límites.                          |
| <code>ERR_NORUNUNIT</code>      | Se ha perdido el contacto con el dispositivo de E/S.   |

### Limitaciones

Los argumentos `HighValue` y `LowValue` deben estar dentro del rango: valor lógico máximo y valor lógico mínimo definidos para la señal.

`HighValue` debe ser superior a `LowValue`.

`DeltaValue` debe ser 0 o un valor positivo.

Las limitaciones de la identidad de la interrupción son las mismas que en el caso de `ISignalDI`.

### Sintaxis

```
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> ';'

```

### Información relacionada

| Para obtener más información sobre                     | Consulte  |
|--|---|
| Resumen de interrupciones y gestión de interrupciones  | <i>Manual de referencia técnica - RAPID Overview</i>  |
| Definición de constantes                               | <a href="#">aiotrigg - Condición de disparo con E/S analógica en la página 1607</a>                           |
| Interrupción a partir de una señal analógica de salida | <a href="#">ISignalAO - Interrupciones a partir de una señal analógica de salida en la página 319</a>         |
| Interrupción a partir de una señal digital de entrada  | <a href="#">ISignalDI - Solicita interrupciones a partir de una señal digital de entrada en la página 323</a> |
| Interrupción a partir de una señal digital de salida   | <a href="#">ISignalDO - Interrupciones a partir de una señal digital de salida en la página 326</a>           |
| Identidad de interrupción                              | <a href="#">intnum - Identidad de interrupción en la página 1687</a>  |
| Parámetros del sistema relacionados (filtro)           | <i>Manual de referencia técnica - Parámetros del sistema</i>  |

## 1.116 ISignalAO - Interrupciones a partir de una señal analógica de salida

### Utilización

ISignalAO (*Interrupt Signal Analog Output*) se utiliza para solicitar y activar interrupciones a partir de una señal analógica de salida.

### Ejemplos básicos

Los ejemplos siguientes ilustran la instrucción ISignalAO.

#### Ejemplo 1

```
VAR intnum siglint;
PROC main()
  CONNECT siglint WITH iroutine1;
  ISignalAO \Single, ao1, AIO_BETWEEN, 1.5, 0.5, 0, siglint;
```

Solicita una interrupción que debe producirse la primera vez que el valor lógico de la señal analógica de salida `ao1` se encuentre entre 0.5 y 1.5. En este caso, se realiza una llamada a la rutina TRAP `iroutine1`.

#### Ejemplo 2

```
ISignalAO ao1, AIO_BETWEEN, 1.5, 0.5, 0.1, siglint;
```

Solicita una interrupción que debe producirse cada vez que el valor lógico de la señal de salida analógica `ao1` se encuentre entre 0.5 y 1.5 y cuando la diferencia absoluta de la señal respecto del valor de referencia almacenado anteriormente sea superior a 0,1.

#### Ejemplo 3

```
ISignalAO ao1, AIO_OUTSIDE, 1.5, 0.5, 0.1, siglint;
```

Se solicita una interrupción que debe producirse cada vez que el valor lógico de la señal de salida analógica `ao1` sea inferior a 0.5 o superior a 1.5 y cuando la diferencia absoluta de la señal respecto del valor de referencia almacenado sea superior a 0,1.

### Argumentos

```
ISignalAO [\Single] | [\SingleSafe] Signal Condition HighValue
          LowValue DeltaValue [\DPos] | [\DNeg] Interrupt
```

`[\Single]`

Tipo de dato: `switch`

Especifica si la interrupción debe producirse una sola vez o de forma cíclica. Si se utiliza el argumento `Single`, la interrupción se produce como máximo una sola vez. Si se omiten los argumentos `Single` y `SingleSafe`, se genera una interrupción cada vez que se cumpla la condición.

`[\SingleSafe]`

Tipo de dato: `switch`

Especifica que la interrupción es única y segura. Para la definición de única, consulte la descripción del argumento `Single`. Una interrupción segura no puede ponerse en reposo con la instrucción `ISleep`. El evento de interrupción segura se coloca en la cola en caso de paro del programa y de ejecución paso a paso. La

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.116 ISignalAO - Interrupciones a partir de una señal analógica de salida

RobotWare - OS

Continuación

interrupción se ejecuta al iniciar nuevamente el modo continuo. El único momento en el que una interrupción segura se desecha es cuando la cola de interrupciones está llena. En este caso se genera un error. La interrupción no sobrevive al restablecimiento del programa, por ejemplo, PP a main.

Signal

Tipo de dato: `signalao`

El nombre de la señal a partir de la cual deben generarse las interrupciones.

Condition

Tipo de dato: `aiotrigg`

Especifica cómo `HighValue` y `LowValue` definen la condición que debe cumplirse:

| Valor | Constante simbólica | Comentario   |
|-------|---------------------|--|
| 1     | AIO_ABOVE_HIGH      | La señal genera interrupciones si se encuentra por encima del valor máximo especificado  |
| 2     | AIO_BELOW_HIGH      | La señal genera interrupciones si se encuentra por debajo del valor máximo especificado  |
| 3     | AIO_ABOVE_LOW       | La señal genera interrupciones si se encuentra por encima del valor mínimo especificado  |
| 4     | AIO_BELOW_LOW       | La señal genera interrupciones si se encuentra por debajo del valor mínimo especificado  |
| 5     | AIO_BETWEEN         | La señal genera interrupciones si se encuentra entre los valores mínimo y máximo especificados                                     |
| 6     | AIO_OUTSIDE         | La señal genera interrupciones si se encuentra por debajo del valor mínimo especificado o por encima del valor máximo especificado |
| 7     | AIO_ALWAYS          | La señal siempre genera interrupciones   |

HighValue

Tipo de dato: `num`

El valor lógico de límite máximo utilizado para definir la condición.

LowValue

Tipo de dato: `num`

El valor lógico de límite mínimo utilizado para definir la condición.

DeltaValue

Tipo de dato: `num`

Define la diferencia mínima entre dos señales lógicas antes de que se genere una nueva interrupción. El valor actual de la señal, comparado con el valor de referencia almacenado anteriormente, debe ser mayor que el valor especificado en `DeltaValue` para que se genere una nueva interrupción.

[`\DPos`]

Tipo de dato: `switch`

Especifica que sólo las diferencias positivas entre señales lógicas suponen la generación de nuevas interrupciones.

Continúa en la página siguiente

1.116 ISignalAO - Interrupciones a partir de una señal analógica de salida

RobotWare - OS

Continuación

[ \DNeg ]

Tipo de dato: switch

Especifica que sólo las diferencias negativas entre señales lógicas suponen la generación de nuevas interrupciones.

Si no se utilizan los argumentos \DPos ni \DNeg, tanto las diferencias positivas como las negativas suponen la generación de nuevas interrupciones.

Interrupt

Tipo de dato: intnum

La identidad de la interrupción. La interrupción debe estar ya conectada a una rutina TRAP mediante la instrucción CONNECT.

Ejecución de programas

Consulte la instrucción ISignalAI para obtener más información sobre:

- Ejecución de programas
- Condiciones para la generación de interrupciones
- Más ejemplos

Los principios en los que se basa ISignalAO son los mismos que se aplican a ISignalAI.

Gestión de errores

Pueden generarse los errores recuperables enumerados a continuación. Estos errores pueden ser gestionados en un gestor de errores. El valor de la variable de sistema ERRNO cambia a:

| Nombre             | Causa del error   |
|--------------------|---|
| ERR_NO_ALIASIO_DEF | La variable de señal es una variable declarada en RAPID y no se ha conectado a una señal de E/S definida en la configuración de E/S con la instrucción AliasIO. |
| ERR_AO_LIM         | El argumento programado HighValue o LowValue para la señal analógica de entrada especificada Signal está fuera de límites.                                      |
| ERR_NORUNUNIT      | Se ha perdido el contacto con el dispositivo de E/S.  |

Limitaciones

Los argumentos HighValue y LowValue deben estar dentro del rango: valor lógico máximo y valor lógico mínimo definidos para la señal.

HighValue debe ser superior a LowValue.

DeltaValue debe ser 0 o un valor positivo.

Las limitaciones de la identidad de la interrupción son las mismas que en el caso de ISignalDO.

Sintaxis

```
ISignalAO
[ '\' Single ] | [ '\' SingleSafe ] ','
[ Signal':=' ]<variable (VAR) of signalao>','
[ Condition':=' ]<expression (IN) of aiotrigg>','
```

Continúa en la página siguiente

# 1 Instrucciones

## 1.116 ISignalAO - Interrupciones a partir de una señal analógica de salida

RobotWare - OS

Continuación

```
[ HighValue':=' ]<expression (IN) of num>','  
[ LowValue':=' ]<expression (IN) of num>','  
[ DeltaValue':=' ]<expression (IN) of num>  
[[ '\DPos' | [ '\DNeg' ],'  
[ Interrupt':=' ]<variable (VAR) of intnum>';'
```

### Información relacionada

| Para obtener más información sobre                      | Consulte  |
|---|---|
| Resumen de interrupciones y gestión de interrupciones   | <a href="#">Manual de referencia técnica - RAPID Overview</a>   |
| Definición de constantes                                | <a href="#">aiotrigg - Condición de disparo con E/S analógica en la página 1607</a>                           |
| Interrupción a partir de una señal analógica de entrada | <a href="#">ISignalAI - Interrupciones a partir de una señal analógica de entrada en la página 309</a>        |
| Interrupción a partir de una señal digital de entrada   | <a href="#">ISignalDI - Solicita interrupciones a partir de una señal digital de entrada en la página 323</a> |
| Interrupción a partir de una señal digital de salida    | <a href="#">ISignalDO - Interrupciones a partir de una señal digital de salida en la página 326</a>           |
| Identidad de interrupción                               | <a href="#">intnum - Identidad de interrupción en la página 1687</a>  |
| Parámetros del sistema relacionados (filtro)            | <a href="#">Manual de referencia técnica - Parámetros del sistema</a>   |

---

## 1.117 ISignalDI - Solicita interrupciones a partir de una señal digital de entrada

---

### Utilización

ISignalDI (*Interrupt Signal Digital In*) se utiliza para solicitar y activar interrupciones a partir de una señal digital de entrada.

---

### Ejemplos básicos

Los ejemplos siguientes ilustran la instrucción ISignalDI.

#### Ejemplo 1

```
VAR intnum siglint;
PROC main()
  CONNECT siglint WITH iroutine1;
  ISignalDI di1,1,siglint;
```

Solicita una interrupción que debe producirse cada vez que la señal digital de entrada di1 cambie a 1. En este caso, se realiza una llamada a la rutina TRAP iroutine1.

#### Ejemplo 2

```
ISignalDI di1,0,siglint;
```

Solicita una interrupción que debe producirse cada vez que la señal digital de entrada di1 cambie a 0.

#### Ejemplo 3

```
ISignalDI \Single, di1,1,siglint;
```

Solicita una interrupción que debe producirse sólo la primera vez que la señal digital de entrada di1 cambie a 1.

---

### Argumentos

```
ISignalDI [ \Single ] | [ \SingleSafe ] Signal TriggValue Interrupt
```

[ \Single ]

Tipo de dato: switch

Especifica si la interrupción debe producirse una sola vez o de forma cíclica.

Si se utiliza el argumento `Single`, la interrupción se produce como máximo una sola vez. Si se omiten los argumentos `Single` y `SingleSafe`, se genera una interrupción cada vez que se cumpla la condición.

[ \SingleSafe ]

Tipo de dato: switch

Especifica que la interrupción es única y segura. Para la definición de única, consulte la descripción del argumento `Single`. Una interrupción segura no puede ponerse en reposo con la instrucción `ISleep`. El evento de interrupción segura se coloca en la cola en caso de paro del programa y de ejecución paso a paso. La interrupción se ejecuta al iniciar nuevamente el modo continuo. El único momento en el que una interrupción segura se desecha es cuando la cola de interrupciones está llena. En este caso se genera un error. La interrupción no sobrevive al restablecimiento del programa, por ejemplo, PP a main.

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.117 ISignalDI - Solicita interrupciones a partir de una señal digital de entrada

RobotWare - OS

Continuación

Signal

Tipo de dato: `signalDI`

El nombre de la señal a partir de la cual deben generarse las interrupciones.

TriggValue

Tipo de dato: `dionum`

El valor al que debe cambiar la señal para que se produzca la interrupción.

El valor se especifica como 0 ó 1 o como un valor simbólico (por ejemplo `high/low`). La señal se dispara en el momento del cambio a 0 ó 1.

En `TriggValue` es posible usar el valor 2 o el valor simbólico `edge` para la generación de interrupciones tanto en el flanco positivo (0 -> 1) como en el flanco negativo (1 -> 0).

Interrupt

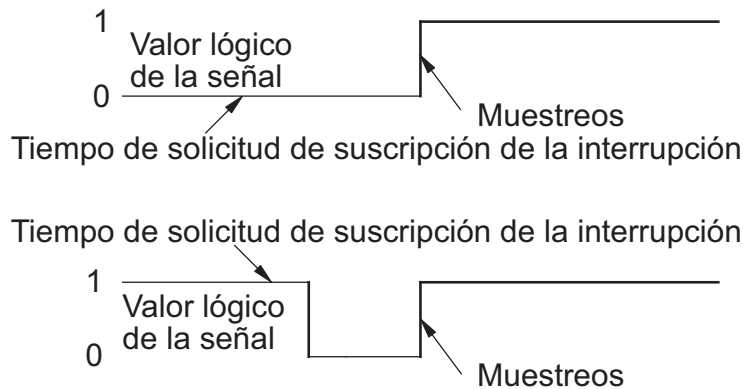
Tipo de dato: `intnum`

La identidad de la interrupción. La interrupción debe estar ya conectada a una rutina TRAP mediante la instrucción `CONNECT`.

### Ejecución de programas

En el momento en el que la señal recibe el valor especificado, se realiza una llamada a la rutina TRAP correspondiente. Una vez ejecutada la rutina, la ejecución del programa continúa a partir del punto en el que se produjo la interrupción.

Si la señal cambia al valor especificado antes de la solicitud de la interrupción, no se produce ninguna interrupción. Las interrupciones de una señal digital de entrada a partir del nivel de señal 1 se ilustran en la figura siguiente.



xx0500002189

### Gestión de errores

Pueden generarse los errores recuperables enumerados a continuación. Estos errores pueden ser gestionados en un gestor de errores. El valor de la variable de sistema `ERRNO` cambia a:

| Nombre                          | Causa del error  |
|---------------------------------|--|
| <code>ERR_NO_ALIASIO_DEF</code> | La variable de señal es una variable declarada en <code>RAPID</code> y no se ha conectado a una señal de E/S definida en la configuración de E/S con la instrucción <code>AliasIO</code> . |

Continúa en la página siguiente



## 1.117 ISignalDI - Solicita interrupciones a partir de una señal digital de entrada

RobotWare - OS

Continuación

| Nombre        | Causa del error                                      |
|---------------|--|
| ERR_NORUNUNIT | Se ha perdido el contacto con el dispositivo de E/S. |

### Limitaciones

No es posible utilizar más de una vez la identidad de la interrupción sin eliminarla previamente. Por tanto, las interrupciones deben tratarse de la forma mostrada, con una de las alternativas siguientes.

```
VAR intnum siglint;
PROC main ()
  CONNECT siglint WITH iroutinel;
  ISignalDI dil, 1, siglint;
  WHILE TRUE DO
    ...
  ENDWHILE
ENDPROC
```

Al principio del programa se produce la activación de todas las interrupciones. En este caso, las instrucciones iniciales se mantienen fuera del flujo principal del programa.

```
VAR intnum siglint;
PROC main ()
  CONNECT siglint WITH iroutinel;
  ISignalDI dil, 1, siglint;
  ...
  IDelete siglint;
ENDPROC
```

La interrupción se elimina al final del programa y se activa de nuevo. En este caso, recuerde que la interrupción permanece inactiva durante un periodo breve.

### Sintaxis

```
ISignalDI
  [ '\ ' Single ] | [ '\ ' SingleSafe ] ', '
  [ Signal ' := ' ] < variable (VAR) of signaldi > ', '
  [ TriggValue ' := ' ] < expression (IN) of dionum > ', '
  [ Interrupt ' := ' ] < variable (VAR) of intnum > ';'
```

### Información relacionada

| Para obtener más información sobre                    | Consulte  |
|---|---|
| Resumen de interrupciones y gestión de interrupciones | <i>Manual de referencia técnica - RAPID Overview</i>  |
| Interrupción a partir de una señal de salida          | <a href="#">ISignalDO - Interrupciones a partir de una señal digital de salida en la página 326</a> |
| Identidad de interrupción                             | <a href="#">intnum - Identidad de interrupción en la página 1687</a>                                |

## 1 Instrucciones

---

### 1.118 ISignalDO - Interrupciones a partir de una señal digital de salida

RobotWare - OS

### 1.118 ISignalDO - Interrupciones a partir de una señal digital de salida

---

#### Utilización

ISignalDO (*Interrupt Signal Digital Out*) se utiliza para solicitar y activar interrupciones a partir de una señal digital de salida.

---

#### Ejemplos básicos

Los ejemplos siguientes ilustran la instrucción ISignalDO.

##### Ejemplo 1

```
VAR intnum siglint;  
PROC main()  
    CONNECT siglint WITH iroutine1;  
    ISignalDO do1,1,siglint;
```

Solicita una interrupción que debe producirse cada vez que la señal de salida digital do1 cambie a 1. En este caso, se realiza una llamada a la rutina TRAP iroutine1.

##### Ejemplo 2

```
ISignalDO do1,0,siglint;
```

Solicita una interrupción que debe producirse cada vez que la señal de salida digital do1 cambie a 0.

##### Ejemplo 3

```
ISignalDO\Single, do1,1,siglint;
```

Solicita una interrupción que debe producirse sólo la primera vez que la señal de salida digital do1 cambie a 1.

---

#### Argumentos

```
ISignalDO [ \Single ] | [ \SingleSafe ] Signal TriggValue Interrupt
```

[ \Single ]

Tipo de dato: switch

Especifica si la interrupción debe producirse una sola vez o de forma cíclica.

Si se utiliza el argumento Single, la interrupción se produce como máximo una sola vez. Si se omiten los argumentos Single y SingleSafe, se genera una interrupción cada vez que se cumpla la condición.

[ \SingleSafe ]

Tipo de dato: switch

Especifica que la interrupción es única y segura. Para la definición de única, consulte la descripción del argumento Single. Una interrupción segura no puede ponerse en reposo con la instrucción ISleep. El evento de interrupción segura se coloca en la cola en caso de paro del programa y de ejecución paso a paso. La interrupción se ejecuta al iniciar nuevamente el modo continuo. El único momento en el que una interrupción segura se desecha es cuando la cola de interrupciones está llena. En este caso se genera un error. La interrupción no sobrevive al restablecimiento del programa, por ejemplo, PP a main.

*Continúa en la página siguiente*

Signal

Tipo de dato: `signaldo`

El nombre de la señal a partir de la cual deben generarse las interrupciones.

TriggValue

Tipo de dato: `dionum`

El valor al que debe cambiar la señal para que se produzca la interrupción.

El valor se especifica como 0 ó 1 o como un valor simbólico (por ejemplo `high/low`). La señal se dispara en el momento del cambio a 0 ó 1.

En `TriggValue` es posible usar el valor 2 o el valor simbólico `edge` para la generación de interrupciones tanto en el flanco positivo (0 -> 1) como en el flanco negativo (1 -> 0).

Interrupt

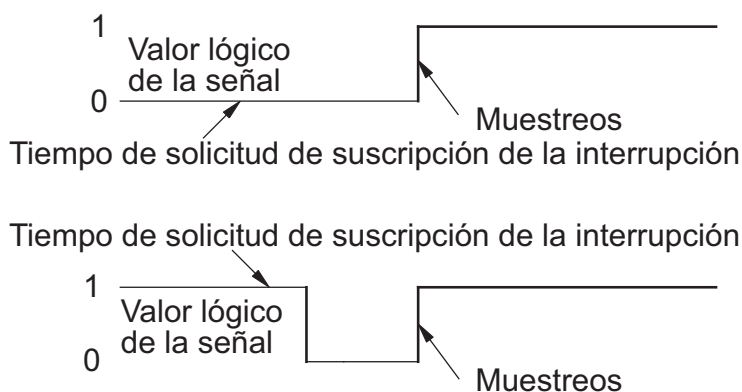
Tipo de dato: `intnum`

La identidad de la interrupción. La interrupción debe estar ya conectada a una rutina TRAP mediante la instrucción `CONNECT`.

### Ejecución de programas

En el momento en el que la señal recibe el valor especificado, 0 ó 1, se realiza una llamada a la rutina TRAP correspondiente. Una vez ejecutada la rutina, la ejecución del programa continúa a partir del punto en el que se produjo la interrupción.

Si la señal cambia al valor especificado antes de la solicitud de la interrupción, no se produce ninguna interrupción. Las interrupciones de una señal digital de salida a partir del nivel de señal 1 se ilustran en la figura siguiente.



xx0500002190

### Gestión de errores

Pueden generarse los errores recuperables enumerados a continuación. Estos errores pueden ser gestionados en un gestor de errores. El valor de la variable de sistema `ERRNO` cambia a:

| Nombre                          | Causa del error  |
|---------------------------------|--|
| <code>ERR_NO_ALIASIO_DEF</code> | La variable de señal es una variable declarada en <code>RAPID</code> y no se ha conectado a una señal de E/S definida en la configuración de E/S con la instrucción <code>AliasIO</code> . |

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.118 ISignalDO - Interrupciones a partir de una señal digital de salida

RobotWare - OS

Continuación

| Nombre        | Causa del error                                      |
|---------------|--|
| ERR_NORUNUNIT | Se ha perdido el contacto con el dispositivo de E/S. |

### Limitaciones

No es posible utilizar más de una vez la identidad de la interrupción sin eliminarla previamente. Por tanto, las interrupciones deben tratarse de la forma mostrada, con una de las alternativas siguientes.

```
VAR intnum siglint;  
PROC main ()  
  CONNECT siglint WITH iroutinel;  
  ISignalDO do1, 1, siglint;  
  WHILE TRUE DO  
    ...  
  ENDWHILE  
ENDPROC
```

Al principio del programa se produce la activación de todas las interrupciones. En este caso, las instrucciones iniciales se mantienen fuera del flujo principal del programa.

```
VAR intnum siglint;  
PROC main ()  
  CONNECT siglint WITH iroutinel;  
  ISignalDO do1, 1, siglint;  
  ...  
  IDelete siglint;  
ENDPROC
```

La interrupción se elimina al final del programa y se activa de nuevo. En este caso, recuerde que la interrupción permanece inactiva durante un periodo breve.

### Sintaxis

```
ISignalDO  
  [ '\ ' Single ] | [ '\ ' SingleSafe ] ', '  
  [ Signal ' := ' ] < variable (VAR) of signaldo > ', '  
  [ TriggValue ' := ' ] < expression (IN) of dionum > ', '  
  [ Interrupt ' := ' ] < variable (VAR) of intnum > ';'
```

### Información relacionada

| Para obtener más información sobre                    | Consulte  |
|---|---|
| Resumen de interrupciones y gestión de interrupciones | <i>Manual de referencia técnica - RAPID Overview</i>  |
| Interrupción a partir de una señal de entrada         | <a href="#">ISignalDI - Solicita interrupciones a partir de una señal digital de entrada en la página 323</a> |
| Identidad de interrupción                             | <a href="#">intnum - Identidad de interrupción en la página 1687</a>  |

**1.119 ISignalGI - Solicita interrupciones de un grupo de señales digitales de entrada****Utilización**

ISignalGI (*Interrupt Signal Group Digital In*) se utiliza para solicitar y activar interrupciones a partir de un grupo de señales digitales de entrada.

**Ejemplos básicos**

El ejemplo que aparece a continuación ilustra la instrucción ISignalGI:

**Ejemplo 1**

```
VAR intnum siglint;
PROC main()
  CONNECT siglint WITH iroutinel;
  ISignalGI gil,siglint;
```

Solicita una interrupción ante un cambio de valor de una señal de un grupo de entradas digitales.

**Argumentos**

```
ISignalGI [ \Single ] | [ \SingleSafe ] Signal Interrupt
```

[ \Single ]

Tipo de dato: switch

Especifica si la interrupción debe producirse una sola vez o de forma cíclica.

Si se utiliza el argumento `Single`, la interrupción se produce como máximo una sola vez. Si se omiten los argumentos `Single` y `SingleSafe`, se genera una interrupción cada vez que se cumpla la condición.

[ \SingleSafe ]

Tipo de dato: switch

Especifica que la interrupción es única y segura. Para la definición de única, consulte la descripción del argumento `Single`. Una interrupción segura no puede ponerse en reposo con la instrucción `ISleep`. El evento de interrupción segura se coloca en la cola en caso de paro del programa y de ejecución paso a paso. La interrupción se ejecuta al iniciar nuevamente el modo continuo. El único momento en el que una interrupción segura se desecha es cuando la cola de interrupciones está llena. En este caso se genera un error. La interrupción no sobrevive al restablecimiento del programa, por ejemplo, PP a main.

Signal

Tipo de dato: signalgi

El nombre de la señal de entrada de grupo a partir de la cual deben generarse las interrupciones.

Interrupt

Tipo de dato: intnum

La identidad de la interrupción. La interrupción debe estar ya conectada a una rutina TRAP mediante la instrucción `CONNECT`.

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.119 ISignalGI - Solicita interrupciones de un grupo de señales digitales de entrada

RobotWare - OS

Continuación

---

### Ejecución de programas

En el momento en el que la señal de grupo cambia de valor, se realiza una llamada a la rutina TRAP correspondiente. Una vez ejecutada la rutina, la ejecución del programa continúa a partir del punto en el que se produjo la interrupción.

Si la señal cambia de valor antes de la solicitud de la interrupción, no se produce ninguna interrupción.

Al cambiar una señal de grupo de entradas digitales a un valor, pueden generarse varias interrupciones. El motivo es que los cambios de los bits individuales incluidos en la señal de grupo no se detectan simultáneamente en el sistema de robot. Para evitar la generación de múltiples interrupciones para un solo cambio de señal de grupo, es posible definir un tiempo de filtro para la señal.

---

### Gestión de errores

Pueden generarse los errores recuperables enumerados a continuación. Estos errores pueden ser gestionados en un gestor de errores. El valor de la variable de sistema `ERRNO` cambia a:

| Nombre                          | Causa del error  |
|---------------------------------|--|
| <code>ERR_NO_ALIASIO_DEF</code> | La variable de señal es una variable declarada en <code>RAPID</code> y no se ha conectado a una señal de E/S definida en la configuración de E/S con la instrucción <code>AliasIO</code> . |
| <code>ERR_NORUNUNIT</code>      | Se ha perdido el contacto con el dispositivo de E/S.   |

---

### Limitaciones

El número máximo de señales que pueden usarse con un grupo determinado es de 32.

No es posible usar una condición con valor numérico en la instrucción para especificar que debe producirse una interrupción si el valor cambia a ese valor específico. Esta funcionalidad debe ser creada en el programa de usuario, mediante la lectura del valor de la señal de grupo en el momento de la ejecución de la rutina TRAP.

Las interrupciones se generan como interrupciones de bits, es decir, las interrupciones ante señales digitales de entrada individuales cambian dentro del grupo. Si los bits de la señal de grupo cambian de valor con un retardo que está dentro de los valores especificados, se generarán varias interrupciones. Es necesario contar con conocimientos sobre cómo funciona la tarjeta de E/S, con el fin de obtener la función correcta al utilizar `ISignalGI`. Si se generan varias interrupciones en los valores de entrada del grupo, utilice en su lugar `ISignalDI` con una señal de muestreo que se activa cuando todos los bits de la señal de grupo están activados.

No es posible utilizar más de una vez la identidad de la interrupción sin eliminarla previamente. Por tanto, las interrupciones deben tratarse de la forma mostrada, con una de las alternativas siguientes.

```
VAR intnum siglint;  
PROC main ()  
    CONNECT siglint WITH iroutinel;  
    ISignalGI gil, siglint;
```

Continúa en la página siguiente

---

## 1.119 ISignalGI - Solicita interrupciones de un grupo de señales digitales de entrada

RobotWare - OS

Continuación

```

WHILE TRUE DO
  ...
ENDWHILE
ENDPROC

```

Al principio del programa se produce la activación de todas las interrupciones. En este caso, las instrucciones iniciales se mantienen fuera del flujo principal del programa.

```

VAR intnum siglint;
PROC main ()
  CONNECT siglint WITH iroutine1;
  ISignalGI gil, siglint;
  ...
  IDelete siglint;
ENDPROC

```

La interrupción se elimina al final del programa y se activa de nuevo. En este caso, es importante recordar que la interrupción permanece inactiva durante un periodo breve.

### Sintaxis

```

ISignalGI
  ['\' Single] | ['\' SingleSafe'],'
  [Signal ':= ' ] < variable (VAR) of signalgi>',
  [Interrupt ':= ' ] <variable (VAR) of intnum>';

```

### Información relacionada

| Para obtener más información sobre                    | Consulte  |
|---|---|
| Resumen de interrupciones y gestión de interrupciones | <i>Manual de referencia técnica - RAPID Overview</i>  |
| Interrupción a partir de una señal de entrada         | <a href="#">ISignalDI - Solicita interrupciones a partir de una señal digital de entrada en la página 323</a>   |
| Interrupción a partir de señales de salida de grupo   | <a href="#">ISignalGO - Solicita interrupciones de un grupo de señales digitales de salida en la página 332</a> |
| Identidad de interrupción                             | <a href="#">intnum - Identidad de interrupción en la página 1687</a>  |
| Tiempo de filtro                                      | <i>Manual de referencia técnica - Parámetros del sistema</i>  |

## 1 Instrucciones

---

1.120 ISignalGO - Solicita interrupciones de un grupo de señales digitales de salida  
RobotWare - OS

### 1.120 ISignalGO - Solicita interrupciones de un grupo de señales digitales de salida

---

#### Utilización

ISignalGO (*Interrupt Signal Group Digital Out*) se utiliza para solicitar y activar interrupciones a partir de un grupo de señales digitales de salida.

---

#### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción ISignalGO:

##### Ejemplo 1

```
VAR intnum siglint;  
PROC main()  
    CONNECT siglint WITH iroutinel;  
    ISignalGO gol,siglint;
```

Solicita una interrupción ante un cambio de valor de una señal de un grupo de salidas digitales.

---

#### Argumentos

```
ISignalGO [ \Single ] | [ \SingleSafe ] Signal Interrupt
```

[ \Single ]

Tipo de dato: switch

Especifica si la interrupción debe producirse una sola vez o de forma cíclica.

Si se utiliza el argumento `\Single`, la interrupción se produce como máximo una sola vez. Si se omiten los argumentos `Single` y `SingleSafe`, se genera una interrupción cada vez que se cumpla la condición.

[ \SingleSafe ]

Tipo de dato: switch

Especifica que la interrupción es única y segura. Para la definición de única, consulte la descripción del argumento `Single`. Una interrupción segura no puede ponerse en reposo con la instrucción `ISleep`. El evento de interrupción segura se coloca en la cola en caso de paro del programa y de ejecución paso a paso. La interrupción se ejecuta al iniciar nuevamente el modo continuo. El único momento en el que una interrupción segura se desecha es cuando la cola de interrupciones está llena. En este caso se genera un error. La interrupción no sobrevive al restablecimiento del programa, por ejemplo, PP a main.

Signal

Tipo de dato: signalgo

El nombre de la señal de salida de grupo a partir de la cual deben generarse las interrupciones.

Interrupt

Tipo de dato: intnum

La identidad de la interrupción. La interrupción debe estar ya conectada a una rutina TRAP mediante la instrucción `CONNECT`.

*Continúa en la página siguiente*



### Ejecución de programas

En el momento en el que la señal de grupo cambia de valor, se realiza una llamada a la rutina TRAP correspondiente. Una vez ejecutada la rutina, la ejecución del programa continúa a partir del punto en el que se produjo la interrupción.

Si la señal cambia de valor antes de la solicitud de la interrupción, no se produce ninguna interrupción.

### Gestión de errores

Pueden generarse los errores recuperables enumerados a continuación. Estos errores pueden ser gestionados en un gestor de errores. El valor de la variable de sistema `ERRNO` cambia a:

| Nombre                          | Causa del error   |
|---------------------------------|---|
| <code>ERR_NO_ALIASIO_DEF</code> | La variable de señal es una variable declarada en RAPID y no se ha conectado a una señal de E/S definida en la configuración de E/S con la instrucción <code>AliasIO</code> . |
| <code>ERR_NORUNUNIT</code>      | Se ha perdido el contacto con el dispositivo de E/S.  |

### Limitaciones

El número máximo de señales que pueden usarse con un grupo determinado es de 32.

No es posible usar una condición con valor numérico en la instrucción para especificar que debe producirse una interrupción si el valor cambia a ese valor específico. Esta funcionalidad debe ser creada en el programa de usuario, mediante la lectura del valor de la señal de grupo en el momento de la ejecución de la rutina TRAP.

No es posible utilizar más de una vez la identidad de la interrupción sin eliminarla previamente. Por tanto, las interrupciones deben tratarse de la forma mostrada, con una de las alternativas siguientes.

```
VAR intnum siglint;  
PROC main ()  
  CONNECT siglint WITH iroutine1;  
  ISignalGO go1, siglint;  
  WHILE TRUE DO  
    ...  
  ENDWHILE  
ENDPROC
```

Al principio del programa se produce la activación de todas las interrupciones. En este caso, las instrucciones iniciales se mantienen fuera del flujo principal del programa.

```
VAR intnum siglint;  
PROC main ()  
  CONNECT siglint WITH iroutine1;  
  ISignalGO go1, siglint;  
  ...  
  IDelete siglint;  
ENDPROC
```

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.120 ISignalGO - Solicita interrupciones de un grupo de señales digitales de salida

RobotWare - OS

Continuación

La interrupción se elimina al final del programa y se activa de nuevo. En este caso, recuerde que la interrupción permanece inactiva durante un periodo breve.

### Sintaxis

```
ISignalGO  
[ '\ ' Single ] | [ '\ ' SingleSafe ] ','  
[ Signal ':=' ] < variable (VAR) of signalgo > ','  
[ Interrupt':=' ] < variable (VAR) of intnum > ';' 
```

### Información relacionada

| Para obtener más información sobre                    | Consulte   |
|---|--|
| Resumen de interrupciones y gestión de interrupciones | <i>Manual de referencia técnica - RAPID Overview</i>   |
| Interrupción a partir de una señal de salida          | <a href="#">ISignalDO - Interrupciones a partir de una señal digital de salida en la página 326</a>              |
| Interrupción a partir de señales de entrada de grupo  | <a href="#">ISignalGI - Solicita interrupciones de un grupo de señales digitales de entrada en la página 329</a> |
| Identidad de interrupción                             | <a href="#">intnum - Identidad de interrupción en la página 1687</a>   |

---

## 1.121 ISleep - Desactiva una interrupción

---

### Utilización

ISleep(*Interrupt Sleep*) se utiliza para desactivar temporalmente una interrupción determinada.

Durante el periodo en el que una interrupción está desactivada, su aparición no se detecta y se desecha, sin ejecutar ninguna rutina TRAP.

---

### Ejemplos básicos

El ejemplo siguiente ilustra la instrucción ISleep.

Consulte también [Más ejemplos en la página 335](#).

#### Ejemplo 1

```
ISleep siglint;
```

Se desactiva la interrupción siglint.

---

### Argumentos

```
ISleep Interrupt
```

Interrupt

Tipo de dato: intnum

La variable (la identidad) de la interrupción.

---

### Ejecución de programas

Cualquier interrupción del tipo especificado que se genere después de ejecutar la instrucción se desecha sin ejecutar ninguna rutina TRAP, hasta que se reactive la interrupción mediante la instrucción IWatch. No se procesa ninguna interrupción generada mientras ISleep.

---

### Más ejemplos

A continuación aparecen más ejemplos de la instrucción ISleep.

#### Ejemplo 1

```
VAR intnum timeint;
VAR iodev binfile;

PROC ISleep_example()
  CONNECT timeint WITH write_binfile;
  ITimer 6, timeint;
  !...
  ISleep timeint;
  WriteBin binfile, buffer, 30;
  WriteBin binfile, buffer2, 30;
  IWatch timeint;
  !...
ENDPROC

TRAP write_binfile
  WriteBin binfile, buffer3, 1;
```

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.121 ISleep - Desactiva una interrupción

RobotWare - OS

Continuación

ENDTRAP

La escritura en el archivo `binfile` se realiza cada 6 segundos. Las interrupciones no se permiten si hay una comunicación en curso.

---

### Gestión de errores

Pueden generarse los errores recuperables enumerados a continuación. Estos errores pueden ser gestionados en un gestor de errores. El valor de la variable de sistema `ERRNO` cambia a:

| Nombre                     | Causa del error  |
|----------------------------|--|
| <code>ERR_UNKINO</code>    | El número de interrupción es desconocido.<br>No se permite ninguna interrupción que no se haya solicitado ni activado. |
| <code>ERR_INOISSAFE</code> | Si se intenta desactivar temporalmente una interrupción segura con <code>ISleep</code> .                               |

---

### Sintaxis

```
ISleep  
  [ Interrupt ':= ' ] < variable (VAR) of intnum > ';' 
```

---

### Información relacionada

| Para obtener más información sobre        | Consulte   |
|---|--|
| Resumen de interrupciones                 | <i>Manual de referencia técnica - RAPID Overview</i>                           |
| Activación de interrupciones              | <a href="#">IWatch - Activar una interrupción en la página 344</a>             |
| Desactivación de todas las interrupciones | <a href="#">IDisable - Desactiva todas las interrupciones en la página 255</a> |
| Cancelación de una interrupción           | <a href="#">IDelete - Cancela una interrupción en la página 254</a>            |

## 1.122 ITimer - Solicita una interrupción temporizada

### Utilización

`ITimer` (*Interrupt Timer*) se utiliza para solicitar y activar una interrupción temporizada.

Por ejemplo, esta instrucción puede utilizarse para comprobar el estado de los equipos periféricos una vez por minuto.

### Ejemplos básicos

Los ejemplos siguientes ilustran la instrucción `ITimer`.

Consulte también [Más ejemplos en la página 338](#).

#### Ejemplo 1

```
VAR intnum timeint;
PROC main()
  CONNECT timeint WITH iroutinel;
  ITimer 60, timeint;
```

Solicita una interrupción que debe producirse cíclicamente cada 60 segundos. En este caso, se realiza una llamada a la rutina TRAP `iroutinel`.

#### Ejemplo 2

```
ITimer \Single, 60, timeint;
```

Solicita una interrupción que debe producirse una vez cada 60 segundos.

### Argumentos

```
ITimer [ \Single ] | [ \SingleSafe ] Time Interrupt
```

[ \Single ]

Tipo de dato: `switch`

Especifica si la interrupción debe producirse una sola vez o de forma cíclica.

Si se utiliza el argumento `Single`, la interrupción se produce una sola vez. Si se omiten los argumentos `Single` y `SingleSafe`, se genera una interrupción cada vez que se alcanza el momento especificado.

[ \SingleSafe ]

Tipo de dato: `switch`

Especifica que la interrupción es única y segura. Para la definición de única, consulte la descripción del argumento `Single`. Una interrupción segura no puede ponerse en reposo con la instrucción `ISleep`. El evento de interrupción segura se coloca en la cola en caso de paro del programa y de ejecución paso a paso. La interrupción se ejecuta al iniciar nuevamente el modo continuo.

Time

Tipo de dato: `num`

La cantidad de tiempo que debe transcurrir antes de que se produzca la interrupción.

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.122 ITimer - Solicita una interrupción temporizada

RobotWare - OS

Continuación

El valor se especifica en segundos. Si se ha definido `Single` o `SingleSafe`, este tiempo no puede ser inferior a los 0,01 segundos. El tiempo correspondiente para las interrupciones cíclicas es de 0,1 segundos.

Interrupt

Tipo de dato: `intnum`

La variable (la identidad) de la interrupción. La interrupción debe estar ya conectada a una rutina TRAP mediante la instrucción `CONNECT`.

---

### Ejecución de programas

Se llama automáticamente a la rutina TRAP correspondiente en un momento determinado a partir de la solicitud de la interrupción. Una vez ejecutada la rutina, la ejecución del programa continúa a partir del punto en el que se produjo la interrupción.

Si la interrupción se produce de forma cíclica, comienza un nuevo cálculo del tiempo a partir del momento en que se produce.

---

### Más ejemplos

A continuación aparecen más ejemplos de la instrucción `ITimer`.

#### Ejemplo 1

```
VAR intnum timeint;
VAR iodev binfile;

PROC ISleep_example()
  CONNECT timeint WITH write_binfile;
  ITimer 6, timeint;
  !...
  ISleep timeint;
  WriteBin binfile, buffer, 30;
  WriteBin binfile, buffer2, 30;
  IWatch timeint;
  !...
ENDPROC

TRAP write_binfile
  WriteBin binfile, buffer3, 1;
ENDTRAP
```

La escritura en el archivo `binfile` se realiza cada 6 segundos. Las interrupciones no se permiten si hay una comunicación en curso.

---

### Limitaciones

No es posible utilizar más de una vez la identidad de la interrupción sin eliminarla previamente. Consulte la instrucción `ISignalDI`.

---

### Sintaxis

```
ITimer
  [ '\ ' Single ] | [ '\ ' SingleSafe ] ', '
  [ Time := ] < expression (IN) of num > ', '
```

Continúa en la página siguiente

## 1.122 ITimer - Solicita una interrupción temporizada

*RobotWare - OS*

*Continuación*

```
[ Interrupt' :=' ] < variable (VAR) of intnum > ';' 
```

### Información relacionada

| Para obtener más información sobre                    | Consulte   |
|---|--|
| Resumen de interrupciones y gestión de interrupciones | <i>Manual de referencia técnica - RAPID Overview</i> |

## 1 Instrucciones

---

### 1.123 IVarValue - Solicita una interrupción a partir del valor de una variable

#### *Optical Tracking*

### 1.123 IVarValue - Solicita una interrupción a partir del valor de una variable

---

#### Utilización

IVarValue (*Interrupt Variable Value*) se utiliza para solicitar y activar una interrupción cuando cambia al valor de una variable consultada a través de la interfaz de sensores.

Por ejemplo, esta instrucción puede utilizarse para obtener valores de volumen de cordón o valores de huecos de un sistema de seguimiento de cordón.

---

#### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción IVarValue:

##### Ejemplo 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};
```

*Continúa en la página siguiente*



## 1.123 IVarValue - Solicita una interrupción a partir del valor de una variable

*Optical Tracking*

*Continuación*

```
adaptWd.weld_wirefeed:=adptWfd{ArrInd};  
adaptWd.weld_speed:=adptSpd{ArrInd};
```

```
!Request a refresh of AW parameters using the new data i adaptWd  
ArcRefresh;
```

```
ENDTRAP
```

---

### Argumentos

```
IVarValue device VarNo Value Interrupt [\Unit] [\DeadBand]  
[\ReportAtTool] [\SpeedAdapt] [\APTR]
```

device

**Tipo de dato:** string

El nombre del dispositivo de E/S configurado en sio.cfg para el sensor utilizado.

VarNo

**Tipo de dato:** num

El número de la variable a controlar.

Value

**Tipo de dato:** num

Una variable PERS que contendrá el nuevo valor de VarNo.

Interrupt

**Tipo de dato:** intnum

La variable (la identidad) de la interrupción. La interrupción debe estar ya conectada a una rutina TRAP mediante la instrucción CONNECT.

[\Unit]

**Tipo de dato:** num

El factor de escala por el que se multiplicará el valor de sensor de VarNo antes de la comprobación y antes del guardado en Value.

[\DeadBand]

**Tipo de dato:** num

Si el valor de VarNo devuelto por el sensor está dentro de +/- DeadBand, no se genera ninguna interrupción.

[\ReportAtTool]

**Tipo de dato:** switch

Este argumento opcional solo está disponible para los sensores anticipatorios, por ejemplo los sensores de seguimiento óptico. El argumento especifica que el valor de la variable no debe evaluarse de inmediato, sino cuando el TCP del robot alcanza la posición; es decir, se compensa la anticipación.

[\SpeedAdapt]

**Tipo de dato:** num

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.123 IVarValue - Solicita una interrupción a partir del valor de una variable

### Optical Tracking

#### Continuación

\SpeedAdapt es un factor de escala utilizado para cambiar la velocidad de proceso en las instrucciones Arc y Cap. Se multiplica por el valor del sensor de VarNo de acuerdo con:

$$\text{velocidad de proceso} = \text{\SpeedAdapt} * \text{value}(\text{VarNo})$$

[ \APTR ]

Tipo de dato: switch

Especifica que la suscripción de la variable debe acoplarse al seguimiento del punto, por ejemplo WeldGuide, especificado en el argumento device.

### Ejecución de programas

Se llama automáticamente a la rutina TRAP correspondiente en un momento determinado a partir de la solicitud de la interrupción. Una vez ejecutada la rutina, la ejecución del programa continúa a partir del punto en el que se produjo la interrupción.

### Limitaciones

- No es posible utilizar la misma variable para una identidad de interrupción más de cinco veces sin eliminarla primero.
- Todas las interrupciones que estén configuradas con IVarValue se deben configurar de nuevo tras el reinicio de un controlador.



#### ¡CUIDADO!

Una frecuencia de interrupción demasiado elevada bloqueará toda la ejecución de RAPID.

### Sintaxis

```
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 ] ';'
```

### Información relacionada

| Para obtener más información sobre                         | Consulte   |
|--|--|
| Establecimiento de una conexión a un dispositivo de sensor | <a href="#">SenDevice - Establece una conexión a un dispositivo de sensor en la página 687</a> |
| Resumen de interrupciones y gestión de interrupciones      | <a href="#">Manual de referencia técnica - RAPID Overview</a>                                  |
| Optical Tracking   | <a href="#">Application manual - Continuous Application Platform</a>                           |

Continúa en la página siguiente

## 1.123 IVarValue - Solicita una interrupción a partir del valor de una variable

*Optical Tracking*

*Continuación*

|                                    |  |
|------------------------------------|--|
| Para obtener más información sobre | Consulte                                       |
| Optical Tracking Arc               | <i>Application manual - Arc and Arc Sensor</i> |

# 1 Instrucciones

---

## 1.124 IWatch - Activar una interrupción RobotWare - OS

### 1.124 IWatch - Activar una interrupción

---

#### Utilización

`IWatch(Interrupt Watch)` se utiliza para activar una interrupción que se ha solicitado anteriormente pero que ha sido desactivada mediante `ISleep`.

---

#### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción `IWatch`:  
Consulte también [Más ejemplos en la página 344](#).

#### Ejemplo 1

```
IWatch siglint;
```

Se activa la interrupción `siglint` que fue desactivada anteriormente.

---

#### Argumentos

```
IWatch Interrupt
```

#### Interrupt

Tipo de dato: `intnum`  
La variable (la identidad) de la interrupción.

---

#### Ejecución de programas

Reactiva las interrupciones del tipo especificado. Las interrupciones generadas durante el periodo en el que está vigente la instrucción `ISleep` no se procesan.

---

#### Más ejemplos

A continuación aparecen más ejemplos de la instrucción `IWatch`.

#### Ejemplo 1

```
VAR intnum siglint;
PROC main()
  CONNECT siglint WITH iroutine1;
  ISignalDI dil,1,siglint;
  ...
  ISleep siglint;
  weldpart1;
  IWatch siglint;
```

Durante la ejecución de la rutina `weldpart1`, no se permite ninguna interrupción a partir de la señal `dil`.

---

#### Gestión de errores

Pueden generarse los errores recuperables enumerados a continuación. Estos errores pueden ser gestionados en un gestor de errores. El valor de la variable de sistema `ERRNO` cambia a:

| Nombre                  | Causa del error  |
|-------------------------|--|
| <code>ERR_UNKINO</code> | El número de interrupción es desconocido.<br>No se permite ninguna interrupción que no se haya solicitado ni activado. |

---

Continúa en la página siguiente

### Sintaxis

IWatch

```
[ Interrupt ':= ' ] < variable (VAR) of intnum > ';' 
```

### Información relacionada

| Para obtener más información sobre | Consulte  |
|------------------------------------|---|
| Resumen de interrupciones          | <i>Manual de referencia técnica - RAPID Overview</i>                        |
| Desactivación de una interrupción  | <a href="#"><i>ISleep - Desactiva una interrupción en la página 335</i></a> |

# 1 Instrucciones

---

1.125 Etiqueta - Nombre de línea  
*RobotWare - OS*

## 1.125 Etiqueta - Nombre de línea

---

### Utilización

Label se utiliza para asignar un nombre a una línea del programa. Cuando se usa la instrucción `GOTO`, este nombre puede usarse para trasladar la ejecución del programa a otro punto dentro de la misma rutina.

---

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción `Label`:

#### Ejemplo 1

```
GOTO next ;  
...  
next :
```

La ejecución del programa continúa en la instrucción que sigue a la etiqueta `next`.

---

### Argumentos

`Label` :

`Label`

*Identifier*

El nombre que desea asignar a la línea.

---

### Ejecución de programas

Cuando se ejecuta esta instrucción, no ocurre nada.

---

### Limitaciones

La etiqueta no debe tener el mismo nombre que ninguno de los elementos siguientes:

- Cualquier otra etiqueta dentro de la misma rutina.
- Cualquier nombre de dato dentro de la misma rutina

El uso de una etiqueta supone la ocultación de los datos globales y las rutinas que tengan el mismo nombre dentro de la rutina en la que se define.

---

### Sintaxis

```
<identifier>' : '
```

---

### Información relacionada

| Para obtener más información sobre                 | Consulte   |
|--|--|
| Identificadores                                    | <i>Manual de referencia técnica - RAPID Overview</i>             |
| Trasladar la ejecución del programa a una etiqueta | <a href="#">GOTO - Salta a otra instrucción en la página 242</a> |

## 1.126 Load - Carga un módulo de programa durante la ejecución

### Utilización

Load se utiliza para cargar un módulo de programa en la memoria de programas durante la ejecución.

El módulo de programa cargado se añade a los módulos que ya existen en la memoria de programa.

Los programas o módulos de sistema pueden cargarse en el modo estático (predeterminado) o en el modo dinámico.

Tanto los módulos cargados en modo estático como los cargados en modo dinámico se descargan al utilizar la instrucción UnLoad.

### Modo estático

En la tabla siguiente se describe cómo afectan las distintas operaciones a los programas o módulos de sistema cargados en el modo estático.

| Tipo de módulo     | Trasladar el puntero de programa a main desde el FlexPendant | Abrir un nuevo programa de RAPID |
|--------------------|--|----------------------------------|
| Módulo de programa | No se ve afectado  | Descargado                       |
| Módulo de sistema  | No se ve afectado  | No se ve afectado                |

### Modo dinámico

En la tabla siguiente se describe cómo afectan las distintas operaciones a los programas o módulos de sistema cargados en el modo dinámico.

| Tipo de módulo     | Trasladar el puntero de programa a main desde el FlexPendant | Abrir un nuevo programa de RAPID |
|--------------------|--|----------------------------------|
| Módulo de programa | Descargado   | Descargado                       |
| Módulo de sistema  | Descargado   | Descargado                       |

### Ejemplos básicos

Los ejemplos siguientes ilustran la instrucción Load.

Consulte también [Más ejemplos en la página 348](#).

#### Ejemplo 1

```
Load \Dynamic, diskhome \File:="PART_A.MOD";
```

Carga en la memoria de programa el módulo PART\_A.MOD, desde diskhome . diskhome es una constante predefinida de cadena de caracteres con el contenido "HOME:". Cargar el módulo de programa en modo dinámico.

#### Ejemplo 2

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

Carga el módulo de programa PART\_A.MOD en la memoria de programas y a continuación se carga PART\_B.MOD. Si PART\_A.MOD contiene referencias a PART\_B.MOD, puede usarse \CheckRef para comprobar si hay referencias no resueltas sólo cuando se carga el último módulo. Si se usa \CheckRef en PART\_A.MOD, se produciría un error de enlace y el módulo no se cargaría.

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.126 Load - Carga un módulo de programa durante la ejecución

RobotWare - OS

Continuación

---

### Argumentos

```
Load [\Dynamic] FilePath [\File] [\CheckRef]
```

[\Dynamic]

Tipo de dato: `switch`

El modificador permite cargar un módulo en modo dinámico. De lo contrario, la carga se realiza en modo estático.

FilePath

Tipo de dato: `string`

La ruta y el nombre del archivo que se cargará en la memoria de programa. El nombre de archivo se excluye cuando se utiliza el argumento `\File`.

[\File]

Tipo de dato: `string`

Cuando se excluye el nombre del archivo en el argumento `FilePath`, es necesario definirlo con este argumento.

[\CheckRef]

Tipo de dato: `switch`

Busca referencias no resueltas en la tarea de programa después de la carga del módulo. Si no se usa, no se realiza ninguna búsqueda de referencias no resueltas.

---

### Ejecución de programas

La ejecución del programa espera a que el módulo de programa termine de cargarse antes de continuar en la instrucción siguiente.

Las referencias no resueltas se aceptan siempre en la operación de carga, si el parámetro `\CheckRef` no se usa, pero se producirá un error de tiempo de ejecución durante la ejecución de las referencias no resueltas.

Una vez que el módulo de programa queda cargado, se vincula e inicializa. La inicialización del módulo cargado devuelve todas las variables del nivel de módulo a los valores de la unidad.

Si se produce algún error en la operación de carga, incluidas las referencias no resueltas si se usa el modificador `\CheckRef`, el módulo cargado no estará ya disponible en la memoria de programas.

Para conseguir una estructura de programa idónea, fácil de comprender y mantener, todas las operaciones de carga y descarga de módulos de programa deben hacerse en el módulo principal ("main") que siempre está presente en la memoria de programa durante la ejecución.

Para la carga de un programa que contiene un procedimiento principal desde un programa principal (que tiene su propio procedimiento principal), consulte [Más ejemplos en la página 348](#) a continuación.

---

### Más ejemplos

A continuación aparecen más ejemplos de cómo usar la instrucción `Load`.

### Más ejemplos generales

```
Load \Dynamic, "HOME:/DOORDIR/DOOR1.MOD";
```

*Continúa en la página siguiente*

---



## 1.126 Load - Carga un módulo de programa durante la ejecución

*RobotWare - OS*

*Continuación*

**Carga el módulo de programa DOOR1.MOD desde HOME: en el directorio DOORDIR en la memoria de programa. Se carga el módulo de programa en el modo dinámico.**

```
Load "HOME:" \File:="DOORDIR/DOOR1.MOD";
```

**Lo mismo que en el ejemplo anterior pero con otra sintaxis. El módulo se carga en el modo estático.**

```
Load\Dynamic, "HOME:/DOORDIR/DOOR1.MOD";  
%"routine_x"%;  
UnLoad "HOME:/DOORDIR/DOOR1.MOD";
```

**El procedimiento routine\_x se enlazará durante la ejecución (enlazamiento en tiempo de ejecución).**

**El módulo cargado contiene un procedimiento principal**

### **car.mod:**

```
MODULE car  
  PROC main()  
    ...  
  TEST part  
  CASE door_part:  
    Load \Dynamic, "HOME:/door.mod";  
    %door:main%;  
    Unload "HOME:/door.mod";  
  CASE window_part:  
    Load \Dynamic, "HOME:/window.mod";  
    %window:main%;  
    Unload \Save "HOME:/window.mod";  
  ENDTEST  
ENDPROC  
ENDMODULE
```

### **door.mod:**

```
MODULE door  
  PROC main()  
    ...  
ENDPROC  
ENDMODULE
```

### **window.mod:**

```
MODULE window  
  PROC main()  
    ...  
ENDPROC  
ENDMODULE
```

**En el ejemplo anterior se muestra cómo es posible cargar un módulo que incluye un procedimiento main. Este módulo puede haber sido desarrollado y probado por separado para su carga posterior con Load o StartLoad... WaitLoad en**

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.126 Load - Carga un módulo de programa durante la ejecución

RobotWare - OS

Continuación

el sistema, a través de un marco de trabajo del programa principal. En este ejemplo se trata de `car.mod`, que carga los módulos secundarios `door.mod` o `window.mod`.

En el módulo `car.mod` se carga `door.mod` o `window.mod` situados en "HOME:". Dado que los procedimientos `main` de `door.mod` y `window.mod` se consideran como locales LOCAL tras la carga del módulo del sistema, las llamadas a los procedimientos se realizan de la forma siguiente: `% "door:main"%` o `% "window:main"%`. Esta sintaxis se utiliza cuando se desea tener acceso a los procedimientos de tipo LOCAL de otros módulos, en este ejemplo el procedimiento `main` del módulo `door` o el módulo `window`.

La descarga de los módulos con el argumento `\Save` hará que los procedimientos `main` sean de nuevo globales en el programa guardado.

Si tras la carga de los módulos `car` o `window` se traslada el puntero de programa a `main` desde cualquier parte del programa, el puntero de programa siempre se traslada al procedimiento `main` global del programa principal, en este ejemplo `car.mod`.

### Limitaciones

Evite tener movimientos en curso durante la carga.

### Gestión de errores

Pueden generarse los errores recuperables enumerados a continuación. Estos errores pueden ser gestionados en un gestor de errores. El valor de la variable de sistema `ERRNO` cambia a:

| Nombre                      | Causa del error   |
|-----------------------------|---|
| <code>ERR_FILNOTFND</code>  | No se encuentra el archivo especificado en la instrucción <code>Load</code> .   |
| <code>ERR_IOERROR</code>    | Existe un problema al leer el archivo de la instrucción <code>Load</code> .   |
| <code>ERR_PRGMEMFULL</code> | Imposible cargar el módulo porque la memoria de programas está llena.   |
| <code>ERR_LOADED</code>     | El módulo ya está cargado en la memoria de programas.   |
| <code>ERR_SYNTAX</code>     | El módulo cargado contiene errores de sintaxis.   |
| <code>ERR_LINKREF</code>    | <ul style="list-style-type: none"><li>El módulo cargado da lugar a errores de vínculo no recuperables.</li><li>Si se usa <code>Load</code> con el modificador <code>\CheckRef</code> para buscar errores de referencia y la memoria de programas contiene referencias no resueltas.</li></ul> |

Si se produce alguno de estos errores, el módulo en sí será descargado y no estará disponible en el gestor de `ERROR`.

### Sintaxis

```
Load
  ['\Dynamic',']
  [FilePath':=']<expression (IN) of string>
  ['\File':=' <expression (IN) of string>]
  ['\CheckRef'];'
```

Continúa en la página siguiente

### Información relacionada

| Para obtener más información sobre   | Consulte   |
|--|--|
| Descarga de un módulo de programa  | <a href="#">UnLoad - Descargar un módulo de programa durante la ejecución en la página 1021</a>  |
| Carga de módulos de programa en paralelo con la ejecución de otro programa | <a href="#">StartLoad - Carga de programa durante la ejecución en la página 792</a><br><a href="#">WaitLoad - Conectar un módulo cargado a una tarea en la página 1064</a> |
| Comprobar referencias de programa  | <a href="#">CheckProgRef - Comprobar referencias de programa en la página 131</a>  |

# 1 Instrucciones

---

## 1.127 LoadId - Identificación de carga de la herramienta o la carga útil RobotWare-OS

### 1.127 LoadId - Identificación de carga de la herramienta o la carga útil

---

#### Utilización

LoadId (*Load Identification*) puede usarse para la identificación de la carga de una herramienta (también de una herramienta de pinza si tiene un TCP fijo en el espacio) o de una carga útil (se activa con la instrucción GripLoad), mediante la ejecución de un programa de RAPID definido por el usuario.



#### Nota

Una forma más fácil de identificar la carga de la herramienta o la carga útil es utilizar el programa de RAPID de diálogo interactivo LoadIdentify. Este programa puede iniciarse desde el menú ProgramEditor/Debug/Call Routine.../LoadIdentify.

#### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción LoadId:

Consulte también [Más ejemplos en la página 356](#).

#### Ejemplo 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";
```

*Continúa en la página siguiente*

1.127 LoadId - Identificación de carga de la herramienta o la carga útil

RobotWare-OS

Continuación

```
IF slow_test_flag = TRUE THEN
  %"LoadId"% TOOL_LOAD_ID, MASS_WITH_AX3, grip3 \SlowTest;
ENDIF
! 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/mockitl.sys";
```

Identificación de la carga de la herramienta grip3.

Condición

Las condiciones siguientes deben cumplirse antes de las mediciones de carga con LoadId:

- Asegúrese de que todas las cargas estén montadas correctamente en el robot.
- Compruebe si el tipo de robot es válido, con ayuda de ParIdRobValid
- Compruebe si la posición es válida, con ayuda de ParIdPosValid:
  - Los ejes 3, 5 y 6 no deben estar cerca de su área de trabajo correspondiente.
  - Carcasa de inclinación casi horizontal, es decir, con el eje 4 en la posición cero.
- Es necesario definir los datos siguientes en los parámetros del sistema y en los argumentos de LoadId antes de ejecutar LoadId

En la tabla siguiente se ilustra la identificación de carga de la herramienta.

| Modos de identificación de carga / Datos definidos antes de LoadId | Se conoce la masa del TCP móvil | Se desconoce la masa del TCP móvil | Se conoce la masa del TCP fijo | Se desconoce la masa del TCP fijo |
|--|---------------------------------|------------------------------------|--------------------------------|-----------------------------------|
| Carga del brazo superior (parámetro del sistema)                   |                                 | Definido                           |                                | Definido                          |
| Masa de la herramienta   | Definido                        |                                    | Definido                       |                                   |

En la tabla siguiente se ilustra la identificación de carga de la carga útil.

| Modos de identificación de carga / Datos definidos antes de LoadId | Se conoce la masa del TCP móvil | Se desconoce la masa del TCP móvil | Se conoce la masa del TCP fijo | Se desconoce la masa del TCP fijo |
|--|---------------------------------|------------------------------------|--------------------------------|-----------------------------------|
| Carga del brazo superior (parámetros del sistema)                  |                                 | Definido                           |                                | Definido                          |
| Datos de carga de la herramienta                                   | Definido                        | Definido                           | Definido                       | Definido                          |
| Masa de la carga útil  | Definido                        |                                    | Definido                       |                                   |
| Base de coordenadas de herramienta en la herramienta               | Definido                        | Definido                           |                                |                                   |
| Base de coordenadas de usuario en el objeto de trabajo             |                                 |                                    | Definido                       | Definido                          |

Continúa en la página siguiente

# 1 Instrucciones

## 1.127 LoadId - Identificación de carga de la herramienta o la carga útil

RobotWare-OS

Continuación

| Modos de identificación de carga / Datos definidos antes de LoadId | Se conoce la masa del TCP móvil | Se desconoce la masa del TCP móvil | Se conoce la masa del TCP fijo | Se desconoce la masa del TCP fijo |
|--|---------------------------------|------------------------------------|--------------------------------|-----------------------------------|
| Base de coordenadas de objeto en el objeto de trabajo              |                                 |                                    | Definido                       | Definido                          |

- Modo de funcionamiento y redefinición de velocidad:
  - Prueba lenta en el modo manual con velocidad reducida
  - Mediciones de carga en el modo automático (o en el modo manual a máxima velocidad) con una redefinición de velocidad del 100%

### Argumentos

```
LoadId ParIdType LoadIdType Tool [\PayLoad] [\WObj] [\ConfAngle] [\SlowTest] [\Accuracy]
```

ParIdType

Tipo de dato: paridnum

Un tipo de identificación de carga de los definidos en la tabla siguiente.

| Valor | Constante simbólica | Comentario   |
|-------|---------------------|--|
| 1     | TOOL_LOAD_ID        | Identificación de la carga de la herramienta                       |
| 2     | PAY_LOAD_ID         | Identificación de la carga útil (consulte la instrucción GripLoad) |

LoadIdType

Tipo de dato: loadidnum

Un tipo de identificación de carga de los definidos en la tabla siguiente.

| Valor | Constante simbólica | Comentario  |
|-------|---------------------|---|
| 1     | MASS_KNOWN          | Masa conocida de la herramienta o de la carga útil respectivamente. (Debe especificarse la masa de la herramienta o la carga útil especificada)                                 |
| 2     | MASS_WITH_AX3       | Masa de la herramienta o de la carga útil desconocida respectivamente. La identificación de la masa de la herramienta o la carga útil se realiza con los movimientos del eje 3. |

Tool

Tipo de dato: tooldata

Variable persistente de la herramienta que se desea identificar. Si se especifica el argumento `\PayLoad`, la variable persistente de la herramienta en uso.

Para la identificación de la carga de la herramienta, no deben especificarse los argumentos `\PayLoad` ni `\WObj`.

[ \ PayLoad ]

Tipo de dato: loaddata

Variable persistente de la carga útil que se desea identificar.

Este argumento opcional debe especificarse siempre para la identificación de carga de una carga útil.

Continúa en la página siguiente

[ \ WObj ]

Tipo de dato: wobjdata

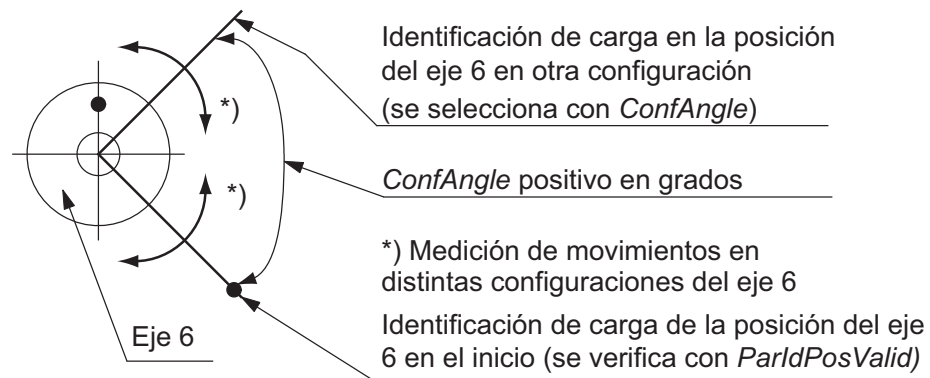
Variable persistente del objeto de trabajo en uso.

Este argumento opcional debe especificarse siempre para la identificación de carga de una carga útil con un TCP fijo en el espacio.

[ \ ConfAngle ]

Tipo de dato: num

Este argumento opcional permite especificar un valor ángulo  $\pm$  grados de configuración determinado para usarlo en la identificación de parámetros.



xx0500002198

Si no se especifica este argumento, el valor predeterminado es +90 grados. Mín. + ó -30 grados. Valor óptimo + o -90 grados.

[ \ SlowTest ]

Tipo de dato: switch

Este argumento opcional permite especificar si debe realizarse una prueba lenta para la comprobación del área de trabajo libre. Consulte la tabla siguiente:

|                      |   |
|----------------------|---|
| LoadId ... \SlowTest | Realizar sólo la prueba lenta   |
| LoadId ...           | Realizar sólo la medición y actualizar la herramienta o la carga útil |

[ \ Accuracy ]

Tipo de dato: num

Una variable para la salida de la exactitud de medición calculada en % para todo el cálculo de identificación de la carga (el 100% significa la máxima exactitud).

### Ejecución de programas

El robot realizará un gran número de movimientos relativos pequeños de transporte y medición en los ejes 5 y 6. Para la identificación de la masa, también se realizarán movimientos con el eje 3.

Después del conjunto de mediciones, movimientos y cálculos de carga, los datos de carga se devuelven en el argumento Tool o PayLoad. Se calculan los datos de carga siguientes:

- Masa en kg (si la masa es desconocida, de lo contrario no se ve afectado)

Continúa en la página siguiente

# 1 Instrucciones

---

## 1.127 LoadId - Identificación de carga de la herramienta o la carga útil

RobotWare-OS

Continuación

- Centro de gravedad x, y, z y eje de momento
- Inercia ix, iy, iz en kgm

---

### Más ejemplos

A continuación aparecen más ejemplos de la instrucción LoadId.

#### Ejemplo 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];
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";
```

Identificación de carga de la carga útil piece5 con una masa conocida en una instalación cuyo TCP está fijo en el espacio.

---

### Limitaciones

Normalmente, la identificación de la carga de la herramienta o de la carga útil se realiza con la rutina de servicio LoadIdentify. También es posible realizar esta identificación con la instrucción de RAPID LoadId. Antes de cargar o ejecutar el programa con LoadId, debe cargar los módulos siguientes en el sistema:

```
Load \Dynamic, "RELEASE:/system/mockit.sys";
Load \Dynamic, "RELEASE:/system/mockit1.sys";
```

En este caso, es posible llamar a LoadId con una llamada con enlazamiento en tiempo de ejecución (consulte el ejemplo 1 anterior).

No es posible reiniciar los movimientos de identificación de la carga después de ningún tipo de paro, como el paro programado, el paro de emergencia o la caída de alimentación. En este caso, los movimientos de identificación de carga deben iniciarse desde el principio.

Continúa en la página siguiente



**Gestión de errores**

Pueden generarse los errores recuperables enumerados a continuación. Estos errores pueden ser gestionados en un gestor de errores. El valor de la variable de sistema ERRNO cambia a:

| Nombre           | Causa del error   |
|------------------|---|
| ERR_PID_MOVESTOP | En caso de cualquier error durante la ejecución de RAPID NOSTEPIN, rutina LoadId.<br>El puntero de programa es elevado hacia la llamada del usuario a LoadId. |
| ERR_PID_RAISE_PP |   |
| ERR_LOADID_FATAL |   |

**Sintaxis**

```
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> ] ';'

```

**Información relacionada**

| Para obtener más información sobre  | Consulte  |
|-------------------------------------|---|
| Programa predefinido Load Identify  | <i>Manual del operador - IRC5 con FlexPendant</i>   |
| Tipo de identificación de parámetro | <a href="#">paridnum - Tipo de identificación de parámetro en la página 1721</a>                                |
| Resultado de ParIdRobValid          | <a href="#">paridvalidnum - Resultado de ParIdRobValid en la página 1723</a>                                    |
| Tipo de identificación de carga     | <a href="#">loadidnum - Tipo de identificación de carga en la página 1700</a>                                   |
| Tipo de robot válido                | <a href="#">ParIdRobValid - Tipo de robot válido para la identificación de parámetros en la página 1403</a>     |
| Posición de robot válida            | <a href="#">ParIdPosValid - Posición de robot válida para la identificación de parámetros en la página 1400</a> |

# 1 Instrucciones

## 1.128 MakeDir - Crea un nuevo directorio

RobotWare - OS

## 1.128 MakeDir - Crea un nuevo directorio

### Utilización

`MakeDir` se utiliza para crear un nuevo directorio. El usuario debe tener permisos de escritura y ejecución del directorio superior debajo del cual se creará el nuevo directorio.

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción `MakeDir`:

#### Ejemplo 1

```
MakeDir "HOME:/newdir";
```

En este ejemplo se crea un nuevo directorio llamado `newdir` dentro de `HOME`:

### Argumentos

```
MakeDir Path
```

Path

Tipo de dato: `string`

El nombre del nuevo directorio, especificado con una ruta completa o relativa.

### Gestión de errores

Pueden generarse los errores recuperables enumerados a continuación. Estos errores pueden ser gestionados en un gestor de errores. El valor de la variable de sistema `ERRNO` cambia a:

| Nombre                   | Causa del error                |
|--------------------------|--------------------------------|
| <code>ERR_FILEACC</code> | Imposible crear el directorio. |

### Sintaxis

```
MakeDir  
[ Path':=' ] < expression (IN) of string>';'
```

### Información relacionada

| Para obtener más información sobre              | Consulte   |
|---|--|
| Eliminación de un directorio                    | <a href="#">RemoveDir - Elimina un directorio en la página 598</a>                     |
| Cambio del nombre de un archivo                 | <a href="#">RenameFile - Cambia el nombre de un archivo en la página 604</a>           |
| Eliminación de un archivo                       | <a href="#">RemoveFile - Elimina un archivo en la página 600</a>                       |
| Copia de un archivo                             | <a href="#">CopyFile - Copia un archivo en la página 169</a>                           |
| Comprobación del tipo del archivo               | <a href="#">IsFile - Comprobar el tipo de un archivo en la página 1351</a>             |
| Comprobación del tamaño del archivo             | <a href="#">FileSize - Obtiene el tamaño de un archivo en la página 1285</a>           |
| Comprobación del tamaño del sistema de archivos | <a href="#">FSSize - Obtiene el tamaño de un sistema de archivos en la página 1291</a> |

Continúa en la página siguiente

| Para obtener más información sobre        | Consulte   |
|---|--|
| Gestión de archivos y dispositivos de E/S | <i>Application manual - Controller software IRC5</i> |

# 1 Instrucciones

## 1.129 ManLoadIdProc - Identificación de carga de los manipuladores IRBP

RobotWare-OS

## 1.129 ManLoadIdProc - Identificación de carga de los manipuladores IRBP

### Utilización

ManLoadIdProc (*Manipulator Load Identification Procedure*) se utiliza para la identificación de carga de la carga útil de los manipuladores externos, mediante la ejecución de un programa de RAPID definido.

Esta instrucción sólo puede usarse en la tarea principal T\_ROB1 o, si se cuenta con un sistema *MultiMove*, en las tareas de movimiento.



#### Nota

Una forma más fácil de identificar la carga útil es utilizar la rutina de servicio **ManLoadIdentify**. La rutina de servicio puede iniciarse desde el menú **Editor de programas, Depurar, Llamar a rutina, ManLoadIdentify**.

### Ejemplos básicos

Los ejemplos siguientes ilustran la instrucción 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;
```

Identificación de carga de la carga útil myload montada sobre la unidad mecánica STN1. El manipulador externo es del tipo IRBP-L. Se establece el ángulo de configuración en 60 grados. El manipulador se activa antes de la identificación de carga y se desactiva a continuación. Tras la identificación, myload se ha actualizado y su definición cambia a TRUE.

### Argumentos

```
ManLoadIdProc [\ParIdType] [\MechUnit] | [\MechUnitName]
[\AxisNumber] [\PayLoad] [\ConfigAngle] [\DeactAll] |
[\AlreadyActive] [DefinedFlag] [DoExit]
```

[ \ ParIdType ]

**Tipo de dato:** paridnum

**Tipo de identificación de parámetro** Las constantes predefinidas se encuentran dentro del tipo de dato paridnum.

[ \ MechUnit ]

**Tipo de dato:** mecunit

La unidad mecánica utilizada para la identificación de carga. No puede utilizarse conjuntamente con el argumento \MechUnitName.

*Continúa en la página siguiente*

[ \ MechUnitName ]

Tipo de dato: `string`

La unidad mecánica utilizada para la identificación de carga, indicada como una cadena. No puede utilizarse conjuntamente con el argumento `\MechUnit`.

[ \ AxisNumber ]

Tipo de dato: `num`

Dentro de la unidad mecánica, el número del eje que sostiene la carga que se desea identificar.

[ \ PayLoad ]

Tipo de dato: `loaddata`

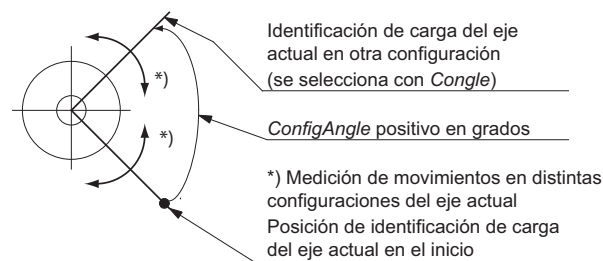
Variable de la carga útil que se desea identificar. Es necesario especificar el componente `mass`.

Esta variable se actualiza después de la identificación.

[ \ ConfigAngle ]

Tipo de dato: `num`

Especificación de un ángulo de configuración específico  $\pm$  grados para su uso en la identificación de parámetros.



xx0500002197

Mín. + ó -30 grados. Valor óptimo + o -90 grados.

[ \ DeactAll ]

Tipo de dato: `switch`

Si se utiliza este modificador, todas las unidades mecánicas del sistema se desactivarán antes de la realización de la identificación. A continuación, se activa la unidad mecánica a identificar. No puede utilizarse conjuntamente con el argumento `\AlreadyActive`.

[ \ AlreadyActive ]

Tipo de dato: `switch`

Este modificador se utiliza si la unidad mecánica a identificar está activa. No puede utilizarse conjuntamente con el argumento `\DeactAll`.

[ \ DefinedFlag ]

Tipo de dato: `bool`

Continúa en la página siguiente

# 1 Instrucciones

## 1.129 ManLoadIdProc - Identificación de carga de los manipuladores IRBP

RobotWare-OS

Continuación

Este argumento tendrá el valor `TRUE` si la identificación ha sido realizada o `FALSE` en caso contrario.

[ \ DoExit ]

Tipo de dato: `bool`

Si tiene el valor `TRUE`, la identificación de la carga terminará con un comando `EXIT` para forzar al usuario a cambiar el PP a Main antes de continuar con la ejecución.

Si no está presente o tiene el valor `FALSE`, no se ejecuta `EXIT`. Recuerde que `ManLoadIdProc` siempre elimina la trayectoria actual.

### Ejecución de programas

Todos los argumentos son opcionales. Si no se indica un argumento, se solicitará al usuario el valor a través del FlexPendant (excepto `\DoExit`).

Siempre se pedirá al usuario que indique la masa y, si el manipulador es del tipo IRBP R, el valor de z en mm.

La unidad mecánica realizará un gran número de movimientos relativos pequeños de transporte y medición.

Después del conjunto de mediciones, movimientos y cálculos de carga, los datos de carga se devuelven en el argumento `Payload` si se usa. Se calculan los datos de carga siguientes.

| Tipo de manipulador/datos de carga calculados  | IRBP-K                   | IRBP-L<br>IRBP-C<br>IRBP_T | IRBP-R  | IRBP-A<br>IRBP-B<br>IRBP-D                            |
|--|--------------------------|----------------------------|---|---|
| Parámetro <code>Payload</code> - <code>cog.x</code> ,<br><code>cog.y</code> , <code>cog.z</code> en <code>loaddata</code> en mm                  | <code>cog.x cog.y</code> | <code>cog.x cog.y</code>   | <code>cog.x cog.y</code>                              | <code>cog.x cog.y cog.z</code>                        |
| Parámetro <code>Payload</code> - <code>ix</code> , <code>iy</code> , <code>iz</code><br>en <code>loaddata</code> en <code>kgm<sup>2</sup></code> | <code>iz</code>          | <code>iz</code>            | <code>ix</code><br><code>iy</code><br><code>iz</code> | <code>ix</code><br><code>iy</code><br><code>iz</code> |

Los datos calculados se muestran en el FlexPendant.

### Limitaciones

Normalmente, la identificación de la carga del manipulador externo se realiza con la rutina de servicio `ManLoadIdentify`. También es posible realizar esta identificación con la instrucción de `RAPID ManLoadIdProc`.

Cualquier trayectoria en curso se borrará antes de la identificación de la carga. El puntero de programa se perderá tras la identificación de la carga si se usa el argumento `\DoExit:=TRUE`.

No es posible reiniciar los movimientos de identificación de la carga después de ningún tipo de paro, como el paro programado, el paro de emergencia o la caída de alimentación. Los movimientos de identificación de carga deben reiniciarse de nuevo desde el principio.

Continúa en la página siguiente

**Gestión de errores**

Pueden generarse los errores recuperables enumerados a continuación. Estos errores pueden ser gestionados en un gestor de errores. El valor de la variable de sistema ERRNO cambia a:

| Nombre           | Causa del error   |
|------------------|---|
| ERR_PID_MOVESTOP | En caso de cualquier error durante la ejecución de RAPID NOSTEPIN, rutina ManLoadIdProc.<br>El puntero de programa es elevado hacia la llamada del usuario a ManLoadIdProc. |
| ERR_PID_RAISE_PP |   |
| ERR_LOADID_FATAL |   |

**Sintaxis**

```
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> ] ';'

```

**Información relacionada**

| Para obtener más información sobre  | Consulte   |
|-------------------------------------|--|
| Tipo de identificación de parámetro | <a href="#">paridnum - Tipo de identificación de parámetro en la página 1721</a> |
| Unidad mecánica                     | <a href="#">mecunit - Unidad mecánica en la página 1703</a>                      |
| Carga útil                          | <a href="#">loaddata - Datos de carga en la página 1694</a>                      |

# 1 Instrucciones

---

## 1.130 MatrixSolve - Soluciona un sistema de ecuaciones lineales

### 1.130 MatrixSolve - Soluciona un sistema de ecuaciones lineales

---

#### Utilización

`MatrixSolve` Se usa para solucionar sistemas de ecuaciones lineales del tipo  $A*x=b$ .

---

#### Ejemplos básicos

Los siguientes ejemplos ilustran la instrucción `MatrixSolve`.

Consulte también [Más ejemplos en la página 365](#).

#### Ejemplo 1

```
VAR dnum A1{3,3}:=[[5, 2, 7],[-3, 1, 1],[1, 10, -3]];
VAR dnum b1{3}:=[-22, 39, 54];
VAR dnum x1{3};
...
MatrixSolve A1, b1, x1;
```

El ejemplo anterior soluciona el sistema de ecuaciones lineales. La matriz `x1` tendrá el valor [-10, 7, 2].

#### Ejemplo 2

```
VAR dnum A2{1,1} := [[5]];
VAR dnum b2{1}:=[35];
VAR dnum x2{1};
...
MatrixSolve A2, b2, x2;
```

El ejemplo anterior soluciona la ecuación trivial  $5x = 35$ . La respuesta es 7.

---

#### Argumentos

`MatrixSolve A [ $\backslash A_m$ ] [ $\backslash A_n$ ] b x`

A

**Tipo de dato:** array of dnum

A es una matriz de dimensiones  $m * n$ , donde  $m \geq n$ . La letra  $m$  indica el número de filas, y la letra  $n$  indica el número de columnas de la matriz. Si  $m > n$ , entonces el sistema de ecuaciones está sobredeterminado y devuelve una solución de mínimos cuadrados.

[ $\backslash A_m$ ]

**Tipo de dato:** num

Con el argumento opcional  $A_m$  es posible especificar cuántas filas ( $m$ ) de la matriz A deben usarse.

[ $\backslash A_n$ ]

**Tipo de dato:** num

Con el argumento opcional  $A_n$  es posible especificar cuántas columnas ( $n$ ) de la matriz A deben usarse.

b

**Tipo de dato:** array of dnum

*Continúa en la página siguiente*

---



### 1.130 MatrixSolve - Soluciona un sistema de ecuaciones lineales

Continuación

$b$  es una matriz con la misma dimensión que las filas ( $m$ ) de la matriz  $A$ . Si utiliza una matriz con una dimensión mayor que las filas ( $m$ ) de la matriz  $A$ , los componentes con índice superior a  $m$  se pondrán a 0.

$x$

Tipo de dato: array of dnum

$x$  es una matriz con la misma dimensión que las columnas ( $n$ ) de la matriz  $A$ . Se trata de una matriz variable en la que queda guardado el resultado del cálculo. Si utiliza una matriz con una dimensión mayor que las columnas ( $n$ ) de la matriz  $A$ , los componentes con índice superior a  $n$  se pondrán a 0.

#### Ejecución de programas

`MatrixSolve` se utiliza para solucionar sistemas de ecuaciones lineales del tipo  $A*x=b$ . Si el sistema de ecuaciones está sobredeterminado, devolverá una solución de mínimos cuadrados.

Si utiliza los argumentos opcionales  $A_m$  y  $A_n$  es posible utilizar la misma matriz para cálculos diferentes con matrices de diferentes tamaños.

#### Gestión de errores

Se generan los siguientes errores recuperables que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` se establecerá en:

|                                  |   |
|----------------------------------|---|
| <code>ERR_ARRAY_SIZE</code>      | Se utilizan dimensiones incorrectas o valores incorrectos en los argumentos opcionales. |
| <code>ERR_MATRIX_SINGULAR</code> | La matriz de entrada es singular.   |

#### Más ejemplos

A continuación aparecen más ejemplos de la función `MatrixSolve`.

##### Ejemplo 1

```
VAR dnum A1{5,5}:=[[5,2,7,0,0], [-3,1,1,0,0], [1,10,-3,0,0]];
VAR dnum b1{8}:=[-22,39,54,0,0,0,0,0];
VAR dnum x1{8};
```

```
MatrixSolve A1 \A_m:=3 \A_n:=3, b1, x1;
```

El ejemplo anterior soluciona el sistema de ecuaciones lineales. La matriz  $x1$  tendrá el valor  $[-10, 7, 2, 0, 0, 0, 0, 0]$ . Este ejemplo es el mismo que [Ejemplo 1 en la página 364](#). La única diferencia es que en este ejemplo se ilustra cómo utilizar los argumentos opcionales  $A_m$  y  $A_n$  y que las matrices más grandes que  $m$  y  $n$  pueden usarse como argumentos  $b$  y  $x$ .  $A_m$  y  $A_n$  pueden usarse para limitar el tamaño de la matriz, por lo que una matriz general grande puede usarse para solucionar diferentes sistemas de ecuaciones.

Continúa en la página siguiente

# 1 Instrucciones

## 1.130 MatrixSolve - Soluciona un sistema de ecuaciones lineales

### Continuación

#### Limitaciones

A la hora de solucionar matrices grandes, el tamaño de la asignación de memoria reservada puede no ser suficiente para completar los cálculos actuales y se generará un informe en el que se registran los eventos. El tamaño de la asignación de memoria reservada es fijo y no puede cambiarse. Inténtelo solucionarlo usando matrices más pequeñas.

#### Sintaxis

```
MatrixSolve
[ A ' := ' ] < array {*}{*} expression (IN) of dnum > ', '
[ '\ ' A_m ' := ' < expression (IN) of num > ]
[ '\ ' A_n ' := ' < expression (IN) of num > ] ', '
[ b ' := ' ] < array {*} expression (IN) of dnum > ', '
[ x ' := ' ] < array variable {*} (VAR) of dnum > ';'

```

#### Información relacionada

| Para obtener más información sobre                | Consulte  |
|---|---|
| Calcula una factorización QR.                     | <a href="#">MatrixSolveQR - Calcula una factorización QR en la página 367</a>                 |
| Calcula una descomposición en valores singulares. | <a href="#">MatrixSVD - Calcula una descomposición en valores singulares en la página 369</a> |
| Instrucciones y funciones matemáticas.            | <a href="#">Manual de referencia técnica - RAPID Overview</a>                                 |

## 1.131 MatrixSolveQR - Calcula una factorización QR

### Utilización

`MatrixSolveQR` Se utiliza para calcular una factorización QR de una matriz A (m x n).

### Ejemplos básicos

El ejemplo siguiente ilustra la instrucción `MatrixSolveQR`.

#### Ejemplo 1

```
VAR dnum A4{3,3}:=[[12,-51,4], [6,167,-68], [-4,24,-41]];
VAR dnum Q4{3,3};
VAR dnum R4{3,3};
MatrixSolveQR A4, Q4, R4;
```

La instrucción `MatrixSolveQR` se utiliza para calcular una factorización QR de una matriz A4 (m x n). El resultado del cálculo es:

```
Q4 := [[-0.857142857142857,0.394285714285714,0.331428571428571],
        [-0.428571428571429,-0.902857142857143,-0.0342857142857143],
        [0.285714285714286,-0.171428571428571,0.942857142857143]];

R4 := [[-14,-21,14], [0,-175,70], [0,0,-35]];
```

### Argumentos

`MatrixSolveQR A [\A_m] [\A_n] Q R`

A

Tipo de dato: array of dnum

A Es una matriz de dimensiones m x n, donde m es el número de filas y n es el número de columnas.

[\A\_m]

Tipo de dato: num

Con el argumento opcional `A_m` es posible especificar cuántas filas m de la matriz A deben usarse.

[\A\_n]

Tipo de dato: num

Con el argumento opcional `A_n` es posible especificar cuántas columnas n de la matriz A deben usarse.

Q

Tipo de dato: array of dnum

Matriz ortogonal (m x m). Es una variable matricial en la que queda guardado el resultado del cálculo.

R

Tipo de dato: array of dnum

(m x n) matriz triangular superior. Es una variable matricial en la que queda guardado el resultado del cálculo.

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.131 MatrixSolveQR - Calcula una factorización QR

Continuación

### Ejecución de programas

`MatrixSolveQR` Calcula una factorización-QR de una matriz  $A$  ( $m \times n$ ) tal que  $A=Q \cdot R$ , donde  $Q$  es una matriz ortogonal ( $m \times m$ ) y  $R$  es una matriz triangular superior ( $m \times n$ ).

Si utiliza los argumentos opcionales  $A_m$  y  $A_n$  es posible utilizar la misma matriz para cálculos diferentes que utilicen diferentes tamaños de la matriz.

### Gestión de errores

Se generan los siguientes errores recuperables que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` se establecerá en:

|                             |   |
|-----------------------------|---|
| <code>ERR_ARRAY_SIZE</code> | Se utilizan dimensiones incorrectas o valores incorrectos en los argumentos opcionales. |
|-----------------------------|---|

### Limitaciones

A la hora de solucionar matrices grandes, el tamaño de la asignación de memoria reservada puede no ser suficiente para completar los cálculos actuales y se generará un informe en el que se registran los eventos. El tamaño de la asignación de memoria reservada es fijo y no puede cambiarse. Inténtelo solucionarlo usando matrices más pequeñas.

### Sintaxis

```
MatrixSolveQR
[ A := ] < array {*}{*} expression (IN) of dnum > ','
[ '\ A_m := ' < expression (IN) of num > ]
[ '\ A_n := ' < expression (IN) of num > ] ','
[ Q := ] < array variable {*}{*} (VAR) of dnum > ','
[ R := ] < array variable {*}{*} (VAR) of dnum > ';'

```

### Información relacionada

| Para obtener más información sobre                | Consulte  |
|---|---|
| Soluciona un sistema de ecuaciones lineales.      | <a href="#">MatrixSolve - Soluciona un sistema de ecuaciones lineales en la página 364</a>    |
| Calcula una descomposición en valores singulares. | <a href="#">MatrixSVD - Calcula una descomposición en valores singulares en la página 369</a> |
| Instrucciones y funciones matemáticas.            | <a href="#">Manual de referencia técnica - RAPID Overview</a>                                 |

## 1.132 MatrixSVD - Calcula una descomposición en valores singulares

### Utilización

**MatrixSVD** Se utiliza para calcular una descomposición en valores singulares (SVD).

La descomposición en valores singulares (SVD) es una factorización de una matriz real, con varias aplicaciones útiles para el procesamiento y la estadística de señales.

### Ejemplos básicos

El ejemplo siguiente ilustra la instrucción **MatrixSVD**.

#### Ejemplo 1

```
VAR dnum A3{7,5}:=[[32,5,30,-47,16], [41,46,-36,35,-33],
  [-38,47,-8,44,21], [42,-35,42,18,-47], [13,48,30,26,-23],
  [-41,46,46,25,-46], [-22,-1,16,-11,-41]];
VAR dnum U3{7,7};
VAR dnum S3{5};
VAR dnum V3{5,5};
```

```
MatrixSVD A3, U3, S3, V3;
```

Se utiliza la instrucción **MatrixSVD** para calcular una descomposición en valores singulares. El resultado del cálculo es:

```
U3 := [ [-0.24489453114765, -0.241308890179438, -0.0602284681243788,
  -0.835993641906923, -0.0767894261551876, 0.240015157740137,
  0.340264541994411], [0.36884312087718, -0.011165754164993,
  0.807113814553714, -0.0174513269190971, -0.383355889416929,
  0.231289898525022, 0.107870591684966], [0.421298684810973,
  0.496825721418653, -0.0754892815524551, -0.0380766666594926,
  0.433352499740395, 0.0439085586237023, 0.615467956914505],
  [0.026022505267863, -0.748510046516493, 0.145750281133738,
  0.297910761636627, 0.491463032720705, 0.125969427524946,
  0.267688931284032], [0.44990665677733, -0.194363365765786,
  0.11415708386435, -0.406351234561054, 0.210650943912061,
  -0.684691075563818, -0.261683066861719], [0.634107267122289,
  -0.191796131356257, -0.458818389285907, -0.058536500937226,
  -0.126605540694026, 0.499569153226195, -0.285627519159172],
  [0.145957672500732, -0.24490688904148, -0.307330264960591,
  0.205089199291144, -0.597836567476733, -0.390807165865741,
  0.521598216836665]];
```

```
S3 := [128.223078192708, 106.345877681972, 86.7728210622664,
  62.5176992467654, 42.2777876032412];
```

```
V3 := [ [-0.241697068016687, -0.449209801318353, 0.774566517264602,
  -0.334347996967586, 0.16748495146732], [0.664865161158152,
  0.281358669789186, 0.148180746944642, -0.633450002424302,
  -0.235743880251818], [0.172346698575578, -0.571920245958167,
  -0.538598865014224, -0.371678506044732, 0.463648787880432],
  [0.541008847671309, 0.114564782377902, 0.291104463782398,
  0.449207178575608, 0.638479004558623], [-0.420883460142759,
  0.61549167032206, -0.0570844141501356, -0.383432536716582,
  0.541985217686462]];
```

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.132 MatrixSVD - Calcula una descomposición en valores singulares

### Continuación

---

#### Argumentos

MatrixSVD A [ $\backslash A_m$ ] [ $\backslash A_n$ ] U S V [ $\backslash Econ$ ]

A

Tipo de dato: array of dnum

A Es una matriz con las dimensiones  $m \times n$ , donde  $m$  es el número de filas, y  $n$  es el número de columnas.

[ $\backslash A_m$ ]

Tipo de dato: num

Con el argumento opcional  $A_m$  es posible especificar cuántas filas  $m$  de la matriz A deben usarse.

[ $\backslash A_n$ ]

Tipo de dato: num

Con el argumento opcional  $A_n$  es posible especificar cuántas columnas  $n$  de la matriz A deben usarse.

U

Tipo de dato: array of dnum

U Son los vectores singulares izquierdos de A, guardados como una matriz  $m \times k$ , donde  $k$  es igual a las columnas de la matriz A (o  $A_n$ ) si se utiliza el interruptor  $\backslash Econ$ , de lo contrario es el mismo que las filas de la matriz A (o  $A_m$ ). Es una variable matricial en la que queda guardado el resultado del cálculo. Si utiliza una matriz de dimensión mayor que las filas ( $m$ ) y las columnas ( $n$ ) de la matriz A, los componentes con índice superior a  $m$  y  $n$  se pondrán a 0.

S

Tipo de dato: array of dnum

S Es una matriz de dimensión  $\text{MIN}(A_m, A_n)$  con los valores  $\geq 0$ . Es una variable matricial en la que queda guardado el resultado del cálculo. Si utiliza una matriz con una dimensión más grande de lo necesario, los componentes con índice superior a  $n$  se pondrán a 0.

V

Tipo de dato: array of dnum

V Son los vectores singulares derechos de A guardados como una matriz  $n \times n$ .

[ $\backslash Econ$ ]

(Economy size)

Tipo de dato: switch

Si se utiliza  $\backslash Econ$  y  $m > n$ , entonces solo se calcularán los primeros  $n$  vectores singulares de U.

---

#### Ejecución de programas

MatrixSVD Se utiliza para calcular una descomposición en valores singulares (SVD) de la matriz de entrada A ( $m \times n$ ).

Continúa en la página siguiente

### 1.132 MatrixSVD - Calcula una descomposición en valores singulares

Continuación

Una SVD de la matriz  $A$  puede escribirse como  $A=U*S*V^T$ , donde  $U$  es ( $m \times m$ ) (vectores singulares izquierdos),  $V$  es ( $n \times n$ ) (vectores singulares derechos) y  $S$  es una matriz diagonal ( $m \times n$ ) con elementos no negativos. Los elementos de la diagonal son los valores singulares de  $A$ .

Para ahorrar espacio, solo se devuelven los valores singulares de  $S$ , y no la matriz completa. Entonces  $S$  se representa como una matriz de longitud  $\text{MIN}(m, n)$ . Los valores singulares se devuelven siempre en orden decreciente.

Con  $m > n$ , puede ahorrar más espacio al calcular solo los primeros  $n$  vectores singulares de  $U$ . Esto se controla con el interruptor `\Econ`. Por lo tanto, si se utiliza  $m > n$  y `\Econ`, entonces  $U$  es ( $m \times n$ ), de lo contrario  $U$  es ( $m \times m$ ).

Si utiliza los argumentos opcionales  $A_m$  y  $A_n$  es posible utilizar la misma matriz para cálculos diferentes que utilicen diferentes tamaños de la matriz.

#### Gestión de errores

Se generan los siguientes errores recuperables que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` se establecerá en:

|                             |   |
|-----------------------------|---|
| <code>ERR_ARRAY_SIZE</code> | Se utilizan dimensiones incorrectas o valores incorrectos en los argumentos opcionales. |
|-----------------------------|---|

#### Limitaciones

A la hora de solucionar matrices grandes, el tamaño de la asignación de memoria reservada puede no ser suficiente para completar los cálculos actuales y se generará un informe en el que se registran los eventos. El tamaño de la asignación de memoria reservada es fijo y no puede cambiarse. Inténtelo solucionarlo usando matrices más pequeñas.

#### Sintaxis

```
MatrixSVD
[ A :=' ] < array {*}{*} expression (IN) of dnum > ','
[ '\ A_m :=' < expression (IN) of num > ]
[ '\ A_n :=' < expression (IN) of num > ] ','
[ U :=' ] < array variable {*}{*} (VAR) of dnum > ','
[ S :=' ] < array variable {*} (VAR) of dnum > ','
[ V :=' ] < array variable {*}{*} (VAR) of dnum > ','
[ '\ Econ ] ';'

```

#### Información relacionada

| Para obtener más información sobre           | Consulte   |
|--|--|
| Soluciona un sistema de ecuaciones lineales. | <a href="#">MatrixSolve - Soluciona un sistema de ecuaciones lineales en la página 364</a> |
| Calcula una factorización QR                 | <a href="#">MatrixSolveQR - Calcula una factorización QR en la página 367</a>              |
| Instrucciones y funciones matemáticas.       | <a href="#">Manual de referencia técnica - RAPID Overview</a>                              |

# 1 Instrucciones

---

1.133 MechUnitLoad - Define una carga útil para una unidad mecánica  
RobotWare - OS

## 1.133 MechUnitLoad - Define una carga útil para una unidad mecánica

---

### Utilización

MechUnitLoad se utiliza para definir una carga útil para una unidad mecánica externa, por ejemplo, posicionadores. La carga útil para un robot se define con la instrucción GripLoad.

Esta instrucción debe usarse con todas las unidades mecánicas que tengan un modelo dinámico (posicionadores ABB y Track Motion), para conseguir el máximo rendimiento de los movimientos.

La instrucción MechUnitLoad debe ejecutarse siempre después de ejecutar la instrucción ActUnit.

Esta instrucción sólo puede usarse en la tarea principal T\_ROB1 o, si se cuenta con un sistema MultiMove, en las tareas de movimiento.

### Descripción

MechUnitLoad especifica qué cargas transporta la unidad mecánica. Las cargas especificadas se utilizan en el sistema de control para poder controlar de la mejor manera posible los movimientos de la unidad mecánica.

La carga útil se conecta y desconecta mediante la instrucción MechUnitLoad, lo que suma o resta el peso de la carga útil al peso de la unidad mecánica.



#### ¡AVISO!

Es importante definir siempre la carga real de la herramienta y, si se usa, la carga útil del robot (por ejemplo, una pieza sujeta por una pinza). Una definición incorrecta de los datos de carga puede dar lugar a la sobrecarga de la estructura mecánica del robot. Existe también el riesgo de que pueda superarse la velocidad en el modo manual a velocidad reducida.

Cuando se especifican datos de carga incorrectos, este hecho suele tener las consecuencias siguientes:

- El robot no puede funcionar a su capacidad máxima.
- Peor exactitud de la trayectoria, con riesgo de sobrepasar posiciones.
- Riesgo de sobrecarga de la estructura mecánica.

El controlador monitoriza continuamente la carga y escribe un registro de eventos si la carga es más elevada que la prevista. Este registro de eventos se guarda y registra en la memoria del controlador.



#### ¡AVISO!

La advertencia anterior también se aplica al definir cargas útiles para unidades mecánicas externas.

Continúa en la página siguiente

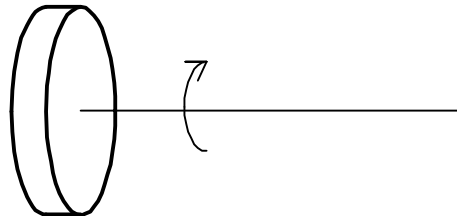


### Ejemplos básicos

Los ejemplos siguientes ilustran la instrucción `MechUnitLoad`.

#### Figura

En la figura que aparece a continuación se muestra el eje 1 de una unidad mecánica con el nombre `STN1` y del tipo `IRBP L`.



xx0500002142

#### Ejemplo 1

```
ActUnit STN1;
MechUnitLoad STN1, 1, load0;
```

Se activa la unidad mecánica `STN1` y se define la carga útil `load0` que corresponde a ninguna carga (en absoluto) montada en el eje 1.

#### Ejemplo 2

```
ActUnit STN1;
MechUnitLoad STN1, 1, fixture1;
```

Se activa la unidad mecánica `STN1` y se define la carga útil `fixture1` que corresponde al accesorio montado en el eje 1.

#### Ejemplo 3

```
ActUnit STN1;
MechUnitLoad STN1, 1, workpiece1;
```

Se activa la unidad mecánica y se define la carga útil `STN1` que corresponde al accesorio y a la pieza de trabajo `workpiece1`, montada en el eje 1.

### Argumentos

```
MechUnitLoad MechUnit AxisNo Load
```

MechUnit

**Mechanical Unit**

Tipo de dato: `mecunit`

El nombre de la unidad mecánica.

AxisNo

**Axis Number**

Tipo de dato: `num`

El número del eje, dentro de la unidad mecánica, que sostiene la carga. La numeración de los ejes comienza en el número 1.

Load

Tipo de dato: `loaddata`

Continúa en la página siguiente

# 1 Instrucciones

## 1.133 MechUnitLoad - Define una carga útil para una unidad mecánica

RobotWare - OS

Continuación

El dato de carga que describe la carga útil actual que se desea definir; es decir, el accesorio o bien el accesorio junto con la pieza de trabajo, en función de si la pieza de trabajo está montada en la unidad mecánica o no.

### Ejecución de programas

La carga especificada se aplica a la siguiente instrucción de movimiento ejecutada y es válida hasta que se ejecute una nueva instrucción MechUnitLoad.

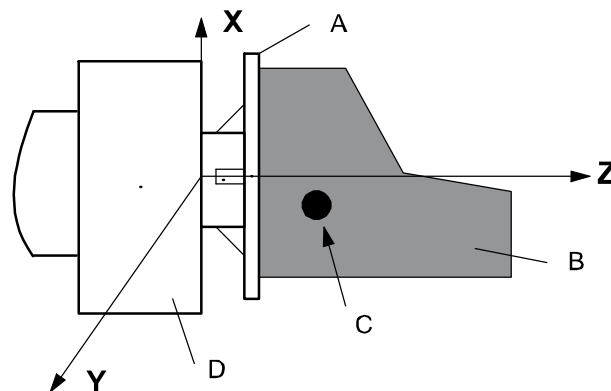
Después de la ejecución de MechUnitLoad, cuando los ejes del robot y los ejes adicionales se han detenido, se define la carga especificada para la unidad mecánica y el eje especificados. Esto significa que la carga útil es controlada y monitorizada por el sistema de control.

La carga útil predeterminada al usar el modo de reinicio **Restablecer sistema** para un tipo de unidad mecánica determinada es la carga útil máxima predefinida para dicho tipo de unidad mecánica.

Cuando se utiliza otra carga útil, es necesario redefinir con esta instrucción la carga útil real de la unidad mecánica y del eje. Esto debe hacerse siempre después de la activación de la unidad mecánica.

La carga útil definida permanece definida después de un reinicio. La carga útil definida también permanece vigente después de reanudar el programa tras la activación manual de otras unidades mecánicas desde la ventana de movimientos.

En el siguiente gráfico se muestra una carga útil montada en el elemento terminal de una unidad mecánica (sistema de coordenadas de elemento terminal de la unidad mecánica).



xx0500002143

|   |  |
|---|--|
| A | Elemento terminal  |
| B | Accesorio y pieza de trabajo                                       |
| C | Centro de gravedad de la carga útil (accesorio + pieza de trabajo) |
| D | Unidad mecánica  |

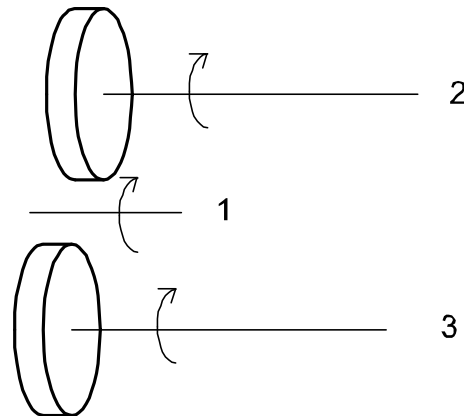
Continúa en la página siguiente

**Más ejemplos**

A continuación aparecen más ejemplos de cómo usar la instrucción `MechUnitLoad`.

**Figura**

En la figura que aparece a continuación se muestra una unidad mecánica con el nombre `INTERCH` y el tipo `IRBP K` con tres ejes (1, 2 y 3).



xx0500002144

**Ejemplo 1**

```
MoveL homeside1, v1000, fine, gun1;
...
ActUnit INTERCH;
```

Se activa la totalidad de la unidad mecánica `INTERCH` .

**Ejemplo 2**

```
MechUnitLoad INTERCH, 2, workpiece1;
```

Define la carga útil `workpiece1` en el eje 2 de la unidad mecánica `INTERCH`.

**Ejemplo 3**

```
MechUnitLoad INTERCH, 3, workpiece2;
```

Define la carga útil `workpiece2` en el eje 3 de la unidad mecánica `INTERCH`.

**Ejemplo 4**

```
MoveL homeside2, v1000, fine, gun1;
```

Los ejes de la unidad mecánica `INTERCH` se mueven a la posición de cambio `homeside2` con la carga útil montada en los dos ejes 2 y 3.

**Ejemplo 5**

```
ActUnit STN1;
MechUnitLoad STN1, 1, workpiece1;
```

Se activa la unidad mecánica `STN1`. Define la carga útil `workpiece1` en el eje 1 de la unidad mecánica `STN1`.

**Limitaciones**

Si esta instrucción va precedida de una instrucción de movimiento, ésta última debe programarse con un punto de paro (`zonedata fine`), no un punto de paso. De lo contrario, no será posible reanudar la ejecución tras una caída de suministro eléctrico.

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.133 MechUnitLoad - Define una carga útil para una unidad mecánica

RobotWare - OS

Continuación

MechUnitLoad no puede ejecutarse en rutinas de RAPID que estén conectadas a los siguientes eventos especiales del sistema: PowerOn, Stop, QStop, Restart o Step.

---

### Sintaxis

```
MechUnitLoad
  [MechUnit ':= ' ] <variable (VAR) of mecunit> ','
  [AxisNo ':= ' ] <expression (IN) of num> ','
  [Load ':= ' ] <persistent (PERS) of loaddata> ';'

```

---

### Información relacionada

| Para obtener más información sobre                                 | Consulte   |
|--|--|
| Identificación de la carga útil de las unidades mecánicas externas | <a href="#">Manual del producto - IRBP /D2009</a>                            |
| Definición de datos de unidades mecánicas                          | <a href="#">mecunit - Unidad mecánica en la página 1703</a>                  |
| Definición de datos de carga                                       | <a href="#">loaddata - Datos de carga en la página 1694</a>                  |
| Define la carga útil del robot.                                    | <a href="#">GripLoad - Define la carga útil de un robot en la página 244</a> |

### 1.134 MotionProcessModeSet - Configuración del modo de proceso de movimientos

#### Utilización

`MotionProcessModeSet` se utiliza para configurar el modo de proceso de movimientos (*Motion Process Mode*) para un robot con TCP.

Esta instrucción sólo puede usarse en la tarea principal `T_ROB1` o, si se cuenta con un sistema *MultiMove*, en las tareas de movimiento.

#### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción

`MotionProcessModeSet`:

```
MotionProcessModeSet OPTIMAL_CYCLE_TIME_MODE;
! Do cycle-time critical movement
..
MotionProcessModeSet ACCURACY_MODE;
! Do cutting with high accuracy
..
```

Cambio del modo del proceso de movimiento utilizado para el robot con TCP en tiempo de ejecución.

#### Argumentos

`MotionProcessModeSet Mode`

Mode

**Tipo de dato:** `motionprocessmode`

El modo de proceso de movimientos que se usará. Se trata de una constante entera del tipo de dato `motionprocessmode`.

#### Ejecución de programas

El modo del proceso de movimiento se aplica al robot con TCP hasta que se ejecuta una nueva instrucción `MotionProcessModeSet`; consulte [Información relacionada en la página 378](#).

El valor predeterminado configurado para el modo de proceso de movimiento se ajusta automáticamente:

- cuando se utiliza el modo de reinicio **Restablecer RAPID**
- al cargar un nuevo programa o un nuevo módulo
- al iniciar la ejecución del programa desde el principio
- al mover el puntero del programa a `main`
- al mover el puntero del programa a una rutina
- al mover el puntero de programa de una forma que se pierde el orden de la ejecución.

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.134 MotionProcessModeSet - Configuración del modo de proceso de movimientos

RobotWare - OS

Continuación

### Datos predefinidos

Se han predefinido las constantes simbólicas siguientes para el tipo de dato `motionprocessmode`. Puede usarlas para especificar el entero en el argumento `Mode`.

El modo predeterminado es definido por el parámetro de sistema *Use Motion Process Mode* del tipo *Robot*, tema *Motion*.

| Constante simbólica     | Valor constante | Descripción  |
|-------------------------|-----------------|--|
| OPTIMAL_CYCLE_TIME_MODE | 1               | Este modo proporciona el tiempo de ciclo más breve posible.  |
| LOW_SPEED_ACCURACY_MODE | 2               | Este modo mejora la exactitud de la trayectoria principalmente para robots grandes.  |
| LOW_SPEED_STIFF_MODE    | 3               | Este modo se recomienda para aplicaciones de contacto de baja velocidad en las que es importante una máxima rigidez del servo. |
| ACCURACY_MODE           | 4               | Este modo mejora la exactitud de la trayectoria principalmente para robots pequeños.   |
| MPM_USER_MODE_1         | 5               | Modos definidos por el usuario.  |
| MPM_USER_MODE_2         | 6               |  |
| MPM_USER_MODE_3         | 7               |  |
| MPM_USER_MODE_4         | 8               |  |
| PRESS_TENDING_MODE      | 9               | Dirigido principalmente a muñequeras flexibles en aplicaciones de tending de prensa.   |

### Limitaciones

Este modo sólo puede modificarse con el robot en reposo; de lo contrario, se impone un punto fino.

### Sintaxis

```
MotionProcessModeSet  
  [Mode ':='] <expression (IN) of motionprocessmode>;'
```

### Información relacionada

| Para obtener más información sobre                          | Consulte  |
|---|---|
| <i>Advanced robot motion</i>                                | <i>Application manual - Controller software IRC5</i>          |
| Configuración de parámetros de <i>Motion Process Mode</i> . | <i>Manual de referencia técnica - Parámetros del sistema</i>  |
| Ajuste de servos.   | <a href="#">TuneServo - Ajuste de servos en la página 994</a> |

**1.135 MotionSup - Desactiva/activa la supervisión del movimiento****Utilización**

MotionSup (*Motion Supervision*) se utiliza para desactivar o activar la función de supervisión del movimiento para los movimientos del robot durante la ejecución del programa.

Esta instrucción sólo puede usarse en la tarea principal T\_ROB1 o, si se cuenta con un sistema *MultiMove*, en las tareas de movimiento.

**Descripción**

Supervisión de movimiento es el nombre que recibe un grupo de funciones que permiten supervisar el robot con una alta sensibilidad y en base a modelos. Contiene la función de supervisión de cargas del eje, supervisión de colisiones del eje y detección de colisiones. Dado que la supervisión se ha diseñado de forma que sea muy sensible, puede dispararse si sobre el robot actúan grandes fuerzas en el proceso.

Si la carga no está definida correctamente, utilice la rutina del servicio de identificación de carga para especificarla. Si existen grandes fuerzas externas al proceso en la mayor parte de la aplicación, como por ejemplo durante la eliminación de rebabas, utilice los parámetros del sistema para aumentar el nivel de la supervisión de movimientos hasta que deje de dispararse. Si las fuerzas externas son sólo temporales, como por ejemplo cuando se cierra una pistola de soldadura por puntos de gran tamaño, se debe utilizar la instrucción MotionSup para aumentar el nivel de supervisión (o desactivar la función) en las partes de la aplicación sobre las que actúa esta alteración.

**Ejemplos básicos**

El ejemplo que aparece a continuación ilustra la instrucción MotionSup:

**Ejemplo 1**

```
! If the motion supervision is active in the system parameters,
! then it is active by default during program execution
...
! If motion supervision is deactivated in the system parameters
! then it cannot be activated using 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;
...
! Activate motion supervision again.
```

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.135 MotionSup - Desactiva/activa la supervisión del movimiento

### *Collision Detection*

#### *Continuación*

```
! No back off at a motion collision  
MotionSup \On \NoBackoff;
```

---

#### Argumentos

```
MotionSup[\On] | [\Off] [\TuneValue] [\NoBackoff]
```

[\On]

**Tipo de dato:** switch

Activación de la función de supervisión del movimiento durante la ejecución del programa (si ya ha sido activada en los parámetros del sistema).

[\Off]

**Tipo de dato:** switch

Desactivación de la función de supervisión del movimiento durante la ejecución del programa.

Es imprescindible especificar uno de los argumentos \On o \Off.

[\TuneValue]

**Tipo de dato:** num

Ajuste del nivel de sensibilidad de la supervisión de movimientos en porcentaje (del 1% al 300%) del nivel de los parámetros del sistema. Un nivel superior proporciona una sensibilidad más robusta. Este argumento sólo puede ser combinado con el argumento \On.

[\NoBackoff]

**Tipo de dato:** switch

Si se utiliza este interruptor, el robot no retrocede después de una colisión de movimiento. Este argumento solo puede combinarse con el argumento \On.

---

#### Ejecución de programas

La supervisión de movimiento especificada se aplica a la siguiente instrucción de movimiento ejecutada y es válida hasta que se ejecute una nueva instrucción MotionSup.

Si la función de supervisión del movimiento está activada tanto en los parámetros del sistema como en el programa RAPID y se dispara la supervisión del movimiento debido a una colisión, entonces

- El robot se detendrá en el menor tiempo posible.
- el robot retrocederá para eliminar cualquier fuerza residual (si no se ha utilizado el interruptor \NoBackoff en la última instrucción MotionSup)
- La ejecución del programa se detiene y se genera un mensaje de error.

Si la supervisión del movimiento está activada en los parámetros del sistema, está también activada de forma predeterminada durante la ejecución del programa (TuneValue:=100, y retrocederá para eliminar cualquier fuerza residual). Estos valores se establecen automáticamente

- cuando se utiliza el modo de reinicio **Restablecer RAPID**
- al cargar un nuevo programa o un nuevo módulo
- al iniciar la ejecución del programa desde el principio

*Continúa en la página siguiente*



## 1.135 MotionSup - Desactiva/activa la supervisión del movimiento

*Collision Detection*

*Continuación*

- al mover el puntero del programa a `main`
- al mover el puntero del programa a una rutina
- al mover el puntero de programa de una forma que se pierde el orden de la ejecución.

### Limitaciones

- La supervisión del movimiento no está nunca activa en el caso de los ejes externos ni al mover uno o varios ejes en el modo de ejes independientes. Al utilizar el robot en el modo de servo suave, es posible que sea necesario desactivar la supervisión del movimiento para evitar que salte accidentalmente.
- Si la supervisión de movimiento está desactivada en los parámetros del sistema, no se podrá activar con la instrucción MotionSup.

### Sintaxis

```
MotionSup
[ '\ On ] | [ '\ Off ]
[ '\ Tunevalue':='< expression (IN) of num> ]
[ '\ NoBackoff ] ';'

```

### Información relacionada

| Para obtener más información sobre                          | Consulte   |
|---|--|
| Datos de parámetros de movimiento                           | <a href="#">motsetdata - Datos de parámetros de movimiento en la página 1705</a> |
| Descripción general de la función                           | <i>Manual de referencia técnica - RAPID Overview</i>                             |
| Ajuste con los parámetros del sistema                       | <i>Manual de referencia técnica - Parámetros del sistema</i>                     |
| LoadIdentify, rutina de servicio de identificación de carga | <i>Manual del operador - IRC5 con FlexPendant</i>                                |

# 1 Instrucciones

---

1.136 MoveAbsJ - Mueve el robot a una posición de ejes absoluta  
RobotWare - OS

## 1.136 MoveAbsJ - Mueve el robot a una posición de ejes absoluta

---

### Utilización

MoveAbsJ (*Move Absolute Joint*) se utiliza para mover el robot hacia una posición absoluta definida en posiciones de ejes.

Ejemplos de uso:

- El punto final es un punto singular.
- Para posiciones ambiguas en el caso del IRB 6400C, por ejemplo para movimientos con la herramienta situada sobre el robot.

La posición final del robot durante un movimiento con MoveAbsJ no se ve afectada por la herramienta, por el objeto de trabajo ni por el desplazamiento de programa. El robot utiliza estos datos para calcular la carga, la velocidad del TCP y la trayectoria de esquina. Es posible usar las mismas herramientas en instrucciones de movimiento adyacentes.

Los ejes del robot y los ejes externos se desplazan hasta la posición de destino a lo largo de una trayectoria no lineal. Todos los ejes alcanzan la posición de destino al mismo tiempo.

Esta instrucción sólo puede usarse en la tarea principal T\_ROB1 o, si se cuenta con un sistema *MultiMove*, en las tareas de movimiento.

---

### Ejemplos básicos

Los ejemplos siguientes ilustran la instrucción MoveAbsJ.

Consulte también [Más ejemplos en la página 386](#).

#### Ejemplo 1

```
MoveAbsJ p50, v1000, z50, tool2;
```

El robot que tiene montada la herramienta tool2 se mueve a lo largo de una trayectoria no lineal hacia la posición absoluta de ejes, p50, con los datos de velocidad v1000 y los datos de zona z50.

#### Ejemplo 2

```
MoveAbsJ *, v1000\T:=5, fine, grip3;
```

El robot que tiene montada la herramienta grip3 se mueve siguiendo una trayectoria lineal hasta un punto de paro que se almacena como una posición absoluta de ejes en la instrucción (se marca con \*). Todo el movimiento requiere 5 segundos.

---

### Argumentos

```
MoveAbsJ [\Conc] ToJointPos [\ID] [\NoEOffs] Speed [\V] | [\T] Zone  
[\Z] [\Inpos] Tool [\WObj] [\TLoad]
```

[**\Conc**]

**Concurrent**

Tipo de dato:switch

Distintas instrucciones consecutivas se ejecutan mientras el robot está en movimiento. El argumento no se utiliza normalmente para acortar el tiempo de

---

Continúa en la página siguiente

## 1.136 MoveAbsJ - Mueve el robot a una posición de ejes absoluta

*RobotWare - OS*

*Continuación*

ciclo si, por ejemplo se realiza una comunicación con equipos externos y no se requiere sincronización.

Cuando se utiliza el argumento `\Conc`, el número de instrucciones de movimiento seguidas está limitado a 5. En secciones de programa que incluyen `StorePath-RestoPath`, no se permite el uso de instrucciones con el argumento `\Conc`.

Si se omite este argumento y `ToJointPos` no es un punto de paro, la instrucción siguiente se ejecuta algún tiempo antes de que el robot alcance la zona programada.

Este argumento no puede usarse en los movimientos sincronizados coordinados en un sistema `MultiMove`.

`ToJointPos`

### *To Joint Position*

Tipo de dato: `jointtarget`

La posición absoluta de destino de los ejes del robot y de los ejes externos Se define como una posición con nombre o se almacena directamente en la instrucción (marcada con un asterisco \* en la instrucción).

[ `\ID` ]

### *Synchronization id*

Tipo de dato: `identno`

El argumento [ `\ID` ] es obligatorio en los sistemas `MultiMove`, si el movimiento es sincronizado o sincronizado coordinado. Este argumento no está permitido en ningún otro caso. El número de ID especificado debe ser el mismo en todas las tareas de programa que cooperan entre sí. Al usar el número de ID los movimientos no se mezclan en tiempo de ejecución.

[ `\NoEOffs` ]

### *No External Offsets*

Tipo de dato: `switch`

Si se utiliza el argumento `\NoEOffs`, el movimiento con `MoveAbsJ` no se ve afectado por los offsets activos para los ejes externos.

`Speed`

Tipo de dato: `speeddata`

Los datos de velocidad que se aplican a los movimientos. Los datos de velocidad definen la velocidad del punto central de la herramienta, la reorientación de la misma y los ejes externos.

[ `\V` ]

### *Velocity*

Tipo de dato: `num`

Este argumento se utiliza para especificar la velocidad del TCP en mm/s directamente en la instrucción. A continuación, se sustituye por la velocidad correspondiente, especificada en los datos de velocidad.

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.136 MoveAbsJ - Mueve el robot a una posición de ejes absoluta

RobotWare - OS

Continuación

[ \T ]

### *Time*

Tipo de dato: num

Este argumento se utiliza para especificar el tiempo total en segundos que dura el movimiento del robot. A continuación, se sustituye por los datos de velocidad correspondientes.

Zone

Tipo de dato: zonedata

Los datos de zona del movimiento. Los datos de zona describen el tamaño de la trayectoria de esquina generada.

[ \Z ]

Zone

Tipo de dato: num

Este argumento se utiliza para especificar la exactitud de la posición del TCP del robot, directamente en la instrucción. La longitud de la trayectoria de esquina se indica en mm y es sustituida por la zona correspondiente especificada en los datos de zona.

[ \Inpos ]

### *In position*

Tipo de dato: stoppointdata

Este argumento se utiliza para especificar los criterios de convergencia para la posición del TCP del robot en el punto de paro. Los datos de puntos de paro sustituyen a la zona especificada en el parámetro *Zone*.

Tool

Tipo de dato: tooldata

La herramienta en uso durante el movimiento.

La posición del TCP y la carga de la herramienta se definen en los datos de la herramienta. La posición del TCP se utiliza para calcular la velocidad y la trayectoria de esquina del movimiento.

[ \WObj ]

### *Work Object*

Tipo de dato: wobjdata

El objeto de trabajo en uso durante el movimiento.

Este argumento puede ser omitido si la herramienta es sostenida por el robot. Si el robot sostiene el objeto de trabajo, es decir, si la herramienta es estacionaria, o si se utilizan ejes externos coordinados, es necesario especificar este argumento.

En el caso de las herramientas estacionarias o los ejes externos coordinados, los datos utilizados por el sistema para calcular la velocidad y la trayectoria de esquinas del movimiento se definen en el objeto de trabajo.

[ \TLoad ]

### *Total load*

Continúa en la página siguiente

Tipo de dato: `loaddata`

El argumento `\TLoad` describe la carga total usada durante el movimiento. La carga total es la carga de la herramienta más la carga útil transportada por la herramienta. Si se utiliza el argumento `\TLoad`, no se tiene en cuenta el valor de `loaddata` en los `tooldata` actuales.

Si el argumento `\TLoad` tiene el valor `load0`, el argumento `\TLoad` no se tiene en cuenta y se utilizan en su lugar los `loaddata` de los `tooldata`.

Para poder utilizar el argumento `\TLoad`, es necesario cambiar el valor del parámetro de sistema `ModalPayLoadMode` a 0. Si `ModalPayLoadMode` tiene el valor 0, ya no es posible utilizar la instrucción `GripLoad`.

La carga total puede identificarse con la rutina de servicio `LoadIdentify`. Si el parámetro de sistema `ModalPayLoadMode` tiene el valor 0, el operador tiene la posibilidad de copiar los `loaddata` de la herramienta a una variable persistente `loaddata` existente o nueva al ejecutar la rutina de servicio.

Es posible realizar una ejecución de prueba del programa sin ninguna carga útil utilizando una señal de entrada digital conectada a la entrada de sistema `SimMode` (modo simulado). Si la señal de entrada digital tiene el valor 1, los `loaddata` del argumento opcional `\TLoad` no se tienen en cuenta y se utilizan en su lugar los `loaddata` de los `tooldata` actuales.



### Nota

La funcionalidad predeterminada de manejo de la carga útil es utilizar la instrucción `GripLoad`. Por tanto, el valor predeterminado del parámetro de sistema `ModalPayLoadMode` es 1.

## Ejecución de programas

Los movimientos realizados con `MoveAbsJ` no se ven afectados por el desplazamiento activo del programa. Además, si se ejecutan con el modificador `\NoEOffs`, no se utilizará ningún `offset` para los ejes externos. Si no se utiliza el modificador `\NoEOffs`, los ejes externos del destino del movimiento se ven afectados por el `offset` activo en los ejes externos.

La herramienta se traslada a la posición absoluta de destino de los ejes, con interpolación de los ángulos de los ejes. Esto significa que cada eje se mueve a una velocidad constante y que todos los ejes alcanzan al mismo tiempo el punto de destino de ejes, lo que da lugar a una trayectoria no lineal.

En términos generales, el TCP se mueve a una velocidad aproximada a la programada. La herramienta se reorienta y los ejes externos se mueven al mismo tiempo que se mueve el TCP. Si no es posible alcanzar la velocidad programada para la orientación o para los ejes externos, se reduce la velocidad del TCP.

Normalmente se generan trayectorias de esquina cuando se transfiere el movimiento a la siguiente sección de la trayectoria. Si se especifica un punto de paro en los datos de zona, la ejecución del programa sólo continúa una vez que los ejes del robot y los ejes externos han alcanzado la posición de ejes adecuada.

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.136 MoveAbsJ - Mueve el robot a una posición de ejes absoluta

RobotWare - OS

Continuación

---

### Más ejemplos

A continuación aparecen más ejemplos de cómo usar la instrucción `MoveAbsJ`.

#### Ejemplo 1

```
MoveAbsJ *, v2000\V:=2200, z40 \Z:=45, grip3;
```

La herramienta, `grip3`, se mueve siguiendo una trayectoria no lineal hasta una posición absoluta de ejes almacenada en la instrucción. El movimiento se realiza con los datos establecidos en `v2000` y `z40`. La velocidad y el tamaño de la zona del TCP son de 2200 mm/s y 45 mm respectivamente.

#### Ejemplo 2

```
MoveAbsJ p5, v2000, fine \Inpos := inpos50, grip3;
```

La herramienta, `grip3`, se mueve siguiendo una trayectoria no lineal hasta la posición absoluta de ejes `p5`. El robot considera que se encuentra en el punto cuando se satisface el 50% de la condición de posición y el 50% de la condición de velocidad de un punto de paro `fine`. Espera al menos 2 segundos a que se satisfagan las condiciones. Consulte los datos predefinidos `inpos50` del tipo de dato `stoppointdata`.

#### Ejemplo 3

```
MoveAbsJ \Conc, *, v2000, z40, grip3;
```

La herramienta, `grip3`, se mueve siguiendo una trayectoria no lineal hasta una posición absoluta de ejes almacenada en la instrucción. Las instrucciones lógicas posteriores se ejecutan mientras el robot está en movimiento.

#### Ejemplo 4

```
MoveAbsJ \Conc, * \NoEOffs, v2000, z40, grip3;
```

El mismo movimiento que el anterior, pero con la diferencia de que no se ve afectado por los offsets activos para los ejes externos.

#### Ejemplo 5

```
GripLoad obj_mass;  
MoveAbsJ start, v2000, z40, grip3 \Wobj:= obj;
```

El robot mueve el objeto de trabajo `obj` respecto de la herramienta fija `grip3`, siguiendo una trayectoria no lineal hasta una posición absoluta de ejes, `start`.

---

### Gestión de errores

Pueden generarse los errores recuperables enumerados a continuación. Estos errores pueden ser gestionados en un gestor de errores. El valor de la variable de sistema `ERRNO` cambia a:

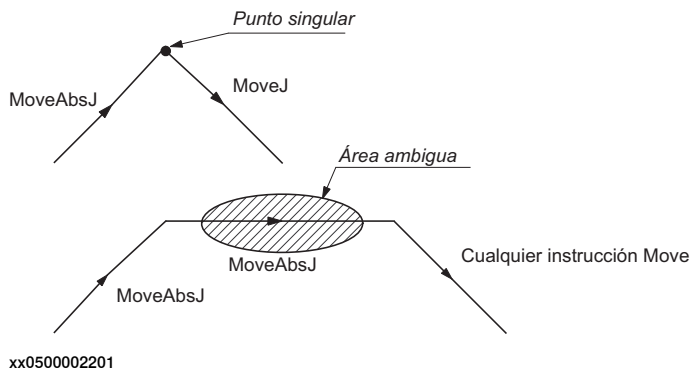
| Nombre                    | Causa del error  |
|---------------------------|--|
| <code>ERR_CONC_MAX</code> | Se ha superado el número de instrucciones de movimiento seguidas con el argumento <code>\Conc</code> . |

Continúa en la página siguiente

**Limitaciones**

Para poder ejecutar hacia atrás la instrucción `MoveAbsJ` y evitar problemas con puntos singulares o áreas ambiguas, resulta esencial que las instrucciones posteriores cumplan determinados requisitos, como se indica a continuación (consulte la figura siguiente).

La figura muestra la limitación para la ejecución hacia atrás con `MoveAbsJ`.



`MoveAbsJ` no puede ejecutarse en un gestor UNDO o en rutinas de RAPID que estén conectadas a los siguientes eventos especiales del sistema: PowerOn, Stop, QStop, Restart, Reset o Step.

**Sintaxis**

```

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>'];'
    
```

**Información relacionada**

| Para obtener más información sobre     | Consulte   |
|--|--|
| Otras instrucciones de posicionamiento | <a href="#">Manual de referencia técnica - RAPID Overview</a>            |
| Definición de jointtarget              | <a href="#">jointtarget - Datos de posición de eje en la página 1691</a> |
| Definición de carga                    | <a href="#">loaddata - Datos de carga en la página 1694</a>              |
| Definición de velocidad                | <a href="#">speeddata - Datos de velocidad en la página 1767</a>         |
| Definición de datos de punto de paro   | <a href="#">stoppointdata - Datos de punto de paro en la página 1771</a> |
| Definición de herramientas             | <a href="#">tooldata - Datos de herramienta en la página 1795</a>        |

Continúa en la página siguiente

# 1 Instrucciones

## 1.136 MoveAbsJ - Mueve el robot a una posición de ejes absoluta

RobotWare - OS

Continuación

| Para obtener más información sobre  | Consulte   |
|---|--|
| Definición de objetos de trabajo  | <a href="#">wobjdata - Datos del objeto de trabajo en la página 1823</a>                 |
| Definición de datos de zona   | <a href="#">zonedata - Datos de zonas en la página 1831</a>                              |
| Movimiento en general   | <i>Manual de referencia técnica - RAPID Overview</i>                                     |
| Ejecución simultánea de programas   | <i>Manual de referencia técnica - RAPID Overview</i>                                     |
| Ejemplo de cómo usar TLoad, carga total.  | <a href="#">MoveL - Mueve el robot siguiendo una trayectoria lineal en la página 447</a> |
| Definición de la carga útil de un robot   | <a href="#">GripLoad - Define la carga útil de un robot en la página 244</a>             |
| LoadIdentify, rutina de servicio de identificación de carga                                   | <i>Manual del operador - IRC5 con FlexPendant</i>  |
| Señal de entrada de sistema SimMode para mover el robot en el modo simulado sin carga útil.   | <i>Manual de referencia técnica - Parámetros del sistema</i>                             |
| Parámetro de sistema ModalPayloadMode para la activación y la desactivación de la carga útil. | <i>Manual de referencia técnica - Parámetros del sistema</i>                             |



## 1.137 MoveC - Mueve el robot en círculo

### Utilización

MoveC se utiliza para trasladar el punto central de la herramienta (TCP) en sentido circular hacia un punto de destino determinado. Durante el movimiento, la orientación suele permanecer sin cambios respecto del círculo.

Esta instrucción sólo puede usarse en la tarea principal T\_ROB1 o, si se cuenta con un sistema *MultiMove*, en las tareas de movimiento.

### Ejemplos básicos

Los ejemplos siguientes ilustran la instrucción MoveC.

Consulte también [Más ejemplos en la página 393](#).

#### Ejemplo 1

```
MoveC p1, p2, v500, z30, tool2;
```

El TCP de la herramienta, *tool2*, se mueve en círculo hacia la posición *p2* con los datos de velocidad *v500* y los datos de zona *z30*. El círculo se define a partir de la posición inicial, el punto de círculo *p1* y el punto de destino *p2*.

#### Ejemplo 2

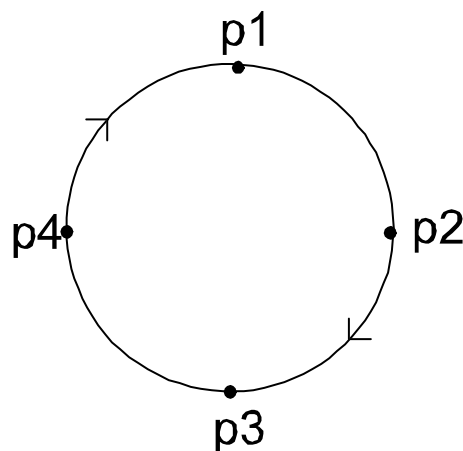
```
MoveC *, *, v500 \T:=5, fine, grip3;
```

El TCP de la herramienta, *grip3*, se mueve en círculo hasta un punto fino almacenado en la instrucción (marcado por el segundo asterisco \*). El punto del círculo también está almacenado en la instrucción (marcado por el primer asterisco \*). Todo el movimiento requiere 5 segundos.

#### Ejemplo 3

```
MoveL p1, v500, fine, tool1;
MoveC p2, p3, v500, z20, tool1;
MoveC p4, p1, v500, fine, tool1;
```

La figura muestra cómo se realiza un círculo completo mediante dos instrucciones MoveC .



xx0500002212

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.137 MoveC - Mueve el robot en círculo

RobotWare - OS

Continuación

---

### Argumentos

```
MoveC [\Conc] CirPoint ToPoint [\ID] Speed [\V] | [\T] Zone [\Z]  
[\Inpos] Tool [\WObj] [\Corr] [\TLoad]
```

[ \Conc ]

#### *Concurrent*

Tipo de dato:switch

Distintas instrucciones consecutivas se ejecutan mientras el robot está en movimiento. Este argumento no se utiliza normalmente, pero puede utilizarse para evitar los paros no deseados, causados por la sobrecarga de la CPU al utilizar puntos de paso. Esto resulta útil cuando los puntos programados están muy cercanos entre sí y se trabaja a velocidades elevadas. Este argumento también resulta útil si, por ejemplo, no se requiere la comunicación con equipos externos ni la sincronización entre los equipos externos y los movimientos del robot.

Cuando se utiliza el argumento \Conc, el número de instrucciones de movimiento seguidas está limitado a 5. En secciones de programa que incluyen StorePath-RestoPath, no se permite el uso de instrucciones con el argumento \Conc.

Si se omite este argumento y ToPoint no es un punto de paro, la instrucción siguiente se ejecuta algún tiempo antes de que el robot alcance la zona programada.

Este argumento no puede usarse en los movimientos sincronizados coordinados en un sistema MultiMove.

CirPoint

Tipo de dato:robtarget

El punto de círculo del robot. El punto de círculo es una posición del círculo entre el punto de inicio y el punto de destino. Para conseguir la máxima exactitud, debe estar situado a mitad de camino entre los puntos inicial y de destino. Si lo sitúa demasiado cerca del punto de inicio o del punto de destino, es posible que el robot genere una advertencia. El punto de círculo se define como una posición con nombre o se almacena directamente en la instrucción (marcada con un asterisco \* en la instrucción). No se utiliza la posición de los ejes externos.

ToPoint

Tipo de dato:robtarget

El punto de destino de los ejes del robot y de los ejes externos. Se define como una posición con nombre o se almacena directamente en la instrucción (marcada con un asterisco \* en la instrucción).

[ \ID ]

#### *Synchronization id*

Tipo de dato:identno

El argumento [ \ID ] es obligatorio en los sistemas MultiMove, si el movimiento es sincronizado o sincronizado coordinado. Este argumento no está permitido en ningún otro caso. El número de ID especificado debe ser el mismo en todas las

*Continúa en la página siguiente*

tareas de programa que cooperan entre sí. Al usar el número de ID los movimientos no se mezclan en tiempo de ejecución.

Speed

Tipo de dato: speeddata

Los datos de velocidad que se aplican a los movimientos. Los datos de velocidad definen la velocidad del TCP, la reorientación de la herramienta y los ejes externos.

[ \V ]

*Velocity*

Tipo de dato: num

Este argumento se utiliza para especificar la velocidad del TCP en mm/s directamente en la instrucción. A continuación, se sustituye por la velocidad correspondiente, especificada en los datos de velocidad.

[ \T ]

*Time*

Tipo de dato: num

Este argumento se utiliza para especificar el tiempo total en segundos que dura el movimiento del robot y de los ejes externos. A continuación, se sustituye por los datos de velocidad correspondientes.

Zone

Tipo de dato: zonedata

Los datos de zona del movimiento. Los datos de zona describen el tamaño de la trayectoria de esquina generada.

[ \Z ]

*Zone*

Tipo de dato: num

Este argumento se utiliza para especificar la exactitud de la posición del TCP del robot, directamente en la instrucción. La longitud de la trayectoria de esquina se indica en mm y es sustituida por la zona correspondiente especificada en los datos de zona.

[ \Inpos ]

*In position*

Tipo de dato: stoppoint data

Este argumento se utiliza para especificar los criterios de convergencia para la posición del TCP del robot en el punto de paro. Los datos de puntos de paro sustituyen a la zona especificada en el parámetro *Zone*.

Tool

Tipo de dato: tooldata

La herramienta en uso durante el movimiento del robot. El punto central de la herramienta es el punto que se mueve hacia el punto de destino especificado.

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.137 MoveC - Mueve el robot en círculo

RobotWare - OS

Continuación

[ \WObj ]

### *Work Object*

Tipo de dato: wobjdata

El objeto de trabajo (sistema de coordenadas de objeto) con el que está relacionada la posición de robot indicada en la instrucción.

Es posible omitir este argumento. Si se omite, la posición depende del sistema de coordenadas mundo. Si por otro lado se usa un TCP estacionario o ejes externos coordinados, es necesario especificar este argumento para que se ejecute un círculo respecto del objeto de trabajo.

[ \Corr ]

### *Correction*

Tipo de dato: switch

Los datos de corrección escritos en una entrada de corrección mediante una instrucción `CorrWrite` se añaden a la trayectoria y a la posición de destino si se utiliza este argumento.

Se requiere RobotWare, opción *Path Offset*, cuando se utiliza este argumento.

[ \TLoad ]

### *Total load*

Tipo de dato: loaddata

El argumento `\TLoad` describe la carga total usada durante el movimiento. La carga total es la carga de la herramienta más la carga útil transportada por la herramienta. Si se utiliza el argumento `\TLoad`, no se tiene en cuenta el valor de `loaddata` en los `tooldata` actuales.

Si el argumento `\TLoad` tiene el valor `load0`, el argumento `\TLoad` no se tiene en cuenta y se utilizan en su lugar los `loaddata` de los `tooldata`.

Para poder utilizar el argumento `\TLoad`, es necesario cambiar el valor del parámetro de sistema `ModalPayLoadMode` a 0. Si `ModalPayLoadMode` tiene el valor 0, ya no es posible utilizar la instrucción `GripLoad`.

La carga total puede identificarse con la rutina de servicio `LoadIdentify`. Si el parámetro de sistema `ModalPayLoadMode` tiene el valor 0, el operador tiene la posibilidad de copiar los `loaddata` de la herramienta a una variable persistente `loaddata` existente o nueva al ejecutar la rutina de servicio.

Es posible realizar una ejecución de prueba del programa sin ninguna carga útil utilizando una señal de entrada digital conectada a la entrada de sistema `SimMode` (modo simulado). Si la señal de entrada digital tiene el valor 1, los `loaddata` del argumento opcional `\TLoad` no se tienen en cuenta y se utilizan en su lugar los `loaddata` de los `tooldata` actuales.



### **Nota**

La funcionalidad predeterminada de manejo de la carga útil es utilizar la instrucción `GripLoad`. Por tanto, el valor predeterminado del parámetro de sistema `ModalPayLoadMode` es 1.

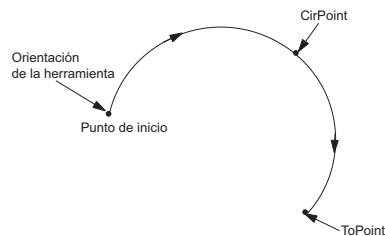
Continúa en la página siguiente

### Ejecución de programas

Las unidades del robot y las externas se trasladan hacia el punto de destino de la forma siguiente:

- El TCP de la herramienta se mueve en círculo a una velocidad constante programada.
- La herramienta se reorienta a una velocidad constante, desde la orientación de la posición inicial hasta la orientación del punto de destino.
- La reorientación se realiza respecto de la trayectoria circular. Por tanto, si la orientación respecto de la trayectoria es la misma en los puntos inicial y final, la orientación relativa permanece sin cambios durante el movimiento (consulte la figura siguiente).

La figura muestra la orientación de la herramienta durante un movimiento circular.



xx0500002214

La orientación del punto circular no se alcanza. Sólo se usa para distinguir entre dos sentidos de orientación posibles. La exactitud de la reorientación a lo largo de la trayectoria sólo depende de la orientación en los puntos inicial y de destino.

Los distintos modos de orientación de la herramienta durante una trayectoria circular se describen en la instrucción `CirPathMode`.

Los ejes externos no coordinados se ejecutan a una velocidad constante para que alcancen el punto de destino al mismo tiempo que los ejes del robot. No se utiliza la posición de la posición de círculo.

Si no es posible alcanzar la velocidad programada para la reorientación o para los ejes externos, se reduce la velocidad del TCP.

Normalmente se general trayectorias de esquina cuando se transfiere el movimiento a la siguiente sección de una trayectoria. Si se especifica un punto de paro en los datos de zona, la ejecución del programa sólo continúa una vez que los ejes del robot y los ejes externos han alcanzado la posición adecuada.

Si el punto de inicio, punto de círculo y el punto de destino son colineales, entonces la instrucción `MoveC` dará lugar a un movimiento lineal.

### Más ejemplos

A continuación aparecen más ejemplos de cómo usar la instrucción `MoveC`.

#### Ejemplo 1

```
MoveC *, *, v500 \V:=550, z40 \Z:=45, grip3;
```

El TCP de la herramienta, `grip3`, se mueve en círculo hacia una posición almacenada en la instrucción. El movimiento se realiza con los datos establecidos

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.137 MoveC - Mueve el robot en círculo

RobotWare - OS

Continuación

en  $v500$  y  $z40$ . La velocidad y el tamaño de zona del TCP son  $550$  mm/s y  $45$  mm, respectivamente.

### Ejemplo 2

```
MoveC p5, p6, v2000, fine \Inpos := inpos50, grip3;
```

El TCP de la herramienta, `grip3`, se mueve en círculo hacia un punto de paro `p6`. El robot considera que se encuentra en el punto cuando se satisface el 50% de la condición de posición y el 50% de la condición de velocidad de un punto de paro `fine`. Espera al menos 2 segundos a que se satisfagan las condiciones. Consulte los datos predefinidos `inpos50` del tipo de dato `stoppointdata..`

### Ejemplo 3

```
MoveC \Conc, *, *, v500, z40, grip3;
```

El TCP de la herramienta, `grip3`, se mueve en círculo hacia una posición almacenada en la instrucción. El punto del círculo también está almacenado en la instrucción. Las instrucciones lógicas posteriores se ejecutan mientras el robot está en movimiento.

### Ejemplo 4

```
MoveC cir1, p15, v500, z40, grip3 \WObj:=fixture;
```

El TCP de la herramienta, `grip3`, se mueve en círculo hacia una posición, `p15`, a través del punto de círculo `cir1`. Estas posiciones se especifican en el sistema de coordenadas de objeto de `fixture`.

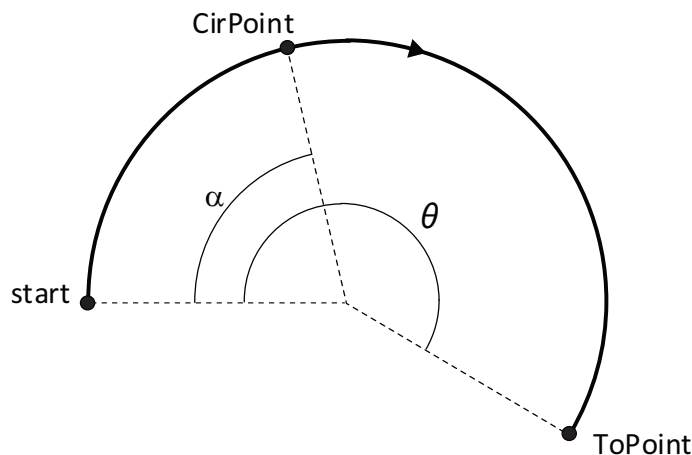
## Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                    | Causa del error  |
|---------------------------|--|
| <code>ERR_CONC_MAX</code> | Se ha superado el número de instrucciones de movimiento seguidas con el argumento <code>\Conc</code> . |

## Limitaciones

Existen algunas limitaciones en cuanto a las posibilidades de posicionamiento de `CirPoint` y `ToPoint`.



xx1700001575

Continúa en la página siguiente

- La distancia mínima entre el punto de inicio y `ToPoint` es de 0,1 mm.
- La distancia mínima entre el punto de inicio y `CirPoint` es de 0,1 mm.
- La distancia mínima entre `CirPoint` y `ToPoint` es 0,1 mm
- Si el parámetro de sistema *Restrict placing of circle points* se establece en **Yes**, entonces se activan las siguientes limitaciones adicionales:
  - El ángulo de la trayectoria circular ( $\theta$  en la imagen anterior) no puede ser mayor de 240°.
  - El punto circular debe estar en la parte intermedia de la trayectoria circular ( $\alpha$  debe ser 25-75% de  $\theta$ , de acuerdo con la imagen anterior).

La exactitud puede ser baja cerca de los límites, es decir, si el punto de inicio y el punto `ToPoint` del círculo están muy cerca el uno del otro, el error causado por la inclinación del círculo puede ser mucho mayor que la exactitud con la que se programaron los puntos.

Asegúrese de que el robot pueda alcanzar el punto de círculo durante la ejecución del programa y dividir la orden de movimientos de círculo programada si es necesario.

Los cambios del modo de ejecución de la ejecución hacia adelante a la ejecución hacia atrás, o viceversa, mientras el robot se detiene en una trayectoria circular no se permiten y generan un mensaje de error.



### ¡AVISO!

La instrucción `MoveC` (o cualquier otra instrucción que incluya un movimiento circular) no debe empezarse en ningún caso desde el principio, con un TCP entre el punto de círculo y el punto final. De lo contrario, el robot no toma la trayectoria programada (posicionamiento alrededor de la trayectoria circular en otra dirección, en comparación con la programada).

Para minimizar el riesgo, establezca el parámetro de sistema *Restrict placing of circlepoints* en **Yes** (tipo *Motion Planner*, tema *Motion*).

`MoveC` no puede ejecutarse en un gestor UNDO o en rutinas de RAPID que estén conectadas a los siguientes eventos especiales del sistema: **PowerOn**, **Stop**, **QStop**, **Restart**, **Reset** o **Step**.

## Sintaxis

```

MoveC
  ['\ ' Conc ' ',']
  [CirPoint ':=' ] <expression (IN) of robtarg>','
  [ToPoint ':=' ] <expression (IN) of robtarg>','
  ['\ ' 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>
    
```

Continúa en la página siguiente

# 1 Instrucciones

## 1.137 MoveC - Mueve el robot en círculo

RobotWare - OS

Continuación

```
[ '\ ' WObj ' := ' <persistent ( PERS ) of wobjdata > ]  
[ '\ ' Corr ]  
[ '\ ' TLoad ' := ' <persistent ( PERS ) of loaddata > ] ; '
```

### Información relacionada

| Para obtener más información sobre  | Consulte   |
|---|--|
| Otras instrucciones de posicionamiento  | <i>Manual de referencia técnica - RAPID Overview</i>   |
| Definición de carga   | <a href="#">loaddata - Datos de carga en la página 1694</a>  |
| Definición de velocidad   | <a href="#">speeddata - Datos de velocidad en la página 1767</a>   |
| Definición de datos de punto de paro  | <a href="#">stoppointdata - Datos de punto de paro en la página 1771</a>                                       |
| Definición de herramientas  | <a href="#">tooldata - Datos de herramienta en la página 1795</a>  |
| Definición de objetos de trabajo  | <a href="#">wobjdata - Datos del objeto de trabajo en la página 1823</a>                                       |
| Definición de datos de zona   | <a href="#">zonedata - Datos de zonas en la página 1831</a>  |
| Escritura en una entrada de corrección  | <a href="#">CorrWrite - Escribe en un generador de correcciones en la página 183</a>                           |
| Reorientación de la herramienta durante trayectorias circulares                               | <a href="#">CirPathMode - Reorientación de la herramienta durante trayectorias circulares en la página 133</a> |
| Movimiento en general   | <i>Manual de referencia técnica - RAPID Overview</i>   |
| Sistemas de coordenadas   | <i>Manual de referencia técnica - RAPID Overview</i>   |
| Ejecución simultánea de programas   | <i>Manual de referencia técnica - RAPID Overview</i>   |
| Parámetros del sistema  | <i>Manual de referencia técnica - Parámetros del sistema</i>   |
| Ejemplo de cómo usar TLoad, carga total.  | <a href="#">MoveL - Mueve el robot siguiendo una trayectoria lineal en la página 447</a>                       |
| Definición de la carga útil de un robot   | <a href="#">GripLoad - Define la carga útil de un robot en la página 244</a>                                   |
| LoadIdentify, rutina de servicio de identificación de carga                                   | <i>Manual del operador - IRC5 con FlexPendant</i>  |
| Señal de entrada de sistema SimMode para mover el robot en el modo simulado sin carga útil.   | <i>Manual de referencia técnica - Parámetros del sistema</i>   |
| Parámetro de sistema ModalPayLoadMode para la activación y la desactivación de la carga útil. | <i>Manual de referencia técnica - Parámetros del sistema</i>   |
| Path Offset   | <i>Application manual - Controller software IRC5</i>   |



1.138 MoveCAO - Mueve el robot en una trayectoria circular y establece una salida analógica en la esquina  
*RobotWare - OS*

## 1.138 MoveCAO - Mueve el robot en una trayectoria circular y establece una salida analógica en la esquina

### Utilización

MoveCAO (*Move Circular Analog Output*) se utiliza para trasladar el punto central de la herramienta (TCP) en una trayectoria circular hacia un punto de destino determinado. La salida analógica especificada se activa en el centro de la trayectoria de esquina del punto de destino. Durante el movimiento, la orientación suele permanecer sin cambios respecto del círculo.

Esta instrucción sólo puede usarse en la tarea principal T\_ROB1 o, si se cuenta con un sistema *MultiMove*, en las tareas de movimiento.

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción MoveCAO:

#### Ejemplo 1

```
MoveCAO p1, p2, v500, z30, tool2, ao1, 1.1;
```

El TCP de la herramienta, *tool2*, se mueve en círculo hacia la posición *p2* con los datos de velocidad *v500* y los datos de zona *z30*. El círculo se define a partir de la posición inicial, el punto de círculo *p1* y el punto de destino *p2*. La salida *ao1* se activa en el centro de la trayectoria de esquina de *p2*.

### Argumentos

```
MoveCAO CirPoint ToPoint [\ID] Speed [\T] Zone Tool [\WObj] Signal  
Value [\TLoad]
```

#### CirPoint

Tipo de dato: *robtarget*

El punto de círculo del robot. El punto de círculo es una posición del círculo entre el punto de inicio y el punto de destino. Para conseguir la máxima exactitud, debe estar situado a mitad de camino entre los puntos inicial y de destino. Si lo sitúa demasiado cerca del punto de inicio o del punto de destino, es posible que el robot genere una advertencia. El punto de círculo se define como una posición con nombre o se almacena directamente en la instrucción (marcada con un asterisco \* en la instrucción). No se utiliza la posición de los ejes externos.

#### ToPoint

Tipo de dato: *robtarget*

El punto de destino de los ejes del robot y de los ejes externos. Se define como una posición con nombre o se almacena directamente en la instrucción (marcada con un asterisco \* en la instrucción).

#### [ \ID ]

*Synchronization id*

Tipo de dato: *identno*

El argumento [ \ID ] es obligatorio en los sistemas *MultiMove*, si el movimiento es sincronizado o sincronizado coordinado. Este argumento no está permitido en ningún otro caso. El número de ID especificado debe ser el mismo en todas las

*Continúa en la página siguiente*

# 1 Instrucciones

---

1.138 MoveCAO - Mueve el robot en una trayectoria circular y establece una salida analógica en la esquina

RobotWare - OS

Continuación

tareas de programa que cooperan entre sí. Al usar el número de ID los movimientos no se mezclan en tiempo de ejecución.

Speed

Tipo de dato: speeddata

Los datos de velocidad que se aplican a los movimientos. Los datos de velocidad definen la velocidad del TCP, la reorientación de la herramienta y los ejes externos.

[ \T ]

*Time*

Tipo de dato: num

Este argumento se utiliza para especificar el tiempo total en segundos que dura el movimiento del robot y de los ejes externos. A continuación, se sustituye por los datos de velocidad correspondientes.

Zone

Tipo de dato: zonedata

Los datos de zona del movimiento. Los datos de zona describen el tamaño de la trayectoria de esquina generada.

Tool

Tipo de dato: tooldata

La herramienta en uso durante el movimiento del robot. El punto central de la herramienta es el punto que se mueve hacia el punto de destino especificado.

[ \Wobj ]

*Work Object*

Tipo de dato: wobjdata

El objeto de trabajo (sistema de coordenadas de objeto) con el que está relacionada la posición de robot indicada en la instrucción.

Es posible omitir este argumento. Si se omite, la posición depende del sistema de coordenadas mundo. Si por otro lado se usa un TCP estacionario o ejes externos coordinados, es necesario especificar este argumento para que se ejecute un círculo respecto del objeto de trabajo.

Signal

Tipo de dato: signalao

El nombre de la señal analógica de salida que debe cambiar de valor.

Value

Tipo de dato: num

El valor deseado para la señal.

[ \TLoad ]

*Total load*

Tipo de dato: loaddata

El argumento \TLoad describe la carga total usada durante el movimiento. La carga total es la carga de la herramienta más la carga útil transportada por la

*Continúa en la página siguiente*

## 1.138 MoveCAO - Mueve el robot en una trayectoria circular y establece una salida analógica en la esquina

RobotWare - OS

Continuación

herramienta. Si se utiliza el argumento `\TLoad`, no se tiene en cuenta el valor de `loaddata` en los `tooldata` actuales.

Si el argumento `\TLoad` tiene el valor `load0`, el argumento `\TLoad` no se tiene en cuenta y se utilizan en su lugar los `loaddata` de los `tooldata`.

Para poder utilizar el argumento `\TLoad`, es necesario cambiar el valor del parámetro de sistema `ModalPayLoadMode` a 0. Si `ModalPayLoadMode` tiene el valor 0, ya no es posible utilizar la instrucción `GripLoad`.

La carga total puede identificarse con la rutina de servicio `LoadIdentify`. Si el parámetro de sistema `ModalPayLoadMode` tiene el valor 0, el operador tiene la posibilidad de copiar los `loaddata` de la herramienta a una variable persistente `loaddata` existente o nueva al ejecutar la rutina de servicio.

Es posible realizar una ejecución de prueba del programa sin ninguna carga útil utilizando una señal de entrada digital conectada a la entrada de sistema `SimMode` (modo simulado). Si la señal de entrada digital tiene el valor 1, los `loaddata` del argumento opcional `\TLoad` no se tienen en cuenta y se utilizan en su lugar los `loaddata` de los `tooldata` actuales.



### Nota

La funcionalidad predeterminada de manejo de la carga útil es utilizar la instrucción `GripLoad`. Por tanto, el valor predeterminado del parámetro de sistema `ModalPayLoadMode` es 1.

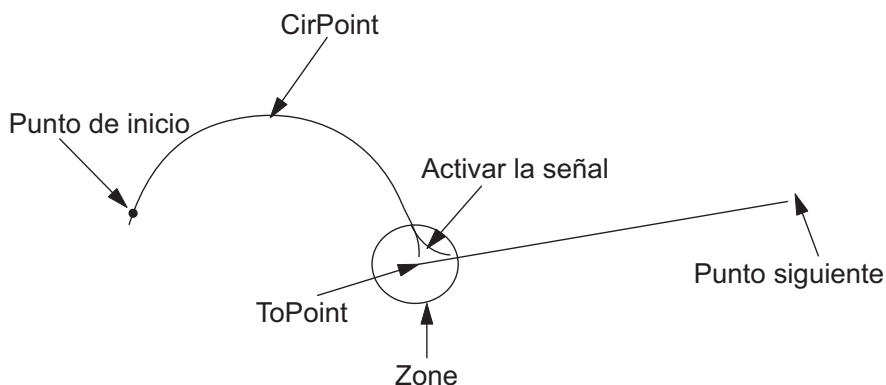
## Ejecución de programas

Consulte la instrucción `MoveC` para obtener más información acerca del movimiento circular, [MoveC - Mueve el robot en círculo en la página 389](#).

La señal analógica de salida se activa en el centro de la trayectoria de esquina en el caso de los puntos de paso, como se muestra en la figura siguiente.

La figura muestra la activación de una señal analógica de salida en la trayectoria de esquina con `MoveCAO`.

```
MoveCAO p2, p2, v500, z30, tool2, ao1, 1.1;
```



xx1400001116

En el caso de los puntos de paro, recomendamos el uso de una secuencia de programa "normal", con el uso de `MoveC` y `SetAO`. Sin embargo, cuando se utiliza

Continúa en la página siguiente

# 1 Instrucciones

1.138 MoveCAO - Mueve el robot en una trayectoria circular y establece una salida analógica en la esquina

RobotWare - OS

Continuación

un punto de paro en la instrucción MoveCAO, la señal analógica de salida se activa cuando el robot alcanza el punto de paro.

La señal de E/S especificada se activa en el modo de ejecución continua y paso a paso hacia delante, pero no en la ejecución paso a paso hacia atrás.

## Limitaciones

Limitaciones generales de acuerdo con la instrucción MoveC, consulte [MoveC - Mueve el robot en círculo en la página 389](#).

MoveCAO no puede ejecutarse en un gestor UNDO o en rutinas de RAPID que estén conectadas a los siguientes eventos especiales del sistema: PowerOn, Stop, QStop, Restart, Reset o Step.

## Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema ERRNO cambiará a:

| Nombre             | Causa del error   |
|--------------------|---|
| ERR_AO_LIM         | El argumento programado Value para la señal analógica de salida especificada Signal está fuera de límites.  |
| ERR_NO_ALIASIO_DEF | La variable de señal es una variable declarada en RAPID. No se ha conectado a una señal E/S definida en la configuración de E/S con la instrucción AliasIO.   |
| ERR_NORUNUNIT      | Se ha perdido el contacto con el dispositivo de E/S.  |
| ERR_SIG_NOT_VALID  | La señal de E/S no está disponible. Los motivos pueden ser que el dispositivo de E/S no está en funcionamiento o que exista un error en la configuración (válido solamente para el bus de campo ICI). |

## Sintaxis

```
MoveCAO
  [CirPoint ':='] <expression (IN) of robtarg>','
  [ToPoint ':='] <expression (IN) of robtarg>','
  ['\ 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>'];'
```

## Información relacionada

| Para obtener más información sobre     | Consulte   |
|--|--|
| Otras instrucciones de posicionamiento | <i>Manual de referencia técnica - RAPID Overview</i>               |
| Movimiento del robot en círculo        | <a href="#">MoveC - Mueve el robot en círculo en la página 389</a> |

Continúa en la página siguiente

## 1.138 MoveCAO - Mueve el robot en una trayectoria circular y establece una salida analógica en la esquina

RobotWare - OS

Continuación

| Para obtener más información sobre  | Consulte   |
|---|--|
| Definición de carga   | <a href="#">loaddata - Datos de carga en la página 1694</a>                              |
| Definición de velocidad   | <a href="#">speeddata - Datos de velocidad en la página 1767</a>                         |
| Definición de herramientas  | <a href="#">tooldata - Datos de herramienta en la página 1795</a>                        |
| Definición de objetos de trabajo  | <a href="#">wobjdata - Datos del objeto de trabajo en la página 1823</a>                 |
| Definición de datos de zona   | <a href="#">zonedata - Datos de zonas en la página 1831</a>                              |
| Movimiento en general   | <i>Manual de referencia técnica - RAPID Overview</i>                                     |
| Sistemas de coordenadas   | <i>Manual de referencia técnica - RAPID Overview</i>                                     |
| Movimientos con parámetros de E/S   | <i>Manual de referencia técnica - RAPID Overview</i>                                     |
| Ejemplo de cómo usar TLoad, carga total.  | <a href="#">MoveL - Mueve el robot siguiendo una trayectoria lineal en la página 447</a> |
| Definición de la carga útil de un robot   | <a href="#">GripLoad - Define la carga útil de un robot en la página 244</a>             |
| LoadIdentify, rutina de servicio de identificación de carga                                   | <i>Manual del operador - IRC5 con FlexPendant</i>  |
| Señal de entrada de sistema SimMode para mover el robot en el modo simulado sin carga útil.   | <i>Manual de referencia técnica - Parámetros del sistema</i>                             |
| Parámetro de sistema ModalPayloadMode para la activación y la desactivación de la carga útil. | <i>Manual de referencia técnica - Parámetros del sistema</i>                             |

## 1 Instrucciones

---

1.139 MoveCDO - Mueve el robot en una trayectoria circular y establece una salida digital en la esquina  
*RobotWare - OS*

### 1.139 MoveCDO - Mueve el robot en una trayectoria circular y establece una salida digital en la esquina

---

#### Utilización

MoveCDO (*Move Circular Digital Output*) se utiliza para trasladar el punto central de la herramienta (TCP) en sentido circular hacia un punto de destino determinado. La salida digital especificada se activa o desactiva en el centro de la trayectoria de esquina del punto de destino. Durante el movimiento, la orientación suele permanecer sin cambios respecto del círculo.

Esta instrucción sólo puede usarse en la tarea principal T\_ROB1 o, si se cuenta con un sistema *MultiMove*, en las tareas de movimiento.

#### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción MoveCDO:

##### Ejemplo 1

```
MoveCDO p1, p2, v500, z30, tool2, do1,1;
```

El TCP de la herramienta, *tool2*, se mueve en círculo hacia la posición *p2* con los datos de velocidad *v500* y los datos de zona *z30*. El círculo se define a partir de la posición inicial, el punto de círculo *p1* y el punto de destino *p2*. La salida *do1* se activa en el centro de la trayectoria de esquina de *p2*.

#### Argumentos

```
MoveCDO CirPoint ToPoint [\ID] Speed [\T] Zone Tool [\WObj] Signal  
Value [\TLoad]
```

##### CirPoint

Tipo de dato: *robtarget*

El punto de círculo del robot. El punto de círculo es una posición del círculo entre el punto de inicio y el punto de destino. Para conseguir la máxima exactitud, debe estar situado a mitad de camino entre los puntos inicial y de destino. Si lo sitúa demasiado cerca del punto de inicio o del punto de destino, es posible que el robot genere una advertencia. El punto de círculo se define como una posición con nombre o se almacena directamente en la instrucción (marcada con un asterisco \* en la instrucción). No se utiliza la posición de los ejes externos.

##### ToPoint

Tipo de dato: *robtarget*

El punto de destino de los ejes del robot y de los ejes externos. Se define como una posición con nombre o se almacena directamente en la instrucción (marcada con un asterisco \* en la instrucción).

##### [ \ID ]

*Synchronization id*

Tipo de dato: *identno*

El argumento [ \ID ] es obligatorio en los sistemas *MultiMove*, si el movimiento es sincronizado o sincronizado coordinado. Este argumento no está permitido en ningún otro caso. El número de ID especificado debe ser el mismo en todas las

*Continúa en la página siguiente*

## 1.139 MoveCDO - Mueve el robot en una trayectoria circular y establece una salida digital en la esquina RobotWare - OS Continuación

tareas de programa que cooperan entre sí. Al usar el número de ID los movimientos no se mezclan en tiempo de ejecución.

Speed

Tipo de dato: speeddata

Los datos de velocidad que se aplican a los movimientos. Los datos de velocidad definen la velocidad del TCP, la reorientación de la herramienta y los ejes externos.

[ \T ]

*Time*

Tipo de dato: num

Este argumento se utiliza para especificar el tiempo total en segundos que dura el movimiento del robot y de los ejes externos. A continuación, se sustituye por los datos de velocidad correspondientes.

Zone

Tipo de dato: zonedata

Los datos de zona del movimiento. Los datos de zona describen el tamaño de la trayectoria de esquina generada.

Tool

Tipo de dato: tooldata

La herramienta en uso durante el movimiento del robot. El punto central de la herramienta es el punto que se mueve hacia el punto de destino especificado.

[ \Wobj ]

*Work Object*

Tipo de dato: wobjdata

El objeto de trabajo (sistema de coordenadas de objeto) con el que está relacionada la posición de robot indicada en la instrucción.

Es posible omitir este argumento. Si se omite, la posición depende del sistema de coordenadas mundo. Si por otro lado se usa un TCP estacionario o ejes externos coordinados, es necesario especificar este argumento para que se ejecute un círculo respecto del objeto de trabajo.

Signal

Tipo de dato: signaldo

El nombre de la señal digital de salida que debe cambiar de valor.

Value

Tipo de dato: dionum

El valor deseado para la señal (0 ó 1).

[ \TLoad ]

*Total load*

Tipo de dato: loaddata

El argumento \TLoad describe la carga total usada durante el movimiento. La carga total es la carga de la herramienta más la carga útil transportada por la

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.139 MoveCDO - Mueve el robot en una trayectoria circular y establece una salida digital en la esquina RobotWare - OS Continuación

herramienta. Si se utiliza el argumento `\TLoad`, no se tiene en cuenta el valor de `loaddata` en los `tooldata` actuales.

Si el argumento `\TLoad` tiene el valor `load0`, el argumento `\TLoad` no se tiene en cuenta y se utilizan en su lugar los `loaddata` de los `tooldata`.

Para poder utilizar el argumento `\TLoad`, es necesario cambiar el valor del parámetro de sistema `ModalPayLoadMode` a 0. Si `ModalPayLoadMode` tiene el valor 0, ya no es posible utilizar la instrucción `GripLoad`.

La carga total puede identificarse con la rutina de servicio `LoadIdentify`. Si el parámetro de sistema `ModalPayLoadMode` tiene el valor 0, el operador tiene la posibilidad de copiar los `loaddata` de la herramienta a una variable persistente `loaddata` existente o nueva al ejecutar la rutina de servicio.

Es posible realizar una ejecución de prueba del programa sin ninguna carga útil utilizando una señal de entrada digital conectada a la entrada de sistema `SimMode` (modo simulado). Si la señal de entrada digital tiene el valor 1, los `loaddata` del argumento opcional `\TLoad` no se tienen en cuenta y se utilizan en su lugar los `loaddata` de los `tooldata` actuales.



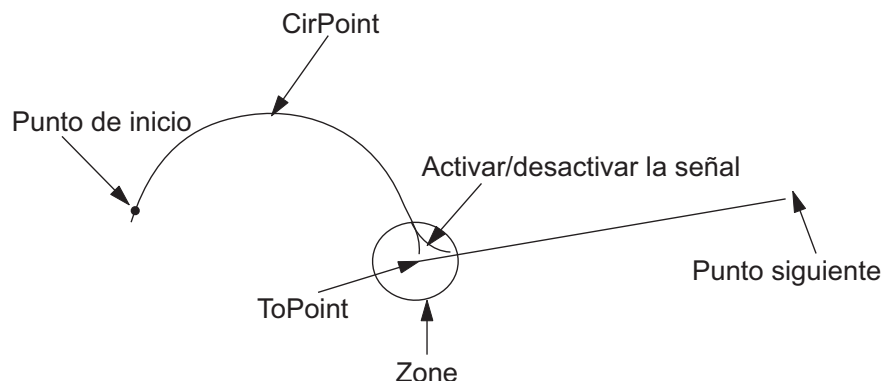
### Nota

La funcionalidad predeterminada de manejo de la carga útil es utilizar la instrucción `GripLoad`. Por tanto, el valor predeterminado del parámetro de sistema `ModalPayLoadMode` es 1.

## Ejecución de programas

Consulte la instrucción `MoveC` para obtener más información acerca del movimiento circular.

La señal digital de salida se activa o desactiva en el centro de la trayectoria de esquina en el caso de los puntos de paso, como se muestra en la figura siguiente. La figura muestra la activación/desactivación de una señal digital de salida en la trayectoria de esquina con `MoveCDO`.



xx0500002215

En el caso de los puntos de paro, recomendamos el uso de una secuencia de programa "normal", con el uso de `MoveC + SetDO`. Sin embargo, cuando se utiliza

Continúa en la página siguiente



## 1.139 MoveCDO - Mueve el robot en una trayectoria circular y establece una salida digital en la esquina

*RobotWare - OS*  
*Continuación*

un punto de paro en la instrucción `MoveCDO`, la señal digital de salida se activa o desactiva cuando el robot alcanza el punto de paro.

La señal de E/S especificada se activa o desactiva en el modo de ejecución continua y paso a paso hacia adelante, pero no en la ejecución paso a paso hacia atrás.

### Limitaciones

Limitaciones generales acorde con la instrucción `MoveC`.

`MoveCDO` no puede ejecutarse en un gestor UNDO o en rutinas de RAPID que estén conectadas a los siguientes eventos especiales del sistema: PowerOn, Stop, QStop, Restart, Reset o Step.

### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                          | Causa del error   |
|---------------------------------|---|
| <code>ERR_NO_ALIASIO_DEF</code> | La variable de señal es una variable declarada en RAPID. No se ha conectado a una señal E/S definida en la configuración de E/S con la instrucción <code>AliasIO</code> .                             |
| <code>ERR_NORUNUNIT</code>      | Se ha perdido el contacto con el dispositivo de E/S.  |
| <code>ERR_SIG_NOT_VALID</code>  | La señal de E/S no está disponible. Los motivos pueden ser que el dispositivo de E/S no está en funcionamiento o que exista un error en la configuración (válido solamente para el bus de campo ICI). |

### Sintaxis

```
MoveCDO
  [CirPoint ':='] <expression (IN) of robtargt>','
  [ToPoint ':='] <expression (IN) of robtargt>','
  ['\ ' 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>'];'
```

### Información relacionada

| Para obtener más información sobre     | Consulte   |
|--|--|
| Otras instrucciones de posicionamiento | <i>Manual de referencia técnica - RAPID Overview</i>               |
| Movimiento del robot en círculo        | <a href="#">MoveC - Mueve el robot en círculo en la página 389</a> |
| Definición de carga                    | <a href="#">loaddata - Datos de carga en la página 1694</a>        |
| Definición de velocidad                | <a href="#">speeddata - Datos de velocidad en la página 1767</a>   |
| Definición de herramientas             | <a href="#">tooldata - Datos de herramienta en la página 1795</a>  |

*Continúa en la página siguiente*

# 1 Instrucciones

1.139 MoveCDO - Mueve el robot en una trayectoria circular y establece una salida digital en la esquina

RobotWare - OS

Continuación

| Para obtener más información sobre  | Consulte   |
|---|--|
| Definición de objetos de trabajo  | <a href="#">wobjdata - Datos del objeto de trabajo en la página 1823</a>                 |
| Definición de datos de zona   | <a href="#">zonedata - Datos de zonas en la página 1831</a>                              |
| Movimiento en general   | <i>Manual de referencia técnica - RAPID Overview</i>                                     |
| Sistemas de coordenadas   | <i>Manual de referencia técnica - RAPID Overview</i>                                     |
| Movimientos con parámetros de E/S   | <i>Manual de referencia técnica - RAPID Overview</i>                                     |
| Ejemplo de cómo usar TLoad, carga total.  | <a href="#">MoveL - Mueve el robot siguiendo una trayectoria lineal en la página 447</a> |
| Definición de la carga útil de un robot   | <a href="#">GripLoad - Define la carga útil de un robot en la página 244</a>             |
| LoadIdentify, rutina de servicio de identificación de carga                                   | <i>Manual del operador - IRC5 con FlexPendant</i>  |
| Señal de entrada de sistema SimMode para mover el robot en el modo simulado sin carga útil.   | <i>Manual de referencia técnica - Parámetros del sistema</i>                             |
| Parámetro de sistema ModalPayloadMode para la activación y la desactivación de la carga útil. | <i>Manual de referencia técnica - Parámetros del sistema</i>                             |

1.140 MoveCGO - Mueve el robot en una trayectoria circular y establece una señal de salida de grupo en la esquina  
RobotWare - OS

## 1.140 MoveCGO - Mueve el robot en una trayectoria circular y establece una señal de salida de grupo en la esquina

### Utilización

MoveCGO (*Move Circular Group Output*) se utiliza para trasladar el punto central de la herramienta (TCP) en una trayectoria circular hacia un punto de destino determinado. La señal de salida de grupo especificada se activa en el centro de la trayectoria de esquina del punto de destino. Durante el movimiento, la orientación suele permanecer sin cambios respecto del círculo.

Esta instrucción sólo puede usarse en la tarea principal T\_ROB1 o, si se cuenta con un sistema *MultiMove*, en las tareas de movimiento.

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción MoveCGO:

#### Ejemplo 1

```
MoveCGO p1, p2, v500, z30, tool2, go1 \Value:=5;
```

El TCP de la herramienta, `tool2`, se mueve en una trayectoria circular hacia la posición `p2` con los datos de velocidad `v500` y los datos de zona `z30`. El círculo se define a partir de la posición inicial, el punto de círculo `p1` y el punto de destino `p2`. La señal de salida de grupo `go1` se activa en el centro de la trayectoria de esquina de `p2`.

### Argumentos

```
MoveCGO CirPoint ToPoint [\ID] Speed [\T] Zone Tool [\WObj] Signal
[\Value] | [\DValue] [\TLoad]
```

#### CirPoint

Tipo de dato: `robtarget`

El punto de círculo del robot. El punto de círculo es una posición del círculo entre el punto de inicio y el punto de destino. Para conseguir la máxima exactitud, debe estar situado a mitad de camino entre los puntos inicial y de destino. Si lo sitúa demasiado cerca del punto de inicio o del punto de destino, es posible que el robot genere una advertencia. El punto de círculo se define como una posición con nombre o se almacena directamente en la instrucción (marcada con un asterisco \* en la instrucción). No se utiliza la posición de los ejes externos.

#### ToPoint

Tipo de dato: `robtarget`

El punto de destino de los ejes del robot y de los ejes externos. Se define como una posición con nombre o se almacena directamente en la instrucción (marcada con un asterisco \* en la instrucción).

#### [ \ID ]

*Synchronization id*

Tipo de dato: `identno`

El argumento [ \ID ] es obligatorio en los sistemas *MultiMove*, si el movimiento es sincronizado o sincronizado coordinado. Este argumento no está permitido en

*Continúa en la página siguiente*

## 1 Instrucciones

---

1.140 MoveCGO - Mueve el robot en una trayectoria circular y establece una señal de salida de grupo en la esquina

*RobotWare - OS*

*Continuación*

ningún otro caso. El número de ID especificado debe ser el mismo en todas las tareas de programa que cooperan entre sí. Al usar el número de ID los movimientos no se mezclan en tiempo de ejecución.

Speed

Tipo de dato: speeddata

Los datos de velocidad que se aplican a los movimientos. Los datos de velocidad definen la velocidad del TCP, la reorientación de la herramienta y los ejes externos.

[ \T ]

*Time*

Tipo de dato: num

Este argumento se utiliza para especificar el tiempo total en segundos que dura el movimiento del robot y de los ejes externos. A continuación, se sustituye por los datos de velocidad correspondientes.

Zone

Tipo de dato: zonedata

Los datos de zona del movimiento. Los datos de zona describen el tamaño de la trayectoria de esquina generada.

Tool

Tipo de dato: tooldata

La herramienta en uso durante el movimiento del robot. El punto central de la herramienta es el punto que se mueve hacia el punto de destino especificado.

[ \WObj ]

*Work Object*

Tipo de dato: wobjdata

El objeto de trabajo (sistema de coordenadas de objeto) con el que está relacionada la posición de robot indicada en la instrucción.

Es posible omitir este argumento. Si se omite, la posición depende del sistema de coordenadas mundo. Si por otro lado se usa un TCP estacionario o ejes externos coordinados, es necesario especificar este argumento para que se ejecute un círculo respecto del objeto de trabajo.

Signal

Tipo de dato: signalgo

El nombre de la señal de salida de grupo que debe cambiar de valor.

[ \Value ]

Tipo de dato: num

El valor deseado para la señal.

[ \DValue ]

Tipo de dato: dnum

El valor deseado para la señal.

*Continúa en la página siguiente*

1.140 MoveCGO - Mueve el robot en una trayectoria circular y establece una señal de salida de grupo en la esquina  
 RobotWare - OS

Si no se introduce ninguno de los argumentos \Value ni \DValue, aparece un mensaje de error.

[ \TLoad ]

### Total load

Tipo de dato: loaddata

El argumento \TLoad describe la carga total usada durante el movimiento. La carga total es la carga de la herramienta más la carga útil transportada por la herramienta. Si se utiliza el argumento \TLoad, no se tiene en cuenta el valor de loaddata en los tooldata actuales.

Si el argumento \TLoad tiene el valor load0, el argumento \TLoad no se tiene en cuenta y se utilizan en su lugar los loaddata de los tooldata.

Para poder utilizar el argumento \TLoad, es necesario cambiar el valor del parámetro de sistema ModalPayLoadMode a 0. Si ModalPayLoadMode tiene el valor 0, ya no es posible utilizar la instrucción GripLoad.

La carga total puede identificarse con la rutina de servicio LoadIdentify. Si el parámetro de sistema ModalPayLoadMode tiene el valor 0, el operador tiene la posibilidad de copiar los loaddata de la herramienta a una variable persistente loaddata existente o nueva al ejecutar la rutina de servicio.

Es posible realizar una ejecución de prueba del programa sin ninguna carga útil utilizando una señal de entrada digital conectada a la entrada de sistema SimMode (modo simulado). Si la señal de entrada digital tiene el valor 1, los loaddata del argumento opcional \TLoad no se tienen en cuenta y se utilizan en su lugar los loaddata de los tooldata actuales.



### Nota

La funcionalidad predeterminada de manejo de la carga útil es utilizar la instrucción GripLoad. Por tanto, el valor predeterminado del parámetro de sistema ModalPayLoadMode es 1.

## Ejecución de programas

Consulte la instrucción MoveC para obtener más información acerca del movimiento circular, [MoveC - Mueve el robot en círculo en la página 389](#).

La señal de salida de grupo se activa en el centro de la trayectoria de esquina en el caso de los puntos de paso, como se muestra en la figura siguiente.

La figura muestra la activación de una señal de salida de grupo en la trayectoria de esquina con MoveCGO.

```
MoveCGO p2, p2, v500, z30, tool2, go1 \Value:=5;
```

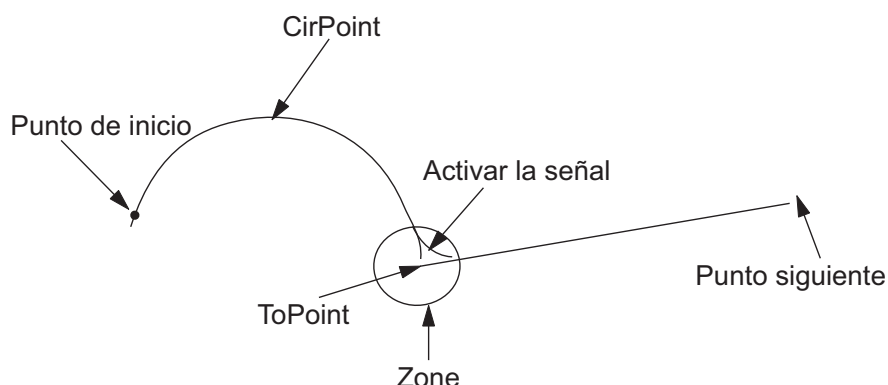
Continúa en la página siguiente

# 1 Instrucciones

1.140 MoveCGO - Mueve el robot en una trayectoria circular y establece una señal de salida de grupo en la esquina

RobotWare - OS

Continuación



xx1400001116

En el caso de los puntos de paro, recomendamos el uso de una secuencia de programa “normal”, con el uso de `MoveC` y `SetGO`. Sin embargo, cuando se utiliza un punto de paro en la instrucción `MoveCGO`, la señal de salida de grupo se activa cuando el robot alcanza el punto de paro.

La señal de E/S especificada se activa en el modo de ejecución continua y paso a paso hacia delante, pero no en la ejecución paso a paso hacia atrás.

## Limitaciones

Limitaciones generales de acuerdo con la instrucción `MoveC`, consulte [MoveC - Mueve el robot en círculo en la página 389](#).

`MoveCGO` no puede ejecutarse en un gestor UNDO o en rutinas de RAPID que estén conectadas a los siguientes eventos especiales del sistema: PowerOn, Stop, QStop, Restart, Reset o Step.

## Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                          | Causa del error   |
|---------------------------------|---|
| <code>ERR_GO_LIM</code>         | El argumento <code>Value</code> o <code>DValue</code> para la señal de salida de grupo especificada está fuera de límites.  |
| <code>ERR_NO_ALIASIO_DEF</code> | La variable de señal es una variable declarada en RAPID. No se ha conectado a una señal E/S definida en la configuración de E/S con la instrucción <code>AliasIO</code> .                             |
| <code>ERR_NORUNUNIT</code>      | Se ha perdido el contacto con el dispositivo de E/S.  |
| <code>ERR_SIG_NOT_VALID</code>  | La señal de E/S no está disponible. Los motivos pueden ser que el dispositivo de E/S no está en funcionamiento o que exista un error en la configuración (válido solamente para el bus de campo ICI). |

## Sintaxis

```
MoveCGO
[CirPoint ':='] <expression (IN) of robtarget>',
[ToPoint ':='] <expression (IN) of robtarget>',
['\' ID ':=' <expression (IN) of identno>'],'
[Speed ':='] <expression (IN) of speeddata>
```

Continúa en la página siguiente

## 1.140 MoveCGO - Mueve el robot en una trayectoria circular y establece una señal de salida de grupo en la esquina

RobotWare - OS

Continuación

```
[['\ ' 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>'];'
```

### Información relacionada

| Para obtener más información sobre  | Consulte   |
|---|--|
| Otras instrucciones de posicionamiento  | <i>Manual de referencia técnica - RAPID Overview</i>                                     |
| Movimiento del robot en círculo   | <a href="#">MoveC - Mueve el robot en círculo en la página 389</a>                       |
| Definición de carga   | <a href="#">loaddata - Datos de carga en la página 1694</a>                              |
| Definición de velocidad   | <a href="#">speeddata - Datos de velocidad en la página 1767</a>                         |
| Definición de herramientas  | <a href="#">tooldata - Datos de herramienta en la página 1795</a>                        |
| Definición de objetos de trabajo  | <a href="#">wobjdata - Datos del objeto de trabajo en la página 1823</a>                 |
| Definición de datos de zona   | <a href="#">zonedata - Datos de zonas en la página 1831</a>                              |
| Movimiento en general   | <i>Manual de referencia técnica - RAPID Overview</i>                                     |
| Sistemas de coordenadas   | <i>Manual de referencia técnica - RAPID Overview</i>                                     |
| Movimientos con parámetros de E/S   | <i>Manual de referencia técnica - RAPID Overview</i>                                     |
| Ejemplo de cómo usar TLoad, carga total.  | <a href="#">MoveL - Mueve el robot siguiendo una trayectoria lineal en la página 447</a> |
| Definición de la carga útil de un robot   | <a href="#">GripLoad - Define la carga útil de un robot en la página 244</a>             |
| LoadIdentify, rutina de servicio de identificación de carga                                   | <i>Manual del operador - IRC5 con FlexPendant</i>  |
| Señal de entrada de sistema SimMode para mover el robot en el modo simulado sin carga útil.   | <i>Manual de referencia técnica - Parámetros del sistema</i>                             |
| Parámetro de sistema ModalPayloadMode para la activación y la desactivación de la carga útil. | <i>Manual de referencia técnica - Parámetros del sistema</i>                             |

## 1 Instrucciones

---

1.141 MoveCSync - Mueve el robot en una trayectoria circular y ejecuta un procedimiento de RAPID RobotWare - OS

### 1.141 MoveCSync - Mueve el robot en una trayectoria circular y ejecuta un procedimiento de RAPID

---

#### Utilización

MoveCSync (*Move Circular Synchronously*) se utiliza para trasladar el punto central de la herramienta (TCP) en sentido circular hacia un punto de destino determinado. En el centro de la trayectoria de esquina del punto de destino, se ordena la ejecución del procedimiento de RAPID especificado. Durante el movimiento, la orientación suele permanecer sin cambios respecto del círculo.

Esta instrucción sólo puede usarse en la tarea principal T\_ROB1 o, si se cuenta con un sistema *MultiMove*, en las tareas de movimiento.

---

#### Ejemplos básicos

Los ejemplos siguientes ilustran la instrucción MoveCSync.

##### Ejemplo 1

```
MoveCSync p1, p2, v500, z30, tool2, "procl";
```

El TCP de la herramienta, tool2, se mueve en círculo hacia la posición p2 con los datos de velocidad v500 y los datos de zona z30. El círculo se define a partir de la posición inicial, el punto de círculo p1 y el punto de destino p2. El procedimiento procl se ejecuta en el centro de la trayectoria de esquina de p2.

##### Ejemplo 2

```
MoveCSync p1, p2, v500, z30, tool2, "MyModule:procl";
```

Igual que en el ejemplo 1 anterior, pero en este caso el procedimiento declarado localmente procl del módulo MyModule será llamado en medio de la trayectoria de esquina.

---

#### Argumentos

```
MoveCSync CirPoint ToPoint [\ID] Speed [\T] Zone Tool [\WObj]  
ProcName [\TLoad]
```

CirPoint

Tipo de dato: robtarget

El punto de círculo del robot. El punto de círculo es una posición del círculo entre el punto de inicio y el punto de destino. Para conseguir la máxima exactitud, debe estar situado a mitad de camino entre los puntos inicial y de destino. Si lo sitúa demasiado cerca del punto de inicio o del punto de destino, es posible que el robot genere una advertencia. El punto de círculo se define como una posición con nombre o se almacena directamente en la instrucción (marcada con un asterisco \* en la instrucción). No se utiliza la posición de los ejes externos.

ToPoint

Tipo de dato: robtarget

El punto de destino de los ejes del robot y de los ejes externos. Se define como una posición con nombre o se almacena directamente en la instrucción (marcada con un asterisco \* en la instrucción).

---

*Continúa en la página siguiente*



## 1.141 MoveCSync - Mueve el robot en una trayectoria circular y ejecuta un procedimiento de RAPID

RobotWare - OS

Continuación

[ \ID ]

### *Synchronization id*

Tipo de dato: `identno`

El argumento [ \ID ] es obligatorio en los sistemas MultiMove, si el movimiento es sincronizado o sincronizado coordinado. Este argumento no está permitido en ningún otro caso. El número de ID especificado debe ser el mismo en todas las tareas de programa que cooperan entre sí. Al usar el número de ID los movimientos no se mezclan en tiempo de ejecución.

Speed

Tipo de dato: `speeddata`

Los datos de velocidad que se aplican a los movimientos. Los datos de velocidad definen la velocidad del TCP, la reorientación de la herramienta y los ejes externos.

[ \T ]

### *Time*

Tipo de dato: `num`

Este argumento se utiliza para especificar el tiempo total en segundos que dura el movimiento del robot y de los ejes externos. A continuación, se sustituye por los datos de velocidad correspondientes.

Zone

Tipo de dato: `zonedata`

Los datos de zona del movimiento. Los datos de zona describen el tamaño de la trayectoria de esquina generada.

Tool

Tipo de dato: `tooldata`

La herramienta en uso durante el movimiento del robot. El punto central de la herramienta es el punto que se mueve hacia el punto de destino especificado.

[ \WObj ]

### *Work Object*

Tipo de dato: `wobjdata`

El objeto de trabajo (sistema de coordenadas de objeto) con el que está relacionada la posición de robot indicada en la instrucción.

Es posible omitir este argumento. Si se omite, la posición depende del sistema de coordenadas mundo. Si, por otro lado, se utiliza un TCP estacionario o ejes externos coordinados, es necesario especificar este argumento.

ProcName

### *Procedure Name*

Tipo de dato: `string`

El nombre del procedimiento de RAPID que debe ejecutarse en el centro de la trayectoria de esquina del punto de destino.

El procedimiento se ejecutará en el nivel de rutina TRAP (consulte Ejecución de programas más abajo).

*Continúa en la página siguiente*

## 1 Instrucciones

---

### 1.141 MoveCSync - Mueve el robot en una trayectoria circular y ejecuta un procedimiento de RAPID RobotWare - OS Continuación

[ \TLoad ]

#### **Total load**

Tipo de dato: loaddata

El argumento \TLoad describe la carga total usada durante el movimiento. La carga total es la carga de la herramienta más la carga útil transportada por la herramienta. Si se utiliza el argumento \TLoad, no se tiene en cuenta el valor de loaddata en los tooldata actuales.

Si el argumento \TLoad tiene el valor load0, el argumento \TLoad no se tiene en cuenta y se utilizan en su lugar los loaddata de los tooldata.

Para poder utilizar el argumento \TLoad, es necesario cambiar el valor del parámetro de sistema ModalPayLoadMode a 0. Si ModalPayLoadMode tiene el valor 0, ya no es posible utilizar la instrucción GripLoad.

La carga total puede identificarse con la rutina de servicio LoadIdentify. Si el parámetro de sistema ModalPayLoadMode tiene el valor 0, el operador tiene la posibilidad de copiar los loaddata de la herramienta a una variable persistente loaddata existente o nueva al ejecutar la rutina de servicio.

Es posible realizar una ejecución de prueba del programa sin ninguna carga útil utilizando una señal de entrada digital conectada a la entrada de sistema SimMode (modo simulado). Si la señal de entrada digital tiene el valor 1, los loaddata del argumento opcional \TLoad no se tienen en cuenta y se utilizan en su lugar los loaddata de los tooldata actuales.



#### **Nota**

La funcionalidad predeterminada de manejo de la carga útil es utilizar la instrucción GripLoad. Por tanto, el valor predeterminado del parámetro de sistema ModalPayLoadMode es 1.

---

### Ejecución de programas

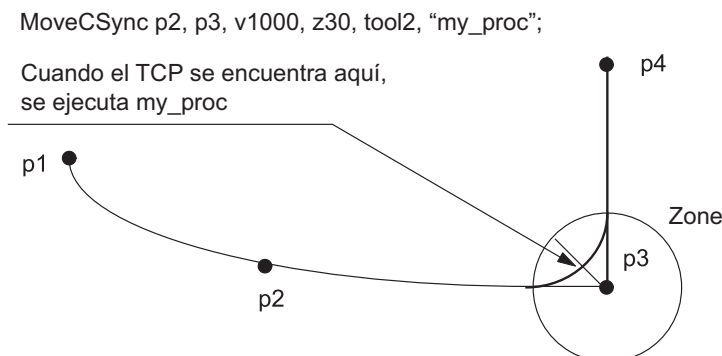
Consulte la instrucción MoveC para obtener más información acerca de los movimientos circulares.

El procedimiento de RAPID especificado se ejecuta cuando el TCP alcanza el centro de la trayectoria de esquina del punto de destino de la instrucción MoveCSync, como se muestra en la figura siguiente.

*Continúa en la página siguiente*

1.141 MoveCSync - Mueve el robot en una trayectoria circular y ejecuta un procedimiento de RAPID  
*RobotWare - OS*  
*Continuación*

La figura muestra el orden de ejecución del procedimiento de RAPID definido por el usuario al llegar al centro de la trayectoria de esquina.



xx0500002216

En el caso de los puntos de paro, recomendamos el uso de una secuencia de programa "normal", con el uso de MoveC y otras instrucciones de RAPID una tras otra.

En la tabla se describe la ejecución del procedimiento de RAPID especificado, con distintos modos de ejecución:

| Modo de ejecución  | Ejecución del procedimiento de RAPID |
|--------------------|--------------------------------------|
| Continuous o Cycle | De acuerdo con esta descripción      |
| Paso hacia delante | En el punto de paro                  |
| Paso hacia atrás   | Ninguno en absoluto                  |

MoveCSync es una encapsulación de las instrucciones TriggInt y TriggC. La llamada al procedimiento se ejecuta en el nivel de rutina TRAP.

Si el centro de la trayectoria de esquina del punto de destino se alcanza durante la deceleración posterior a un paro de programa, no se realizará la llamada al procedimiento (la ejecución del programa se para). La llamada al procedimiento se ejecuta en el siguiente inicio de programa.

**Limitaciones**

Limitaciones generales acorde con la instrucción MoveC.

Cuando el robot alcanza el centro de la trayectoria de esquina, existe normalmente un retardo de 2 a 30 ms hasta que se ejecuta la rutina de RAPID especificada, en función de qué tipo de movimiento se realice en el momento.

Si tras un punto de paro se cambia el modo de ejecución del modo continuo o cíclico al modo de ejecución paso a paso hacia adelante o hacia atrás, se genera un error. Este error informa al usuario de que el cambio de modo de ejecución puede dar lugar a que no llegue a ejecutarse un procedimiento de RAPID que está en cola para su ejecución en la trayectoria.

No es posible usar la instrucción MoveCSync en rutinas TRAP. No es posible comprobar el procedimiento de RAPID con la ejecución paso a paso.

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.141 MoveCSync - Mueve el robot en una trayectoria circular y ejecuta un procedimiento de RAPID RobotWare - OS Continuación

MoveCSync no puede ejecutarse en un gestor UNDO o en rutinas de RAPID que estén conectadas a los siguientes eventos especiales del sistema: PowerOn, Stop, QStop, Restart, Reset o Step.

### Sintaxis

```
MoveCSync
  [CirPoint ':='] <expression (IN) of robtarg>','
  [ToPoint ':='] <expression (IN) of robtarg>','
  ['\ 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>'];'
```

### Información relacionada

| Para obtener más información sobre  | Consulte  |
|---|---|
| Otras instrucciones de posicionamiento  | <a href="#">Manual de referencia técnica - RAPID Overview</a>                                   |
| Mueve el robot en círculo   | <a href="#">MoveC - Mueve el robot en círculo en la página 389</a>                              |
| Definición de carga   | <a href="#">loaddata - Datos de carga en la página 1694</a>                                     |
| Definición de velocidad   | <a href="#">speeddata - Datos de velocidad en la página 1767</a>                                |
| Definición de herramientas  | <a href="#">tooldata - Datos de herramienta en la página 1795</a>                               |
| Definición de objetos de trabajo  | <a href="#">wobjdata - Datos del objeto de trabajo en la página 1823</a>                        |
| Definición de datos de zona   | <a href="#">zonedata - Datos de zonas en la página 1831</a>                                     |
| Movimiento en general   | <a href="#">Manual de referencia técnica - RAPID Overview</a>                                   |
| Sistemas de coordenadas   | <a href="#">Manual de referencia técnica - RAPID Overview</a>                                   |
| Define una interrupción dependiente de una posición   | <a href="#">TriggInt - Define una interrupción dependiente de una posición en la página 921</a> |
| Movimiento circular del robot con eventos   | <a href="#">TriggC - Movimiento circular del robot con eventos en la página 895</a>             |
| Ejemplo de cómo usar TLoad, carga total.  | <a href="#">MoveL - Mueve el robot siguiendo una trayectoria lineal en la página 447</a>        |
| Definición de la carga útil de un robot   | <a href="#">GripLoad - Define la carga útil de un robot en la página 244</a>                    |
| LoadIdentify, rutina de servicio de identificación de carga                                   | <a href="#">Manual del operador - IRC5 con FlexPendant</a>                                      |
| Señal de entrada de sistema SimMode para mover el robot en el modo simulado sin carga útil.   | <a href="#">Manual de referencia técnica - Parámetros del sistema</a>                           |
| Parámetro de sistema ModalPayloadMode para la activación y la desactivación de la carga útil. |   |

## 1.142 MoveExtJ - Mueve una o varias unidades mecánicas sin TCP

### Utilización

MoveExtJ (*Move External Joints*) se utiliza para mover ejes externos sólo lineales o de rotación. Estos ejes externos pueden pertenecer a una o varias unidades mecánicas sin TCP.

Esta instrucción sólo puede usarse con una tarea de programa real definida como tarea de movimiento y siempre y cuando la tarea controle una o varias unidades mecánicas sin TCP.

### Ejemplos básicos

Los ejemplos siguientes ilustran la instrucción MoveExtJ.

Consulte también [Más ejemplos en la página 419](#).

#### Ejemplo 1

```
MoveExtJ jpos10, vrot10, z50;
```

Mueve el eje externo de rotación a la posición de eje jpos10 a una velocidad de 10 grados/s con los datos de zona z50.

#### Ejemplo 2

```
MoveExtJ \Conc, jpos20, vrot10 \T:=5, fine \InPos:=inpos20;
```

Mueve los ejes externos a la posición de eje jpos20 en 5 s. La ejecución del programa avanza de una vez pero el eje externo se detiene en la posición jpos20 hasta que se cumplan los criterios de convergencia en inpos20.

### Argumentos

```
MoveExtJ [\Conc] ToJointPos [\ID] [\UseEOffs] Speed [\T] Zone
[\Inpos]
```

[ \Conc ]

#### *Concurrent*

Tipo de dato: switch

Las instrucciones posteriores se ejecutan mientras el eje externo está en movimiento. Este argumento no se utiliza normalmente, pero puede utilizarse para evitar los paros no deseados, causados por la sobrecarga de la CPU al utilizar puntos de paso. Esto resulta útil cuando los puntos programados están muy cercanos entre sí y se trabaja a velocidades elevadas. Este argumento también resulta útil si, por ejemplo, no se requiere la comunicación con equipos externos ni la sincronización entre los equipos externos y los movimientos del robot.

Cuando se utiliza el argumento \Conc, el número de instrucciones de movimiento seguidas está limitado a 5. En secciones de programa que incluyen StorePath-RestoPath, no se permite el uso de instrucciones con el argumento \Conc.

Si se omite este argumento y ToJointPos no es un punto de paro, la instrucción siguiente se ejecuta algún tiempo antes de que los ejes externos alcancen la zona programada.

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.142 MoveExtJ - Mueve una o varias unidades mecánicas sin TCP

RobotWare - OS

Continuación

Este argumento no puede usarse en los movimientos sincronizados coordinados en un sistema MultiMove.

ToJointPos

*To Joint Position*

Tipo de dato: jointtarget

La posición absoluta de destino de los ejes externos. Se define como una posición con nombre o se almacena directamente en la instrucción (marcada con un asterisco \* en la instrucción).

[ \ID ]

*Synchronization ID*

Tipo de dato: identno

El argumento [ \ID ] es obligatorio en los sistemas MultiMove, si el movimiento es sincronizado o sincronizado coordinado. Este argumento no está permitido en ningún otro caso. El número de ID especificado debe ser el mismo en todas las tareas de programa que cooperan entre sí. Al usar el número de ID los movimientos no se mezclan en tiempo de ejecución.

[ \UseEOffs ]

*Use External Offset*

Tipo de dato: switch

El offset de los ejes externos, configurado por la instrucción EOffsSet, se activa para la instrucción MoveExtJ cuando se utiliza el argumento UseEOffs. Consulte la instrucción EOffsSet para obtener más información acerca del offset externo.

Speed

Tipo de dato: speeddata

Los datos de velocidad que se aplican a los movimientos. Los datos de velocidad definen la velocidad del eje externo lineal o de rotación.

[ \T ]

*Time*

Tipo de dato: num

Este argumento se utiliza para especificar el tiempo total en segundos que dura el movimiento de los ejes externos. A continuación, se sustituye por los datos de velocidad correspondientes.

Zone

Tipo de dato: zonedata

Los datos de zona del movimiento. Los datos de zona definen un punto de paro o un punto de paso. En un punto de paso, el tamaño de la zona describe la deceleración y aceleración de los ejes externos lineales o de rotación.

[ \Inpos ]

*In position*

Tipo de dato: stoppointdata

*Continúa en la página siguiente*

## 1.142 MoveExtJ - Mueve una o varias unidades mecánicas sin TCP

RobotWare - OS

Continuación

Este argumento se utiliza para especificar los criterios de convergencia para la posición del eje externo en el punto de paro. Los datos de puntos de paro sustituyen a la zona especificada en el parámetro Zone.

### Ejecución de programas

Los ejes externos lineales o de rotación se mueven hasta el punto programado a la velocidad programada.

### Limitaciones

MoveExtJ no puede ejecutarse en un gestor UNDO o en rutinas de RAPID que estén conectadas a los siguientes eventos especiales del sistema: PowerOn, Stop, QStop, Restart, Reset o Step.

### Más ejemplos

```

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;

```

En este ejemplo, el eje de rotación sencillo se mueve hasta las posiciones de eje 0, 30, 60 y 90 grados a la velocidad de 45 grados/s.

### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema ERRNO cambiará a:

| Nombre       | Causa del error  |
|--------------|--|
| ERR_CONC_MAX | Se ha superado el número de instrucciones de movimiento seguidas con el argumento \Conc. |

### Sintaxis

```

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 > ] ';'

```

Continúa en la página siguiente

# 1 Instrucciones

---

## 1.142 MoveExtJ - Mueve una o varias unidades mecánicas sin TCP

RobotWare - OS

Continuación

---

### Información relacionada

| Para obtener más información sobre     | Consulte   |
|--|--|
| Otras instrucciones de posicionamiento | <i>Manual de referencia técnica - RAPID Overview</i>                     |
| Definición de jointtarget              | <a href="#">jointtarget - Datos de posición de eje en la página 1691</a> |
| Definición de velocidad                | <a href="#">speeddata - Datos de velocidad en la página 1767</a>         |
| Definición de datos de zona            | <a href="#">zonedata - Datos de zonas en la página 1831</a>              |
| Movimiento en general                  | <i>Manual de referencia técnica - RAPID Overview</i>                     |
| Ejecución simultánea de programas      | <i>Manual de referencia técnica - RAPID Overview</i>                     |



---

## 1.143 MoveJ - Mueve el robot mediante un movimiento de ejes

### Utilización

MoveJ se utiliza para mover el robot rápidamente de un punto a otro cuando no es imprescindible que el movimiento siga una línea recta.

Los ejes del robot y los ejes externos se desplazan hasta la posición de destino a lo largo de una trayectoria no lineal. Todos los ejes alcanzan la posición de destino al mismo tiempo.

Esta instrucción sólo puede usarse en la tarea principal T\_ROB1 o, si se cuenta con un sistema *MultiMove*, en las tareas de movimiento.

---

### Ejemplos básicos

Los ejemplos siguientes ilustran la instrucción MoveJ.

Consulte también [Más ejemplos en la página 424](#).

#### Ejemplo 1

```
MoveJ p1, vmax, z30, tool2;
```

El punto central de la herramienta (TCP), tool2, se mueve siguiendo una trayectoria no lineal hacia la posición p1, con los datos de velocidad vmax y los datos de zona z30.

#### Ejemplo 2

```
MoveJ *, vmax \T:=5, fine, grip3;
```

El TCP de la herramienta, grip3, se mueve siguiendo una trayectoria no lineal hacia un punto de paro almacenado en la instrucción (marcado con un asterisco \*). Todo el movimiento requiere 5 segundos.

---

### Argumentos

```
MoveJ [\Conc] ToPoint [\ID] Speed [\V] | [\T] Zone [\Z] [\Inpos]
      Tool [\WObj] [\TLoad]
```

[ \Conc ]

#### *Concurrent*

Tipo de dato: switch

Distintas instrucciones consecutivas se ejecutan mientras el robot está en movimiento. Este argumento no se utiliza normalmente, pero puede utilizarse para evitar los paros no deseados, causados por la sobrecarga de la CPU al utilizar puntos de paso. Esto resulta útil cuando los puntos programados están muy cercanos entre sí y se trabaja a velocidades elevadas. Este argumento también resulta útil si, por ejemplo, no se requiere la comunicación con equipos externos ni la sincronización entre los equipos externos y los movimientos del robot.

Cuando se utiliza el argumento \Conc, el número de instrucciones de movimiento seguidas está limitado a 5. En secciones de programa que incluyen StorePath-RestoPath, no se permite el uso de instrucciones con el argumento \Conc.

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.143 MoveJ - Mueve el robot mediante un movimiento de ejes

RobotWare - OS

Continuación

Si se omite este argumento y `ToPoint` no es un punto de paro, la instrucción siguiente se ejecuta algún tiempo antes de que el robot alcance la zona programada.

Este argumento no puede usarse en los movimientos sincronizados coordinados en un sistema MultiMove.

`ToPoint`

Tipo de dato: `robtarget`

El punto de destino de los ejes del robot y de los ejes externos. Se define como una posición con nombre o se almacena directamente en la instrucción (marcada con un asterisco \* en la instrucción).

[ `\ID` ]

*Synchronization id*

Tipo de dato: `identno`

Este argumento debe utilizarse en los sistemas MultiMove si el movimiento está sincronizado y coordinado y no está permitido en ningún otro caso.

El número de ID especificado debe ser el mismo en todas las tareas de programa que cooperan entre sí. El número de ID constituye una garantía de que los movimientos no se mezclen en tiempo de ejecución.

`Speed`

Tipo de dato: `speeddata`

Los datos de velocidad que se aplican a los movimientos. Los datos de velocidad definen la velocidad del punto central de la herramienta, la reorientación de la misma y los ejes externos.

[ `\V` ]

*Velocity*

Tipo de dato: `num`

Este argumento se utiliza para especificar la velocidad del TCP en mm/s directamente en la instrucción. A continuación, se sustituye por la velocidad correspondiente, especificada en los datos de velocidad.

[ `\T` ]

*Time*

Tipo de dato: `num`

Este argumento se utiliza para especificar el tiempo total en segundos que dura el movimiento del robot. A continuación, se sustituye por los datos de velocidad correspondientes.

`Zone`

Tipo de dato: `zonedata`

Los datos de zona del movimiento. Los datos de zona describen el tamaño de la trayectoria de esquina generada.

[ `\Z` ]

*Zone*

Continúa en la página siguiente

Tipo de dato: num

Este argumento se utiliza para especificar la exactitud de la posición del TCP del robot, directamente en la instrucción. La longitud de la trayectoria de esquina se indica en mm y es sustituida por la zona correspondiente especificada en los datos de zona.

[ \Inpos ]

#### *In position*

Tipo de dato: stoppointdata

Este argumento se utiliza para especificar los criterios de convergencia para la posición del TCP del robot en el punto de paro. Los datos de puntos de paro sustituyen a la zona especificada en el parámetro Zone.

Tool

Tipo de dato: tooldata

La herramienta en uso durante el movimiento del robot. El punto central de la herramienta es el punto que se mueve hacia el punto de destino especificado.

[ \Wobj ]

#### *Work Object*

Tipo de dato: wobjdata

El objeto de trabajo (sistema de coordenadas) con el que está relacionada la posición de robot indicada en la instrucción.

Es posible omitir este argumento. Si se omite, la posición depende del sistema de coordenadas mundo. Si, por otro lado, se utiliza un TCP estacionario o ejes externos coordinados, es necesario especificar este argumento.

[ \TLoad ]

#### *Total load*

Tipo de dato: loaddata

El argumento \TLoad describe la carga total usada durante el movimiento. La carga total es la carga de la herramienta más la carga útil transportada por la herramienta. Si se utiliza el argumento \TLoad, no se tiene en cuenta el valor de loaddata en los tooldata actuales.

Si el argumento \TLoad tiene el valor load0, el argumento \TLoad no se tiene en cuenta y se utilizan en su lugar los loaddata de los tooldata.

Para poder utilizar el argumento \TLoad, es necesario cambiar el valor del parámetro de sistema ModalPayLoadMode a 0. Si ModalPayLoadMode tiene el valor 0, ya no es posible utilizar la instrucción GripLoad.

La carga total puede identificarse con la rutina de servicio LoadIdentify. Si el parámetro de sistema ModalPayLoadMode tiene el valor 0, el operador tiene la posibilidad de copiar los loaddata de la herramienta a una variable persistente loaddata existente o nueva al ejecutar la rutina de servicio.

Es posible realizar una ejecución de prueba del programa sin ninguna carga útil utilizando una señal de entrada digital conectada a la entrada de sistema SimMode (modo simulado). Si la señal de entrada digital tiene el valor 1, los loaddata del

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.143 MoveJ - Mueve el robot mediante un movimiento de ejes

RobotWare - OS

Continuación

argumento opcional `\TLoad` no se tienen en cuenta y se utilizan en su lugar los `loaddata` de los `tooldata` actuales.



### Nota

La funcionalidad predeterminada de manejo de la carga útil es utilizar la instrucción `GripLoad`. Por tanto, el valor predeterminado del parámetro de sistema `ModalPayLoadMode` es 1.

---

## Ejecución de programas

El punto central de la herramienta se mueve hacia el punto de destino mediante la interpolación de los ángulos de los ejes. Esto significa que cada eje se mueve a una velocidad constante y que todos los ejes alcanzan al mismo tiempo el punto de destino, lo que da lugar a una trayectoria no lineal.

En términos generales, el TCP se traslada a la velocidad programada aproximada (independientemente de si los ejes externos están coordinados). La herramienta se reorienta y los ejes externos se mueven al mismo tiempo que se mueve el TCP. Si no es posible alcanzar la velocidad programada para la orientación o para los ejes externos, se reduce la velocidad del TCP.

Normalmente se generan trayectorias de esquina cuando se transfiere el movimiento a la siguiente sección de la trayectoria. Si se especifica un punto de paro en los datos de zona, la ejecución del programa sólo continúa una vez que los ejes del robot y los ejes externos han alcanzado la posición adecuada.

---

## Limitaciones

MoveJ no puede ejecutarse en un gestor UNDO o en rutinas de RAPID que estén conectadas a los siguientes eventos especiales del sistema: PowerOn, Stop, QStop, Restart, Reset o Step.

---

## Más ejemplos

A continuación aparecen más ejemplos de cómo usar la instrucción `MoveJ`.

### Ejemplo 1

```
MoveJ *, v2000\V:=2200, z40 \Z:=45, grip3;
```

El TCP de la herramienta, `grip3`, se mueve siguiendo una trayectoria no lineal hacia una posición almacenada en la instrucción. El movimiento se realiza con los datos establecidos en `v2000` y `z40`. La velocidad y el tamaño de zona del TCP son 2200 mm/s y 45 mm, respectivamente.

### Ejemplo 2

```
MoveJ p5, v2000, fine \Inpos := inpos50, grip3;
```

El TCP de la herramienta, `grip3`, se mueve siguiendo una trayectoria no lineal hacia un punto de paro `p5`. El robot considera que se encuentra en el punto cuando se satisface el 50% de la condición de posición y el 50% de la condición de velocidad de un punto de paro `fine`. Espera al menos 2 segundos a que se satisfagan las condiciones. Consulte los datos predefinidos `inpos50` del tipo de dato `stoppointdata`.

Continúa en la página siguiente

## 1.143 MoveJ - Mueve el robot mediante un movimiento de ejes

RobotWare - OS

Continuación

## Ejemplo 3

```
MoveJ \Conc, *, v2000, z40, grip3;
```

El TCP de la herramienta, `grip3`, se mueve siguiendo una trayectoria no lineal hacia una posición almacenada en la instrucción. Las instrucciones lógicas posteriores se ejecutan mientras el robot está en movimiento.

## Ejemplo 4

```
MoveJ start, v2000, z40, grip3 \Wobj:=fixture;
```

El TCP de la herramienta, `grip3`, se mueve siguiendo una trayectoria no lineal hacia la posición `start`. Esta posición se especifica en el sistema de coordenadas de objeto de `fixture`.

## Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                    | Causa del error  |
|---------------------------|--|
| <code>ERR_CONC_MAX</code> | Se ha superado el número de instrucciones de movimiento seguidas con el argumento <code>\Conc</code> . |

## Sintaxis

```
MoveJ
['\' Conc ',' ]
[ToPoint ':='] < expression (IN) of robtarg >
['\' 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>]
['\' TLoad ':='] <persistent (PERS) of loaddata>|';'
```

## Información relacionada

| Para obtener más información sobre     | Consulte   |
|--|--|
| Otras instrucciones de posicionamiento | <i>Manual de referencia técnica - RAPID Overview</i>                     |
| Definición de carga                    | <a href="#">loaddata - Datos de carga en la página 1694</a>              |
| Definición de velocidad                | <a href="#">speeddata - Datos de velocidad en la página 1767</a>         |
| Definición de datos de punto de paro   | <a href="#">stoppointdata - Datos de punto de paro en la página 1771</a> |
| Definición de herramientas             | <a href="#">tooldata - Datos de herramienta en la página 1795</a>        |
| Definición de objetos de trabajo       | <a href="#">wobjdata - Datos del objeto de trabajo en la página 1823</a> |
| Definición de datos de zona            | <a href="#">zonedata - Datos de zonas en la página 1831</a>              |

Continúa en la página siguiente

# 1 Instrucciones

## 1.143 MoveJ - Mueve el robot mediante un movimiento de ejes

RobotWare - OS

Continuación

| Para obtener más información sobre  | Consulte   |
|---|--|
| Movimiento en general   | <i>Manual de referencia técnica - RAPID Overview</i>                                     |
| Sistemas de coordenadas   | <i>Manual de referencia técnica - RAPID Overview</i>                                     |
| Ejecución simultánea de programas   | <i>Manual de referencia técnica - RAPID Overview</i>                                     |
| Ejemplo de cómo usar TLoad, carga total.  | <a href="#">MoveL - Mueve el robot siguiendo una trayectoria lineal en la página 447</a> |
| Definición de la carga útil de un robot   | <a href="#">GripLoad - Define la carga útil de un robot en la página 244</a>             |
| LoadIdentify, rutina de servicio de identificación de carga                                   | <i>Manual del operador - IRC5 con FlexPendant</i>  |
| Señal de entrada de sistema SimMode para mover el robot en el modo simulado sin carga útil.   | <i>Manual de referencia técnica - Parámetros del sistema</i>                             |
| Parámetro de sistema ModalPayloadMode para la activación y la desactivación de la carga útil. | <i>Manual de referencia técnica - Parámetros del sistema</i>                             |

1.144 MoveJAO - Mueve el robot mediante el movimiento de los ejes y activa una salida analógica en la esquina  
RobotWare - OS

## 1.144 MoveJAO - Mueve el robot mediante el movimiento de los ejes y activa una salida analógica en la esquina

### Utilización

MoveJAO (*Move Joint Analog Output*) se utiliza para mover el robot rápidamente de un punto a otro cuando no es imprescindible que el movimiento siga una línea recta. La señal analógica de salida especificada se activa en el centro de la trayectoria de esquina.

Los ejes del robot y los ejes externos se desplazan hasta la posición de destino a lo largo de una trayectoria no lineal. Todos los ejes alcanzan la posición de destino al mismo tiempo.

Esta instrucción sólo puede usarse en la tarea principal T\_ROB1 o, si se cuenta con un sistema *MultiMove*, en las tareas de movimiento.

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción MoveJAO:

#### Ejemplo 1

```
MoveJAO p1, vmax, z30, tool2, a01, 1.1;
```

El punto central de la herramienta (TCP), tool2, se mueve siguiendo una trayectoria no lineal hacia la posición p1, con los datos de velocidad vmax y los datos de zona z30. La salida a01 se activa en el centro de la trayectoria de esquina de p1.

### Argumentos

```
MoveJAO ToPoint [\ID] Speed [\T] Zone Tool [\WObj] Signal Value
[\TLoad]
```

ToPoint

Tipo de dato: robtarget

El punto de destino de los ejes del robot y de los ejes externos. Se define como una posición con nombre o se almacena directamente en la instrucción (marcada con un asterisco \* en la instrucción).

[ \ID ]

*Synchronization id*

Tipo de dato: identno

El argumento [ \ID ] es obligatorio en los sistemas MultiMove, si el movimiento es sincronizado o sincronizado coordinado. Este argumento no está permitido en ningún otro caso. El número de ID especificado debe ser el mismo en todas las tareas de programa que cooperan entre sí. Al usar el número de ID los movimientos no se mezclan en tiempo de ejecución.

Speed

Tipo de dato: speeddata

*Continúa en la página siguiente*

# 1 Instrucciones

---

1.144 MoveJAO - Mueve el robot mediante el movimiento de los ejes y activa una salida analógica en la esquina

RobotWare - OS

Continuación

Los datos de velocidad que se aplican a los movimientos. Los datos de velocidad definen la velocidad del punto central de la herramienta, la reorientación de la misma y los ejes externos.

[ \T ]

*Time*

Tipo de dato: num

Este argumento se utiliza para especificar el tiempo total en segundos que dura el movimiento del robot. A continuación, se sustituye por los datos de velocidad correspondientes.

Zone

Tipo de dato: zonedata

Los datos de zona del movimiento. Los datos de zona describen el tamaño de la trayectoria de esquina generada.

Tool

Tipo de dato: tooldata

La herramienta en uso durante el movimiento del robot. El punto central de la herramienta es el punto que se mueve hacia el punto de destino especificado.

[ \Wobj ]

*Work Object*

Tipo de dato: wobjdata

El objeto de trabajo (sistema de coordenadas) con el que está relacionada la posición de robot indicada en la instrucción.

Es posible omitir este argumento. Si se omite, la posición depende del sistema de coordenadas mundo. Si, por otro lado, se utiliza un TCP estacionario o ejes externos coordinados, es necesario especificar este argumento.

Signal

Tipo de dato: signalao

El nombre de la señal analógica de salida que debe cambiar de valor.

Value

Tipo de dato: num

El valor deseado para la señal.

[ \TLoad ]

*Total load*

Tipo de dato: loaddata

El argumento \TLoad describe la carga total usada durante el movimiento. La carga total es la carga de la herramienta más la carga útil transportada por la herramienta. Si se utiliza el argumento \TLoad, no se tiene en cuenta el valor de loaddata en los tooldata actuales.

Si el argumento \TLoad tiene el valor load0, el argumento \TLoad no se tiene en cuenta y se utilizan en su lugar los loaddata de los tooldata.

*Continúa en la página siguiente*



## 1.144 MoveJAO - Mueve el robot mediante el movimiento de los ejes y activa una salida analógica en la esquina

RobotWare - OS

Continuación

Para poder utilizar el argumento `\TLoad`, es necesario cambiar el valor del parámetro de sistema `ModalPayloadMode` a 0. Si `ModalPayloadMode` tiene el valor 0, ya no es posible utilizar la instrucción `GripLoad`.

La carga total puede identificarse con la rutina de servicio `LoadIdentify`. Si el parámetro de sistema `ModalPayloadMode` tiene el valor 0, el operador tiene la posibilidad de copiar los `loaddata` de la herramienta a una variable persistente `loaddata` existente o nueva al ejecutar la rutina de servicio.

Es posible realizar una ejecución de prueba del programa sin ninguna carga útil utilizando una señal de entrada digital conectada a la entrada de sistema `SimMode` (modo simulado). Si la señal de entrada digital tiene el valor 1, los `loaddata` del argumento opcional `\TLoad` no se tienen en cuenta y se utilizan en su lugar los `loaddata` de los `tooldata` actuales.

**Nota**

La funcionalidad predeterminada de manejo de la carga útil es utilizar la instrucción `GripLoad`. Por tanto, el valor predeterminado del parámetro de sistema `ModalPayloadMode` es 1.

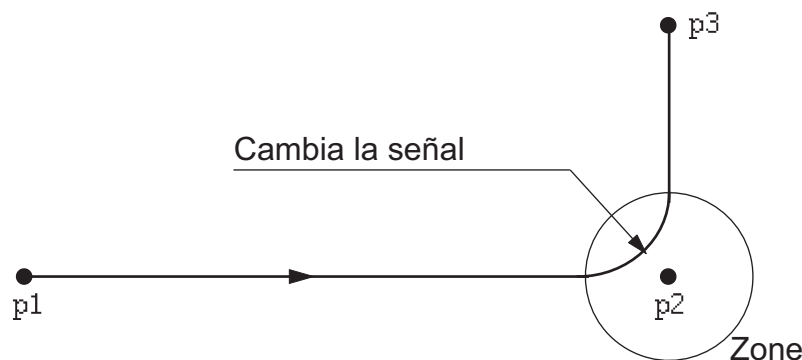
**Ejecución de programas**

Consulte la instrucción `MoveJ` para obtener más información acerca del movimiento de ejes, [MoveJ - Mueve el robot mediante un movimiento de ejes en la página 421](#).

La señal analógica de salida se activa en el centro de la trayectoria de esquina en el caso de los puntos de paso, como se muestra en la figura siguiente.

La figura muestra la activación de una señal analógica de salida en la trayectoria de esquina con `MoveJAO`.

```
MoveJAO p2, vmax, z30, tool2, a01, 1.1;
```



xx1400001118

En el caso de los puntos de paro, recomendamos el uso de una secuencia de programa "normal", con el uso de `MoveJ` y `SetA0`. Sin embargo, cuando se utiliza un punto de paro en la instrucción `MoveJAO`, la señal analógica de salida se activa cuando el robot alcanza el punto de paro.

La señal de E/S especificada se activa en el modo de ejecución continua y paso a paso hacia delante, pero no en la ejecución paso a paso hacia atrás.

Continúa en la página siguiente

# 1 Instrucciones

1.144 MoveJAO - Mueve el robot mediante el movimiento de los ejes y activa una salida analógica en la esquina

RobotWare - OS

Continuación

Limitaciones

MoveJAO no puede ejecutarse en un gestor UNDO o en rutinas de RAPID que estén conectadas a los siguientes eventos especiales del sistema: PowerOn, Stop, QStop, Restart, Reset o Step.

## Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre             | Causa del error   |
|--------------------|---|
| ERR_AO_LIM         | El argumento programado <code>Value</code> para la señal analógica de salida especificada <code>Signal</code> está fuera de límites.  |
| ERR_NO_ALIASIO_DEF | La variable de señal es una variable declarada en RAPID. No se ha conectado a una señal E/S definida en la configuración de E/S con la instrucción <code>AliasIO</code> .                             |
| ERR_NORUNUNIT      | Se ha perdido el contacto con el dispositivo de E/S.  |
| ERR_SIG_NOT_VALID  | La señal de E/S no está disponible. Los motivos pueden ser que el dispositivo de E/S no está en funcionamiento o que exista un error en la configuración (válido solamente para el bus de campo ICI). |

## Sintaxis

```
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>';'
```

## Información relacionada

| Para obtener más información sobre                  | Consulte   |
|---|--|
| Otras instrucciones de posicionamiento              | <i>Manual de referencia técnica - RAPID Overview</i>                                   |
| Movimiento del robot mediante un movimiento de ejes | <a href="#">MoveJ - Mueve el robot mediante un movimiento de ejes en la página 421</a> |
| Definición de carga                                 | <a href="#">loaddata - Datos de carga en la página 1694</a>                            |
| Definición de velocidad                             | <a href="#">speeddata - Datos de velocidad en la página 1767</a>                       |
| Definición de herramientas                          | <a href="#">tooldata - Datos de herramienta en la página 1795</a>                      |
| Definición de objetos de trabajo                    | <a href="#">wobjdata - Datos del objeto de trabajo en la página 1823</a>               |
| Definición de datos de zona                         | <a href="#">zonedata - Datos de zonas en la página 1831</a>                            |
| Movimiento en general                               | <i>Manual de referencia técnica - RAPID Overview</i>                                   |

Continúa en la página siguiente

1.144 MoveJAO - Mueve el robot mediante el movimiento de los ejes y activa una salida analógica en la esquina

RobotWare - OS

Continuación

| Para obtener más información sobre   | Consulte   |
|--|--|
| Sistemas de coordenadas  | <i>Manual de referencia técnica - RAPID Overview</i>                                     |
| Movimientos con parámetros de E/S  | <i>Manual de referencia técnica - RAPID Overview</i>                                     |
| Ejemplo de cómo usar <code>TLoad</code> , carga total.   | <a href="#">MoveL - Mueve el robot siguiendo una trayectoria lineal en la página 447</a> |
| Definición de la carga útil de un robot  | <a href="#">GripLoad - Define la carga útil de un robot en la página 244</a>             |
| LoadIdentify, rutina de servicio de identificación de carga  | <i>Manual del operador - IRC5 con FlexPendant</i>  |
| Señal de entrada de sistema <i>SimMode</i> para mover el robot en el modo simulado sin carga útil.   | <i>Manual de referencia técnica - Parámetros del sistema</i>                             |
| Parámetro de sistema <i>ModalPayloadMode</i> para la activación y la desactivación de la carga útil. | <i>Manual de referencia técnica - Parámetros del sistema</i>                             |

## 1 Instrucciones

---

1.145 MoveJDO - Mueve el robot mediante el movimiento de los ejes y activa una salida digital en la esquina

RobotWare - OS

### 1.145 MoveJDO - Mueve el robot mediante el movimiento de los ejes y activa una salida digital en la esquina

---

#### Utilización

MoveJDO(*Move Joint Digital Output*) se utiliza para mover el robot rápidamente de un punto a otro cuando no es imprescindible que el movimiento siga una línea recta. La señal digital de salida especificada se activa o desactiva en el centro de la trayectoria de esquina.

Los ejes del robot y los ejes externos se desplazan hasta la posición de destino a lo largo de una trayectoria no lineal. Todos los ejes alcanzan la posición de destino al mismo tiempo.

Esta instrucción sólo puede usarse en la tarea principal T\_ROB1 o, si se cuenta con un sistema *MultiMove*, en las tareas de movimiento.

---

#### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción MoveJDO:

##### Ejemplo 1

```
MoveJDO p1, vmax, z30, tool2, do1, 1;
```

El punto central de la herramienta (TCP), tool2, se mueve siguiendo una trayectoria no lineal hacia la posición p1, con los datos de velocidad vmax y los datos de zona z30. La salida do1 se activa en el centro de la trayectoria de esquina de p1.

---

#### Argumentos

```
MoveJDO ToPoint [\ID] Speed [\T] Zone Tool [\WObj] Signal Value  
[\TLoad]
```

ToPoint

Tipo de dato: robtarget

El punto de destino de los ejes del robot y de los ejes externos. Se define como una posición con nombre o se almacena directamente en la instrucción (marcada con un asterisco \* en la instrucción).

[ \ID ]

*Synchronization id*

Tipo de dato: identno

El argumento [ \ID ] es obligatorio en los sistemas MultiMove, si el movimiento es sincronizado o sincronizado coordinado. Este argumento no está permitido en ningún otro caso. El número de ID especificado debe ser el mismo en todas las tareas de programa que cooperan entre sí. Al usar el número de ID los movimientos no se mezclan en tiempo de ejecución.

Speed

Tipo de dato: speeddata

---

*Continúa en la página siguiente*

1.145 MoveJDO - Mueve el robot mediante el movimiento de los ejes y activa una salida digital en la esquina

RobotWare - OS

*Continuación*

Los datos de velocidad que se aplican a los movimientos. Los datos de velocidad definen la velocidad del punto central de la herramienta, la reorientación de la misma y los ejes externos.

[ \T ]

*Time*

Tipo de dato: num

Este argumento se utiliza para especificar el tiempo total en segundos que dura el movimiento del robot. A continuación, se sustituye por los datos de velocidad correspondientes.

Zone

Tipo de dato: zonedata

Los datos de zona del movimiento. Los datos de zona describen el tamaño de la trayectoria de esquina generada.

Tool

Tipo de dato: tooldata

La herramienta en uso durante el movimiento del robot. El punto central de la herramienta es el punto que se mueve hacia el punto de destino especificado.

[ \Wobj ]

*Work Object*

Tipo de dato: wobjdata

El objeto de trabajo (sistema de coordenadas) con el que está relacionada la posición de robot indicada en la instrucción.

Es posible omitir este argumento. Si se omite, la posición depende del sistema de coordenadas mundo. Si, por otro lado, se utiliza un TCP estacionario o ejes externos coordinados, es necesario especificar este argumento.

Signal

Tipo de dato: signaldo

El nombre de la señal digital de salida que debe cambiar de valor.

Value

Tipo de dato: dionum

El valor deseado para la señal (0 ó 1).

[ \TLoad ]

*Total load*

Tipo de dato: loaddata

El argumento \TLoad describe la carga total usada durante el movimiento. La carga total es la carga de la herramienta más la carga útil transportada por la herramienta. Si se utiliza el argumento \TLoad, no se tiene en cuenta el valor de loaddata en los tooldata actuales.

Si el argumento \TLoad tiene el valor load0, el argumento \TLoad no se tiene en cuenta y se utilizan en su lugar los loaddata de los tooldata.

*Continúa en la página siguiente*

# 1 Instrucciones

1.145 MoveJDO - Mueve el robot mediante el movimiento de los ejes y activa una salida digital en la esquina

RobotWare - OS

Continuación

Para poder utilizar el argumento `\TLoad`, es necesario cambiar el valor del parámetro de sistema `ModalPayLoadMode` a 0. Si `ModalPayLoadMode` tiene el valor 0, ya no es posible utilizar la instrucción `GripLoad`.

La carga total puede identificarse con la rutina de servicio `LoadIdentify`. Si el parámetro de sistema `ModalPayLoadMode` tiene el valor 0, el operador tiene la posibilidad de copiar los `loaddata` de la herramienta a una variable persistente `loaddata` existente o nueva al ejecutar la rutina de servicio.

Es posible realizar una ejecución de prueba del programa sin ninguna carga útil utilizando una señal de entrada digital conectada a la entrada de sistema `SimMode` (modo simulado). Si la señal de entrada digital tiene el valor 1, los `loaddata` del argumento opcional `\TLoad` no se tienen en cuenta y se utilizan en su lugar los `loaddata` de los `tooldata` actuales.



## Nota

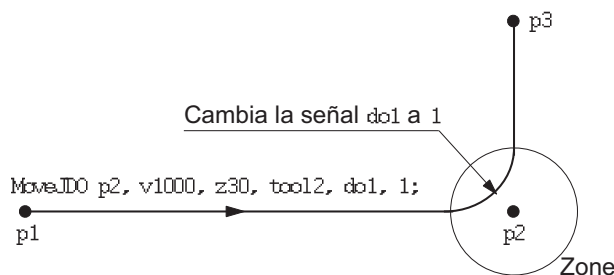
La funcionalidad predeterminada de manejo de la carga útil es utilizar la instrucción `GripLoad`. Por tanto, el valor predeterminado del parámetro de sistema `ModalPayLoadMode` es 1.

## Ejecución de programas

Consulte la instrucción `MoveJ` para obtener más información acerca del movimiento de ejes.

La señal digital de salida se activa o desactiva en el centro de la trayectoria de esquina en el caso de los puntos de paso, como se muestra en la figura siguiente.

La figura muestra la activación/desactivación de una señal digital de salida en la trayectoria de esquina con `MoveJDO`.



xx0500002196

En el caso de los puntos de paro, recomendamos el uso de una secuencia de programa "normal", con el uso de `MoveJ + SetDO`. Sin embargo, cuando se utiliza un punto de paro en la instrucción `MoveJDO`, la señal digital de salida se activa o desactiva cuando el robot alcanza el punto de paro.

La señal de E/S especificada se activa o desactiva en el modo de ejecución continua y paso a paso hacia adelante, pero no en la ejecución paso a paso hacia atrás.

Continúa en la página siguiente

## 1.145 MoveJDO - Mueve el robot mediante el movimiento de los ejes y activa una salida digital en la esquina

RobotWare - OS

Continuación

## Limitaciones

MoveJDO no puede ejecutarse en un gestor UNDO o en rutinas de RAPID que estén conectadas a los siguientes eventos especiales del sistema: PowerOn, Stop, QStop, Restart, Reset o Step.

## Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre             | Causa del error   |
|--------------------|---|
| ERR_NO_ALIASIO_DEF | La variable de señal es una variable declarada en RAPID. No se ha conectado a una señal E/S definida en la configuración de E/S con la instrucción <code>AliasIO</code> .                             |
| ERR_NORUNUNIT      | Se ha perdido el contacto con el dispositivo de E/S.  |
| ERR_SIG_NOT_VALID  | La señal de E/S no está disponible. Los motivos pueden ser que el dispositivo de E/S no está en funcionamiento o que exista un error en la configuración (válido solamente para el bus de campo ICI). |

## Sintaxis

```

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 > ] ';'

```

## Información relacionada

| Para obtener más información sobre                  | Consulte   |
|---|--|
| Otras instrucciones de posicionamiento              | <i>Manual de referencia técnica - RAPID Overview</i>                                   |
| Movimiento del robot mediante un movimiento de ejes | <a href="#">MoveJ - Mueve el robot mediante un movimiento de ejes en la página 421</a> |
| Definición de carga                                 | <a href="#">loaddata - Datos de carga en la página 1694</a>                            |
| Definición de velocidad                             | <a href="#">speeddata - Datos de velocidad en la página 1767</a>                       |
| Definición de herramientas                          | <a href="#">tooldata - Datos de herramienta en la página 1795</a>                      |
| Definición de objetos de trabajo                    | <a href="#">wobjdata - Datos del objeto de trabajo en la página 1823</a>               |
| Definición de datos de zona                         | <a href="#">zonedata - Datos de zonas en la página 1831</a>                            |
| Movimiento en general                               | <i>Manual de referencia técnica - RAPID Overview</i>                                   |
| Sistemas de coordenadas                             | <i>Manual de referencia técnica - RAPID Overview</i>                                   |
| Movimientos con parámetros de E/S                   | <i>Manual de referencia técnica - RAPID Overview</i>                                   |

Continúa en la página siguiente

# 1 Instrucciones

1.145 MoveJDO - Mueve el robot mediante el movimiento de los ejes y activa una salida digital en la esquina

RobotWare - OS

Continuación

| Para obtener más información sobre   | Consulte   |
|--|--|
| Ejemplo de cómo usar <code>TLoad</code> , carga total.   | <a href="#">MoveL</a> - Mueve el robot siguiendo una trayectoria lineal en la página 447 |
| Definición de la carga útil de un robot  | <a href="#">GripLoad</a> - Define la carga útil de un robot en la página 244             |
| <code>LoadIdentify</code> , rutina de servicio de identificación de carga                                  | Manual del operador - IRC5 con FlexPendant   |
| Señal de entrada de sistema <code>SimMode</code> para mover el robot en el modo simulado sin carga útil.   | Manual de referencia técnica - Parámetros del sistema                                    |
| Parámetro de sistema <code>ModalPayloadMode</code> para la activación y la desactivación de la carga útil. | Manual de referencia técnica - Parámetros del sistema                                    |



1.146 MoveJGO - Mueve el robot mediante un movimiento de ejes y establece una señal de salida de grupo en la esquina  
*RobotWare - OS*

## 1.146 MoveJGO - Mueve el robot mediante un movimiento de ejes y establece una señal de salida de grupo en la esquina

### Utilización

MoveJGO (*Move Joint Group Output*) se utiliza para mover el robot rápidamente de un punto a otro cuando no es imprescindible que el movimiento siga una línea recta. La señal de salida de grupo especificada se activa en el centro de la trayectoria de esquina.

Los ejes del robot y los ejes externos se desplazan hasta la posición de destino a lo largo de una trayectoria no lineal. Todos los ejes alcanzan la posición de destino al mismo tiempo.

Esta instrucción sólo puede usarse en la tarea principal T\_ROB1 o, si se cuenta con un sistema *MultiMove*, en las tareas de movimiento.

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción MoveJGO:

#### Ejemplo 1

```
MoveJGO p1, vmax, z30, tool2, go1 \Value:=5;
```

El punto central de la herramienta (TCP), tool2 , se mueve siguiendo una trayectoria no lineal hacia la posición p1, con los datos de velocidad vmax y los datos de zona z30. La señal de salida de grupo go1 se activa en el centro de la trayectoria de esquina de p1.

### Argumentos

```
MoveJGO ToPoint [\ID] Speed [\T] Zone Tool [\WObj] Signal [\Value]  
| [\DValue] [\TLoad]
```

ToPoint

Tipo de dato: robtarget

El punto de destino de los ejes del robot y de los ejes externos. Se define como una posición con nombre o se almacena directamente en la instrucción (marcada con un asterisco \* en la instrucción).

[ \ID ]

*Synchronization id*

Tipo de dato: identno

El argumento [ \ID ] es obligatorio en los sistemas MultiMove, si el movimiento es sincronizado o sincronizado coordinado. Este argumento no está permitido en ningún otro caso. El número de ID especificado debe ser el mismo en todas las tareas de programa que cooperan entre sí. Al usar el número de ID los movimientos no se mezclan en tiempo de ejecución.

Speed

Tipo de dato: speeddata

*Continúa en la página siguiente*

# 1 Instrucciones

---

1.146 MoveJGO - Mueve el robot mediante un movimiento de ejes y establece una señal de salida de grupo en la esquina

*RobotWare - OS*

*Continuación*

Los datos de velocidad que se aplican a los movimientos. Los datos de velocidad definen la velocidad del punto central de la herramienta, la reorientación de la misma y los ejes externos.

[ \T ]

*Time*

Tipo de dato: num

Este argumento se utiliza para especificar el tiempo total en segundos que dura el movimiento del robot. A continuación, se sustituye por los datos de velocidad correspondientes.

Zone

Tipo de dato: zonedata

Los datos de zona del movimiento. Los datos de zona describen el tamaño de la trayectoria de esquina generada.

Tool

Tipo de dato: tooldata

La herramienta en uso durante el movimiento del robot. El punto central de la herramienta es el punto que se mueve hacia el punto de destino especificado.

[ \Wobj ]

*Work Object*

Tipo de dato: wobjdata

El objeto de trabajo (sistema de coordenadas) con el que está relacionada la posición de robot indicada en la instrucción.

Es posible omitir este argumento. Si se omite, la posición depende del sistema de coordenadas mundo. Si, por otro lado, se utiliza un TCP estacionario o ejes externos coordinados, es necesario especificar este argumento.

Signal

Tipo de dato: signalgo

El nombre de la señal de salida de grupo que debe cambiar de valor.

[ \Value ]

Tipo de dato: num

El valor deseado para la señal.

[ \DValue ]

Tipo de dato: dnum

El valor deseado para la señal.

Si no se introduce ninguno de los argumentos \Value ni \DValue, aparece un mensaje de error.

[ \TLoad ]

*Total load*

Tipo de dato: loaddata

*Continúa en la página siguiente*

## 1.146 MoveJGO - Mueve el robot mediante un movimiento de ejes y establece una señal de salida de grupo en la esquina

RobotWare - OS

Continuación

El argumento `\TLoad` describe la carga total usada durante el movimiento. La carga total es la carga de la herramienta más la carga útil transportada por la herramienta. Si se utiliza el argumento `\TLoad`, no se tiene en cuenta el valor de `loaddata` en los `tooldata` actuales.

Si el argumento `\TLoad` tiene el valor `load0`, el argumento `\TLoad` no se tiene en cuenta y se utilizan en su lugar los `loaddata` de los `tooldata`.

Para poder utilizar el argumento `\TLoad`, es necesario cambiar el valor del parámetro de sistema `ModalPayLoadMode` a 0. Si `ModalPayLoadMode` tiene el valor 0, ya no es posible utilizar la instrucción `GripLoad`.

La carga total puede identificarse con la rutina de servicio `LoadIdentify`. Si el parámetro de sistema `ModalPayLoadMode` tiene el valor 0, el operador tiene la posibilidad de copiar los `loaddata` de la herramienta a una variable persistente `loaddata` existente o nueva al ejecutar la rutina de servicio.

Es posible realizar una ejecución de prueba del programa sin ninguna carga útil utilizando una señal de entrada digital conectada a la entrada de sistema `SimMode` (modo simulado). Si la señal de entrada digital tiene el valor 1, los `loaddata` del argumento opcional `\TLoad` no se tienen en cuenta y se utilizan en su lugar los `loaddata` de los `tooldata` actuales.



### Nota

La funcionalidad predeterminada de manejo de la carga útil es utilizar la instrucción `GripLoad`. Por tanto, el valor predeterminado del parámetro de sistema `ModalPayLoadMode` es 1.

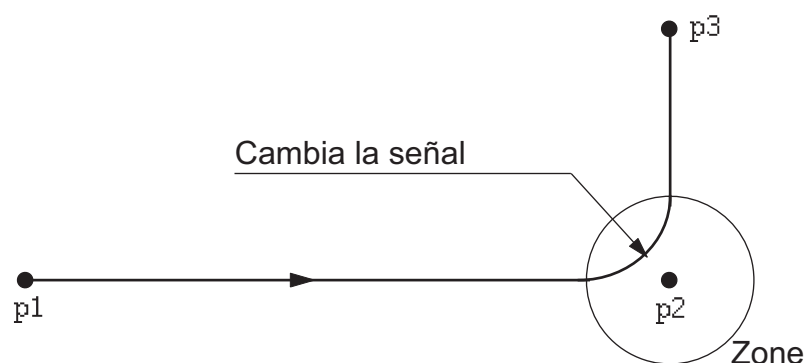
## Ejecución de programas

Consulte la instrucción `MoveJ` para obtener más información acerca del movimiento de ejes.

La señal de salida de grupo se activa en el centro de la trayectoria de esquina en el caso de los puntos de paso, como se muestra en la figura siguiente.

La figura muestra la activación de una señal de salida de grupo en la trayectoria de esquina con `MoveJGO`.

```
MoveJGO p2, vmax, z30, tool2, go1 \Value:=5;
```



xx1400001118

Continúa en la página siguiente

# 1 Instrucciones

1.146 MoveJGO - Mueve el robot mediante un movimiento de ejes y establece una señal de salida de grupo en la esquina

RobotWare - OS

Continuación

En el caso de los puntos de paro, recomendamos el uso de una secuencia de programa "normal", con el uso de MoveJ + SetGO. Sin embargo, cuando se utiliza un punto de paro en la instrucción MoveJGO, la señal de salida de grupo se activa cuando el robot alcanza el punto de paro.

La señal de E/S especificada se activa en el modo de ejecución continua y paso a paso hacia delante, pero no en la ejecución paso a paso hacia atrás.

## Limitaciones

MoveJGO no puede ejecutarse en un gestor UNDO o en rutinas de RAPID que estén conectadas a los siguientes eventos especiales del sistema: PowerOn, Stop, QStop, Restart, Reset o Step.

## Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema ERRNO cambiará a:

| Nombre             | Causa del error   |
|--------------------|---|
| ERR_GO_LIM         | El argumento Value o DValue para la señal de salida de grupo especificada está fuera de límites.  |
| ERR_NO_ALIASIO_DEF | La variable de señal es una variable declarada en RAPID. No se ha conectado a una señal E/S definida en la configuración de E/S con la instrucción AliasIO.   |
| ERR_NORUNUNIT      | se ha perdido el contacto con el dispositivo de E/S.  |
| ERR_SIG_NOT_VALID  | La señal de E/S no está disponible. Los motivos pueden ser que el dispositivo de E/S no está en funcionamiento o que exista un error en la configuración (válido solamente para el bus de campo ICI). |

## Sintaxis

```
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 > ] ';'
```

## Información relacionada

| Para obtener más información sobre                  | Consulte   |
|---|--|
| Otras instrucciones de posicionamiento              | <a href="#">Manual de referencia técnica - RAPID Overview</a>                          |
| Movimiento del robot mediante un movimiento de ejes | <a href="#">MoveJ - Mueve el robot mediante un movimiento de ejes en la página 421</a> |
| Definición de carga                                 | <a href="#">loaddata - Datos de carga en la página 1694</a>                            |

Continúa en la página siguiente

1.146 MoveJGO - Mueve el robot mediante un movimiento de ejes y establece una señal de salida de grupo en la esquina

RobotWare - OS

Continuación

| Para obtener más información sobre  | Consulte   |
|---|--|
| Definición de velocidad   | <a href="#">speeddata - Datos de velocidad en la página 1767</a>                         |
| Definición de herramientas  | <a href="#">tooldata - Datos de herramienta en la página 1795</a>                        |
| Definición de objetos de trabajo  | <a href="#">wobjdata - Datos del objeto de trabajo en la página 1823</a>                 |
| Definición de datos de zona   | <a href="#">zonedata - Datos de zonas en la página 1831</a>                              |
| Movimiento en general   | <i>Manual de referencia técnica - RAPID Overview</i>                                     |
| Sistemas de coordenadas   | <i>Manual de referencia técnica - RAPID Overview</i>                                     |
| Movimientos con parámetros de E/S   | <i>Manual de referencia técnica - RAPID Overview</i>                                     |
| Ejemplo de cómo usar TLoad, carga total.  | <a href="#">MoveL - Mueve el robot siguiendo una trayectoria lineal en la página 447</a> |
| Definición de la carga útil de un robot   | <a href="#">GripLoad - Define la carga útil de un robot en la página 244</a>             |
| LoadIdentify, rutina de servicio de identificación de carga                                   | <i>Manual del operador - IRC5 con FlexPendant</i>  |
| Señal de entrada de sistema SimMode para mover el robot en el modo simulado sin carga útil.   | <i>Manual de referencia técnica - Parámetros del sistema</i>                             |
| Parámetro de sistema ModalPayloadMode para la activación y la desactivación de la carga útil. | <i>Manual de referencia técnica - Parámetros del sistema</i>                             |

# 1 Instrucciones

---

1.147 MoveJSync - Mueve el robot con un movimiento de ejes y ejecuta un procedimiento de RAPID RobotWare - OS

## 1.147 MoveJSync - Mueve el robot con un movimiento de ejes y ejecuta un procedimiento de RAPID

---

### Utilización

MoveJSync (*Move Joint Synchronously*) se utiliza para mover el robot rápidamente de un punto a otro cuando no es imprescindible que el movimiento siga una línea recta. En el centro de la trayectoria de esquina del punto de destino, se ordena la ejecución del procedimiento de RAPID especificado.

Los ejes del robot y los ejes externos se desplazan hasta la posición de destino a lo largo de una trayectoria no lineal. Todos los ejes alcanzan la posición de destino al mismo tiempo.

Esta instrucción sólo puede usarse en la tarea principal T\_ROB1 o, si se cuenta con un sistema *MultiMove*, en las tareas de movimiento.

---

### Ejemplos básicos

Los ejemplos siguientes ilustran la instrucción MoveJSync.

#### Ejemplo 1

```
MoveJSync p1, vmax, z30, tool2, "proc1";
```

El punto central de la herramienta (TCP), tool2, se mueve siguiendo una trayectoria no lineal hacia la posición p1, con los datos de velocidad vmax y los datos de zona z30. El procedimiento proc1 se ejecuta en el centro de la trayectoria de esquina de p1.

#### Ejemplo 2

```
MoveJSync p1, vmax, z30, tool2, "MyModule:proc1";
```

Igual que en el ejemplo 1 anterior, pero en este caso el procedimiento declarado localmente proc1 del módulo MyModule será llamado en medio de la trayectoria de esquina.

---

### Argumentos

```
MoveJSync ToPoint [\ID] Speed [\T] Zone Tool [\WObj] ProcName  
[\TLoad]
```

ToPoint

Tipo de dato: robtarget

El punto de destino de los ejes del robot y de los ejes externos. Se define como una posición con nombre o se almacena directamente en la instrucción (marcada con un asterisco \* en la instrucción).

[ \ID ]

*Synchronization id*

Tipo de dato: identno

El argumento [ \ID ] es obligatorio en los sistemas MultiMove, si el movimiento es sincronizado o sincronizado coordinado. Este argumento no está permitido en ningún otro caso. El número de ID especificado debe ser el mismo en todas las tareas de programa que cooperan entre sí. Al usar el número de ID los movimientos no se mezclan en tiempo de ejecución.

---

*Continúa en la página siguiente*

## 1.147 MoveJSync - Mueve el robot con un movimiento de ejes y ejecuta un procedimiento de RAPID

RobotWare - OS

Continuación

Speed

Tipo de dato: speeddata

Los datos de velocidad que se aplican a los movimientos. Los datos de velocidad definen la velocidad del punto central de la herramienta, la reorientación de la misma y los ejes externos.

[ \T ]

*Time*

Tipo de dato: num

Este argumento se utiliza para especificar el tiempo total en segundos que dura el movimiento del robot. A continuación, se sustituye por los datos de velocidad correspondientes.

Zone

Tipo de dato: zonedata

Los datos de zona del movimiento. Los datos de zona describen el tamaño de la trayectoria de esquina generada.

Tool

Tipo de dato: tooldata

La herramienta en uso durante el movimiento del robot. El punto central de la herramienta es el punto que se mueve hacia el punto de destino especificado.

[ \WObj ]

*Work Object*

Tipo de dato: wobjdata

El objeto de trabajo (sistema de coordenadas) con el que está relacionada la posición de robot indicada en la instrucción.

Es posible omitir este argumento. Si se omite, la posición depende del sistema de coordenadas mundo. Si, por otro lado, se utiliza un TCP estacionario o ejes externos coordinados, es necesario especificar este argumento.

ProcName

*Procedure Name*

Tipo de dato: string

El nombre del procedimiento de RAPID que debe ejecutarse en el centro de la trayectoria de esquina del punto de destino. La llamada al procedimiento es una llamada con enlazamiento en tiempo de ejecución y por tanto hereda sus propiedades.

El procedimiento se ejecutará en el nivel de rutina TRAP (consulte Ejecución de programas más abajo).

[ \TLoad ]

*Total load*

Tipo de dato: loaddata

El argumento \TLoad describe la carga total usada durante el movimiento. La carga total es la carga de la herramienta más la carga útil transportada por la

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.147 MoveJSync - Mueve el robot con un movimiento de ejes y ejecuta un procedimiento de RAPID RobotWare - OS Continuación

herramienta. Si se utiliza el argumento `\TLoad`, no se tiene en cuenta el valor de `loaddata` en los `tooldata` actuales.

Si el argumento `\TLoad` tiene el valor `load0`, el argumento `\TLoad` no se tiene en cuenta y se utilizan en su lugar los `loaddata` de los `tooldata`.

Para poder utilizar el argumento `\TLoad`, es necesario cambiar el valor del parámetro de sistema `ModalPayLoadMode` a 0. Si `ModalPayLoadMode` tiene el valor 0, ya no es posible utilizar la instrucción `GripLoad`.

La carga total puede identificarse con la rutina de servicio `LoadIdentify`. Si el parámetro de sistema `ModalPayLoadMode` tiene el valor 0, el operador tiene la posibilidad de copiar los `loaddata` de la herramienta a una variable persistente `loaddata` existente o nueva al ejecutar la rutina de servicio.

Es posible realizar una ejecución de prueba del programa sin ninguna carga útil utilizando una señal de entrada digital conectada a la entrada de sistema `SimMode` (modo simulado). Si la señal de entrada digital tiene el valor 1, los `loaddata` del argumento opcional `\TLoad` no se tienen en cuenta y se utilizan en su lugar los `loaddata` de los `tooldata` actuales.



### Nota

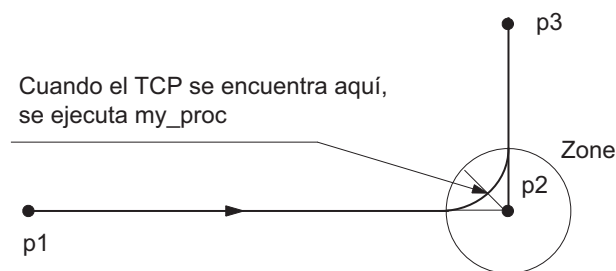
La funcionalidad predeterminada de manejo de la carga útil es utilizar la instrucción `GripLoad`. Por tanto, el valor predeterminado del parámetro de sistema `ModalPayLoadMode` es 1.

## Ejecución de programas

Consulte la instrucción `MoveJ` para obtener más información acerca de los movimientos de ejes.

El procedimiento de RAPID especificado se ejecuta cuando el TCP alcanza el centro de la trayectoria de esquina del punto de destino de la instrucción `MoveJSync`, como se muestra en la figura siguiente debajo.

```
MoveJSync p2, v1000, z30, tool2, "my_proc";
```



xx0500002195

En el caso de los puntos de paro, recomendamos el uso de una secuencia de programa "normal", con el uso de `MoveJ` y otras instrucciones de RAPID una tras otra.

Continúa en la página siguiente



## 1.147 MoveJSync - Mueve el robot con un movimiento de ejes y ejecuta un procedimiento de RAPID

RobotWare - OS

Continuación

En la tabla se describe la ejecución del procedimiento de RAPID especificado, con distintos modos de ejecución:

| Modo de ejecución  | Ejecución del procedimiento de RAPID |
|--------------------|--------------------------------------|
| Continuous o Cycle | De acuerdo con esta descripción      |
| Paso hacia delante | En el punto de paro                  |
| Paso hacia atrás   | Ninguno en absoluto                  |

MoveJSync es una encapsulación de las instrucciones TriggInt y TriggJ. La llamada al procedimiento se ejecuta en el nivel de rutina TRAP.

Si el centro de la trayectoria de esquina del punto de destino se alcanza durante la deceleración posterior a un paro de programa, no se realizará la llamada al procedimiento (la ejecución del programa se para). La llamada al procedimiento se ejecuta en el siguiente inicio de programa.

### Limitaciones

Cuando el robot alcanza el centro de la trayectoria de esquina, suele ser un retardo de entre 2 y 30 ms hasta que se ejecuta la rutina de RAPID especificada, en función del tipo de movimiento que se esté realizando en ese momento.

Si tras un punto de paro se cambia el modo de ejecución del modo continuo o cíclico al modo de ejecución paso a paso hacia adelante o hacia atrás, se genera un error. Este error informa al usuario de que el cambio de modo de ejecución puede dar lugar a que no llegue a ejecutarse un procedimiento de RAPID que está en cola para su ejecución en la trayectoria.

No es posible usar la instrucción MoveJSync en rutinas TRAP. No es posible comprobar el procedimiento de RAPID con la ejecución paso a paso.

MoveJSync no puede ejecutarse en un gestor UNDO o en rutinas de RAPID que estén conectadas a los siguientes eventos especiales del sistema: PowerOn, Stop, QStop, Restart, Reset o Step.

### Sintaxis

```
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 > ] ';'

```

### Información relacionada

| Para obtener más información sobre     | Consulte                                      |
|--|---|
| Otras instrucciones de posicionamiento | Manual de referencia técnica - RAPID Overview |

Continúa en la página siguiente

# 1 Instrucciones

## 1.147 MoveJSync - Mueve el robot con un movimiento de ejes y ejecuta un procedimiento de RAPID RobotWare - OS Continuación

| Para obtener más información sobre  | Consulte  |
|---|---|
| Movimiento del robot mediante un movimiento de ejes   | <a href="#">MoveJ - Mueve el robot mediante un movimiento de ejes en la página 421</a>          |
| Definición de carga   | <a href="#">loaddata - Datos de carga en la página 1694</a>                                     |
| Definición de velocidad   | <a href="#">speeddata - Datos de velocidad en la página 1767</a>                                |
| Definición de herramientas  | <a href="#">tooldata - Datos de herramienta en la página 1795</a>                               |
| Definición de objetos de trabajo  | <a href="#">wobjdata - Datos del objeto de trabajo en la página 1823</a>                        |
| Definición de datos de zona   | <a href="#">zonedata - Datos de zonas en la página 1831</a>                                     |
| Movimiento en general   | <i>Manual de referencia técnica - RAPID Overview</i>  |
| Sistemas de coordenadas   | <i>Manual de referencia técnica - RAPID Overview</i>  |
| Define una interrupción dependiente de una posición   | <a href="#">TriggInt - Define una interrupción dependiente de una posición en la página 921</a> |
| Movimientos de robot eje por eje con eventos  | <a href="#">TriggJ - Movimientos de ejes del robot a partir de eventos en la página 932</a>     |
| Ejemplo de cómo usar TLoad, carga total.  | <a href="#">MoveL - Mueve el robot siguiendo una trayectoria lineal en la página 447</a>        |
| Definición de la carga útil de un robot   | <a href="#">GripLoad - Define la carga útil de un robot en la página 244</a>                    |
| LoadIdentify, rutina de servicio de identificación de carga                                   | <i>Manual del operador - IRC5 con FlexPendant</i>   |
| Señal de entrada de sistema SimMode para mover el robot en el modo simulado sin carga útil.   | <i>Manual de referencia técnica - Parámetros del sistema</i>                                    |
| Parámetro de sistema ModalPayloadMode para la activación y la desactivación de la carga útil. | <i>Manual de referencia técnica - Parámetros del sistema</i>                                    |

## 1.148 MoveL - Mueve el robot siguiendo una trayectoria lineal

### Utilización

MoveL se utiliza para trasladar el punto central de la herramienta (TCP) en sentido lineal hacia un punto de destino determinado. Cuando se desea que el TCP permanezca estacionario, esta instrucción puede usarse también para reorientar la herramienta.

Esta instrucción sólo puede usarse en la tarea principal T\_ROB1 o, si se cuenta con un sistema MultiMove, en las tareas de movimiento.

### Ejemplos básicos

Los ejemplos siguientes ilustran la instrucción MoveL.

Consulte también [Más ejemplos en la página 450](#).

#### Ejemplo 1

```
MoveL p1, v1000, z30, tool2;
```

El TCP de la herramienta, tool2, se mueve a lo largo de una línea hacia la posición p1 con los datos de velocidad v1000 y los datos de zona z30.

#### Ejemplo 2

```
MoveL *, v1000\T:=5, fine, grip3;
```

El TCP de la herramienta, grip3, se mueve linealmente hacia un punto fino almacenado en la instrucción (marcado con un asterisco \*). Todo el movimiento requiere 5 segundos.

### Argumentos

```
MoveL [\Conc] ToPoint [\ID] Speed [\V] | [ \T] Zone [\Z] [\Inpos]
      Tool [\WObj] [\Corr] [\TLoad]
```

[ \Conc ]

#### **Concurrent**

Tipo de dato: switch

Distintas instrucciones consecutivas se ejecutan mientras el robot está en movimiento. Este argumento no se utiliza normalmente, pero puede utilizarse para evitar los paros no deseados, causados por la sobrecarga de la CPU al utilizar puntos de paso. Esto resulta útil cuando los puntos programados están muy cercanos entre sí y se trabaja a velocidades elevadas. Este argumento también resulta útil si, por ejemplo, no se requiere la comunicación con equipos externos ni la sincronización entre los equipos externos y los movimientos del robot.

Cuando se utiliza el argumento \Conc, el número de instrucciones de movimiento seguidas está limitado a 5. En secciones de programa que incluyen StorePath-RestoPath, no se permite el uso de instrucciones con el argumento \Conc.

Si se omite este argumento y ToPoint no es un punto de paro, la instrucción siguiente se ejecuta algún tiempo antes de que el robot alcance la zona programada.

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.148 MoveL - Mueve el robot siguiendo una trayectoria lineal

RobotWare - OS

Continuación

Este argumento no puede usarse en los movimientos sincronizados coordinados en un sistema MultiMove.

ToPoint

Tipo de dato: `robtarget`

El punto de destino de los ejes del robot y de los ejes externos. Se define como una posición con nombre o se almacena directamente en la instrucción (marcada con un asterisco \* en la instrucción).

[ \ID ]

*Synchronization id*

Tipo de dato: `identno`

El argumento [ \ID ] es obligatorio en los sistemas MultiMove, si el movimiento es sincronizado o sincronizado coordinado. Este argumento no está permitido en ningún otro caso. El número de ID especificado debe ser el mismo en todas las tareas de programa que cooperan entre sí. Al usar el número de ID los movimientos no se mezclan en tiempo de ejecución.

Speed

Tipo de dato: `speeddata`

Los datos de velocidad que se aplican a los movimientos. Los datos de velocidad definen la velocidad del punto central de la herramienta, la reorientación de la misma y los ejes externos.

[ \V ]

*Velocity*

Tipo de dato: `num`

Este argumento se utiliza para especificar la velocidad del TCP en mm/s directamente en la instrucción. A continuación, se sustituye por la velocidad correspondiente, especificada en los datos de velocidad.

[ \T ]

*Time*

Tipo de dato: `num`

Este argumento se utiliza para especificar el tiempo total en segundos que dura el movimiento del robot. A continuación, se sustituye por los datos de velocidad correspondientes.

Zone

Tipo de dato: `zonedata`

Los datos de zona del movimiento. Los datos de zona describen el tamaño de la trayectoria de esquina generada.

[ \Z ]

*Zone*

Tipo de dato: `num`

Este argumento se utiliza para especificar la exactitud de la posición del TCP del robot, directamente en la instrucción. La longitud de la trayectoria de esquina se

*Continúa en la página siguiente*

## 1.148 MoveL - Mueve el robot siguiendo una trayectoria lineal

*RobotWare - OS*

*Continuación*

indica en mm y es sustituida por la zona correspondiente especificada en los datos de zona.

[ \Inpos ]

### *In position*

Tipo de dato: stoppointdata

Este argumento se utiliza para especificar los criterios de convergencia para la posición del TCP del robot en el punto de paro. Los datos de puntos de paro sustituyen a la zona especificada en el parámetro *Zone*.

Tool

Tipo de dato: tooldata

La herramienta en uso durante el movimiento del robot. El punto central de la herramienta es el punto que se mueve hacia la posición de destino especificada.

[ \WObj ]

### *Work Object*

Tipo de dato: wobjdata

El objeto de trabajo (sistema de coordenadas) con el que está relacionada la posición de robot indicada en la instrucción.

Es posible omitir este argumento. Si se omite, la posición depende del sistema de coordenadas mundo. Si por otro lado se usa una herramienta estacionaria o ejes externos coordinados, es necesario especificar el argumento para realizar un movimiento lineal respecto del objeto de trabajo.

[ \Corr ]

### *Correction*

Tipo de dato: switch

Los datos de corrección escritos en una entrada de corrección mediante una instrucción *CorrWrite* se añaden a la trayectoria y a la posición de destino si se utiliza este argumento.

Se requiere RobotWare, opción *Path Offset*, cuando se utiliza este argumento.

[ \TLoad ]

### *Total load*

Tipo de dato: loaddata

El argumento *\TLoad* describe la carga total usada durante el movimiento. La carga total es la carga de la herramienta más la carga útil transportada por la herramienta. Si se utiliza el argumento *\TLoad*, no se tiene en cuenta el valor de *loaddata* en los *tooldata* actuales.

Si el argumento *\TLoad* tiene el valor *load0*, el argumento *\TLoad* no se tiene en cuenta y se utilizan en su lugar los *loaddata* de los *tooldata*.

Para poder utilizar el argumento *\TLoad*, es necesario cambiar el valor del parámetro de sistema *ModalPayLoadMode* a 0. Si *ModalPayLoadMode* tiene el valor 0, ya no es posible utilizar la instrucción *GripLoad*.

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.148 MoveL - Mueve el robot siguiendo una trayectoria lineal

RobotWare - OS

Continuación

La carga total puede identificarse con la rutina de servicio LoadIdentify. Si el parámetro de sistema ModalPayloadMode tiene el valor 0, el operador tiene la posibilidad de copiar los loaddata de la herramienta a una variable persistente loaddata existente o nueva al ejecutar la rutina de servicio.

Es posible realizar una ejecución de prueba del programa sin ninguna carga útil utilizando una señal de entrada digital conectada a la entrada de sistema SimMode (modo simulado). Si la señal de entrada digital tiene el valor 1, los loaddata del argumento opcional \TLoad no se tienen en cuenta y se utilizan en su lugar los loaddata de los tooldata actuales.



### Nota

La funcionalidad predeterminada de manejo de la carga útil es utilizar la instrucción GripLoad. Por tanto, el valor predeterminado del parámetro de sistema ModalPayloadMode es 1.

---

## Ejecución de programas

Las unidades del robot y las externas se trasladan hacia la posición de destino de la forma siguiente:

- El TCP de la herramienta se mueve linealmente a una velocidad constante programada.
- La herramienta se reorienta en intervalos iguales a lo largo de la trayectoria.
- Los ejes externos no coordinados se ejecutan a una velocidad constante para que alcancen el punto de destino al mismo tiempo que los ejes del robot.

Si no es posible alcanzar la velocidad programada para la reorientación o para los ejes externos, se reduce la velocidad del TCP.

Normalmente se general trayectorias de esquina cuando se transfiere el movimiento a la siguiente sección de una trayectoria. Si se especifica un punto de paro en los datos de zona, la ejecución del programa sólo continúa una vez que los ejes del robot y los ejes externos han alcanzado la posición adecuada.

---

## Más ejemplos

A continuación aparecen más ejemplos de cómo usar la instrucción MoveL.

### Ejemplo 1

```
MoveL *, v2000 \V:=2200, z40 \Z:=45, grip3;
```

El TCP de la herramienta, grip3, se mueve linealmente hacia una posición almacenada en la instrucción. El movimiento se realiza con los datos establecidos en v2000 y z40. La velocidad y el tamaño de la zona del TCP son de 2200 mm/s y 45 mm respectivamente.

### Ejemplo 2

```
MoveL p5, v2000, fine \Inpos := inpos50, grip3;
```

El TCP de la herramienta, grip3, se mueve linealmente hacia un punto de paro p5. El robot considera que se encuentra en el punto cuando se satisface el 50% de la condición de posición y el 50% de la condición de velocidad de un punto de

Continúa en la página siguiente

paro *fine*. Espera al menos 2 segundos a que se satisfagan las condiciones. Consulte los datos predefinidos *inpos50* del tipo de dato *stoppointdata..*

### Ejemplo 3

```
MoveL \Conc, *, v2000, z40, grip3;
```

El TCP de la herramienta, *grip3*, se mueve linealmente hacia una posición almacenada en la instrucción. Las instrucciones lógicas posteriores se ejecutan mientras el robot está en movimiento.

### Ejemplo 4

```
MoveL start, v2000, z40, grip3 \Wobj:=fixture;
```

El TCP de la herramienta, *grip3*, se mueve linealmente hacia una posición, *start*. Esta posición se especifica en el sistema de coordenadas de objeto de *fixture*.

### Ejemplo con TLoad

```
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;
```

El TCP de la herramienta, *tool2*, se mueve linealmente hasta la posición *p1*, donde se recoge una carga útil. Desde esa posición, el TCP se mueve hasta las posiciones *p2* y *p3* con la carga total *tool2piece*. Los *loaddata* de los *tooldata* actuales no se tienen en cuenta. La carga útil se libera y al moverse hasta la posición *p4*, se vuelve a considerar la carga de la herramienta.

### Limitaciones

*MoveL* no puede ejecutarse en un gestor UNDO o en rutinas de RAPID que estén conectadas a los siguientes eventos especiales del sistema: *PowerOn*, *Stop*, *QStop*, *Restart*, *Reset* o *Step*.

### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema *ERRNO* cambiará a:

| Nombre       | Causa del error  |
|--------------|--|
| ERR_CONC_MAX | Se ha superado el número de instrucciones de movimiento seguidas con el argumento <i>\Conc</i> . |

### Sintaxis

```
MoveL
[ '\ ' Conc ' , ' ]
[ ToPoint ' := ' ] < expression (IN) of robtarg >
[ '\ ' ID ' := ' < expression (IN) of identno > ] ' , '
[ Speed ' := ' ] < expression (IN) of speeddata >
[ '\ ' V ' := ' < expression (IN) of num > ]
| [ '\ ' T ' := ' < expression (IN) of num > ] ' , '

```

Continúa en la página siguiente

# 1 Instrucciones

## 1.148 MoveL - Mueve el robot siguiendo una trayectoria lineal

RobotWare - OS

Continuación

```
[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 > ] ';' ;
```

### Información relacionada

| Para obtener más información sobre   | Consulte   |
|--|--|
| Otras instrucciones de posicionamiento   | <i>Manual de referencia técnica - RAPID Overview</i>                                 |
| Definición de carga  | <a href="#">loaddata - Datos de carga en la página 1694</a>                          |
| Definición de velocidad  | <a href="#">speeddata - Datos de velocidad en la página 1767</a>                     |
| Definición de datos de punto de paro   | <a href="#">stoppointdata - Datos de punto de paro en la página 1771</a>             |
| Definición de herramientas   | <a href="#">tooldata - Datos de herramienta en la página 1795</a>                    |
| Definición de objetos de trabajo   | <a href="#">wobjdata - Datos del objeto de trabajo en la página 1823</a>             |
| Definición de datos de zona  | <a href="#">zonedata - Datos de zonas en la página 1831</a>                          |
| Escritura en una entrada de corrección   | <a href="#">CorrWrite - Escribe en un generador de correcciones en la página 183</a> |
| Movimiento en general  | <i>Manual de referencia técnica - RAPID Overview</i>                                 |
| Sistemas de coordenadas  | <i>Manual de referencia técnica - RAPID Overview</i>                                 |
| Ejecución simultánea de programas  | <i>Manual de referencia técnica - RAPID Overview</i>                                 |
| Definición de la carga útil de un robot  | <a href="#">GripLoad - Define la carga útil de un robot en la página 244</a>         |
| LoadIdentify, rutina de servicio de identificación de carga  | <i>Manual del operador - IRC5 con FlexPendant</i>                                    |
| Señal de entrada de sistema <i>SimMode</i> para mover el robot en el modo simulado sin carga útil.   | <i>Manual de referencia técnica - Parámetros del sistema</i>                         |
| Parámetro de sistema <i>ModalPayloadMode</i> para la activación y la desactivación de la carga útil. | <i>Manual de referencia técnica - Parámetros del sistema</i>                         |
| <i>Path Offset</i>   | <i>Application manual - Controller software IRC5</i>                                 |



1.149 MoveLAO - Mueve el robot siguiendo una trayectoria lineal y establece una salida analógica en la esquina  
*RobotWare - OS*

## 1.149 MoveLAO - Mueve el robot siguiendo una trayectoria lineal y establece una salida analógica en la esquina

### Utilización

MoveLAO (*Move Linearly Analog Output*) se utiliza para trasladar el punto central de la herramienta (TCP) en sentido lineal hacia un punto de destino determinado. La señal analógica de salida especificada se activa en el centro de la trayectoria de esquina.

Cuando se desea que el TCP permanezca estacionario, esta instrucción puede usarse también para reorientar la herramienta.

Esta instrucción sólo puede usarse en la tarea principal T\_ROB1 o, si se cuenta con un sistema *MultiMove*, en las tareas de movimiento.

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción MoveLAO:

#### Ejemplo 1

```
MoveLAO p1, v1000, z30, tool2, ao1, 1.1;
```

El TCP de la herramienta, `tool2`, se mueve a lo largo de una línea hacia la posición `p1` con los datos de velocidad `v1000` y los datos de zona `z30`. La salida `ao1` se activa en el centro de la trayectoria de esquina de `p1`.

### Argumentos

```
MoveLAO ToPoint [\ID] Speed [\T] Zone Tool [\WObj] Signal Value
[\TLoad]
```

#### ToPoint

Tipo de dato: `robtarget`

El punto de destino de los ejes del robot y de los ejes externos. Se define como una posición con nombre o se almacena directamente en la instrucción (marcada con un asterisco \* en la instrucción).

#### [ \ID ]

*Synchronization id*

Tipo de dato: `identno`

El argumento [ \ID ] es obligatorio en los sistemas *MultiMove*, si el movimiento es sincronizado o sincronizado coordinado. Este argumento no está permitido en ningún otro caso. El número de ID especificado debe ser el mismo en todas las tareas de programa que cooperan entre sí. Al usar el número de ID los movimientos no se mezclan en tiempo de ejecución.

#### Speed

Tipo de dato: `speeddata`

Los datos de velocidad que se aplican a los movimientos. Los datos de velocidad definen la velocidad del punto central de la herramienta, la reorientación de la misma y los ejes externos.

*Continúa en la página siguiente*

## 1 Instrucciones

---

1.149 MoveLAO - Mueve el robot siguiendo una trayectoria lineal y establece una salida analógica en la esquina

RobotWare - OS

Continuación

[ \T ]

### *Time*

Tipo de dato: num

Este argumento se utiliza para especificar el tiempo total en segundos que dura el movimiento del robot. A continuación, se sustituye por los datos de velocidad correspondientes.

Zone

Tipo de dato: zonedata

Los datos de zona del movimiento. Los datos de zona describen el tamaño de la trayectoria de esquina generada.

Tool

Tipo de dato: tooldata

La herramienta en uso durante el movimiento del robot. El punto central de la herramienta es el punto que se mueve hacia la posición de destino especificada.

[ \Wobj ]

### *Work Object*

Tipo de dato: wobjdata

El objeto de trabajo (sistema de coordenadas) con el que está relacionada la posición de robot indicada en la instrucción.

Es posible omitir este argumento. Si se omite, la posición depende del sistema de coordenadas mundo. Si, por otro lado, se utiliza un TCP estacionario o ejes externos coordinados, es necesario especificar este argumento.

Signal

Tipo de dato: signalao

El nombre de la señal analógica de salida que debe cambiar de valor.

Value

Tipo de dato: num

El valor deseado para la señal.

[ \TLoad ]

### *Total load*

Tipo de dato: loaddata

El argumento \TLoad describe la carga total usada durante el movimiento. La carga total es la carga de la herramienta más la carga útil transportada por la herramienta. Si se utiliza el argumento \TLoad, no se tiene en cuenta el valor de loaddata en los tooldata actuales.

Si el argumento \TLoad tiene el valor load0, el argumento \TLoad no se tiene en cuenta y se utilizan en su lugar los loaddata de los tooldata.

Para poder utilizar el argumento \TLoad, es necesario cambiar el valor del parámetro de sistema ModalPayLoadMode a 0. Si ModalPayLoadMode tiene el valor 0, ya no es posible utilizar la instrucción GripLoad.

Continúa en la página siguiente

## 1.149 MoveLAO - Mueve el robot siguiendo una trayectoria lineal y establece una salida analógica en la esquina

RobotWare - OS

Continuación

La carga total puede identificarse con la rutina de servicio LoadIdentify. Si el parámetro de sistema ModalPayloadMode tiene el valor 0, el operador tiene la posibilidad de copiar los loaddata de la herramienta a una variable persistente loaddata existente o nueva al ejecutar la rutina de servicio.

Es posible realizar una ejecución de prueba del programa sin ninguna carga útil utilizando una señal de entrada digital conectada a la entrada de sistema SimMode (modo simulado). Si la señal de entrada digital tiene el valor 1, los loaddata del argumento opcional \TLoad no se tienen en cuenta y se utilizan en su lugar los loaddata de los tooldata actuales.



### Nota

La funcionalidad predeterminada de manejo de la carga útil es utilizar la instrucción GripLoad. Por tanto, el valor predeterminado del parámetro de sistema ModalPayloadMode es 1.

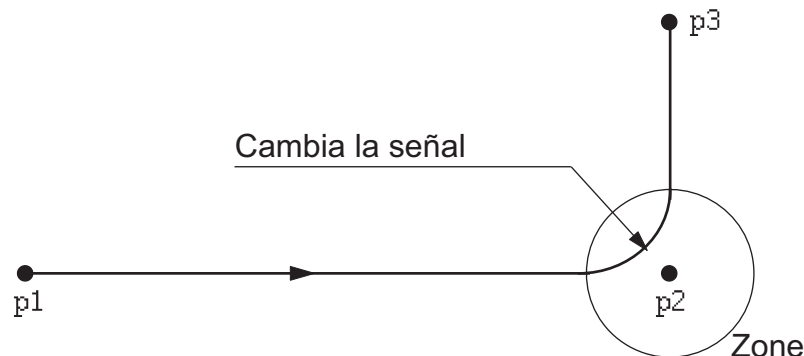
## Ejecución de programas

Consulte la instrucción MoveL para obtener más información acerca de los movimientos lineales.

La señal analógica de salida se activa en el centro de la trayectoria de esquina en el caso de los puntos de paso, como se muestra en la figura siguiente.

La figura muestra la activación de una señal analógica de salida en la trayectoria de esquina con MoveLAO.

```
MoveLAO p2, v1000, z30, tool2, ao1, 1.1;
```



xx1400001118

En el caso de los puntos de paro, recomendamos el uso de una secuencia de programa "normal", con el uso de MoveL y SetAO. Sin embargo, cuando se utiliza un punto de paro en la instrucción MoveLAO, la señal analógica de salida se activa cuando el robot alcanza el punto de paro.

La señal de E/S especificada se activa en el modo de ejecución continua y paso a paso hacia delante, pero no en la ejecución paso a paso hacia atrás.

Continúa en la página siguiente

# 1 Instrucciones

1.149 MoveLAO - Mueve el robot siguiendo una trayectoria lineal y establece una salida analógica en la esquina

RobotWare - OS

Continuación

Limitaciones

MoveLAO no puede ejecutarse en un gestor UNDO o en rutinas de RAPID que estén conectadas a los siguientes eventos especiales del sistema: PowerOn, Stop, QStop, Restart, Reset o Step.

## Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre             | Causa del error   |
|--------------------|---|
| ERR_AO_LIM         | El argumento programado <code>Value</code> para la señal analógica de salida especificada <code>Signal</code> está fuera de límites.  |
| ERR_NO_ALIASIO_DEF | La variable de señal es una variable declarada en RAPID. No se ha conectado a una señal E/S definida en la configuración de E/S con la instrucción <code>AliasIO</code> .                             |
| ERR_NORUNUNIT      | Se ha perdido el contacto con el dispositivo de E/S.  |
| ERR_SIG_NOT_VALID  | La señal de E/S no está disponible. Los motivos pueden ser que el dispositivo de E/S no está en funcionamiento o que exista un error en la configuración (válido solamente para el bus de campo ICI). |

## Sintaxis

```
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 > ] ';'
```

## Información relacionada

| Para obtener más información sobre     | Consulte   |
|--|--|
| Otras instrucciones de posicionamiento | <a href="#">Manual de referencia técnica - RAPID Overview</a>                            |
| Movimiento lineal del robot            | <a href="#">MoveL - Mueve el robot siguiendo una trayectoria lineal en la página 447</a> |
| Definición de carga                    | <a href="#">loaddata - Datos de carga en la página 1694</a>                              |
| Definición de velocidad                | <a href="#">speeddata - Datos de velocidad en la página 1767</a>                         |
| Definición de herramientas             | <a href="#">tooldata - Datos de herramienta en la página 1795</a>                        |
| Definición de objetos de trabajo       | <a href="#">wobjdata - Datos del objeto de trabajo en la página 1823</a>                 |
| Definición de datos de zona            | <a href="#">zonedata - Datos de zonas en la página 1831</a>                              |
| Movimiento en general                  | <a href="#">Manual de referencia técnica - RAPID Overview</a>                            |

Continúa en la página siguiente

1.149 MoveLAO - Mueve el robot siguiendo una trayectoria lineal y establece una salida analógica en la esquina

RobotWare - OS

Continuación

| Para obtener más información sobre   | Consulte   |
|--|--|
| Sistemas de coordenadas  | <i>Manual de referencia técnica - RAPID Overview</i>                                     |
| Movimientos con parámetros de E/S  | <i>Manual de referencia técnica - RAPID Overview</i>                                     |
| Ejemplo de cómo usar <code>TLoad</code> , carga total.   | <a href="#">MoveL - Mueve el robot siguiendo una trayectoria lineal en la página 447</a> |
| Definición de la carga útil de un robot  | <a href="#">GripLoad - Define la carga útil de un robot en la página 244</a>             |
| LoadIdentify, rutina de servicio de identificación de carga  | <i>Manual del operador - IRC5 con FlexPendant</i>  |
| Señal de entrada de sistema <i>SimMode</i> para mover el robot en el modo simulado sin carga útil.   | <i>Manual de referencia técnica - Parámetros del sistema</i>                             |
| Parámetro de sistema <i>ModalPayloadMode</i> para la activación y la desactivación de la carga útil. | <i>Manual de referencia técnica - Parámetros del sistema</i>                             |

## 1 Instrucciones

---

1.150 MoveLDO - Mueve el robot linealmente y establece una salida digital en la esquina  
*RobotWare - OS*

### 1.150 MoveLDO - Mueve el robot linealmente y establece una salida digital en la esquina

---

#### Utilización

MoveLDO (*Move Linearly Digital Output*) se utiliza para trasladar el punto central de la herramienta (TCP) en sentido lineal hacia un punto de destino determinado. La señal digital de salida especificada se activa o desactiva en el centro de la trayectoria de esquina.

Cuando se desea que el TCP permanezca estacionario, esta instrucción puede usarse también para reorientar la herramienta.

Esta instrucción sólo puede usarse en la tarea principal T\_ROB1 o, si se cuenta con un sistema *MultiMove*, en las tareas de movimiento.

#### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción MoveLDO:

##### Ejemplo 1

```
MoveLDO p1, v1000, z30, tool2, do1,1;
```

El TCP de la herramienta, *tool2*, se mueve a lo largo de una línea hacia la posición *p1* con los datos de velocidad *v1000* y los datos de zona *z30*. La salida *do1* se activa en el centro de la trayectoria de esquina de *p1*.

#### Argumentos

```
MoveLDO ToPoint [\ID] Speed [\T] Zone Tool [\WObj] Signal Value  
[\TLoad]
```

ToPoint

Tipo de dato: *robtarget*

El punto de destino de los ejes del robot y de los ejes externos. Se define como una posición con nombre o se almacena directamente en la instrucción (marcada con un asterisco \* en la instrucción).

[ \ID ]

*Synchronization id*

Tipo de dato: *identno*

El argumento [ \ID ] es obligatorio en los sistemas *MultiMove*, si el movimiento es sincronizado o sincronizado coordinado. Este argumento no está permitido en ningún otro caso. El número de ID especificado debe ser el mismo en todas las tareas de programa que cooperan entre sí. Al usar el número de ID los movimientos no se mezclan en tiempo de ejecución.

Speed

Tipo de dato: *speeddata*

Los datos de velocidad que se aplican a los movimientos. Los datos de velocidad definen la velocidad del punto central de la herramienta, la reorientación de la misma y los ejes externos.

*Continúa en la página siguiente*

## 1.150 MoveLDO - Mueve el robot linealmente y establece una salida digital en la esquina *RobotWare - OS* *Continuación*

[ \T ]

### *Time*

Tipo de dato: num

Este argumento se utiliza para especificar el tiempo total en segundos que dura el movimiento del robot. A continuación, se sustituye por los datos de velocidad correspondientes.

Zone

Tipo de dato: zonedata

Los datos de zona del movimiento. Los datos de zona describen el tamaño de la trayectoria de esquina generada.

Tool

Tipo de dato: tooldata

La herramienta en uso durante el movimiento del robot. El punto central de la herramienta es el punto que se mueve hacia la posición de destino especificada.

[ \Wobj ]

### *Work Object*

Tipo de dato: wobjdata

El objeto de trabajo (sistema de coordenadas) con el que está relacionada la posición de robot indicada en la instrucción.

Es posible omitir este argumento. Si se omite, la posición depende del sistema de coordenadas mundo. Si, por otro lado, se utiliza un TCP estacionario o ejes externos coordinados, es necesario especificar este argumento.

Signal

Tipo de dato: signaldo

El nombre de la señal digital de salida que debe cambiar de valor.

Value

Tipo de dato: dionum

El valor deseado para la señal (0 ó 1).

[ \TLoad ]

### *Total load*

Tipo de dato: loaddata

El argumento \TLoad describe la carga total usada durante el movimiento. La carga total es la carga de la herramienta más la carga útil transportada por la herramienta. Si se utiliza el argumento \TLoad, no se tiene en cuenta el valor de loaddata en los tooldata actuales.

Si el argumento \TLoad tiene el valor load0, el argumento \TLoad no se tiene en cuenta y se utilizan en su lugar los loaddata de los tooldata.

Para poder utilizar el argumento \TLoad, es necesario cambiar el valor del parámetro de sistema ModalPayLoadMode a 0. Si ModalPayLoadMode tiene el valor 0, ya no es posible utilizar la instrucción GripLoad.

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.150 MoveLDO - Mueve el robot linealmente y establece una salida digital en la esquina

RobotWare - OS

Continuación

La carga total puede identificarse con la rutina de servicio LoadIdentify. Si el parámetro de sistema ModalPayloadMode tiene el valor 0, el operador tiene la posibilidad de copiar los loaddata de la herramienta a una variable persistente loaddata existente o nueva al ejecutar la rutina de servicio.

Es posible realizar una ejecución de prueba del programa sin ninguna carga útil utilizando una señal de entrada digital conectada a la entrada de sistema SimMode (modo simulado). Si la señal de entrada digital tiene el valor 1, los loaddata del argumento opcional \TLoad no se tienen en cuenta y se utilizan en su lugar los loaddata de los tooldata actuales.



### Nota

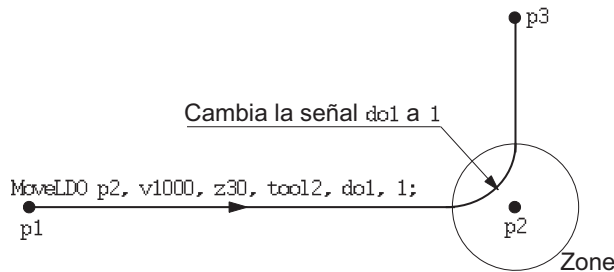
La funcionalidad predeterminada de manejo de la carga útil es utilizar la instrucción GripLoad. Por tanto, el valor predeterminado del parámetro de sistema ModalPayloadMode es 1.

## Ejecución de programas

Consulte la instrucción MoveL para obtener más información acerca de los movimientos lineales.

La señal digital de salida se activa o desactiva en el centro de la trayectoria de esquina en el caso de los puntos de paso, como se muestra en la figura siguiente.

La figura muestra la activación/desactivación de una señal digital de salida en la trayectoria de esquina con MoveLDO.



En el caso de los puntos de paro, recomendamos el uso de una secuencia de programa "normal", con el uso de MoveL y SetDO. Sin embargo, cuando se utiliza un punto de paro en la instrucción MoveLDO, la señal digital de salida se activa o desactiva cuando el robot alcanza el punto de paro.

La señal de E/S especificada se activa o desactiva en el modo de ejecución continua y paso a paso hacia adelante, pero no en la ejecución paso a paso hacia atrás.

## Limitaciones

MoveLDO no puede ejecutarse en un gestor UNDO o en rutinas de RAPID que estén conectadas a los siguientes eventos especiales del sistema: PowerOn, Stop, QStop, Restart, Reset o Step.

Continúa en la página siguiente



## 1.150 MoveLDO - Mueve el robot linealmente y establece una salida digital en la esquina

RobotWare - OS

Continuación

## Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre             | Causa del error   |
|--------------------|---|
| ERR_NO_ALIASIO_DEF | La variable de señal es una variable declarada en RAPID. No se ha conectado a una señal E/S definida en la configuración de E/S con la instrucción <code>AliasIO</code> .                             |
| ERR_NORUNUNIT      | Se ha perdido el contacto con el dispositivo de E/S.  |
| ERR_SIG_NOT_VALID  | La señal de E/S no está disponible. Los motivos pueden ser que el dispositivo de E/S no está en funcionamiento o que exista un error en la configuración (válido solamente para el bus de campo ICI). |

## Sintaxis

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 > ] ';'

```

## Información relacionada

| Para obtener más información sobre                     | Consulte   |
|--|--|
| Otras instrucciones de posicionamiento                 | <i>Manual de referencia técnica - RAPID Overview</i>                                     |
| Movimiento lineal del robot                            | <a href="#">MoveL - Mueve el robot siguiendo una trayectoria lineal en la página 447</a> |
| Definición de carga                                    | <a href="#">loaddata - Datos de carga en la página 1694</a>                              |
| Definición de velocidad                                | <a href="#">speeddata - Datos de velocidad en la página 1767</a>                         |
| Definición de herramientas                             | <a href="#">tooldata - Datos de herramienta en la página 1795</a>                        |
| Definición de objetos de trabajo                       | <a href="#">wobjdata - Datos del objeto de trabajo en la página 1823</a>                 |
| Definición de datos de zona                            | <a href="#">zonedata - Datos de zonas en la página 1831</a>                              |
| Movimiento en general                                  | <i>Manual de referencia técnica - RAPID Overview</i>                                     |
| Sistemas de coordenadas                                | <i>Manual de referencia técnica - RAPID Overview</i>                                     |
| Movimientos con parámetros de E/S                      | <i>Manual de referencia técnica - RAPID Overview</i>                                     |
| Ejemplo de cómo usar <code>TLoad</code> , carga total. | <a href="#">MoveL - Mueve el robot siguiendo una trayectoria lineal en la página 447</a> |
| Definición de la carga útil de un robot                | <a href="#">GripLoad - Define la carga útil de un robot en la página 244</a>             |

Continúa en la página siguiente

# 1 Instrucciones

---

1.150 MoveLDO - Mueve el robot linealmente y establece una salida digital en la esquina

RobotWare - OS

Continuación

| Para obtener más información sobre   | Consulte   |
|--|--|
| LoadIdentify, rutina de servicio de identificación de carga  | <i>Manual del operador - IRC5 con FlexPendant</i>            |
| Señal de entrada de sistema <i>SimMode</i> para mover el robot en el modo simulado sin carga útil.   | <i>Manual de referencia técnica - Parámetros del sistema</i> |
| Parámetro de sistema <i>ModalPayloadMode</i> para la activación y la desactivación de la carga útil. | <i>Manual de referencia técnica - Parámetros del sistema</i> |

## 1.151 MoveLGO - Mueve el robot linealmente y establece una señal de salida de grupo en la esquina

*RobotWare - OS*

### 1.151 MoveLGO - Mueve el robot linealmente y establece una señal de salida de grupo en la esquina

#### Utilización

MoveLGO (*Move Linearly Group Output*) se utiliza para trasladar el punto central de la herramienta (TCP) en sentido lineal hacia un punto de destino determinado. La señal de salida de grupo especificada se activa en el centro de la trayectoria de esquina.

Cuando se desea que el TCP permanezca estacionario, esta instrucción puede usarse también para reorientar la herramienta.

Esta instrucción sólo puede usarse en la tarea principal T\_ROB1 o, si se cuenta con un sistema *MultiMove*, en las tareas de movimiento.

#### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción MoveLGO:

##### Ejemplo 1

```
MoveLGO p1, v1000, z30, tool2, go1 \Value:=5;
```

El TCP de la herramienta, `tool2`, se mueve a lo largo de una línea hacia la posición `p1` con los datos de velocidad `v1000` y los datos de zona `z30`. La señal de salida de grupo `go1` se activa en el centro de la trayectoria de esquina de `p1`.

#### Argumentos

```
MoveLGO ToPoint [\ID] Speed [\T] Zone Tool [\WObj] Signal [\Value]
      | [\DValue] [\TLoad]
```

##### ToPoint

Tipo de dato: `robtarget`

El punto de destino de los ejes del robot y de los ejes externos. Se define como una posición con nombre o se almacena directamente en la instrucción (marcada con un asterisco \* en la instrucción).

##### [ \ID ]

*Synchronization id*

Tipo de dato: `identno`

El argumento [ \ID ] es obligatorio en los sistemas *MultiMove*, si el movimiento es sincronizado o sincronizado coordinado. Este argumento no está permitido en ningún otro caso. El número de ID especificado debe ser el mismo en todas las tareas de programa que cooperan entre sí. Al usar el número de ID los movimientos no se mezclan en tiempo de ejecución.

##### Speed

Tipo de dato: `speeddata`

Los datos de velocidad que se aplican a los movimientos. Los datos de velocidad definen la velocidad del punto central de la herramienta, la reorientación de la misma y los ejes externos.

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.151 MoveLGO - Mueve el robot linealmente y establece una señal de salida de grupo en la esquina *RobotWare - OS*

### *Continuación*

[ \T ]

#### *Time*

Tipo de dato: num

Este argumento se utiliza para especificar el tiempo total en segundos que dura el movimiento del robot. A continuación, se sustituye por los datos de velocidad correspondientes.

Zone

Tipo de dato: zonedata

Los datos de zona del movimiento. Los datos de zona describen el tamaño de la trayectoria de esquina generada.

Tool

Tipo de dato: tooldata

La herramienta en uso durante el movimiento del robot. El punto central de la herramienta es el punto que se mueve hacia la posición de destino especificada.

[ \WObj ]

#### *Work Object*

Tipo de dato: wobjdata

El objeto de trabajo (sistema de coordenadas) con el que está relacionada la posición de robot indicada en la instrucción.

Es posible omitir este argumento. Si se omite, la posición depende del sistema de coordenadas mundo. Si, por otro lado, se utiliza un TCP estacionario o ejes externos coordinados, es necesario especificar este argumento.

Signal

Tipo de dato: signalgo

El nombre de la señal de salida de grupo que debe cambiar de valor.

[ \Value ]

Tipo de dato: num

El valor deseado para la señal.

[ \DValue ]

Tipo de dato: dnum

El valor deseado para la señal.

Si no se introduce ninguno de los argumentos \Value ni \DValue, aparece un mensaje de error.

[ \TLoad ]

#### *Total load*

Tipo de dato: loaddata

El argumento \TLoad describe la carga total usada durante el movimiento. La carga total es la carga de la herramienta más la carga útil transportada por la herramienta. Si se utiliza el argumento \TLoad, no se tiene en cuenta el valor de loaddata en los tooldata actuales.

*Continúa en la página siguiente*

## 1.151 MoveLGO - Mueve el robot linealmente y establece una señal de salida de grupo en la esquina

*RobotWare - OS*  
*Continuación*

Si el argumento `\TLoad` tiene el valor `load0`, el argumento `\TLoad` no se tiene en cuenta y se utilizan en su lugar los `loaddata` de los `tooldata`.

Para poder utilizar el argumento `\TLoad`, es necesario cambiar el valor del parámetro de sistema `ModalPayLoadMode` a 0. Si `ModalPayLoadMode` tiene el valor 0, ya no es posible utilizar la instrucción `GripLoad`.

La carga total puede identificarse con la rutina de servicio `LoadIdentify`. Si el parámetro de sistema `ModalPayLoadMode` tiene el valor 0, el operador tiene la posibilidad de copiar los `loaddata` de la herramienta a una variable persistente `loaddata` existente o nueva al ejecutar la rutina de servicio.

Es posible realizar una ejecución de prueba del programa sin ninguna carga útil utilizando una señal de entrada digital conectada a la entrada de sistema `SimMode` (modo simulado). Si la señal de entrada digital tiene el valor 1, los `loaddata` del argumento opcional `\TLoad` no se tienen en cuenta y se utilizan en su lugar los `loaddata` de los `tooldata` actuales.



### Nota

La funcionalidad predeterminada de manejo de la carga útil es utilizar la instrucción `GripLoad`. Por tanto, el valor predeterminado del parámetro de sistema `ModalPayLoadMode` es 1.

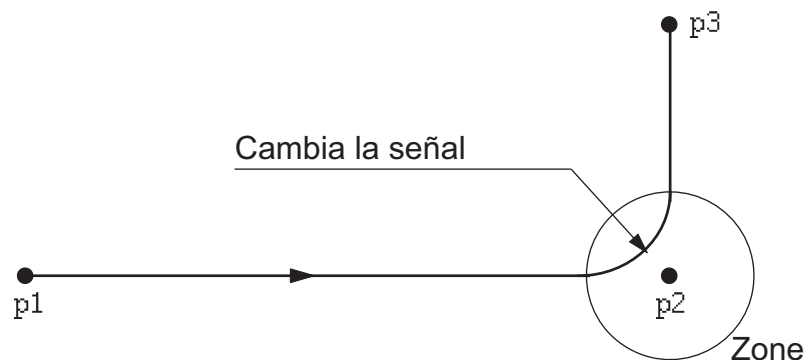
## Ejecución de programas

Consulte la instrucción `MoveL` para obtener más información acerca de los movimientos lineales.

La señal de salida de grupo se activa en el centro de la trayectoria de esquina en el caso de los puntos de paso, como se muestra en la figura siguiente.

La figura muestra la activación de una señal de salida de grupo en la trayectoria de esquina con `MoveLGO`.

```
MoveLGO p2, v1000, z30, tool2, go1 \Value:=5;
```



xx1400001118

En el caso de los puntos de paro, recomendamos el uso de una secuencia de programa "normal", con el uso de `MoveL` + `SetGO`. Sin embargo, cuando se utiliza un punto de paro en la instrucción `MoveLGO`, la señal de salida de grupo se activa cuando el robot alcanza el punto de paro.

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.151 MoveLGO - Mueve el robot linealmente y establece una señal de salida de grupo en la esquina RobotWare - OS Continuación

La señal de E/S especificada se activa en el modo de ejecución continua y paso a paso hacia delante, pero no en la ejecución paso a paso hacia atrás.

### Limitaciones

MoveLGO no puede ejecutarse en un gestor UNDO o en rutinas de RAPID que estén conectadas a los siguientes eventos especiales del sistema: PowerOn, Stop, QStop, Restart, Reset o Step.

### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema ERRNO cambiará a:

| Nombre             | Causa del error   |
|--------------------|---|
| ERR_NO_ALIASIO_DEF | La variable de señal es una variable declarada en RAPID. No se ha conectado a una señal E/S definida en la configuración de E/S con la instrucción AliasIO.   |
| ERR_NORUNUNIT      | Se ha perdido el contacto con el dispositivo de E/S.  |
| ERR_SIG_NOT_VALID  | La señal de E/S no está disponible. Los motivos pueden ser que el dispositivo de E/S no está en funcionamiento o que exista un error en la configuración (válido solamente para el bus de campo ICI). |

### Sintaxis

```
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 > ] ';'
```

### Información relacionada

| Para obtener más información sobre     | Consulte   |
|--|--|
| Otras instrucciones de posicionamiento | <a href="#">Manual de referencia técnica - RAPID Overview</a>                            |
| Movimiento lineal del robot            | <a href="#">MoveL - Mueve el robot siguiendo una trayectoria lineal en la página 447</a> |
| Definición de carga                    | <a href="#">loaddata - Datos de carga en la página 1694</a>                              |
| Definición de velocidad                | <a href="#">speeddata - Datos de velocidad en la página 1767</a>                         |
| Definición de herramientas             | <a href="#">tooldata - Datos de herramienta en la página 1795</a>                        |
| Definición de objetos de trabajo       | <a href="#">wobjdata - Datos del objeto de trabajo en la página 1823</a>                 |
| Definición de datos de zona            | <a href="#">zonedata - Datos de zonas en la página 1831</a>                              |

Continúa en la página siguiente

## 1.151 MoveLGO - Mueve el robot linealmente y establece una señal de salida de grupo en la esquina

RobotWare - OS

Continuación

| Para obtener más información sobre   | Consulte   |
|--|--|
| Movimiento en general  | <i>Manual de referencia técnica - RAPID Overview</i>                                     |
| Sistemas de coordenadas  | <i>Manual de referencia técnica - RAPID Overview</i>                                     |
| Movimientos con parámetros de E/S  | <i>Manual de referencia técnica - RAPID Overview</i>                                     |
| Ejemplo de cómo usar TLoad, carga total.   | <a href="#">MoveL - Mueve el robot siguiendo una trayectoria lineal en la página 447</a> |
| Definición de la carga útil de un robot  | <a href="#">GripLoad - Define la carga útil de un robot en la página 244</a>             |
| LoadIdentify, rutina de servicio de identificación de carga                                    | <i>Manual del operador - IRC5 con FlexPendant</i>  |
| Señal de entrada de sistema SimMode para mover el robot en el modo simulado sin carga útil.    | <i>Manual de referencia técnica - Parámetros del sistema</i>                             |
| Parámetro de sistema ModalPayload-Mode para la activación y la desactivación de la carga útil. | <i>Manual de referencia técnica - Parámetros del sistema</i>                             |

# 1 Instrucciones

---

1.152 MoveLSync - Mueve el robot de forma lineal y ejecuta un procedimiento de RAPID  
RobotWare - OS

## 1.152 MoveLSync - Mueve el robot de forma lineal y ejecuta un procedimiento de RAPID

---

### Utilización

MoveLSync (*Move Linearly Synchronously*) se utiliza para trasladar el punto central de la herramienta (TCP) en sentido lineal hacia un punto de destino determinado. En el centro de la trayectoria de esquina del punto de destino, se ordena la ejecución del procedimiento de RAPID especificado.

Cuando se desea que el TCP permanezca estacionario, esta instrucción puede usarse también para reorientar la herramienta.

Esta instrucción sólo puede usarse en la tarea principal T\_ROB1 o, si se cuenta con un sistema *MultiMove*, en las tareas de movimiento.

### Ejemplos básicos

Los ejemplos siguientes ilustran la instrucción MoveLSync.

#### Ejemplo 1

```
MoveLSync p1, v1000, z30, tool2, "proc1";
```

El TCP de la herramienta, tool2, se mueve a lo largo de una línea hacia la posición p1 con los datos de velocidad v1000 y los datos de zona z30. El procedimiento proc1 se ejecuta en el centro de la trayectoria de esquina de p1.

#### Ejemplo 2

```
MoveLSync p1, v1000, z30, tool2, "proc1";
```

Igual que en el ejemplo 1 anterior, pero en este caso el procedimiento declarado localmente proc1 del módulo MyModule será llamado en medio de la trayectoria de esquina.

### Argumentos

```
MoveLSync ToPoint [\ID] Speed [\T] Zone Tool [\WObj] ProcName  
[\TLoad]
```

ToPoint

Tipo de dato: robtarget

El punto de destino de los ejes del robot y de los ejes externos. Se define como una posición con nombre o se almacena directamente en la instrucción (marcada con un asterisco \* en la instrucción).

[ \ID ]

*Synchronization id*

Tipo de dato: identno

El argumento [ \ID ] es obligatorio en los sistemas MultiMove, si el movimiento es sincronizado o sincronizado coordinado. Este argumento no está permitido en ningún otro caso. El número de ID especificado debe ser el mismo en todas las tareas de programa que cooperan entre sí. Al usar el número de ID los movimientos no se mezclan en tiempo de ejecución.

*Continúa en la página siguiente*



## 1.152 MoveLSync - Mueve el robot de forma lineal y ejecuta un procedimiento de RAPID

RobotWare - OS

Continuación

Speed

Tipo de dato: speeddata

Los datos de velocidad que se aplican a los movimientos. Los datos de velocidad definen la velocidad del punto central de la herramienta, la reorientación de la misma y los ejes externos.

[ \T ]

*Time*

Tipo de dato: num

Este argumento se utiliza para especificar el tiempo total en segundos que dura el movimiento del robot. A continuación, se sustituye por los datos de velocidad correspondientes.

Zone

Tipo de dato: zonedata

Los datos de zona del movimiento. Los datos de zona describen el tamaño de la trayectoria de esquina generada.

Tool

Tipo de dato: tooldata

La herramienta en uso durante el movimiento del robot. El punto central de la herramienta es el punto que se mueve hacia la posición de destino especificada.

[ \WObj ]

*Work Object*

Tipo de dato: wobjdata

El objeto de trabajo (sistema de coordenadas) con el que está relacionada la posición de robot indicada en la instrucción.

Es posible omitir este argumento. Si se omite, la posición depende del sistema de coordenadas mundo. Si, por otro lado, se utiliza un TCP estacionario o ejes externos coordinados, es necesario especificar este argumento.

ProcName

*Procedure Name*

Tipo de dato: string

El nombre del procedimiento de RAPID que debe ejecutarse en el centro de la trayectoria de esquina del punto de destino. La llamada al procedimiento es una llamada con enlazamiento en tiempo de ejecución y por tanto hereda sus propiedades.

El procedimiento se ejecutará en el nivel de rutina TRAP (consulte Ejecución de programas más abajo).

[ \TLoad ]

*Total load*

Tipo de dato: loaddata

El argumento \TLoad describe la carga total usada durante el movimiento. La carga total es la carga de la herramienta más la carga útil transportada por la

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.152 MoveLSync - Mueve el robot de forma lineal y ejecuta un procedimiento de RAPID

RobotWare - OS

Continuación

herramienta. Si se utiliza el argumento `\TLoad`, no se tiene en cuenta el valor de `loaddata` en los `tooldata` actuales.

Si el argumento `\TLoad` tiene el valor `load0`, el argumento `\TLoad` no se tiene en cuenta y se utilizan en su lugar los `loaddata` de los `tooldata`.

Para poder utilizar el argumento `\TLoad`, es necesario cambiar el valor del parámetro de sistema `ModalPayLoadMode` a 0. Si `ModalPayLoadMode` tiene el valor 0, ya no es posible utilizar la instrucción `GripLoad`.

La carga total puede identificarse con la rutina de servicio `LoadIdentify`. Si el parámetro de sistema `ModalPayLoadMode` tiene el valor 0, el operador tiene la posibilidad de copiar los `loaddata` de la herramienta a una variable persistente `loaddata` existente o nueva al ejecutar la rutina de servicio.

Es posible realizar una ejecución de prueba del programa sin ninguna carga útil utilizando una señal de entrada digital conectada a la entrada de sistema `SimMode` (modo simulado). Si la señal de entrada digital tiene el valor 1, los `loaddata` del argumento opcional `\TLoad` no se tienen en cuenta y se utilizan en su lugar los `loaddata` de los `tooldata` actuales.



### Nota

La funcionalidad predeterminada de manejo de la carga útil es utilizar la instrucción `GripLoad`. Por tanto, el valor predeterminado del parámetro de sistema `ModalPayLoadMode` es 1.

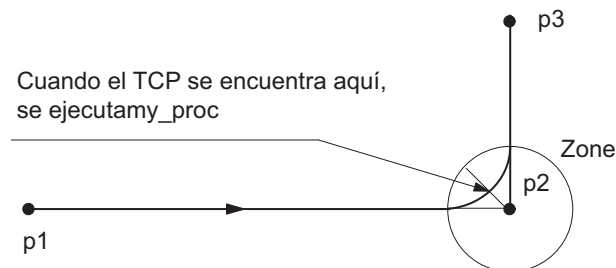
## Ejecución de programas

Consulte la instrucción `MoveL` para obtener más información acerca de los movimientos lineales.

El procedimiento de RAPID especificado se ejecuta cuando el TCP alcanza el centro de la trayectoria de esquina del punto de destino de la instrucción `MoveLSync`, como se muestra en la figura siguiente.

La figura muestra el orden de ejecución del procedimiento de RAPID definido por el usuario al llegar al centro de la trayectoria de esquina.

```
MoveLSync p2, v1000, z30, tool2, "my_proc";
```



xx0500002194

En el caso de los puntos de paro, recomendamos el uso de una secuencia de programa "normal", con el uso de `MoveL` y otras instrucciones de RAPID en secuencia.

Continúa en la página siguiente

## 1.152 MoveLSync - Mueve el robot de forma lineal y ejecuta un procedimiento de RAPID

RobotWare - OS

Continuación

En la tabla se describe la ejecución del procedimiento de RAPID especificado, con distintos modos de ejecución:

| Modo de ejecución: | Ejecución del procedimiento de RAPID: |
|--------------------|---------------------------------------|
| Continuous o Cycle | De acuerdo con esta descripción       |
| Paso hacia delante | En el punto de paro                   |
| Paso hacia atrás   | Ninguno en absoluto                   |

MoveLSync es una encapsulación de las instrucciones TriggInt y TriggL. La llamada al procedimiento se ejecuta en el nivel de rutina TRAP.

Si el centro de la trayectoria de esquina del punto de destino se alcanza durante la deceleración posterior a un paro de programa, no se realizará la llamada al procedimiento (la ejecución del programa se para). La llamada al procedimiento se ejecuta en el siguiente inicio de programa.

### Limitaciones

Cuando el robot alcanza el centro de la trayectoria de esquina, suele ser un retardo de entre 2 y 30 ms hasta que se ejecuta la rutina de RAPID especificada, en función del tipo de movimiento que se esté realizando en ese momento.

Si tras un punto de paro se cambia el modo de ejecución del modo continuo o cíclico al modo de ejecución paso a paso hacia adelante o hacia atrás, se genera un error. Este error informa al usuario de que el cambio de modo de ejecución puede dar lugar a que no llegue a ejecutarse un procedimiento de RAPID que está en cola para su ejecución en la trayectoria.

No es posible usar la instrucción MoveLSync en rutinas TRAP. No es posible comprobar el procedimiento de RAPID con la ejecución paso a paso.

MoveLSync no puede ejecutarse en un gestor UNDO o en rutinas de RAPID que estén conectadas a los siguientes eventos especiales del sistema: PowerOn, Stop, QStop, Restart, Reset o Step.

### Sintaxis

```
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 > ] ';'
```

### Información relacionada

| Para obtener más información sobre     | Consulte   |
|--|--|
| Otras instrucciones de posicionamiento | <i>Manual de referencia técnica - RAPID Overview</i>                                     |
| Movimiento lineal del robot            | <a href="#">MoveL - Mueve el robot siguiendo una trayectoria lineal en la página 447</a> |

Continúa en la página siguiente

# 1 Instrucciones

## 1.152 MoveLSync - Mueve el robot de forma lineal y ejecuta un procedimiento de RAPID

RobotWare - OS

Continuación

| Para obtener más información sobre   | Consulte  |
|--|---|
| Definición de carga  | <a href="#">loaddata - Datos de carga en la página 1694</a>                                     |
| Definición de velocidad  | <a href="#">speeddata - Datos de velocidad en la página 1767</a>                                |
| Definición de herramientas   | <a href="#">tooldata - Datos de herramienta en la página 1795</a>                               |
| Definición de objetos de trabajo   | <a href="#">wobjdata - Datos del objeto de trabajo en la página 1823</a>                        |
| Definición de datos de zona  | <a href="#">zonedata - Datos de zonas en la página 1831</a>                                     |
| Movimiento en general  | <i>Manual de referencia técnica - RAPID Overview</i>  |
| Sistemas de coordenadas  | <i>Manual de referencia técnica - RAPID Overview</i>  |
| Define una interrupción dependiente de una posición  | <a href="#">TriggInt - Define una interrupción dependiente de una posición en la página 921</a> |
| Movimientos de robot lineales con eventos  | <a href="#">TriggL - Movimiento lineal del robot con eventos en la página 940</a>               |
| Ejemplo de cómo usar TLoad, carga total.   | <a href="#">MoveL - Mueve el robot siguiendo una trayectoria lineal en la página 447</a>        |
| Definición de la carga útil de un robot  | <a href="#">GripLoad - Define la carga útil de un robot en la página 244</a>                    |
| LoadIdentify, rutina de servicio de identificación de carga                                    | <i>Manual del operador - IRC5 con FlexPendant</i>   |
| Señal de entrada de sistema SimMode para mover el robot en el modo simulado sin carga útil.    | <i>Manual de referencia técnica - Parámetros del sistema</i>                                    |
| Parámetro de sistema ModalPayload-Mode para la activación y la desactivación de la carga útil. | <i>Manual de referencia técnica - Parámetros del sistema</i>                                    |

1.153 MovePnP: Mueve el robot a lo largo de una trayectoria de elección y colocación.  
SCARA robots

### 1.153 MovePnP: Mueve el robot a lo largo de una trayectoria de elección y colocación.

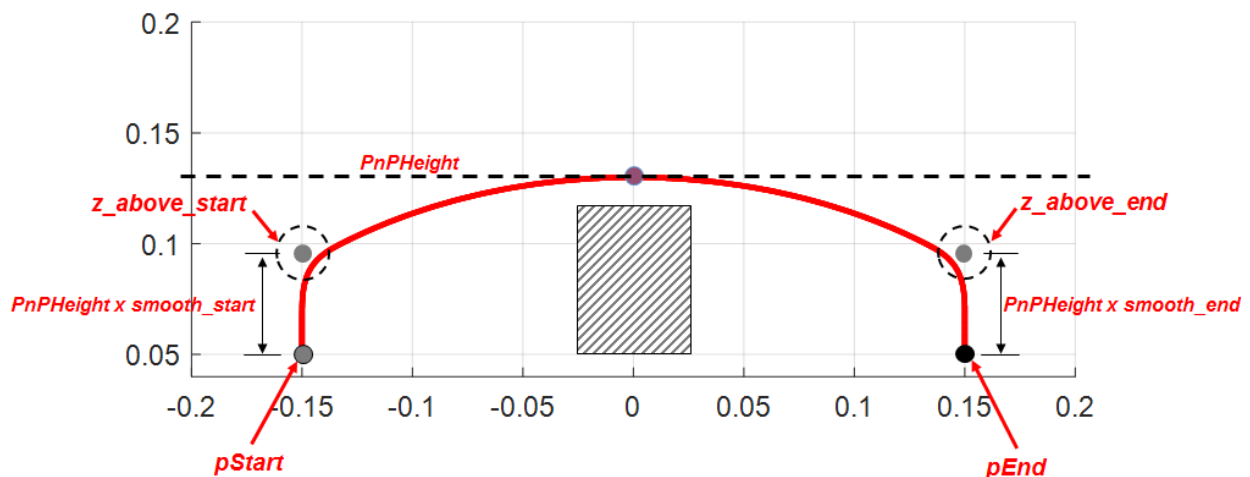
#### Utilización

MovePnP se utiliza para mover el punto central de la herramienta (TCP) rápidamente a lo largo de una trayectoria de elección y colocación según se ilustra en la siguiente imagen.

La trayectoria incluye dos movimientos verticales conectados por un punto superior. La altura del punto superior, así como las alturas de los movimientos verticales, es configurable para hacer que la trayectoria tome distintas formas.

Los distintos tipos de forma son:

- Forma de cinco puntos, como se ve en la siguiente imagen.
- Forma de cuatro puntos, en forma de cuadro.
- Forma de cuatro puntos, en forma de arco.
- Asimétrica, cualquiera de las combinaciones de arriba.



xx1700001194

La trayectoria en el plano horizontal parece un arco cuya curvatura se aparta de un movimiento lineal mediante un parámetro de desplazamiento que no puede

Continúa en la página siguiente

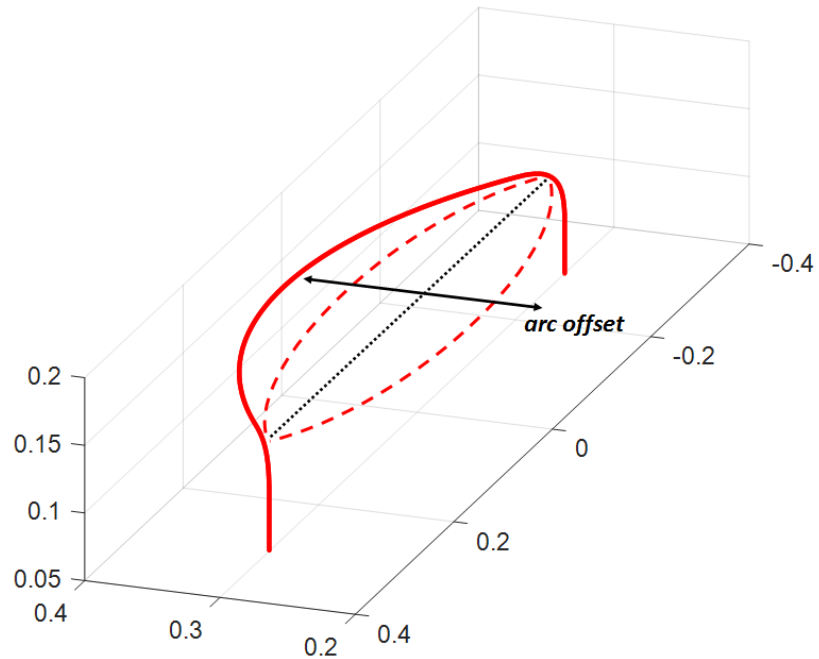
# 1 Instrucciones

1.153 MovePnP: Mueve el robot a lo largo de una trayectoria de elección y colocación.

SCARA robots

Continuación

especificarse explícitamente. Este parámetro se calcula internamente con el objetivo específico de minimizar el tiempo de ciclo.



xx1800002597

## Ejemplos básicos

El ejemplo siguiente ilustra la instrucción MovePnP.

Consulte también [Más ejemplos en la página 479](#).

### Ejemplo 1

```
VAR num my_pnp_height := 130;
VAR pnpdata my_pnpdata

my_pnpdata.smooth_start := 50;
my_pnpdata.smooth_end := 50;
MoveL pStart, v300, fine, tool2;
MovePnP pEnd, v300, \PnPHeight:=my_pnp_height, fine, tool2
      \PnPDataIN:=my_pnpdata;
```

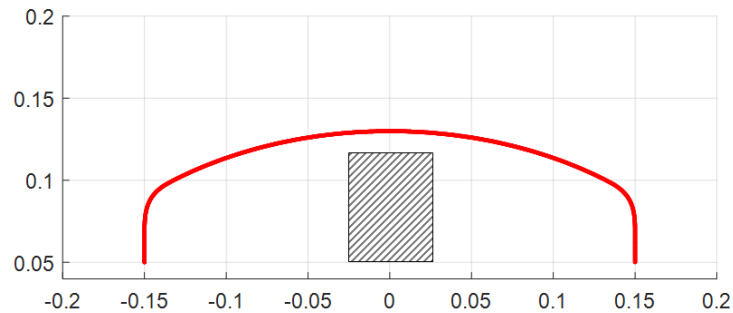
Continúa en la página siguiente

## 1.153 MovePnP: Mueve el robot a lo largo de una trayectoria de elección y colocación.

SCARA robots

Continuación

Trayectoria con forma de cinco puntos. Se configura ajustando los parámetros opcionales `smooth_start` y `smooth_end` a un valor entre 0 y 100.



xx1700001195

### Argumentos

```
MovePnP ToPoint [\ID] Speed [\PnPHeight] Zone [\Inpos] Tool [\WObj]
[\TLoad] [\PnPDataIN] [\SignalIN] [\Value] [\MaxTime]
[\TimeFlag] [\PnPTrigg] [\PnPTriggOption]
```

ToPoint

Tipo de dato: `robtarget`

El punto de destino de los ejes del robot y de los ejes externos. Se define como una posición con nombre o se almacena directamente en la instrucción (marcada con un asterisco \* en la instrucción).

[\ID]

*Synchronization id*

Tipo de dato: `identno`

El argumento `[\ID]` es obligatorio en los sistemas MultiMove si el movimiento es sincronizado o sincronizado coordinado. Este argumento no está permitido en ningún otro caso. El número de ID especificado debe ser el mismo en todas las tareas cooperativas del programa. Al usar el mismo número de ID, los movimientos no se mezclan en tiempo de ejecución.

Speed

Tipo de dato: `speeddata`

Los datos de velocidad que se aplican a los movimientos. Los datos de velocidad definen la velocidad del punto central de la herramienta, la reorientación de la herramienta y los ejes adicionales. Los datos de velocidad no se utilizan para generar la trayectoria óptima. La trayectoria ejecuta con la instrucción `MovePnP` se optimiza para la velocidad máxima del robot.

[\PnPHeight]

Tipo de dato: `num`

La altura del punto superior especificada en mm en coordenadas absolutas (no en relación con el punto de inicio o el punto final) en la base de coordenadas de

*Continúa en la página siguiente*

# 1 Instrucciones

---

1.153 MovePnP: Mueve el robot a lo largo de una trayectoria de elección y colocación.

SCARA robots

Continuación

referencia `ToPoint`. Si el valor no se especifica, la altura se corresponde a la posición superior (cero) del eje 3.



## Nota

La manera de utilizar `PnPHeight` resulta evidente en caso de alturas de elección y colocación diferentes ([Ejemplo 4 en la página 480](#)), ya que `PnPHeight` debe ser independiente de la altura de inicio y la altura final; es decir, no debe especificarse una con respecto a la otra.

Zone

Tipo de dato: `zonedata`

Los datos de zona del movimiento. Los datos de zona describen el tamaño de la trayectoria de esquina generada.

[`\Inpos`]

### *In position*

Tipo de dato: `stoppointdata`

Este argumento se utiliza para especificar los criterios de convergencia para la posición del TCP del robot en el punto de paro. Los datos de puntos de paro sustituyen a la zona especificada en el parámetro `Zone`.

Tool

Tipo de dato: `tooldata`

La herramienta en uso durante el movimiento del robot. El punto central de la herramienta es el punto que se mueve hacia el punto de destino especificado.

[`\Wobj`]

### *Work Object*

Tipo de dato: `wobjdata`

El objeto de trabajo (sistema de coordenadas) con el que está relacionada la posición de robot indicada en la instrucción.

Es posible omitir este argumento. Si se omite, la posición depende del sistema de coordenadas mundo. Si, por otro lado, se utiliza un TCP estacionario o ejes externos coordinados, es necesario especificar este argumento.

[`\TLoad`]

### *Total load*

Tipo de dato: `loaddata`

El argumento `\TLoad` describe la carga total usada durante el movimiento. La carga total es la carga de la herramienta más la carga útil transportada por la herramienta. Si se utiliza el argumento `\TLoad`, no se tiene en cuenta el valor de `loaddata` en los `tooldata` actuales.

Si el argumento `\TLoad` tiene el valor `load0`, el argumento `\TLoad` no se tiene en cuenta y se utilizan en su lugar los `loaddata` de los `tooldata`.

Continúa en la página siguiente



## 1.153 MovePnP: Mueve el robot a lo largo de una trayectoria de elección y colocación.

SCARA robots

Continuación

Para poder utilizar el argumento `\TLoad`, es necesario cambiar el valor del parámetro de sistema `ModalPayLoadMode` a 0. Si `ModalPayLoadMode` tiene el valor 0, ya no es posible utilizar la instrucción `GripLoad`.

La carga total puede identificarse con la rutina de servicio `LoadIdentify`. Si el parámetro de sistema `ModalPayLoadMode` tiene el valor 0, el operador tiene la posibilidad de copiar los `loaddata` de la herramienta a una variable persistente `loaddata` existente o nueva al ejecutar la rutina de servicio.

Es posible realizar una ejecución de prueba del programa sin ninguna carga útil utilizando una señal de entrada digital conectada a la entrada de sistema `SimMode` (modo simulado). Si la señal de entrada digital tiene el valor 1, los `loaddata` del argumento opcional `\TLoad` no se tienen en cuenta y se utilizan en su lugar los `loaddata` de los `tooldata` actuales.

**Nota**

La funcionalidad predeterminada de manejo de la carga útil es utilizar la instrucción `GripLoad`. Por tanto, el valor predeterminado del parámetro de sistema `ModalPayLoadMode` es 1.

[\PnPDataIN]

Tipo de dato: `pnpdata`

`pnpdata` es la estructura de datos utilizada para configurar las trayectorias de elección y colocación. Si no se especifica, la trayectoria se configurará con los valores predeterminados.

**Nota**

Si se declara la estructura pero no se asignan todos los campos, los campos no asignados tendrán valores cero de forma predeterminada.

[\SignalIN]

Tipo de dato: `signaldi`

El nombre de la señal.

[\Value]

Tipo de dato: `dionum`

El valor deseado para la señal.

[\MaxTime]

Tipo de dato: `num`

El periodo máximo permitido para el tiempo de espera, expresado en segundos. Si el tiempo se agota antes de que se cumpla la condición, se llama al gestor de errores si lo hay, con el código de error `ERR_WAIT_MAXTIME`. Si no hay ningún gestor de errores, se detiene la ejecución.

[\TimeFlag]

Tipo de dato: `bool`

*Continúa en la página siguiente*

# 1 Instrucciones

1.153 MovePnP: Mueve el robot a lo largo de una trayectoria de elección y colocación.

SCARA robots

Continuación

El parámetro de salida contiene el valor `TRUE` si se agota el tiempo máximo de espera permitido antes de que se cumpla la condición. Si se incluye este parámetro en la instrucción, no se considera un error si llega a agotarse el tiempo límite. Este argumento no se tiene en cuenta si el `MaxTime` se agota. Este argumento no se tiene en cuenta si el argumento `MaxTime` no se incluye en la instrucción.

`[\PnPTrigg]`

Tipo de dato: `triggdata`

La variable que hace referencia a las condiciones de disparo y la actividad de disparo, definida anteriormente en el programa usando las instrucciones `TriggIO`, `TriggEquip`, `TriggInt`, `TriggCheckIO`, `TriggSpeed` o `TriggRampAO`.



## Nota

La condición de disparo se establece cuando el TCP del robot está en un punto de la trayectoria de acuerdo al parámetro `PnPTriggOption`.

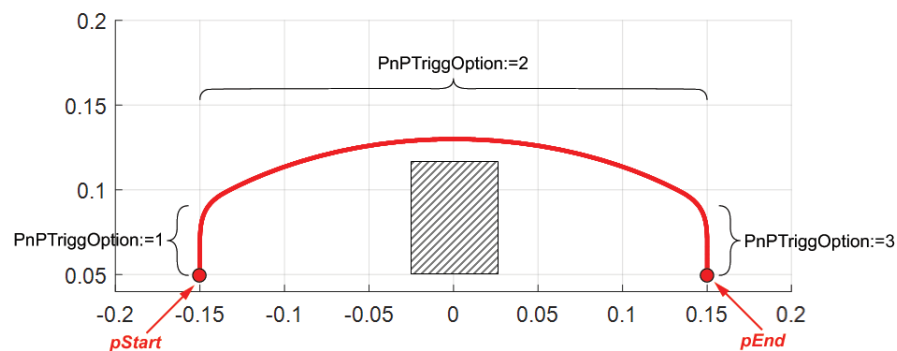
`[\PnPTriggOption]`

Tipo de dato: `num`

Especifica qué parte del movimiento está asociada con las condiciones de disparo especificadas en `PnPTrigg`.

Las opciones válidas son:

| Valor | Descripción  |
|-------|--|
| 1     | El primer movimiento vertical, desde el punto de inicio. |
| 2     | El movimiento horizontal.                                |
| 3     | El segundo movimiento vertical, hacia el punto final.    |



xx1800001532

Figure 1.1:

El valor predeterminado es 2 (es decir, si se omite `\PnPTriggOption`, se especificará el disparo a lo largo el movimiento horizontal).

Continúa en la página siguiente

**Ejecución de programas**

El robot de TCP se traslada a la posición de destino de la siguiente forma:

- 1 El TCP se mueve de forma vertical sobre la posición actual del robot con una distancia especificada por el porcentaje de altura de la trayectoria.
- 2 El TCP alcanza la punta superior computada como el punto medio de la trayectoria con una altura de trayectoria especificada por un parámetro `\PnPHeight`.
- 3 El TCP continúa moviéndose de forma vertical sobre la posición final descrita por la distancia especificada por el porcentaje de altura de la trayectoria.
- 4 El TCP se mueve en vertical hasta la posición final.  
(El parámetro opcional `\SignalIN` se puede utilizar para esperar una señal de entrada digital antes de empezar el movimiento hacia la posición final).
- 5 El argumento opcional `\PnPTrigg` se puede utilizar para sincronizar el movimiento con un dispositivo externo mientras el TCP se desplaza por la trayectoria horizontal o el movimiento vertical (dependiendo del argumento opcional `\PnPTriggOption`).

**Más ejemplos**

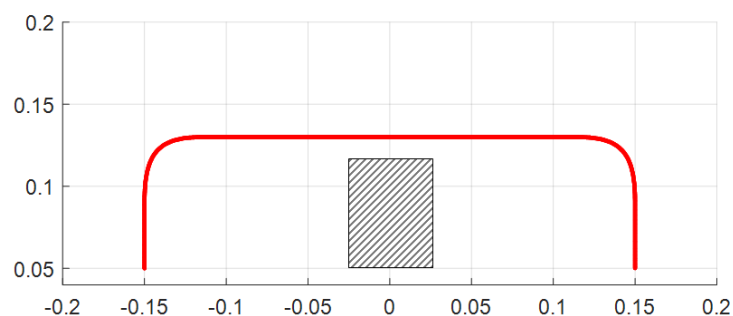
A continuación aparecen más ejemplos de la instrucción `MovePnP`.

**Ejemplo 1**

```
VAR num my_pnp_height := 130;
VAR pnpdata my_pnpdata

my_pnpdata.smooth_start := 100;
my_pnpdata.smooth_end := 100;
MoveL pStart, v300, fine, tool2;
MovePnP pEnd, v300, \PnPHeight:=my_pnp_height, fine, tool2
      \PnPDataIN:=my_pnpdata;
```

Forma de cuatro puntos, forma de cuadro. Está configurada ajustando los parámetros opcionales `smooth_start` y `smooth_end` a 100.



xx1700001196

**Ejemplo 2**

```
VAR num my_pnp_height := 130;
VAR pnpdata my_pnpdata
```

*Continúa en la página siguiente*

# 1 Instrucciones

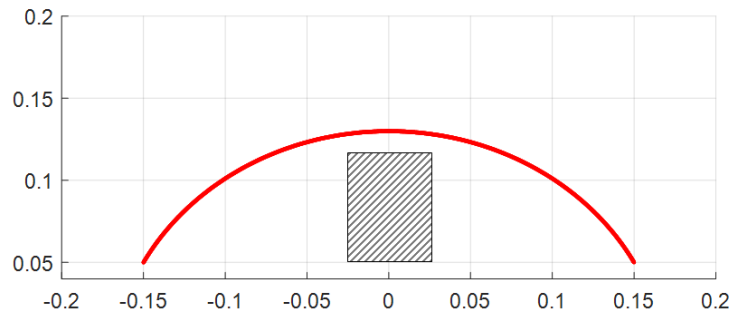
## 1.153 MovePnP: Mueve el robot a lo largo de una trayectoria de elección y colocación.

### SCARA robots

#### Continuación

```
my_pnpdata.smooth_start := 0;  
my_pnpdata.smooth_end := 0;  
MoveL pStart, v300, fine, tool2;  
MovePnP pEnd, v300, \PnPHeight:=my_pnp_height, fine, tool2  
  \PnPDataIN:=my_pnpdata;
```

Forma de tres puntos, forma de arco. Está configurada ajustando los parámetros opcionales `smooth_start` y `smooth_end` a 0.

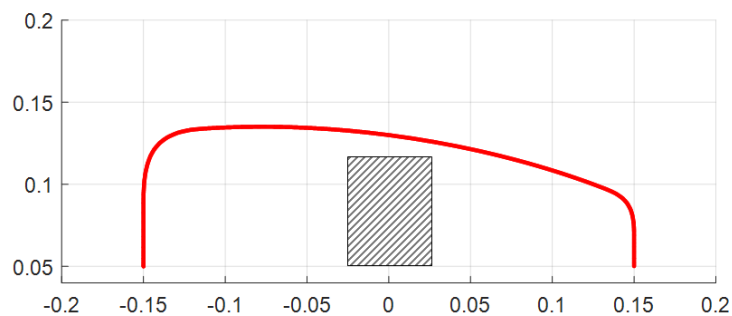


xx1700001197

### Ejemplo 3

```
VAR num my_pnp_height := 130;  
VAR pnpdata my_pnpdata  
  
my_pnpdata.smooth_start := 100;  
my_pnpdata.smooth_end := 50;  
MoveL pStart, v300, fine, tool2;  
MovePnP pEnd, v300, \PnPHeight:=my_pnp_height, fine, tool2  
  \PnPDataIN:=my_pnpdata;
```

Forma asimétrica de cuatro puntos. Está configurada ajustando los parámetros opcionales `smooth_start` y `smooth_end` a diferentes valores.



xx1800000951

### Ejemplo 4

```
VAR num my_pnp_height := 130;  
VAR pnpdata my_pnpdata  
  
my_pnpdata.smooth_start := 100;  
my_pnpdata.smooth_end := 100;
```

Continúa en la página siguiente

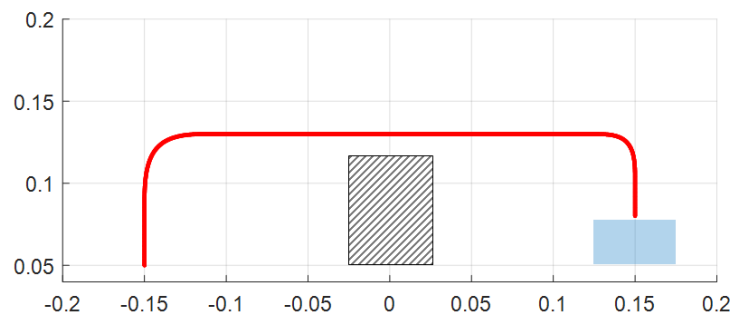
## 1.153 MovePnP: Mueve el robot a lo largo de una trayectoria de elección y colocación.

SCARA robots

Continuación

```
MoveL pStart, v300, fine, tool2;
MovePnP pEnd, v300, \PnPHeight:=my_pnp_height, fine, tool2
\PnPDataIN:=my_pnpdata;
```

Forma de cuatro puntos con diferentes niveles de recogida y colocación. Se configura al establecer los parámetros opcionales `smooth_start` y `smooth_end` hasta 100 (no hay diferencia entre la de recogida y colocación del mismo nivel).



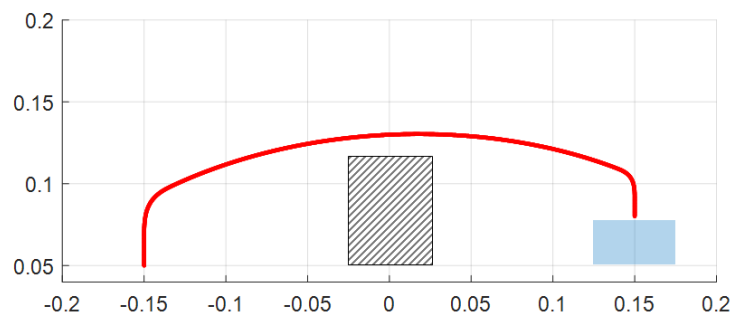
xx180000952

### Ejemplo 5

```
VAR num my_pnp_height := 130;
VAR pnpdata my_pnpdata

my_pnpdata.smooth_start := 50;
my_pnpdata.smooth_end := 50;
MoveL pStart, v300, fine, tool2;
MovePnP pEnd, v300, \PnPHeight:=my_pnp_height, fine, tool2
\PnPDataIN:=my_pnpdata;
```

Forma de cinco puntos con diferentes niveles de recogida y colocación. Recuerde que la forma será asimétrica incluso si se establecen los parámetros `smooth_start` y `smooth_end` al mismo valor.



xx180000953

### Ejemplo 6

```
VAR num my_pnp_height := 130;
VAR pnpdata my_pnpdata
VAR trigdata open_gripper;

my_pnpdata.smooth_start := 50;
```

Continúa en la página siguiente

# 1 Instrucciones

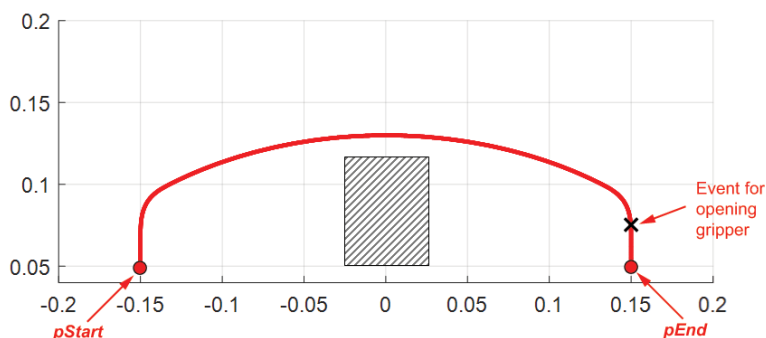
## 1.153 MovePnP: Mueve el robot a lo largo de una trayectoria de elección y colocación.

SCARA robots

Continuación

```
my_pnpdata.smooth_end := 50;
TriggIO open_gripper, 25 \DOp:=doGripper, 0;
MoveL pStart, v300, fine, tool2;
MovePnP pEnd, v300, \PnPHeight:=my_pnp_height, fine, tool2
      \PnPDataIN:=my_pnpdata \PnPTrigg:=open_gripper
      \PnPTriggOption:=3;
```

La señal digital de salida doGripper cambia al valor 0 cuando el TCP se encuentra en una posición en la que la distancia vertical pEnd es 25 mm.



xx1800001531

### Limitaciones

La instrucción MovePnP no es compatible con la ejecución hacia atrás.

La instrucción MovePnP solo se puede aplicar en robots SCARA.

### Sintaxis

```
MovePnP
  [ToPoint ':='] <expression (IN) of robtarger>
  ['\' ID ':='] <expression (IN) of identno>','
  [Speed ':='] <expression (IN) of speeddata>
  ['\' PnPHeight ':='] <expression (IN) of num>','
  [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>
  ['\' PnPDataIN ':='] <expression (IN) of pnpdata>
  ['\' SignalIN ':='] <variable (VAR) of signaldi>
  ['\' Value ':='] <expression (IN) of dionum>
  ['\' MaxTime ':='] <expression (IN) of num>
  ['\' TimeFlag ':='] <variable (VAR) of bool>
  ['\' PnPTrigg ':='] <variable (VAR) of triggdata>
  ['\' PnPTriggOption ':='] <variable (VAR) of num>;'
```

### Información relacionada

| Para obtener más información sobre                         | Consulte  |
|--|---|
| Configuración de las trayectorias de elección y colocación | <a href="#">pnpdata: Configurar las trayectorias de elección y colocación en la página 1727</a> |

---

## 1.154 MToolRotCalib - Calibración de la rotación de una herramienta móvil

---

### Utilización

MToolRotCalib (*Moving Tool Rotation Calibration*) se utiliza para calibrar la rotación de una herramienta móvil.

La posición del robot y sus movimientos dependen siempre de su sistema de coordenadas de herramienta, es decir, del TCP y de la orientación de la herramienta. Para conseguir la máxima exactitud, es importante definir con la mayor corrección posible el sistema de coordenadas de la herramienta.

La calibración también puede realizarse con un método manual, utilizando el FlexPendant, consulte *Manual del operador - IRC5 con FlexPendant*.

---

### Descripción

Para definir la orientación de la herramienta, necesita una punta de referencia fija en el espacio de trabajo del robot.

Antes de usar la instrucción MToolRotCalib, es necesario satisfacer algunas condiciones previas:

- La herramienta que se desea calibrar debe estar montada en el robot y definida con el correspondiente componente `robhold(TRUE)`.
- Si se utiliza el robot con una exactitud total, la carga y el centro de gravedad de la herramienta ya deben estar definidos. Es posible usar `LoadIdentify` para la definición de la carga.
- El valor de TCP de la herramienta debe estar ya definido. La calibración puede hacerse con la instrucción `MToolTCPCalib`.
- `tool0`, `wobj0` y `PDispOff` deben estar activados antes de mover el robot.
- Mueva el TCP de la herramienta actual hasta el lugar más cercano posible de la punta de referencia fija (el origen del sistema de coordenadas de la herramienta) y definir un `jointtarget` para el punto de referencia `RefTip`.
- Mueva el robot sin cambiar la orientación de la herramienta, de forma que la punta de referencia fija apunte hacia algún punto del eje z positivo del sistema de coordenadas de herramienta y defina un `jointtarget` para el punto `ZPos`.
- Opcionalmente, mueva el robot sin cambiar la orientación de la herramienta, de forma que la punta de referencia fija apunte hacia algún punto del eje z positivo del sistema de coordenadas de herramienta y defina un `jointtarget` para el punto `XPos`.

Como ayuda para apuntar hacia el eje x y el eje x positivos, puede usarse algún tipo de herramienta alargadora.

*Continúa en la página siguiente*

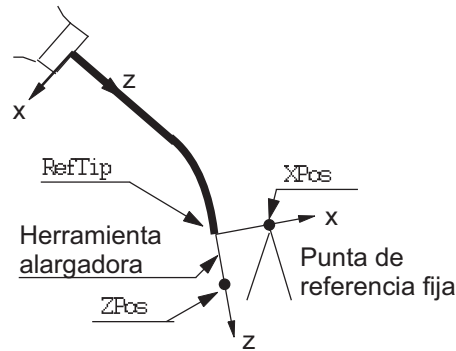
# 1 Instrucciones

## 1.154 MToolRotCalib - Calibración de la rotación de una herramienta móvil

RobotWare - OS

Continuación

Consulte la figura siguiente para obtener una definición del objetivo de ejes para RefTip, ZPos, y opcionalmente XPos.



### Nota

No se recomienda modificar las posiciones de RefTip, ZPos y XPos en la instrucción MToolRotCalib.

## Ejemplos básicos

Los siguientes ejemplos ilustran la función MToolRotCalib:

### Ejemplo 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;
```

Se calcula la orientación de la herramienta (`tframe.rot`) en la dirección z de `tool1`. Las direcciones x e y de la orientación de la herramienta se calculan de forma que coincidan con el sistema de coordenadas de la muñeca.

### Ejemplo 2

```
! Calibration with complete tool orientation
MToolRotCalib pos_tip, pos_z \XPos:=pos_x, tool1;
```

Continúa en la página siguiente



## 1.154 MToolRotCalib - Calibración de la rotación de una herramienta móvil

RobotWare - OS

Continuación

Se calcula la orientación completa (`tframe.rot`) de la herramienta `tool1`.

## Argumentos

```
MToolRotCalib RefTip ZPos [\XPos]Tool
```

RefTip

Tipo de dato: `jointtarget`

El punto al que está apuntando el TCP de la herramienta en la punta fijada a mundo.

ZPos

Tipo de dato: `jointtarget`

El punto de alargador que define la dirección z positiva.

[\XPos]

Tipo de dato: `jointtarget`

El punto de alargador que define la dirección x positiva. Si se omite este punto, las direcciones x e y de la herramienta coincidirán con los ejes correspondientes del sistema de coordenadas de la muñeca.

Tool

Tipo de dato: `tooldata`

La variable persistente de la herramienta a calibrar.

## Ejecución de programas

El sistema calcula y actualiza la orientación de la herramienta (`tframe.rot`) en los datos `tooldata` especificados. El cálculo se basa en los 2 ó 3 `jointtarget` especificados. El resto de los datos de la herramienta, por ejemplo el TCP (`tframe.trans`), permanece sin cambios.

## Sintaxis

```
MToolRotCalib
  [RefTip ':=' ] <expression (IN) of jointtarget>','
  [ZPos ':=' ] <expression (IN) of jointtarget>
  ['\ ' XPos ':=' <expression (IN) of jointtarget>'],'
  [Tool ':=' ] <persistent (PERS) of tooldata>;'
```

## Información relacionada

| Para obtener más información sobre                        | Consulte  |
|---|---|
| Calibración del TCP de una herramienta móvil              | <a href="#">MToolTCPCalib - Calibración del TCP de una herramienta móvil en la página 486</a>                         |
| Calibración del TCP de una herramienta fija               | <a href="#">SToolTCPCalib - Calibración del TCP de una herramienta estacionaria en la página 819</a>                  |
| Calibración del TCP y la rotación de una herramienta fija | <a href="#">SToolRotCalib - Calibración del TCP y de la rotación de una herramienta estacionaria en la página 816</a> |

## 1 Instrucciones

1.155 MToolTCPCalib - Calibración del TCP de una herramienta móvil  
RobotWare - OS

### 1.155 MToolTCPCalib - Calibración del TCP de una herramienta móvil

#### Utilización

MToolTCPCalib (*Moving Tool TCP Calibration*) se usa para calibrar el TCP (punto central de la herramienta) de una herramienta móvil.

La posición del robot y sus movimientos dependen siempre de su sistema de coordenadas de herramienta, es decir, del TCP y de la orientación de la herramienta. Para conseguir la máxima exactitud, es importante definir con la mayor corrección posible el sistema de coordenadas de la herramienta.

La calibración también puede realizarse con un método manual, utilizando el FlexPendant, consulte *Manual del operador - IRC5 con FlexPendant*.

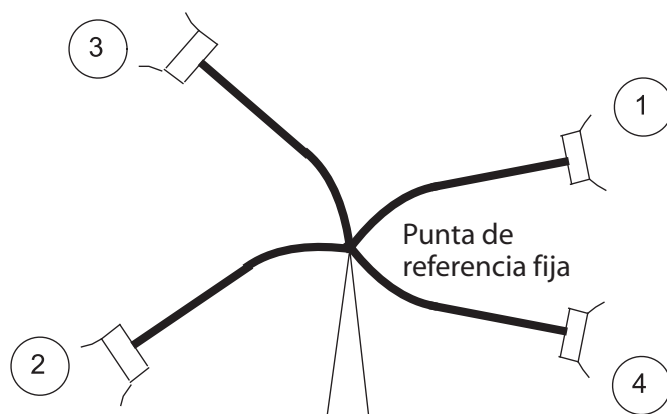
#### Descripción

Para definir el TCP de una herramienta, necesita una punta de referencia fija en el espacio de trabajo del robot.

Antes de usar la instrucción MToolTCPCalib, es necesario satisfacer algunas condiciones previas:

- La herramienta que se desea calibrar debe estar montada en el robot y definida con el correspondiente componente `robhold(TRUE)`.
- Si se utiliza el robot con una exactitud total, la carga y el centro de gravedad de la herramienta ya deben estar definidos. Es posible usar `LoadIdentify` para la definición de la carga.
- `tool0`, `wobj0` y `PDispOff` deben estar activados antes de mover el robot.
- Mueva el TCP de la herramienta actual hasta el lugar más cercano posible de la punta de referencia fija (el origen del sistema de coordenadas de la herramienta) y definir un `jointtarget` para el punto de referencia `RefTip`.
- Defina las tres posiciones adicionales (`p2,p3` y `p4`) todas con orientaciones diferentes.

Definición de 4 `jointtarget`, de `p1` a `p4`. Consulte la figura siguiente.



xx0500002191

Continúa en la página siguiente

**Nota**

No se recomienda modificar las posiciones de Pos1 a Pos4 en la instrucción MToolTCPCalib.

La reorientación entre las 4 posiciones debe ser la mayor posible, poniendo el robot en configuraciones diferentes. También resulta adecuado comprobar la calidad del TCP antes de una calibración. Esto puede realizarse reorientando la herramienta y comprobando si el TCP permanece en la misma posición.

**Ejemplos básicos**

El ejemplo que aparece a continuación ilustra la instrucción MToolTCPCalib:

**Ejemplo 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;
```

Se calibra y actualiza el valor del TCP (`tframe.trans`) de `tool1.max_err` y `mean_err` contendrán el error máximo en mm a partir del TCP calculado y el error medio en mm a partir del TCP calculado, respectivamente.

**Argumentos**

```
MToolTCPCalib Pos1 Pos2 Pos3 Pos4 Tool MaxErr MeanErr
```

Pos1

Tipo de dato: jointtarget

El primer punto de aproximación.

Pos2

Tipo de dato: jointtarget

El segundo punto de aproximación.

Pos3

Tipo de dato: jointtarget

El tercer punto de aproximación.

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.155 MToolTCPCalib - Calibración del TCP de una herramienta móvil

RobotWare - OS

Continuación

Pos4

Tipo de dato: jointtarget

El cuarto punto de aproximación.

Tool

Tipo de dato: tooldata

La variable persistente de la herramienta a calibrar.

MaxErr

Tipo de dato: num

El error máximo en mm para un punto de aproximación.

MeanErr

Tipo de dato: num

La distancia media que separa los puntos de aproximación del TCP calculado, es decir, la exactitud con la que el robot se posicionó respecto de la punta.

### Ejecución de programas

El sistema calcula y actualiza el valor del TCP en el sistema de coordenadas de la muñeca (`tframe.trans`) en el valor especificado `tooldata`. El cálculo se basa en los 4 valores especificados de `jointtarget`. Los demás datos de `tooldata`, por ejemplo la orientación de la herramienta (`tframe.rot`), permanecen sin cambios.

### Sintaxis

```
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>';'
```

### Información relacionada

| Para obtener más información sobre                        | Consulte  |
|---|---|
| Calibración de la rotación de una herramienta móvil       | <a href="#">MToolRotCalib - Calibración de la rotación de una herramienta móvil en la página 483</a>                  |
| Calibración del TCP de una herramienta fija               | <a href="#">SToolTCPCalib - Calibración del TCP de una herramienta estacionaria en la página 819</a>                  |
| Calibración del TCP y la rotación de una herramienta fija | <a href="#">SToolRotCalib - Calibración del TCP y de la rotación de una herramienta estacionaria en la página 816</a> |

## 1.156 Open - Abre un archivo o dispositivo de E/S

### Utilización

Open se utiliza para abrir un archivo o un dispositivo de E/S para su lectura o escritura.

### Ejemplos básicos

Los ejemplos siguientes ilustran la instrucción Open.

Consulte también [Más ejemplos en la página 491](#).

#### Ejemplo 1

```
VAR iodev logfile;
...
Open "HOME:" \File:= "LOGFILE1.DOC", logfile \Write;
```

Se abre el archivo LOGFILE1.DOC de la unidad HOME:, para escritura. El nombre de referencia logfile se utiliza más tarde en el programa al escribir en el archivo.

#### Ejemplo 2

```
VAR iodev logfile;
...
Open "LOGFILE1.DOC", logfile \Write;
```

Mismo resultado que en el ejemplo 1. El directorio predeterminado es HOME:.

### Argumentos

```
Open Object [\File]IODevice [\Read] | [\Write] | [\Append] [\Bin]
```

Object

Tipo de dato: string

El objeto de E/S (dispositivo de E/S) que se desea abrir, por ejemplo "HOME:", "TEMP:", "com1:" o "pc:" (opcional).

En la tabla se describen los distintos dispositivos de E/S del controlador de robot.

| Nombre del dispositivo de E/S   | Tipo de dispositivo de E/S        |
|---|-----------------------------------|
| "HOME:" o diskhome <sup>i</sup>   | Tarjeta SD                        |
| "TEMP:" o disktemp <sup>i</sup>   | Tarjeta SD                        |
| "RemovableDisk1:" o usbdisk1 <sup>i</sup><br>"RemovableDisk2:" o usbdisk2 <sup>i</sup><br>"RemovableDisk3:" o usbdisk3 <sup>i</sup> | Por ejemplo, llave de memoria USB |
| "com1:" <sup>ii</sup>   | Canal serie                       |
| "pc:" <sup>iii</sup>  | Disco montado                     |
| "RAMDISK:" o diskram <sup>i, iv</sup>   | Memoria de disco RAM              |

<sup>i</sup> Cadena de RAPID para definir un nombre del dispositivo.

<sup>ii</sup> Nombre de canal serie definido por el usuario en los parámetros del sistema.

<sup>iii</sup> Protocolo de aplicación, ruta de servidor, definido en los parámetros del sistema.

<sup>iv</sup> La memoria de disco RAM no tiene como fin el almacenamiento permanente de ningún dato. Su tamaño es de alrededor de 100 Mb y se borra con cada apagado.

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.156 Open - Abre un archivo o dispositivo de E/S

RobotWare - OS

Continuación

En la tabla siguiente se describen los distintos dispositivos de E/S del controlador virtual.

| Nombre del dispositivo de E/S  | Tipo de dispositivo de E/S  |
|--|---|
| "HOME:" o diskhome <sup>i</sup>  |   |
| "TEMP:" o disktemp   | Disco duro  |
| "RemovableDisk1:" o usbdisk1<br>"RemovableDisk2:" o usbdisk2<br>"RemovableDisk3:" o usbdisk3 | Por ejemplo, llave de memoria USB   |
| "RAMDISK:" o diskram <sup>i</sup>  | Disco duro<br>..TEMP (apunta a la carpeta TEMP que está situada en la misma carpeta que su sistema virtual) |

<sup>i</sup> Cadena de RAPID para definir un nombre del dispositivo.

[ \File ]

Tipo de dato: string

El nombre del archivo que se desea abrir, por ejemplo "LOGFILE1.DOC" o

"LOGDIR/LOGFILE1.DOC"

La ruta completa también puede especificarse en el argumento Object,

"HOME:/LOGDIR/LOGFILE.DOC".

IODevice

Tipo de dato: iodev

Una referencia al archivo o dispositivo de E/S que se desea abrir. A continuación, esta referencia se utiliza para las operaciones de lectura y escritura del archivo o dispositivo de E/S.

[ \Read ]

Tipo de dato: switch

Abre un archivo o un dispositivo de E/S para lectura. Al leer el archivo, la lectura comienza al principio de éste.

[ \Write ]

Tipo de dato: switch

Abre un archivo o un dispositivo de E/S para escritura. Si el archivo seleccionado ya existe, su contenido se elimina. Cualquier información que se escriba a partir de ese momento se escribe al principio del archivo.

[ \Append ]

Tipo de dato: switch

Abre un archivo o un dispositivo de E/S para escritura. Si el archivo seleccionado ya existe, cualquier información que se escriba a partir de ese momento se escribe al final del archivo.

Para abrir un archivo o un dispositivo de E/S, utilice \Append sin los argumentos \Bin. La instrucción abre un archivo o un dispositivo de E/S alfanumérico para escritura.

Continúa en la página siguiente

Abra un archivo o dispositivo de E/S con los argumentos `\Append` y `\Bin`. La instrucción abre un archivo o un dispositivo de E/S para lectura y escritura. Los argumentos `\Read`, `\Write`, `\Append` son excluyentes entre sí. Si no se especifica ninguno de estos argumentos, la instrucción actúa de la misma forma que el argumento `\Write` en el caso de los archivos o dispositivos de E/S alfanuméricos (instrucción sin el argumento `\Bin`) y de la misma forma que el argumento `\Append` en el caso de los archivos o dispositivos de E/S binarios (instrucción con el argumento `\Bin`).

`[\Bin]`

Tipo de dato: `switch`

El archivo o dispositivo de E/S se abre en el modo binario. Si no se especifica ninguno de los argumentos `\Read`, `\Write` o `\Append`, la instrucción abre un archivo o un dispositivo de E/S binario, tanto para lectura como para escritura, con el puntero del archivo situado en el final de éste.

La instrucción `Rewind` puede usarse para situar el puntero del archivo al principio del archivo, si así se desea.

El conjunto de instrucciones que se usa para el acceso de un archivo o dispositivo de E/S binario es distinto del conjunto de instrucciones que se usa con los archivos alfanuméricos.

## Más ejemplos

A continuación aparecen más ejemplos de cómo usar la instrucción `Open`.

### Ejemplo 1

```
]
VAR iodev printer;
...
Open "com1:", printer \Bin;
WriteStrBin printer, "This is a message to the printer\0D";
Close printer;
```

Se abre el canal serie `com1:` para lectura y escritura binarias. El nombre de referencia `printer` se utiliza más tarde al escribir y cerrar el canal serie.

### Ejemplo 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;
```

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.156 Open - Abre un archivo o dispositivo de E/S

RobotWare - OS

Continuación

```
Close io_device;
```

```
UnpackRawBytes raw_data_in, 1, answer \ISOLatin1Encoding:=10;
```

En este ejemplo, `raw_data_out` se deja sin contenido y se empaqueta con el encabezado de DeviceNet, junto con un valor de coma flotante con el valor 0.2.

Se abre un dispositivo, `"/FCI1/:dsqc328_1"`, y los datos válidos actuales de `raw_data_out` se escriben en el dispositivo. A continuación, el programa espera al menos 1 segundo antes de leer del dispositivo, almacenando en `raw_data_in` la información leída.

Después de cerrar el dispositivo `"/FCI1/:dsqc328_1"`, los datos leídos se desempaquetan dando lugar a una cadena de 10 caracteres que se almacena en la cadena llamada `answer`.

---

## Ejecución de programas

El archivo o dispositivo de E/S especificado se abre de forma que sea posible leer su contenido o escribir en él.

Es posible abrir el mismo archivo físico varias veces simultáneamente, pero cada ejecución de la instrucción `Open` devuelve una referencia distinta al archivo (tipo de dato `iodev`). Por ejemplo, es posible tener un puntero para escribir y otro puntero para leer del mismo archivo de forma simultánea.

La variable `iodev` utilizada al abrir un archivo o dispositivo de E/S debe estar libre para su uso. Si se ha usado anteriormente para abrir un archivo, es necesario cerrar el archivo antes de ejecutar una nueva instrucción `Open` con la misma variable `iodev`.

En caso de un paro de programa y traslado del PP a Main, cualquier archivo o dispositivo de E/S que esté abierto en la tarea de programa se cierra y el descriptor de E/S de la variable del tipo `iodev` se restablece. Una excepción a esta regla la constituyen las variables instaladas como compartidas en el sistema, con los tipos global VAR o LOCAL VAR. Los archivos o dispositivos de E/S de este tipo pertenecientes a todo el sistema seguirán estando abiertos.

En caso de un reinicio tras una caída de alimentación, todos los archivos o dispositivos de E/S abiertos del sistema se cierran y el descriptor de E/S de la variable del tipo `iodev` se restablece.

---

## Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                    | Causa del error                 |
|---------------------------|---------------------------------|
| <code>ERR_FILEOPEN</code> | No es posible abrir un archivo. |

---

## Sintaxis

```
Open  
[ Object '[:=' ] <expression (IN) of string>  
[ '\ ' File '[:=' ] <expression (IN) of string>] ', '  
[ IODevice '[:=' ] <variable (VAR) of iodev>  
[ '\ ' Read] |
```

Continúa en la página siguiente



## 1.156 Open - Abre un archivo o dispositivo de E/S

*RobotWare - OS*

*Continuación*

```
[ '\\' Write] |  
[ '\\' Append]  
[ '\\' Bin] ';' 
```

### Información relacionada

| Para obtener más información sobre                                      | Consulte   |
|---|--|
| Escritura, lectura y cierre de archivos o dispositivos de E/S           | <i>Manual de referencia técnica - RAPID Overview</i> |
| Fieldbus Command Interface<br>Gestión de archivos y dispositivos de E/S | <i>Application manual - Controller software IRC5</i> |

# 1 Instrucciones

---

## 1.157 OpenDir - Abre un directorio

RobotWare - OS

## 1.157 OpenDir - Abre un directorio

---

### Utilización

OpenDir se utiliza para abrir un directorio para su análisis posterior.

---

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción OpenDir:

#### Ejemplo 1

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

Este ejemplo imprime los nombres de todos los archivos o subdirectorios que se encuentran dentro del directorio especificado.

---

### Argumentos

OpenDir Dev Path

Dev

Tipo de dato: dir

Una variable que hace referencia a un directorio, capturada con OpenDir. Esta variable se utiliza posteriormente para hacer lecturas del directorio.

Path

Tipo de dato: string

La ruta del directorio.

---

### Limitaciones

Los directorios abiertos deben ser cerrados siempre por el usuario después de las operaciones de lectura (instrucción CloseDir).

---

### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema ERRNO cambiará a:

| Nombre      | Causa del error   |
|-------------|---|
| ERR_FILEACC | La ruta apunta a un directorio inexistente o existen demasiados directorios abiertos al mismo tiempo. |

---

### Sintaxis

```
OpenDir
  [ Dev ':' ] < variable (VAR) of dir> ','
  [ Path ':' ] < expression (IN) of string> ';'

```

Continúa en la página siguiente

---

#### Información relacionada

| Para obtener más información sobre        | Consulte  |
|---|---|
| Directorio                                | <a href="#">dir - Estructura de directorio de archivos en la página 1659</a>          |
| Creación de un directorio                 | <a href="#">MakeDir - Crea un nuevo directorio en la página 358</a>                   |
| Eliminación de un directorio              | <a href="#">RemoveDir - Elimina un directorio en la página 598</a>                    |
| Lectura de un directorio                  | <a href="#">ReadDir - Lee la siguiente entrada de un directorio en la página 1437</a> |
| Cierre de un directorio                   | <a href="#">CloseDir - Cierra un directorio en la página 153</a>                      |
| Eliminación de un archivo                 | <a href="#">RemoveFile - Elimina un archivo en la página 600</a>                      |
| Cambio del nombre de un archivo           | <a href="#">RenameFile - Cambia el nombre de un archivo en la página 604</a>          |
| Gestión de archivos y dispositivos de E/S | <a href="#">Application manual - Controller software IRC5</a>                         |

# 1 Instrucciones

---

## 1.158 PackDNHeader - Empaqueta un encabezado de DeviceNet en datos rawbytes RobotWare - OS

### 1.158 PackDNHeader - Empaqueta un encabezado de DeviceNet en datos rawbytes

---

#### Utilización

`PackDNHeader` se utiliza para empaquetar el encabezado de un mensaje explícito de DeviceNet en un 'contenedor' del tipo `rawbytes`.

A continuación, la parte de datos del mensaje de DeviceNet puede definirse con la instrucción `PackRawBytes`.

---

#### Ejemplos básicos

Los ejemplos siguientes ilustran la instrucción `PackDNHeader`.

##### Ejemplo 1

```
VAR rawbytes raw_data;
```

```
PackDNHeader "0E", "6,20 01 24 01 30 06,9,4", raw_data;
```

Se empaqueta la cabecera del mensaje explícito de DeviceNet con el código de servicio "0E" y la cadena de ruta "6,2001 24 01 30 06,9,4" en el `raw_data` correspondiente para obtener el número de serie de algún dispositivo de E/S.

Este mensaje está listo para su envío sin necesidad de llenar el mensaje con datos adicionales.

##### Ejemplo 2

```
VAR rawbytes raw_data;
```

```
PackDNHeader "10", "20 1D 24 01 30 64", raw_data;
```

Se empaqueta la cabecera del mensaje explícito de DeviceNet con el código de servicio "10" y la cadena de ruta "201D 24 01 30 64" en el `raw_data` correspondiente para establecer el tiempo de filtro del borde de elevación de la señal de entrada 1 de un dispositivo de E/S.

Este mensaje debe complementarse con datos del tiempo de filtro. Esto puede hacerse con la instrucción `PackRawBytes`, empezando por el número de índice `RawBytesLen(raw_data)+1` (se hace después de `PackDNHeader`).

---

#### Argumentos

```
PackDNHeader Service Path RawData
```

Service

Tipo de dato: `string`

El servicio que debe realizarse, por ejemplo obtener o definir el atributo. Debe especificarse con un código hexadecimal en una cadena, por ejemplo "IF".

|                    |                                 |
|--------------------|---------------------------------|
| Longitud de cadena | 2 caracteres                    |
| Formato            | '0' - '9', 'a' - 'f', 'A' - 'F' |
| Rango              | "00" - "FF"                     |

Los valores de `Service` se almacenan en el archivo EDS. Para obtener descripciones adicionales, consulte el documento *ODVA DeviceNet Specification*

---

Continúa en la página siguiente

## 1.158 PackDNHeader - Empaqueta un encabezado de DeviceNet en datos rawbytes

RobotWare - OS

Continuación

revisión 2.0 (Especificación de DeviceNet ODVA revisión 2.0) de la Open DeviceNet Vendor Association.

Path

Tipo de dato: string

Los valores de Path se almacenan en el archivo EDS. Para obtener descripciones adicionales, consulte el documento *ODVA DeviceNet Specification revision 2.0*. (Especificación de DeviceNet ODVA revisión 2.0) de la Open DeviceNet Vendor Association.

Se admiten tanto el formato de cadena largo (por ejemplo, "6,20 1D 24 01 30 64,8,1") como el corto (por ejemplo "20 1D 24 01 30 64").

RawData

Tipo de dato: rawbytes

El contenedor de variable que se desea empaquetar con los datos del encabezado del mensaje, a partir del número de índice 1 de RawData.

### Ejecución de programas

Durante la ejecución del programa, se hace lo siguiente con el contenedor del mensaje de DeviceNet RawData:

- Se borra completamente en primer lugar.
- A continuación, se empaqueta la parte de encabezado con datos.

### Formato del encabezado de DeviceNet

La instrucción PackDNHeader crea un encabezado de mensaje de DeviceNet con el formato siguiente:

| Formato de encabezado RawData | Nº de bytes | Nota   |
|-------------------------------|-------------|--|
| Formato                       | 1           | Código interno del controlador de robot para DeviceNet |
| Servicio                      | 1           | Código hexadecimal del servicio                        |
| Tamaño de ruta                | 1           | Bytes de entrada                                       |
| Trayectoria                   | x           | Caracteres ASCII                                       |

A continuación, la parte de datos del mensaje de DeviceNet puede definirse con la instrucción PackRawBytes, a partir del número de índice capturado con (RawBytesLen(my\_rawdata)+1).

### Sintaxis

```
PackDNHeader
  [Service ':=' ] <expression (IN) of string> ','
  [Path ':=' ] <expression (IN) of string> ','
  [RawData ':=' ] <variable (VAR) of rawbytes> ';'

```

### Información relacionada

| Para obtener más información sobre | Consulte   |
|------------------------------------|--|
| rawbytes datos                     | <a href="#">rawbytes - Datos sin formato en la página 1736</a> |

Continúa en la página siguiente

# 1 Instrucciones

## 1.158 PackDNHeader - Empaqueta un encabezado de DeviceNet en datos rawbytes

RobotWare - OS

Continuación

| Para obtener más información sobre                | Consulte  |
|---|---|
| Obtención de la longitud de un dato rawbytes      | <a href="#">RawBytesLen</a> - Obtiene la longitud de un dato de tipo rawbytes en la página 1432   |
| Borrado del contenido de un dato de tipo rawbytes | <a href="#">ClearRawBytes</a> - Borra el contenido de un dato de tipo rawbytes en la página 146   |
| Copiado del contenido de un dato de tipo rawbytes | <a href="#">CopyRawBytes</a> - Copia el contenido de un dato de tipo rawbytes en la página 171    |
| Empaquetamiento de datos en datos rawbytes        | <a href="#">PackRawBytes</a> - Empaqueta datos en un dato de tipo rawbytes en la página 499       |
| Escritura de un dato rawbytes                     | <a href="#">WriteRawBytes</a> - Escribe un dato de tipo rawbytes en la página 1111                |
| Lectura de un dato rawbytes                       | <a href="#">ReadRawBytes</a> - Lee datos de tipo rawbytes en la página 589                        |
| Desempaquetamiento de datos de un dato rawbytes   | <a href="#">UnpackRawBytes</a> - Desempaqueta datos de un dato de tipo rawbytes en la página 1024 |
| Funciones para bits/bytes                         | Manual de referencia técnica - <a href="#">RAPID Overview</a>                                     |
| Funciones para cadenas de caracteres              | Manual de referencia técnica - <a href="#">RAPID Overview</a>                                     |
| Gestión de archivos y dispositivos de E/S         | Application manual - Controller software IRC5   |

## 1.159 PackRawBytes - Empaqueta datos en un dato de tipo rawbytes

### Utilización

PackRawBytes se utiliza para empaquetar el contenido de las variables de tipo num, dnum, byte, o string en un contenedor del tipo rawbytes.

### Ejemplos básicos

Los siguientes ejemplos ilustran la instrucción 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;
```

**Empaquetamiento del encabezado de DeviceNet en un dato raw\_data.**

**A continuación, se empaquetan los datos de bus de campo en raw\_data con PackRawBytes. En los ejemplos siguientes se presenta cómo se pueden añadir distintos datos.**

#### Ejemplo 1

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

**El contenido de los 4 bytes siguientes al encabezado de raw\_data será un 8 decimal.**

#### Ejemplo 2

```
PackRawBytes bigInt, raw_data, (RawBytesLen(raw_data)+1) \IntX :=
  UDINT;
```

**El contenido de los 4 bytes siguientes al encabezado de raw\_data será un 4294967295 decimal.**

#### Ejemplo 3

```
PackRawBytes bigInt, raw_data, (RawBytesLen(raw_data)+1) \IntX :=
  LINT;
```

**El contenido de los 8 bytes siguientes al encabezado de raw\_data será un 4294967295 decimal.**

#### Ejemplo 4

```
PackRawBytes float, raw_data, RawBytesLen(raw_data)+1) \Float4;
```

**El contenido de los 4 bytes siguientes de raw\_data será un 13,4 decimal.**

#### Ejemplo 5

```
PackRawBytes data1, raw_data, (RawBytesLen(raw_data)+1) \ASCII;
```

**El contenido del byte siguiente de raw\_data será 122, el código ASCII de "z".**

#### Ejemplo 6

```
PackRawBytes string1, raw_data, (RawBytesLen(raw_data)+1) \ASCII;
```

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.159 PackRawBytes - Empaqueta datos en un dato de tipo rawbytes

RobotWare - OS

Continuación

El contenido de los 7 bytes siguientes de `raw_data` será "abcdefg", codificado en ASCII.

### Ejemplo 7

```
byte1 := StrToByte("1F" \Hex);  
PackRawBytes byte1, raw_data, (RawBytesLen(raw_data)+1) \Hex1;
```

El contenido del byte siguiente de `raw_data` será "1F" hexadecimal.

---

### Argumentos

```
PackRawBytes Value RawData [ \Network ] StartIndex [\Hex1] | [\IntX  
] | [ \Float4 ] | [ \ASCII ]
```

Consulte [Combinación de los argumentos en la página 501](#).

Value

Tipo de dato: anytype

Datos a empaquetar en `RawData`.

Los tipos de datos permitidos son: num, dnum, byte o string. No se permite el uso de matrices.

RawData

Tipo de dato: rawbytes

El contenedor de variable en el que se almacenarán los datos empaquetados.

[ \Network ]

Tipo de dato: switch

Indica que los valores de tipo integer y float deben empaquetarse con la representación big-endian (orden de red) en `RawData`. Tanto ProfiBus como InterBus utilizan big-endian.

Sin este modificador, los valores integer y float se empaquetan con la representación little-endian (sin el orden de la red) en `RawData`. DeviceNet utiliza little-endian.

Sólo relevante junto con el parámetro opcional `\IntX - UINT, UDINT, ULINT, INT, DINT, LINT` y `\Float4`.

StartIndex

Tipo de dato: num

`StartIndex`, entre 1 y 1.024, indica en qué lugar de `Value` debe situarse el primer byte contenido en `RawData`.

[ \Hex1 ]

Tipo de dato: switch

El `Value` a empaquetar tiene el formato byte y debe convertirse al formato hexadecimal y almacenarse en 1 byte en `RawData`.

[ \IntX ]

Tipo de dato: inttypes

Continúa en la página siguiente



## 1.159 PackRawBytes - Empaquetar datos en un dato de tipo rawbytes

RobotWare - OS

Continuación

El valor `Value` a empaquetar tiene el formato `num` o `dnum`. Se trata de un entero y debe almacenarse en `RawData` de acuerdo con esta constante especificada del tipo de dato `inttypes`.

Consulte [Datos predefinidos en la página 502](#).

[ \Float4 ]

Tipo de dato: `switch`

El `Value` a empaquetar tiene el formato `num` y debe almacenarse en `RawData` como flotante en 4 bytes.

[ \ASCII ]

Tipo de dato: `switch`

El `Value` a empaquetar tiene el formato `byte` o `string`.

Si el valor `Value` a empaquetar tiene el formato `byte`, se almacenará en `RawData` como 1 byte que interpreta `Value` como el código ASCII de un carácter.

Si el valor `Value` a empaquetar tiene el formato de cadena (de 1 a 80 caracteres), se almacenará en `RawData` como caracteres ASCII con tantos caracteres como contenga `Value`. Los datos de cadena no son terminados con `NULL` por el sistema en el caso de los datos de tipo `rawbytes`. El programador es el responsable de añadir un encabezado de cadena en caso necesario (obligatorio en el caso de `DeviceNet`).

## Combinación de los argumentos

Se debe utilizar uno de los argumentos `\Hex1`, `\IntX`, `\Float4`, o `\ASCII`.

Se permiten las combinaciones siguientes:

| Tipo de dato de <code>Value</code> : | Parámetros opcionales permitidos:          |
|--------------------------------------|--|
| <code>num</code> <sup>i</sup>        | <code>\IntX</code>                         |
| <code>dnum</code> <sup>ii</sup>      | <code>\IntX</code>                         |
| <code>num</code>                     | <code>\Float4</code>                       |
| <code>string</code>                  | <code>\ASCII (de 1 a 80 caracteres)</code> |
| <code>byte</code>                    | <code>\Hex1 \ASCII</code>                  |

<sup>i</sup> Debe ser un entero dentro del rango de valor de la constante simbólica seleccionada, `USINT`, `UINT`, `UDINT`, `SINT`, `INT` o `DINT`.

<sup>ii</sup> Debe ser un entero dentro del rango de valor de la constante simbólica seleccionada, `USINT`, `UINT`, `UDINT`, `ULINT`, `SINT`, `INT`, `DINT` o `LINT`.

## Ejecución de programas

Durante la ejecución del programa, se empaquetan los datos de la variable de tipo `anytype` en un contenedor de tipo `rawbytes`.

La longitud actual de los bytes válidos de la variable `RawData` cambia a:

- $(\text{StartIndex} + \text{número\_de\_bytes\_empaquetados} - 1)$
- La longitud actual de los bytes válidos de la variable `RawData` no cambia si la totalidad de la operación de empaquetado se realiza dentro de la longitud anterior de bytes válidos de la variable `RawData`.

Continúa en la página siguiente

# 1 Instrucciones

## 1.159 PackRawBytes - Empaqueta datos en un dato de tipo rawbytes

RobotWare - OS

Continuación

### Datos predefinidos

Se han definido las constantes simbólicas siguientes para el tipo de dato `inttypes`. Puede usarlas para especificar el entero en el parámetro `\IntX`.

| Constante simbólica | Valor constante | Formato de entero           | Rango de valores enteros   |
|---------------------|-----------------|-----------------------------|--|
| USINT               | 1               | Entero de 1 byte sin signo  | 0 ... 255  |
| UINT                | 2               | Entero de 2 byte sin signo  | 0 ... 65 535   |
| UDINT               | 4               | Entero de 4 byte sin signo  | 0 ... 8 388 608 <sup>i</sup><br>0 ... 4 294 967 295 <sup>ii</sup>                        |
| ULINT               | 8               | Entero de 8 byte sin signo  | 0 ... 4 503 599 627 370 496 <sup>iii</sup>   |
| SINT                | - 1             | Entero de 1 bytes con signo | - 128... 127   |
| INT                 | - 2             | Entero de 2 bytes con signo | - 32 768 ... 32 767  |
| DINT                | - 4             | Entero de 4 bytes con signo | - 8 388 607 ... 8 388 608 <sup>i</sup><br>-2 147 483 648 ... 2 147 483 647 <sup>iv</sup> |
| LINT                | - 8             | Entero de 8 bytes con signo | - 4 503 599 627 370 496... 4 503 599 627 370 496 <sup>iii</sup>                          |

<sup>i</sup> Limitación de RAPID para el almacenamiento de enteros en el tipo de dato `num`.

<sup>ii</sup> Rango al utilizar una variable `dnum` e `inttype UDINT`.

<sup>iii</sup> Limitación de RAPID para el almacenamiento de enteros en el tipo de dato `dnum`.

<sup>iv</sup> Rango al utilizar una variable `dnum` e `inttype DINT`.

### Sintaxis

```
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] ';' ;'
```

### Información relacionada

| Para obtener más información sobre                              | Consulte   |
|---|--|
| rawbytes datos  | <a href="#">rawbytes - Datos sin formato en la página 1736</a>   |
| Obtención de la longitud de un dato rawbytes                    | <a href="#">RawBytesLen - Obtiene la longitud de un dato de tipo rawbytes en la página 1432</a>        |
| Borrado del contenido de un dato de tipo rawbytes               | <a href="#">ClearRawBytes - Borra el contenido de un dato de tipo rawbytes en la página 146</a>        |
| Copiado del contenido de un dato de tipo rawbytes               | <a href="#">CopyRawBytes - Copia el contenido de un dato de tipo rawbytes en la página 171</a>         |
| Empaquetamiento de un encabezado de DeviceNet en datos rawbytes | <a href="#">PackDNHeader - Empaqueta un encabezado de DeviceNet en datos rawbytes en la página 496</a> |

Continúa en la página siguiente

## 1.159 PackRawBytes - Empaqueta datos en un dato de tipo rawbytes

RobotWare - OS

Continuación

| Para obtener más información sobre                           | Consulte  |
|--|---|
| Escritura de un dato <code>rawbytes</code>                   | <a href="#">WriteRawBytes - Escribe un dato de tipo rawbytes en la página 1111</a>                |
| Lectura de un dato <code>rawbytes</code>                     | <a href="#">ReadRawBytes - Lee datos de tipo rawbytes en la página 589</a>                        |
| Desempaquetamiento de datos de un dato <code>rawbytes</code> | <a href="#">UnpackRawBytes - Desempaqueta datos de un dato de tipo rawbytes en la página 1024</a> |
| Funciones para bits/bytes                                    | <i>Manual de referencia técnica - RAPID Overview</i>  |
| Funciones para cadenas de caracteres                         | <i>Manual de referencia técnica - RAPID Overview</i>  |
| Gestión de archivos y dispositivos de E/S                    | <i>Application manual - Controller software IRC5</i>  |

# 1 Instrucciones

1.160 PathAccLim - Reduce la aceleración del TCP a lo largo de la trayectoria  
RobotWare - OS

## 1.160 PathAccLim - Reduce la aceleración del TCP a lo largo de la trayectoria

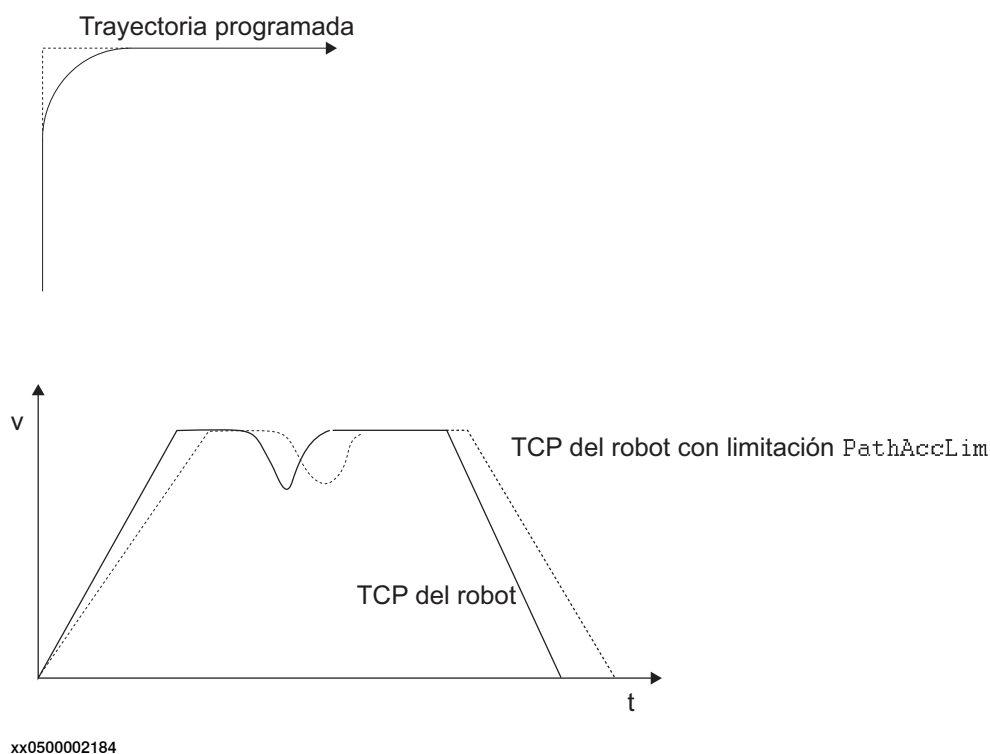
### Utilización

PathAccLim (*Path Acceleration Limitation*) se usa para establecer o restablecer limitaciones de aceleración y/o deceleración de TCP a lo largo de la trayectoria de movimiento.

La limitación se realiza a lo largo de la trayectoria de movimiento, es decir, la aceleración en la base de coordenadas de la trayectoria. La aceleración/deceleración que se limita es la aceleración/deceleración tangencial en la dirección de la trayectoria.

Esta instrucción no limita la aceleración total del equipo, es decir, la aceleración en la base de coordenadas mundo, de forma que no es posible usarla directamente para proteger al equipo de grandes aceleraciones.

Esta instrucción sólo puede usarse en la tarea principal T\_ROB1 o, si se cuenta con un sistema *MultiMove*, en las tareas de movimiento.



### Ejemplos básicos

Los ejemplos siguientes ilustran la instrucción PathAccLim.

Consulte también [Más ejemplos en la página 506](#).

#### Ejemplo 1

```
PathAccLim TRUE \AccMax := 4, TRUE \DecelMax := 4;
```

Se limita a  $4 \text{ m/s}^2$  la aceleración y la deceleración del TCP.

#### Ejemplo 2

```
PathAccLim FALSE, FALSE;
```

Continúa en la página siguiente

## 1.160 PathAccLim - Reduce la aceleración del TCP a lo largo de la trayectoria RobotWare - OS Continuación

Se devuelven la aceleración y deceleración del TCP a sus valores máximos (predeterminados).

### Argumentos

```
PathAccLim AccLim [\AccMax] DecelLim [\DecelMax]
```

AccLim

Tipo de dato: bool

TRUE si debe haber una limitación de la aceleración; de lo contrario, FALSE.

[ \AccMax ]

Tipo de dato: num

El valor absoluto de la limitación de aceleración en  $m/s^2$ . Sólo debe usarse si AccLim tiene el valor TRUE.

DecelLim

Tipo de dato: bool

TRUE si debe haber una limitación de la deceleración; de lo contrario, FALSE.

[ \DecelMax ]

Tipo de dato: num

El valor absoluto de la limitación de deceleración en  $m/s^2$ . Sólo debe usarse si DecelLim tiene el valor TRUE.

### Ejecución de programas

Las limitaciones de aceleración/deceleración se aplican a la siguiente instrucción de movimiento ejecutada y es válida hasta que se ejecute una nueva instrucción PathAccLim.

La aceleración/deceleración máxima (PathAccLim FALSE, FALSE) se establece automáticamente en los casos siguientes:

- cuando se utiliza el modo de reinicio **Restablecer RAPID**
- al cargar un nuevo programa o un nuevo módulo
- al iniciar la ejecución del programa desde el principio
- al mover el puntero del programa a `main`
- al mover el puntero del programa a una rutina
- al mover el puntero de programa de una forma que se pierde el orden de la ejecución.

Si se usa una combinación de instrucciones AccSet y PathAccLim, el sistema reduce la aceleración y deceleración en el orden siguiente:

- Según AccSet
- Según PathAccLim

Continúa en la página siguiente

# 1 Instrucciones

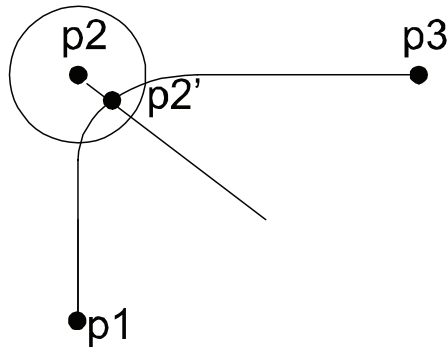
## 1.160 PathAccLim - Reduce la aceleración del TCP a lo largo de la trayectoria

RobotWare - OS

Continuación

### Más ejemplos

A continuación aparecen más ejemplos de cómo usar la instrucción PathAccLim.



xx0500002183

#### Ejemplo 1

```
MoveL p1, v1000, fine, tool0;  
PathAccLim TRUE\AccMax := 4, FALSE;  
MoveL p2, v1000, z30, tool0;  
MoveL p3, v1000, fine, tool0;  
PathAccLim FALSE, FALSE;
```

Se limita la aceleración del TCP a  $4 \text{ m/s}^2$  entre p1 y p3.

#### Ejemplo 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;
```

Se limita la aceleración del TCP a  $3 \text{ m/s}^2$  entre p2' y p3.

La deceleración del TCP se limita a  $4 \text{ m/s}^2$  entre p2' y p3.

### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema ERRNO cambiará a:

| Nombre          | Causa del error  |
|-----------------|--|
| ERR_ACC_TOO_LOW | El parámetro \AccMax o \DecelMax se ha establecido demasiado bajo. |

### Limitaciones

La aceleración/deceleración mínima permitida es de  $0.1 \text{ m/s}^2$ . Se recomienda que el límite de aceleración y deceleración sean simétricos, para que tengan el mismo valor en AccMax y DecelMax.

### Sintaxis

```
PathAccLim  
[ AccLim ':= ' ] < expression (IN) of bool >  
[ '\ ' AccMax ':= ' <expression (IN) of num > ] ', '
```

Continúa en la página siguiente

## 1.160 PathAccLim - Reduce la aceleración del TCP a lo largo de la trayectoria

RobotWare - OS

Continuación

```
[ DecelLim ':=' ] < expression (IN) of bool >  
[ '\ ' DecelMax ':=' <expression (IN) of num > ] ';' 
```

### Información relacionada

| Para obtener más información sobre                        | Consulte  |
|---|---|
| Reducción de la aceleración                               | <a href="#">AccSet - Reduce la aceleración en la página 25</a>  |
| Datos de parámetros de movimiento                         | <a href="#">motsetdata - Datos de parámetros de movimiento en la página 1705</a>                          |
| Control de aceleración en el sistema de coordenadas mundo | <a href="#">WorldAccLim - Control de aceleración en el sistema de coordenadas mundo en la página 1096</a> |
| Instrucciones de posicionamiento                          | <a href="#">Manual de referencia técnica - RAPID Overview, sección Resumen sobre RAPID - Movimiento</a>   |

## 1 Instrucciones

---

1.161 PathLengthReset - Restablece el valor de longitud de trayectoria actual del contador  
*RobotWare - OS*

### 1.161 PathLengthReset - Restablece el valor de longitud de trayectoria actual del contador

---

#### Utilización

`PathLengthReset` se utiliza para restablecer el contador que monitoriza la longitud de trayectoria recorrida por el TCP del robot.

La llamada a esta instrucción puede realizarse en cualquier momento, aunque se recomienda hacerlo cuando el robot esté en reposo para obtener un comportamiento predecible para el contador.

Esta instrucción sólo puede usarse en la tarea principal `T_ROB1` o, si se cuenta con un sistema `MultiMove`, en las tareas de movimiento.

---

#### Ejemplos básicos

El ejemplo siguiente ilustra la instrucción `PathLengthReset`.

##### Ejemplo 1

```
PathLengthStart;  
MoveJ p10, v1000, z50, L10tip;  
...  
MoveL p40, v1000, fine, L10tip;  
PathLengthStop;  
TPWrite "PathLengthGet: "+ValToStr(PathLengthGet());  
PathLengthReset;
```

Este ejemplo realiza la lectura del valor del contador que mide la longitud de trayectoria recorrida por el TCP del robot. El valor se escribe en el `FlexPendant`.

---

#### Ejecución de programas

La medición de longitud de trayectoria se aplica a la siguiente instrucción de movimiento de robot ejecutada, de cualquier tipo, y se mantiene vigente hasta la ejecución de una instrucción `PathLengthStop`.

La medición de longitud de trayectoria se desactiva, y el contador de medición de longitud de trayectoria se establece en cero, cuando se ejecuta una instrucción `PathLengthReset`. El valor predeterminado, medición de longitud de trayectoria desactivada, se establece automáticamente:

- Cuando se utiliza el modo de reinicio *Restablecer RAPID*.
  - al cargar un nuevo programa o un nuevo módulo.
  - Cuando se inicia la ejecución del programa desde el principio.
  - Cuando se mueve el puntero del programa a rutina principal.
  - al mover el puntero del programa a una rutina.
  - al mover el puntero de programa de una forma que se pierde el orden de la ejecución.
- 

#### Limitaciones

Las mediciones de longitud de trayectoria solo son aplicables a robots TCP.

---

*Continúa en la página siguiente*



## 1.161 PathLengthReset - Restablece el valor de longitud de trayectoria actual del contador

RobotWare - OS

Continuación

### Sintaxis

```
PathLengthReset';'
```

### Información relacionada

| Para obtener más información sobre | Consulte  |
|------------------------------------|---|
| PathLengthGet                      | <a href="#">PathLengthGet - Lee el valor de longitud de trayectoria actual del contador en la página 1406</a>   |
| PathLengthStart                    | <a href="#">PathLengthStart - Activa el contador que monitoriza la longitud de trayectoria en la página 510</a> |
| PathLengthStop                     | <a href="#">PathLengthStop - Detiene el contador que monitoriza la longitud de trayectoria en la página 512</a> |

## 1 Instrucciones

---

1.162 PathLengthStart - Activa el contador que monitoriza la longitud de trayectoria  
RobotWare - OS

### 1.162 PathLengthStart - Activa el contador que monitoriza la longitud de trayectoria

---

#### Utilización

`PathLengthStart` se utiliza para activar el contador que monitoriza la longitud de trayectoria recorrida por el TCP del robot, a partir de la siguiente instrucción de movimiento. La longitud de trayectoria se mide siempre en el objeto de trabajo. Tenga en cuenta que esta instrucción no restablece el contador interno para la longitud del TCP. Para restablecer el contador, utilice la instrucción `PathLengthReset`.

Esta instrucción sólo puede usarse en la tarea principal `T_ROB1` o, si se cuenta con un sistema `MultiMove`, en las tareas de movimiento.

---

#### Ejemplos básicos

El ejemplo siguiente ilustra la instrucción `PathLengthStart`.

##### Ejemplo 1

```
PathLengthStart;  
MoveJ p10, v1000, z50, L10tip;  
...  
MoveL p40, v1000, fine, L10tip;  
PathLengthStop;  
TPWrite "PathLengthGet: "+ValToStr(PathLengthGet());  
PathLengthReset;
```

Este ejemplo realiza la lectura del valor del contador que mide la longitud de trayectoria recorrida por el TCP del robot. El valor se escribe en el FlexPendant.

---

#### Ejecución de programas

La medición de longitud de trayectoria se aplica a la siguiente instrucción de movimiento de robot ejecutada, de cualquier tipo, y se mantiene vigente hasta la ejecución de una instrucción `PathLengthStop`.

La medición de longitud de trayectoria se desactiva, y el contador de medición de longitud de trayectoria se establece en cero, cuando se ejecuta una instrucción `PathLengthReset`. El valor predeterminado, medición de longitud de trayectoria desactivada, se establece automáticamente:

- Cuando se utiliza el modo de reinicio *Restablecer RAPID*.
  - al cargar un nuevo programa o un nuevo módulo.
  - Cuando se inicia la ejecución del programa desde el principio
  - Cuando se mueve el puntero del programa a rutina principal.
  - al mover el puntero del programa a una rutina.
  - al mover el puntero de programa de una forma que se pierde el orden de la ejecución.
- 

#### Limitaciones

Las mediciones de longitud de trayectoria solo son aplicables a robots TCP.

---

*Continúa en la página siguiente*

## 1.162 PathLengthStart - Activa el contador que monitoriza la longitud de trayectoria

RobotWare - OS

Continuación

### Sintaxis

```
PathLengthStart';'
```

### Información relacionada

| Para obtener más información sobre | Consulte  |
|------------------------------------|---|
| PathLengthReset                    | <a href="#">PathLengthReset</a> - Restablece el valor de longitud de trayectoria actual del contador en la página 508 |
| PathLengthGet                      | <a href="#">PathLengthGet</a> - Lee el valor de longitud de trayectoria actual del contador en la página 1406         |
| PathLengthStop                     | <a href="#">PathLengthStop</a> - Detiene el contador que monitoriza la longitud de trayectoria en la página 512       |

## 1 Instrucciones

---

1.163 PathLengthStop - Detiene el contador que monitoriza la longitud de trayectoria  
*RobotWare - OS*

### 1.163 PathLengthStop - Detiene el contador que monitoriza la longitud de trayectoria

---

#### Utilización

`PathLengthStop` se utiliza para detener el contador que monitoriza la longitud de trayectoria recorrida por el TCP del robot, a partir de la siguiente instrucción de movimiento.

La longitud de trayectoria se mide siempre en el objeto de trabajo. Tenga en cuenta que esta instrucción no restablece el contador interno para la longitud del TCP. Para restablecer el contador, utilice la instrucción `PathLengthReset`.

Esta instrucción sólo puede usarse en la tarea principal `T_ROB1` o, si se cuenta con un sistema MultiMove, en las tareas de movimiento.

#### Ejemplos básicos

El ejemplo siguiente ilustra la instrucción `PathLengthStop`.

##### Ejemplo 1

```
PathLengthStart;  
MoveJ p10, v1000, z50, L10tip;  
...  
MoveL p40, v1000, fine, L10tip;  
PathLengthStop;  
TPWrite "PathLengthGet: "+ValToStr(PathLengthGet());  
PathLengthReset;
```

Este ejemplo realiza la lectura del valor del contador que mide la longitud de trayectoria recorrida por el TCP del robot. El valor se escribe en el FlexPendant.

#### Ejecución de programas

La medición de longitud de trayectoria se aplica a la siguiente instrucción de movimiento de robot ejecutada, de cualquier tipo, y se mantiene vigente hasta la ejecución de una instrucción `PathLengthStop`.

La medición de longitud de trayectoria se desactiva, y el contador de medición de longitud de trayectoria se establece en cero, cuando se ejecuta una instrucción `PathLengthReset`. El valor predeterminado, medición de longitud de trayectoria desactivada, se establece automáticamente:

- Cuando se utiliza el modo de reinicio *Restablecer RAPID*.
- al cargar un nuevo programa o un nuevo módulo.
- Cuando se inicia la ejecución del programa desde el principio.
- Cuando se mueve el puntero del programa a rutina principal.
- al mover el puntero del programa a una rutina.
- al mover el puntero de programa de una forma que se pierde el orden de la ejecución.

#### Limitaciones

Las mediciones de longitud de trayectoria solo son aplicables a robots TCP.

*Continúa en la página siguiente*

## 1.163 PathLengthStop - Detiene el contador que monitoriza la longitud de trayectoria

RobotWare - OS

Continuación

### Sintaxis

```
PathLengthStop';'
```

### Información relacionada

| Para obtener más información sobre | Consulte  |
|------------------------------------|---|
| PathLengthReset                    | <a href="#">PathLengthReset</a> - Restablece el valor de longitud de trayectoria actual del contador en la página 508 |
| PathLengthGet                      | <a href="#">PathLengthGet</a> - Lee el valor de longitud de trayectoria actual del contador en la página 1406         |
| PathLengthStart                    | <a href="#">PathLengthStart</a> - Activa el contador que monitoriza la longitud de trayectoria en la página 510       |

## 1 Instrucciones

---

### 1.164 PathRecMoveBwd - Hace retroceder la grabadora de trayectorias

*Path Recovery*

### 1.164 PathRecMoveBwd - Hace retroceder la grabadora de trayectorias

---

#### Utilización

PathRecMoveBwd se utiliza para hacer retroceder el robot a lo largo de una trayectoria grabada.

---

#### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción PathRecMoveBwd: Consulte también [Más ejemplos en la página 515](#).

#### Ejemplo 1

```
VAR pathrecid fixture_id;  
PathRecMoveBwd \ID:=fixture_id \ToolOffs:=[0, 0, 10] \Speed:=v500;
```

El robot se mueve hacia atrás hasta la posición del programa en la que la instrucción PathRecStart insertó el identificador fixture\_id. El offset del TCP está situado 10 mm en la dirección Z y la velocidad es de 500 mm/s.

---

#### Argumentos

```
PathRecMoveBwd [\ID] [\ToolOffs] [\Speed]
```

[\ID]

##### Identifier

Tipo de dato: pathrecid

Una variable que especifica la posición de ID hasta la que se desea retroceder. El tipo de dato pathrecid es de un tipo sin valor y sólo se utiliza como un identificador para asignar un nombre a la posición de grabación.

Si no se especifica ninguna posición de ID, el movimiento de retroceso se realiza en un sistema sencillo hasta la posición de ID más cercana que se haya grabado. Sin embargo, en un sistema MultiMove en modo sincronizado, los movimientos de retroceso se realizan hasta la más cercana de las posiciones siguientes:

- Regreso a la posición en la que se inició el movimiento sincronizado
- Regreso hasta la posición de ID más cercana grabada

[\ToolOffs]

##### Tool Offset

Tipo de dato: pos

Proporciona un offset de separación del TCP durante el movimiento. Se aplica una coordenada de offset cartesiana a las coordenadas del TCP. El valor de offset Z positivo indica la separación. Resulta útil si el robot ejecuta un proceso en el que se añade material. Si se está utilizando el movimiento sincronizado, o bien todas o ninguna de las unidades mecánicas deben utilizar este argumento. Si no se desea utilizar ningún offset en algunas unidades mecánicas, es posible aplicar un offset cero. Incluso las unidades mecánicas sin TCP deben utilizar el argumento si se usa un robot con TCP perteneciente a otra tarea.

[\Speed]

Tipo de dato: speeddata

*Continúa en la página siguiente*

---

## 1.164 PathRecMoveBwd - Hace retroceder la grabadora de trayectorias

*Path Recovery*

*Continuación*

Esta velocidad reemplaza a la velocidad original utilizada durante el movimiento de avance. Speeddata define la velocidad del punto central de la herramienta, la reorientación de la misma y los ejes externos. Si está presente, esta velocidad se utiliza durante todo el movimiento de retroceso. Si se omite, el movimiento de retroceso se ejecuta a la velocidad de las instrucciones de movimiento originales.

### Ejecución de programas

La grabadora de trayectorias se activa con la instrucción `PathRecStart`. Una vez que la grabadora se ha iniciado, todas las instrucciones de movimiento quedan grabadas y el robot puede retroceder en cualquier momento a lo largo de la trayectoria grabada, con ayuda de `PathRecMoveBwd`.

### Movimiento sincronizado

La ejecución de la grabadora de trayectorias durante el movimiento sincronizado supone algunas consideraciones adicionales.

- Todas las tareas implicadas en el movimiento grabado sincronizado deben ejecutar `PathRecMoveBwd` antes de que ninguno de los robots empiece a moverse.
- Todo el manejo de la sincronización se graba y ejecuta de forma invertida. Por ejemplo, si se ejecuta `PathRecMoveBwd` dentro de un bloque de sincronización hacia una posición independiente, la grabadora de trayectorias cambiará automáticamente el estado al modo independiente al ejecutarse la instrucción `SyncMoveOn`.
- `SyncMoveOn` se considera como punto de ruptura sin identificador de trayectoria. Es decir, si la grabadora de trayectorias ha sido puesta en marcha por medio de `PathRecStart` y `PathRecMoveBwd` sin el argumento opcional `\ID` dentro de un bloque con movimiento sincronizado, el robot retrocede hasta la posición en la que se encontraba cuando se ejecutó `SyncMoveOn`. Dado que el movimiento hacia atrás se detiene antes de `SyncMoveOn`, el estado cambia al modo independiente.
- `WaitSyncTask` se considera como punto de ruptura sin identificador de trayectoria. Es decir, si la grabadora de trayectorias ha sido puesta en marcha por medio de `PathRecStart` y se ejecuta `PathRecMoveBwd`, el robot retrocederá no más allá de la posición en la que se encontraba en el momento de ejecutarse `WaitSyncTask`.

### Más ejemplos

A continuación aparecen más ejemplos de cómo usar la instrucción `PathRecMoveBwd`.

#### Ejemplo 1 - Movimiento independiente

```
VAR pathrecid safe_id;  
CONST robtarget p0 := [...];  
...  
CONST robtarget p4 := [...];  
VAR num choice;  
  
MoveJ p0, vmax, z50, tool1;
```

*Continúa en la página siguiente*

# 1 Instrucciones

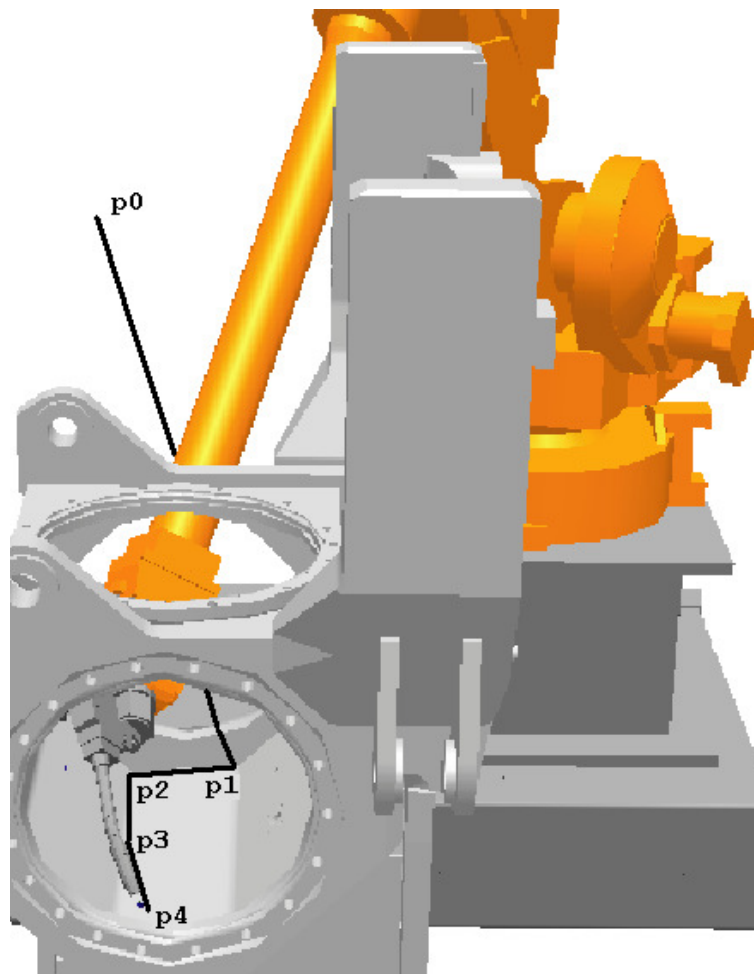
## 1.164 PathRecMoveBwd - Hace retroceder la grabadora de trayectorias

### Path Recovery

#### Continuación

```
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
ENDIF
ENDIF
```



xx0500002135

Continúa en la página siguiente



## 1.164 PathRecMoveBwd - Hace retroceder la grabadora de trayectorias

*Path Recovery*

*Continuación*

En este ejemplo se muestra cómo puede usarse la grabadora de trayectorias para extraer el robot de espacios reducidos en caso de error, sin necesidad de programar una trayectoria designada.

Se fabrica una pieza. En el punto de aproximación, `p0`, la grabadora de trayectorias se inicia y se le asigna el identificador de grabadora de trayectorias `safe_id`. Suponga que durante el movimiento del robot de `p3` a `p4` se produce un error recuperable. En ese punto, la trayectoria se almacena ejecutando `StorePath`. Al almacenar la trayectoria, el gestor de errores puede iniciar un nuevo movimiento y reiniciar más adelante el movimiento original. Una vez que la trayectoria ha quedado almacenada, se usa la grabadora de trayectorias para hacer retroceder el robot hasta una posición segura, `p0`, mediante la ejecución de `PathRecMoveBwd`. Recuerde que se aplica un offset de herramienta para disponer de una separación, por ejemplo respecto de una nueva soldadura realizada. Una vez que el robot ha retrocedido, el operador puede tomar las acciones oportunas para solucionar el error (por ejemplo limpiar la pistola de soldadura). A continuación, el robot es devuelto al punto en el que se produjo el error, con ayuda de `PathRecMoveFwd`. En la ubicación del error, el nivel de la trayectoria se devuelve al nivel básico con `RestoPath` y se realiza un intento de recuperación.

### Ejemplo 2 - Movimiento sincronizado

#### T\_ROB1

```

VAR pathrecid HomeROB1;
CONST robtarget pR1_10:=[...];
...
CONST robtarget pR1_60:=[...];

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:=[...];

```

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.164 PathRecMoveBwd - Hace retroceder la grabadora de trayectorias

### Path Recovery

#### Continuación

```
...
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:=[...];
...
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;
```

**El sistema se compone de tres manipuladores, todos ellos ejecutándose en tareas separadas. Suponga que T\_ROB1 sufre un error ERR\_PATH\_STOP dentro del bloque**

*Continúa en la página siguiente*

## 1.164 PathRecMoveBwd - Hace retroceder la grabadora de trayectorias

*Path Recovery**Continuación*

sincronizado, `sync1`. En el error, se desea retroceder hasta la posición de inicio marcada con el identificador `HomeROB1` de la grabadora de trayectorias, para realizar un servicio en el equipo externo del robot. Esto se realiza mediante `PathRecMoveBwd` e indicando el identificador `pathrecid`.

Dado que el error se produjo durante el movimiento sincronizado, es necesario que el segundo robot con TCP `T_ROB2` y el eje externo `T_POS1` también ejecuten `PathRecMoveBwd`. Estos manipuladores no tienen por qué retroceder más allá del punto en el que se inició el movimiento sincronizado. Al no suministrar `PathRecMoveBwd` en `ERR_PATH_STOP` con un identificador de grabadora de trayectorias, se aprovecha la capacidad de la grabadora de trayectorias para detenerse a continuación de `SyncMoveOn`. Recuerde que el eje externo que no tiene TCP sigue añadiendo un offset cero de herramienta, para permitir la posibilidad de que los robots con TCP lo hagan también.

El comportamiento `DEFAULT` del gestor de `ERROR` de este ejemplo es que todos los manipuladores realicen en primer lugar los movimientos sincronizados hacia atrás y a continuación los movimientos independientes hacia atrás hasta el punto de inicio de la trayectoria grabada. Esto se consigue especificando `\ID` en `PathRecMoveBwd` para todos los manipuladores.

**Limitaciones**

Los movimientos realizados utilizando la grabadora de trayectorias no pueden realizarse en el nivel de base. Es decir, debe ejecutarse `StorePath` antes de ejecutar `PathRecMoveBwd`.

En ningún caso es posible retroceder más allá de una sentencia `SynchMoveOff`.

En ningún caso es posible retroceder más allá de una sentencia `WaitSyncTask`.

la instrucción `SyncMoveOn` debe ir precedida de al menos un movimiento independiente, si se desea retroceder hasta la posición en la que se inició el movimiento sincronizado.

Si no se desea volver al punto en el que se ejecutó `PathRecMoveBwd` (mediante la ejecución de `PathRecMoveFwd`) es necesario detener la grabadora de trayectorias con la instrucción `PathRecStop`. `PathRecStop\Clear` también borra la trayectoria grabada.

`PathRecMoveBwd` no puede ejecutarse en rutinas de `RAPID` que estén conectadas a los siguientes eventos especiales del sistema: `PowerOn`, `Stop`, `QStop`, `Restart`, `Reset` o `Step`.

**Sintaxis**

```
PathRecMoveBwd
[ '\ ID :=' < variable (VAR) of pathrecid > ]
[ '\ ToolOffs :=' <expression (IN) of pos> ]
[ '\ Speed :=' <expression (IN) of speeddata> ]';
```

**Información relacionada**

| Para obtener más información sobre         | Consulte   |
|--|--|
| Identificador de grabadora de trayectorias | <a href="#">pathrecid - Identificador de grabadora de trayectorias en la página 1725</a> |

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.164 PathRecMoveBwd - Hace retroceder la grabadora de trayectorias

### *Path Recovery*

#### *Continuación*

| <b>Para obtener más información sobre</b>                  | <b>Consulte</b>   |
|--|---|
| Inicio y detención de la grabadora de trayectorias         | <a href="#">PathRecStart - Inicia la grabadora de trayectorias en la página 524</a><br><a href="#">PathRecStop - Detiene la grabadora de trayectorias en la página 527</a>  |
| Comprobación de que la trayectoria grabada es correcta     | <a href="#">PathRecValidBwd - Comprueba si existe una trayectoria de retroceso válida guardada en la página 1410</a><br><a href="#">PathRecValidFwd - Comprueba si existe una trayectoria de avance válida guardada en la página 1413</a> |
| Reproducción de la grabación de trayectorias hacia delante | <a href="#">PathRecMoveFwd - Hace avanzar la grabadora de trayectorias en la página 521</a>   |
| Almacenamiento y restauración de trayectorias              | <a href="#">StorePath - Almacena la trayectoria cuando se produce una interrupción en la página 835</a><br><a href="#">RestoPath - Restablece la trayectoria después de una interrupción en la página 615</a>                             |
| Otras instrucciones de posicionamiento                     | <i>Manual de referencia técnica - RAPID Overview</i>  |
| Recuperación en caso de error                              | <i>Manual de referencia técnica - RAPID Overview</i>  |

### 1.165 PathRecMoveFwd - Hace avanzar la grabadora de trayectorias

#### Utilización

`PathRecMoveFwd` se utiliza para mover el robot de nuevo a la posición en la que se ejecutó `PathRecMoveBwd`. También es posible hacer avanzar parcialmente al robot, mediante un identificador que se entrega durante el movimiento de retroceso.

#### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción `PathRecMoveFwd`: Consulte también [Más ejemplos en la página 522](#).

#### Ejemplo 1

```
PathRecMoveFwd;
```

Se devuelve al robot a la posición en la cual la grabadora de trayectorias inició el movimiento de retroceso.

#### Argumentos

```
PathRecMoveFwd [\ID] [\ToolOffs] [\Speed]
```

#### [\ID]

##### *Identifier*

Tipo de dato: `pathrecid`

Una variable que especifica la posición de ID hasta la que se desea avanzar. El tipo de dato `pathrecid` es de un tipo sin valor y sólo se utiliza como un identificador para asignar un nombre a la posición de grabación.

Si no se especifica ninguna posición de ID, el movimiento de avance se realiza siempre hasta la posición de interrupción de la trayectoria original.

#### [\ToolOffs]

##### *Tool Offset*

Tipo de dato: `pos`

Proporciona un offset de separación del TCP durante el movimiento. Se aplica una coordenada cartesiana a las coordenadas del TCP. Resulta útil si el robot ejecuta un proceso en el que se añade material.

#### [\Speed]

Tipo de dato: `speeddata`

Esta velocidad tiene prioridad sobre la velocidad original utilizada durante el movimiento de avance. `Speeddata` define la velocidad del punto central de la herramienta, la reorientación de la misma y los ejes externos. Si está presente, esta velocidad se utiliza durante todo el movimiento de avance. Si se omite, el movimiento de avance se ejecuta a la velocidad de las instrucciones de movimiento originales.

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.165 PathRecMoveFwd - Hace avanzar la grabadora de trayectorias

### PathRecovery

#### Continuación

#### Ejecución de programas

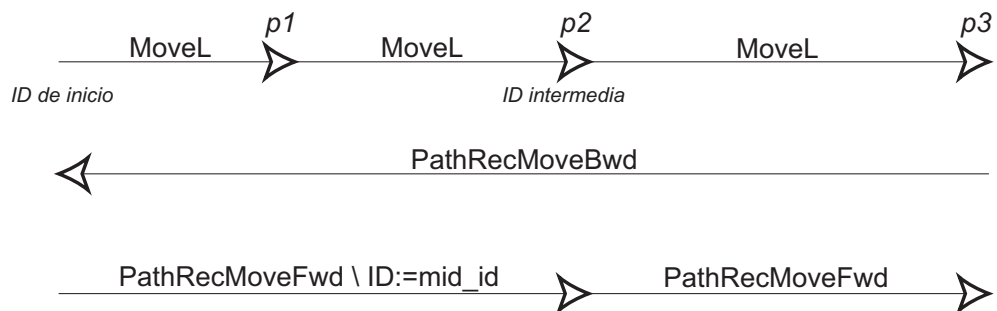
La grabadora de trayectorias se activa con la instrucción `PathRecStart`. Una vez que la grabadora se ha iniciado, el robot puede retroceder a lo largo de la trayectoria ejecutada, con ayuda de `PathRecMoveBwd`. Posteriormente, es posible solicitar al robot que vuelva a la posición en la que se inició la ejecución hacia atrás. Para ello se llama a `PathRecMoveFwd`. También es posible hacer avanzar parcialmente al robot, mediante un identificador que se entrega durante el movimiento de retroceso.

#### Más ejemplos

A continuación aparecen más ejemplos de cómo usar la instrucción `PathRecMoveFwd`.

```
VAR pathrecid start_id;
VAR pathrecid mid_id;
CONST robtarget p1 := [...];
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

En el ejemplo anterior, se inicia la grabadora de trayectorias y se marca el punto inicial con el identificador de trayectoria `start_id`. A partir de ese momento, el robot avanzará con las instrucciones de movimiento tradicionales y volverá al identificador de grabadora de trayectorias `start_id` mediante la trayectoria grabada. Por último, volverá a avanzar en dos pasos mediante `PathRecMoveFwd`.

Continúa en la página siguiente

## 1.165 PathRecMoveFwd - Hace avanzar la grabadora de trayectorias

*PathRecovery**Continuación***Limitaciones**

Los movimientos realizados utilizando la grabadora de trayectorias deben tener lugar en el nivel de la rutina TRAP. Es decir, es necesario ejecutar `StorePath` antes que `PathRecMoveFwd`.

Para poder ejecutar `PathRecMoveFwd`, es necesario haber ejecutado anteriormente una instrucción `PathRecMoveBwd`.

Si no se desea volver al punto en el que se ejecutó `PathRecMoveBwd` (mediante la ejecución de `PathRecMoveFwd`) es necesario detener la grabadora de trayectorias con la instrucción `PathRecStop`. `PathRecStop\Clear` también borra la trayectoria grabada.

`PathRecMoveFwd` no puede ejecutarse en rutinas de RAPID que estén conectadas a los siguientes eventos especiales del sistema: `PowerOn`, `Stop`, `QStop`, `Restart`, `Reset` o `Step`.

**Sintaxis**

```
PathRecMoveFwd '('
  [ '\ ' ID ' := ' < variable (VAR) of pathid > ]
  [ '\ ' ToolOffs ' := ' <expression (IN) of pos> ]
  [ '\ ' Speed ' := ' <expression (IN) of speeddata> ] ;'
```

**Información relacionada**

| Para obtener más información sobre                       | Consulte  |
|--|---|
| Identificadores de grabadora de trayectorias             | <a href="#">pathrecid - Identificador de grabadora de trayectorias en la página 1725</a>  |
| Inicio y detención de la grabadora de trayectorias       | <a href="#">PathRecStart - Inicia la grabadora de trayectorias en la página 524</a><br><a href="#">PathRecStop - Detiene la grabadora de trayectorias en la página 527</a>  |
| Comprobación de que la trayectoria grabada es correcta   | <a href="#">PathRecValidBwd - Comprueba si existe una trayectoria de retroceso válida guardada en la página 1410</a><br><a href="#">PathRecValidFwd - Comprueba si existe una trayectoria de avance válida guardada en la página 1413</a> |
| Reproducción de la grabación de trayectorias hacia atrás | <a href="#">PathRecMoveBwd - Hace retroceder la grabadora de trayectorias en la página 514</a>  |
| Almacenamiento y restauración de trayectorias            | <a href="#">StorePath - Almacena la trayectoria cuando se produce una interrupción en la página 835</a><br><a href="#">RestoPath - Restablece la trayectoria después de una interrupción en la página 615</a>                             |
| Otras instrucciones de posicionamiento                   | <a href="#">Manual de referencia técnica - RAPID Overview</a>   |
| Recuperación en caso de error                            | <a href="#">Manual de referencia técnica - RAPID Overview</a><br><a href="#">Manual de referencia técnica - RAPID Overview</a>  |

# 1 Instrucciones

---

## 1.166 PathRecStart - Inicia la grabadora de trayectorias

*Path Recovery*

## 1.166 PathRecStart - Inicia la grabadora de trayectorias

---

### Utilización

`PathRecStart` se utiliza para iniciar la grabación de la trayectoria del robot. La grabadora de trayectorias almacena la información de las trayectorias durante la ejecución del programa de RAPID.

---

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción `PathRecStart`:

#### Ejemplo 1

```
VAR pathrecid fixture_id;
```

```
PathRecStart fixture_id;
```

Se inicia la grabadora de trayectorias y el punto inicial (la posición de la instrucción en el programa de RAPID) se marca con el identificador `fixture_id`.

---

### Argumentos

```
PathRecStart ID
```

ID

#### *Identifier*

Tipo de dato: `pathrecid`

Una variable que especifica el nombre de la posición de inicio de la grabación. El tipo de dato `pathrecid` es de un tipo sin valor y sólo se utiliza como un identificador para asignar un nombre a la posición de grabación.

---

### Ejecución de programas

Cuando se solicita su inicio, la grabadora de trayectorias inicia la grabación interna de la trayectoria del robot en el controlador del robot. La secuencia de posiciones de programa grabada puede recorrerse en sentido inverso mediante `PathRecMoveBwd`, haciendo que el robot retroceda a lo largo de la trayectoria ejecutada.

---

### Más ejemplos

A continuación aparecen más ejemplos de cómo usar la instrucción `PathRecStart`.

#### Ejemplo 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;
```

*Continúa en la página siguiente*



## 1.166 PathRecStart - Inicia la grabadora de trayectorias

*Path Recovery**Continuación*

```

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;
    ELSE
      PathRecMoveBwd \ID:=corner_id;
    ENDIF
    Stop;
    !Fix problem
    PathRecMoveFwd;
    RestoPath;
    StartMove;
    RETRY;
  ENDIF

```

En el ejemplo anterior, la grabadora de trayectorias se utiliza para mover el robot hasta una posición de servicio en caso de que se produzca un error durante la ejecución normal.

El robot ejecuta el movimiento a lo largo de una trayectoria. Después de la posición  $p_1$ , se inicia la grabadora de trayectorias. Después del punto  $p_2$ , se inserta otro identificador de trayectoria. Suponga que se produce un error recuperable durante el movimiento desde la posición  $p_3$  hasta la posición  $p_4$ . En este momento puede ejecutarse el gestor de errores, y el usuario tiene la opción de extraer el robot hasta las posiciones `Origin` (punto  $p_1$ ) o `Corner` (punto  $p_2$ ). A continuación, se cambia el nivel de la trayectoria con `StorePath`, para poder reiniciar el movimiento más adelante en el lugar del error. Una vez que el robot ha retrocedido desde la posición del error, el usuario tiene la opción de resolver el error (normalmente solucionando el problema en el equipo que rodea al robot).

A continuación, se ordena el retroceso del robot hasta la posición del error. El nivel de la trayectoria se devuelve al nivel normal y se realiza un intento de recuperación.

**Limitaciones**

La grabadora de trayectorias sólo puede iniciarse y sólo grabará la trayectoria en el nivel de trayectoria de base. Es decir, los movimientos del nivel `StorePath` no se graban.

**Sintaxis**

```

PathRecStart
  [ ID := ' ] < variable (VAR) of pathrecid > ';'

```

**Información relacionada**

| Para obtener más información sobre           | Consulte   |
|--|--|
| Identificadores de grabadora de trayectorias | <a href="#">pathrecid - Identificador de grabadora de trayectorias en la página 1725</a> |

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.166 PathRecStart - Inicia la grabadora de trayectorias

*Path Recovery*

*Continuación*

| Para obtener más información sobre                         | Consulte  |
|--|---|
| Detención de la grabadora de trayectorias                  | <a href="#">PathRecStop - Detiene la grabadora de trayectorias en la página 527</a>   |
| Comprobación de que la trayectoria grabada es correcta     | <a href="#">PathRecValidBwd - Comprueba si existe una trayectoria de retroceso válida guardada en la página 1410</a><br><a href="#">PathRecValidFwd - Comprueba si existe una trayectoria de avance válida guardada en la página 1413</a> |
| Reproducción de la grabación de trayectorias hacia atrás   | <a href="#">PathRecMoveBwd - Hace retroceder la grabadora de trayectorias en la página 514</a>  |
| Reproducción de la grabación de trayectorias hacia delante | <a href="#">PathRecMoveFwd - Hace avanzar la grabadora de trayectorias en la página 521</a>   |
| Movimiento en general                                      | <a href="#">Manual de referencia técnica - RAPID Overview</a>   |

## 1.167 PathRecStop - Detiene la grabadora de trayectorias

### Utilización

`PathRecStop` se utiliza para detener la grabación de la trayectoria del robot.

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción `PathRecStop`:  
Consulte también *Más ejemplos*, a continuación.

#### Ejemplo 1

```
PathRecStop \Clear;
```

Se detiene la grabadora de trayectorias y se vacía el búfer de información de trayectorias grabada.

### Argumentos

```
PathRecStop [\Clear]
```

[\Clear]

Tipo de dato: `switch`

Borra la trayectoria grabada.

### Ejecución de programas

Cuando se solicita la detención de la grabadora de trayectorias, la grabación de la trayectoria se detiene. El argumento opcional `\Clear` elimina del búfer la información de la trayectoria grabada, impidiendo la ejecución accidental de la trayectoria grabada.

Una vez que la grabadora ha sido detenida con `PathRecStop`, no es posible utilizar las trayectorias grabadas anteriormente para los movimientos de retroceso (`PathRecMoveBwd`). Sí es posible utilizar trayectorias grabadas anteriormente si se ejecuta de nuevo `PathRecStart` desde la misma posición en la que fue detenida la grabadora de trayectorias. Consulte el ejemplo que aparece a continuación.

### Más ejemplos

A continuación aparecen más ejemplos de cómo usar la instrucción `PathRecStop`.

```
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;
```

*Continúa en la página siguiente*

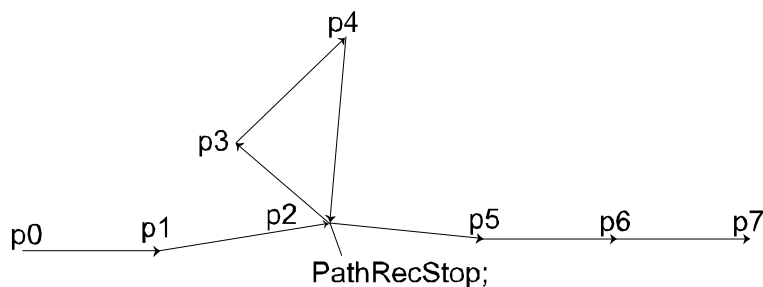
# 1 Instrucciones

## 1.167 PathRecStop - Detiene la grabadora de trayectorias

### Path Recovery

#### Continuación

```
PathRecStart id2;
MoveL p5, vmax, z50, tool1;
MoveL p6, vmax, z50, tool1;
StorePath;
PathRecMoveBwd \ID:=id1;
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\_

En los ejemplos anteriores se describe la grabación de la trayectoria del robot en el momento de la detención de la trayectoria en medio de una secuencia. En *example1*, la orden `PathRecMoveBwd \ID:=id1;` es válida y el robot ejecuta la trayectoria siguiente: p6 -> p5 -> p2 -> p1 -> p0

El motivo de que la orden sea válida es que la grabadora fue detenida e iniciada en la misma posición exacta del robot. Si no es un comportamiento deseado, la orden de detención debe incluir el argumento opcional `\Clear`. De esta forma, la trayectoria grabada se borrará y no volverá a ser posible retroceder hasta los identificadores anteriores de la grabadora de trayectorias.

Continúa en la página siguiente

## 1.167 PathRecStop - Detiene la grabadora de trayectorias

*Path Recovery*

*Continuación*

La única diferencia en el caso de `example2` es el punto en el que se inició la grabadora por segunda vez. En este caso, `PathRecMoveBwd \ID:=id1` genera un error. Esto se debe a que no existe ninguna trayectoria grabada entre `p4`, `p3` y `p2`. Es posible ejecutar `PathRecMoveBwd \ID:=id2`.

### Sintaxis

```
PathRecStop
[ '\switch Clear ] ';'

```

### Información relacionada

| Para obtener más información sobre                         | Consulte  |
|--|---|
| Identificadores de grabadora de trayectorias               | <a href="#">pathrecid - Identificador de grabadora de trayectorias en la página 1725</a>  |
| Inicio de la grabadora de trayectorias                     | <a href="#">PathRecStart - Inicia la grabadora de trayectorias en la página 524</a>   |
| Comprobación de que la trayectoria grabada es correcta     | <a href="#">PathRecValidBwd - Comprueba si existe una trayectoria de retroceso válida guardada en la página 1410</a><br><a href="#">PathRecValidFwd - Comprueba si existe una trayectoria de avance válida guardada en la página 1413</a> |
| Reproducción de la grabación de trayectorias hacia atrás   | <a href="#">PathRecMoveBwd - Hace retroceder la grabadora de trayectorias en la página 514</a>  |
| Reproducción de la grabación de trayectorias hacia delante | <a href="#">PathRecMoveFwd - Hace avanzar la grabadora de trayectorias en la página 521</a>   |
| Movimiento en general                                      | <a href="#">Manual de referencia técnica - RAPID Overview</a>   |

# 1 Instrucciones

---

1.168 PathResol - Ajusta la resolución de la trayectoria  
RobotWare - OS

## 1.168 PathResol - Ajusta la resolución de la trayectoria

---

### Utilización

PathResol (*Path Resolution*) se utiliza para ajustar el tiempo configurado de muestreo de trayectoria geométrica de los parámetros de sistema para las unidades mecánicas controladas desde la tarea de programa actual.

El tiempo de muestreo de la trayectoria geométrica se usa en aplicaciones con entradas de sensor. Observe que no se utiliza para definir la trayectoria geométrica en una aplicación normal, una funcionalidad heredada.

Esta instrucción sólo puede usarse en la tarea principal T\_ROB1 o, si se cuenta con un sistema *MultiMove*, en las tareas de movimiento.

### Descripción

Ejemplo de cuándo usar PathResol:

- Se usa la interpolación coordinada.
- Uso de *Weldguide*.
- Se usa la opción *Conveyor Tracking*.

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción PathResol:

```
MoveJ p1,v1000,fine,tool1;  
PathResol 150;
```

Con el robot en un punto de parada, el tiempo de muestreo de la trayectoria aumenta al 150 % del valor configurado.

### Argumentos

```
PathResol PathSampleTime
```

PathSampleTime

Tipo de dato: num

El ajuste, como porcentaje del tiempo de muestreo de trayectoria configurado. El 100% corresponde al tiempo de muestreo configurado para la trayectoria. Dentro del rango del 25% al 400%.

### Ejecución de programas

Las resoluciones de la trayectoria en todas las instrucciones posteriores de posicionamiento se ven afectadas hasta que se ejecuta una nueva instrucción PathResol. Esto afectará a la resolución de la ruta durante todas las operaciones de movimiento del programa (el nivel de trayectoria predeterminado y el nivel de trayectoria después de StorePath) y también durante el movimiento.

En un sistema MultiMove en el modo coordinado sincronizado son válidos los siguientes puntos:

- Todas las unidades mecánicas incluidas en el modo coordinado sincronizado se ejecutan con la resolución de trayectoria actual para el planificador de movimientos actual (utilizado).

*Continúa en la página siguiente*

### 1.168 PathResol - Ajusta la resolución de la trayectoria RobotWare - OS Continuación

- Un nuevo orden de resolución de trayectoria frente al planificador de movimientos actual afecta al movimiento coordinado sincronizado y al movimiento independiente posterior de ese planificador de movimientos.
- Un nuevo orden de resolución de trayectoria frente a otro planificador de movimientos sólo afecta al movimiento independiente posterior de ese planificador de movimientos.

Para más información acerca de la conexión entre la tarea de programa y el planificador de movimientos, consulte el *Manual de aplicaciones - MultiMove*.

El valor predeterminado para la redefinición del tiempo de muestreo de la trayectoria es el 100%. Este valor se configura automáticamente en los casos siguientes:

- cuando se utiliza el modo de reinicio **Restablecer RAPID**
- al cargar un nuevo programa o un nuevo módulo
- al iniciar la ejecución del programa desde el principio
- al mover el puntero del programa a `main`
- al mover el puntero del programa a una rutina
- al mover el puntero de programa de una forma que se pierde el orden de la ejecución.

El ajuste actual del tiempo de muestreo puede leerse a través de la variable `C_MOTSET` (tipo de dato `motsetdata`) del componente `pathresol`.

#### Limitación

Si esta instrucción va precedida de una instrucción de movimiento, ésta última debe programarse con un punto de paro (zonedata `fine`), no un punto de paso. De lo contrario, no será posible reanudar la ejecución tras una caída de suministro eléctrico.

`PathResol` no puede ejecutarse en rutinas de RAPID que estén conectadas a los siguientes eventos especiales del sistema: `PowerOn`, `Stop`, `QStop`, `Restart` o `Step`.

#### Sintaxis

```
PathResol
  [PathSampleTime ':=' ] <expression (IN) of num>';'
```

#### Información relacionada

| Para obtener más información sobre               | Consulte   |
|--|--|
| Instrucciones de posicionamiento                 | <i>Manual de referencia técnica - RAPID Overview</i>                             |
| Parámetros de movimiento                         | <i>Manual de referencia técnica - RAPID Overview</i>                             |
| Configuración de la resolución de la trayectoria | <i>Manual de referencia técnica - Parámetros del sistema</i>                     |
| Datos de parámetros de movimiento                | <a href="#">motsetdata - Datos de parámetros de movimiento en la página 1705</a> |

# 1 Instrucciones

1.169 PDispOff - Desactiva el desplazamiento de programa  
RobotWare - OS

## 1.169 PDispOff - Desactiva el desplazamiento de programa

### Utilización

`PDispOff` (*Program Displacement Off*) se utiliza para desactivar un desplazamiento de programa.

El desplazamiento de programa se activa mediante la instrucción `PDispSet` o `PDispOn` y se aplica a todos los movimientos hasta que se activa otro desplazamiento de programa o hasta que se desactiva el desplazamiento de programa.

Esta instrucción sólo puede usarse en la tarea principal `T_ROB1` o, si se cuenta con un sistema *MultiMove*, en las tareas de movimiento.

### Ejemplos básicos

Los ejemplos siguientes ilustran la instrucción `PDispOff`.

#### Ejemplo 1

```
PDispOff;
```

Desactivación del desplazamiento de programa.

#### Ejemplo 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;
```

El desplazamiento de programa se define como la diferencia entre las posiciones `p10` y `p11`. El desplazamiento afecta al movimiento hacia `p20` y `p30`, pero no hacia `p40`.

### Ejecución de programas

Se elimina el desplazamiento de programa que está activado. Esto significa que el sistema de coordenadas del desplazamiento de programa es el mismo que el sistema de coordenadas de objeto y por tanto todas las posiciones programadas están relacionadas entre sí.

### Sintaxis

```
PDispOff ' ;'
```

### Información relacionada

| Para obtener más información sobre   | Consulte  |
|--|---|
| Definición del desplazamiento de programa con dos posiciones                   | <a href="#">PDispOn - Activa el desplazamiento de programa en la página 533</a>   |
| Definición del desplazamiento de programa con una base de coordenadas conocida | <a href="#">PDispSet - Activa un desplazamiento de programa a partir de una base de coordenadas conocida en la página 538</a> |



## 1.170 PDispOn - Activa el desplazamiento de programa

### Utilización

`PDispOn` (*Program Displacement On*) se utiliza para definir y activar un nuevo desplazamiento de programa usando dos posiciones del robot.

El desplazamiento de programa se utiliza, por ejemplo, después de realizar una búsqueda o cuando se repiten patrones de movimiento similares en partes distintas del programa.

Esta instrucción sólo puede usarse en la tarea principal `T_ROB1` o, si se cuenta con un sistema *MultiMove*, en las tareas de movimiento.

### Ejemplos básicos

Los ejemplos siguientes ilustran la instrucción `PDispOn`.

Consulte también [Más ejemplos en la página 535](#).

#### Ejemplo 1

```
MoveL p10, v500, z10, tool1;
PDispOn \ExeP:=p10, p20, tool1;
```

Activación de un desplazamiento de programa (desplazamiento en paralelo). El offset se calcula a partir de la diferencia entre las posiciones `p10` y `p20`.

#### Ejemplo 2

```
MoveL p10, v500, fine \Inpos := inpos50, tool1;
PDispOn *, tool1;
```

Activación de un desplazamiento de programa (desplazamiento en paralelo). Dado que en la instrucción anterior se ha utilizado un punto de paro bien definido, no es necesario utilizar el argumento `\ExeP`. El desplazamiento se calcula a partir de la diferencia existente entre la posición real del robot y el punto programado (\*) almacenado en la instrucción.

#### Ejemplo 3

```
PDispOn \Rot \ExeP:=p10, p20, tool1;
```

Activación de un desplazamiento de programa con una rotación. El offset se calcula a partir de la diferencia entre las posiciones `p10` y `p20`.

### Argumentos

```
PDispOn [\Rot] [\ExeP] ProgPoint Tool [\WObj]
```

[ `\Rot` ]

#### *Rotation*

Tipo de dato: switch

La diferencia de orientación de la herramienta se tiene en cuenta, lo cual implica una rotación del programa.

[ `\ExeP` ]

#### *Executed Point*

Tipo de dato: robtarget

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.170 PDispOn - Activa el desplazamiento de programa

RobotWare - OS

Continuación

La nueva posición del robot, utilizada para el cálculo del desplazamiento. Si se omite el argumento, se utiliza la posición actual del robot en el momento de la ejecución del programa.

ProgPoint

**Programmed Point**

Tipo de dato: `robtarget`

La posición original del robot en el momento de la programación.

Tool

Tipo de dato: `tooldata`

La herramienta utilizada durante la programación, es decir, el TCP con el que está relacionada la posición `ProgPoint`.

[ \WObj ]

**Work Object**

Tipo de dato: `wobjdata`

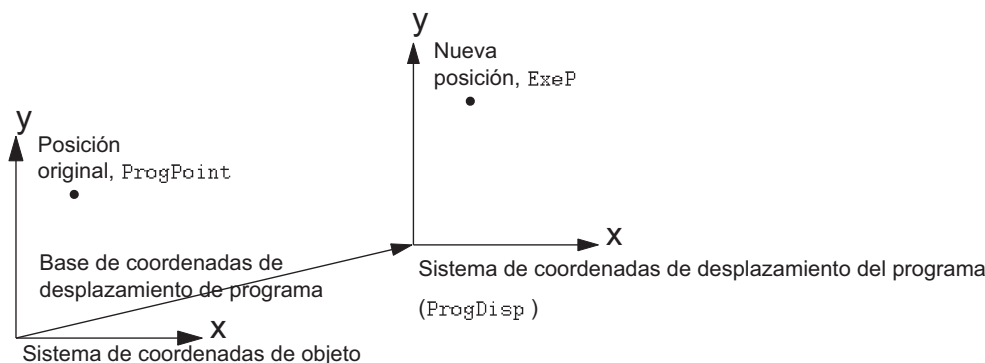
El objeto de trabajo (sistema de coordenadas) con el que está relacionada la posición `ProgPoint`.

Es posible omitir este argumento. Si se omite, la posición depende del sistema de coordenadas mundo. Si se utiliza un TCP estacionario o ejes externos coordinados, es necesario especificar este argumento.

Los argumentos `Tool` y `\WObj` se utilizan en conjunto para calcular la posición `ProgPoint` durante la programación y para calcular la posición actual durante la ejecución del programa si no se programa ningún argumento `\ExeP`.

## Ejecución de programas

El desplazamiento de programa implica que el sistema de coordenadas `ProgDisp` se traslada respecto del sistema de coordenadas de objeto. Dado que todas las posiciones dependen del sistema de coordenadas `ProgDisp`, se desplazan también todas las posiciones programadas. Consulte la figura siguiente, que muestra el desplazamiento lateral de una posición programada, con ayuda del desplazamiento de programa.



xx0500002186

El desplazamiento de programa se activa cuando la instrucción `PDispOn` se ejecuta y permanece activa hasta que se activa otro desplazamiento de programa (la

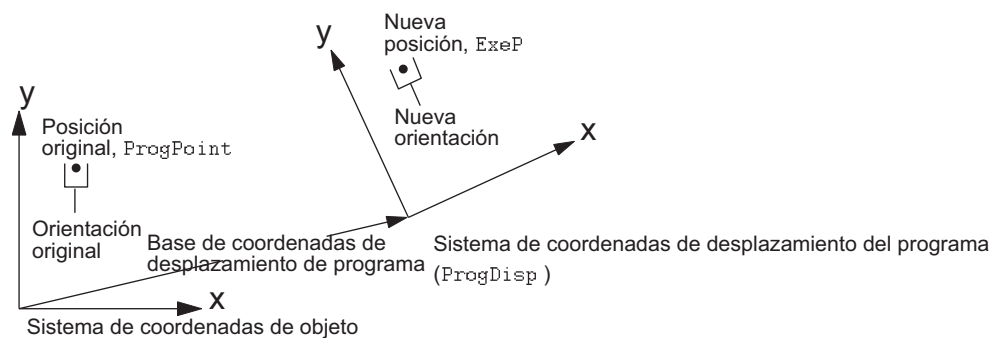
Continúa en la página siguiente

instrucción `PDispSet` o `PDispOn`) o hasta que se desactiva el desplazamiento de programa (la instrucción `PDispOff`).

Sólo puede estar activado un único desplazamiento de programa cada vez. Por otro lado, es posible programar varias instrucciones `PDispOn`, una tras otra y, en este caso, se suman los distintos desplazamientos de programa.

El desplazamiento de programa se calcula como la diferencia entre `ExeP` y `ProgPoint`. Si no se ha especificado el valor de `ExeP`, se utiliza en su lugar la posición actual del robot en el momento de la ejecución del programa. Dado que es la posición real del robot que se utiliza, el robot no debe estar en movimiento cuando se ejecuta `PDispOn`.

Si se utiliza el argumento `\Rot` la rotación se calcula también a partir de la orientación de la herramienta en las dos posiciones. El desplazamiento se calcula de forma que la nueva posición (`ExeP`) tendrá la misma posición y orientación respecto del sistema de coordenadas desplazado, `ProgDisp`, que las que tenía la posición anterior (`ProgPoint`) respecto del sistema de coordenadas original. Consulte la figura siguiente, que muestra la traslación y rotación de una posición programada.



xx0500002187

El desplazamiento de programa se elimina automáticamente en los casos siguientes:

- cuando se utiliza el modo de reinicio **Restablecer RAPID**
- al cargar un nuevo programa o un nuevo módulo
- al iniciar la ejecución del programa desde el principio
- al mover el puntero del programa a `main`
- al mover el puntero del programa a una rutina
- al mover el puntero de programa de una forma que se pierde el orden de la ejecución.

## Más ejemplos

A continuación aparecen más ejemplos de cómo usar la instrucción `PDispOn`.

### Ejemplo 1

```
PROC draw_square()
  PDispOn *, tool1;
  MoveL *, v500, z10, tool1;
  MoveL *, v500, z10, tool1;
```

*Continúa en la página siguiente*

# 1 Instrucciones

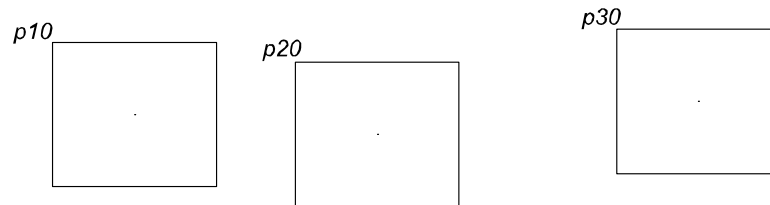
## 1.170 PDispOn - Activa el desplazamiento de programa

RobotWare - OS

Continuación

```
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;
```

La rutina `draw_square` se utiliza para ejecutar el mismo patrón de movimiento en tres posiciones diferentes, basándose en las posiciones `p10`, `p20` y `p30`. Consulte la figura siguiente, que muestra que cuando se usa el desplazamiento de programa, es posible reutilizar patrones de movimiento.



xx0500002185

### Ejemplo 2

```
SearchL sen1, psearch, p10, v100, tool1\WObj:=fixture1;  
PDispOn \ExeP:=psearch, *, tool1 \WObj:=fixture1;
```

Se realiza una búsqueda en la cual la posición buscada por el robot se almacena en la posición `psearch`. Cualquier movimiento realizado a continuación parte de esta posición utilizando un desplazamiento de programa (desplazamiento paralelo). El desplazamiento se calcula a partir de la diferencia existente entre la posición buscada y el punto programado (\*) almacenado en la instrucción. Todas las posiciones se basan en el sistema de coordenadas de objeto de `fixture1`.

### Sintaxis

```
PDispOn  
[[ '\ ' Rot ]  
[ '\ ' ExeP ' := ' <expression (IN) of robtarg> ], '  
[ ProgPoint ' := ' ] <expression (IN) of robtarg> ', '  
[ Tool ' := ' ] <persistent (PERS) of tooldata>  
[ '\ ' WObj ' := ' <persistent (PERS) of wobjdata> ]';'
```

### Información relacionada

| Para obtener más información sobre   | Consulte  |
|--|---|
| Desactivación del desplazamiento de programa                                   | <a href="#">PDispOff - Desactiva el desplazamiento de programa en la página 532</a>   |
| Definición del desplazamiento de programa con una base de coordenadas conocida | <a href="#">PDispSet - Activa un desplazamiento de programa a partir de una base de coordenadas conocida en la página 538</a> |

Continúa en la página siguiente

## 1.170 PDispOn - Activa el desplazamiento de programa

*RobotWare - OS*

*Continuación*

| Para obtener más información sobre | Consulte   |
|------------------------------------|--|
| Sistemas de coordenadas            | <i>Manual de referencia técnica - Parámetros del sistema</i>             |
| Definición de herramientas         | <a href="#">tooldata - Datos de herramienta en la página 1795</a>        |
| Definición de objetos de trabajo   | <a href="#">wobjdata - Datos del objeto de trabajo en la página 1823</a> |

## 1 Instrucciones

1.171 PDispSet - Activa un desplazamiento de programa a partir de una base de coordenadas conocida  
*RobotWare - OS*

### 1.171 PDispSet - Activa un desplazamiento de programa a partir de una base de coordenadas conocida

#### Utilización

PDispSet (*Program Displacement Set*) se usa para definir y activar un desplazamiento de programa usando la base de coordenadas conocida.

El desplazamiento de programa se utiliza, por ejemplo, cuando se repiten patrones de movimiento similares en partes distintas del programa.

Esta instrucción sólo puede usarse en la tarea principal T\_ROB1 o, si se cuenta con un sistema *MultiMove*, en las tareas de movimiento.

#### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción PDispSet:

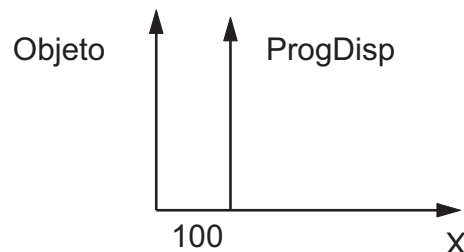
##### Ejemplo 1

```
VAR pose xp100 := [ [100, 0, 0], [1, 0, 0, 0] ];  
...  
PDispSet xp100;
```

Activación del desplazamiento de programa xp100, que implica que:

- El sistema de coordenadas ProgDisp se desplaza 100 mm respecto del sistema de coordenadas del objeto en la dirección del eje x positivo (consulte la figura siguiente).
- Siempre y cuando esté activado este desplazamiento de programa, todas las posiciones se desplazan 100 mm en la dirección del eje x.

La figura muestra un desplazamiento de programa de 100 mm a lo largo del eje x.



#### Argumentos

PDispSet DispFrame

DispFrame

*Displacement Frame*

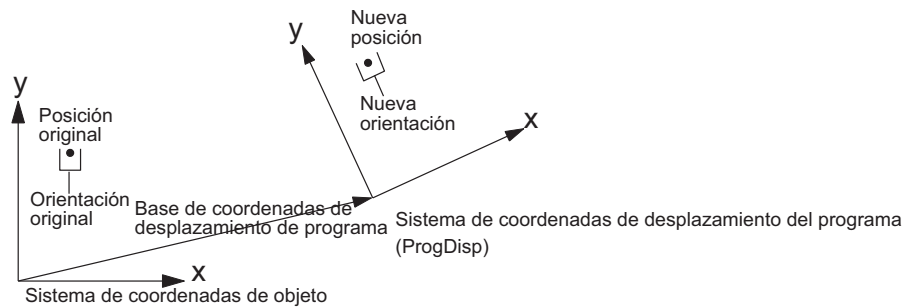
Tipo de dato: pose

El desplazamiento de programa se define mediante un dato del tipo pose.

*Continúa en la página siguiente*

Ejecución de programas

El desplazamiento de programa implica la traslación y/o la rotación del sistema de coordenadas `ProgDisp` respecto del sistema de coordenadas de objeto. Dado que todas las posiciones dependen del sistema de coordenadas `ProgDisp`, se desplazan también todas las posiciones programadas. Consulte la figura siguiente, que muestra la traslación y rotación de una posición programada.



xx0500002204

El desplazamiento de programa se activa cuando la instrucción `PDispOn` se ejecuta y permanece activa hasta que se activa otro desplazamiento de programa (la instrucción `PDispSet` o `PDispOn`) o hasta que se desactiva el desplazamiento de programa (la instrucción `PDispOff`).

Sólo puede estar activado un único desplazamiento de programa cada vez. No es posible sumar desplazamientos de programa mediante `PDispSet`.

El desplazamiento de programa se elimina automáticamente en los casos siguientes:

- cuando se utiliza el modo de reinicio **Restablecer RAPID**
- al cargar un nuevo programa o un nuevo módulo
- al iniciar la ejecución del programa desde el principio
- al mover el puntero del programa a `main`
- al mover el puntero del programa a una rutina
- al mover el puntero de programa de una forma que se pierde el orden de la ejecución.

Sintaxis

```
PDispSet
  [ DispFrame ::= ] < expression (IN) of pose> ';'

```

Información relacionada

| Para obtener más información sobre                           | Consulte  |
|--|---|
| Desactivación del desplazamiento de programa                 | <a href="#">PDispOff - Desactiva el desplazamiento de programa en la página 532</a> |
| Definición del desplazamiento de programa con dos posiciones | <a href="#">PDispOn - Activa el desplazamiento de programa en la página 533</a>     |
| Definición de datos del tipo pose                            | <a href="#">pose - Transformaciones de coordenadas en la página 1731</a>            |

Continúa en la página siguiente

## 1 Instrucciones

---

1.171 PDispSet - Activa un desplazamiento de programa a partir de una base de coordenadas conocida  
*RobotWare - OS*

*Continuación*

| Para obtener más información sobre                          | Consulte   |
|---|--|
| Sistemas de coordenadas                                     | <i>Manual de referencia técnica - RAPID Overview</i>                                   |
| Ejemplos de cómo puede usarse el desplazamiento de programa | <a href="#"><i>PDispOn - Activa el desplazamiento de programa en la página 533</i></a> |



### 1.172 ProcCall - Llama a un nuevo procedimiento

#### Utilización

Las llamadas a procedimientos se utilizan para transferir la ejecución del programa a otro procedimiento. Una vez ejecutado en su totalidad el procedimiento, la ejecución del programa continúa con la instrucción que aparece a continuación de la llamada al procedimiento.

Normalmente, es posible enviar un conjunto de argumentos al nuevo procedimiento. Estos argumentos controlan el comportamiento del procedimiento y permiten utilizar un mismo procedimiento para distintas operaciones.

#### Ejemplos básicos

Los ejemplos siguientes ilustran la instrucción `ProcCall`.

##### Ejemplo 1

```
weldpipe1;
```

Llama al procedimiento `weldpipe1` .

##### Ejemplo 2

```
errormessage;  
Set dol;  
...  
PROC errormessage()  
  TPWrite "ERROR";  
ENDPROC
```

Se llama al procedimiento `errormessage` . Cuando el procedimiento se completa, la ejecución del programa vuelve a la instrucción que aparece a continuación de la llamada al procedimiento, `Set dol`.

#### Argumentos

```
Procedure { Argument }
```

Procedure

##### *Identifier*

El nombre del procedimiento al que se llama.

Argument

Tipo de dato: Según la declaración del procedimiento.

Los argumentos del procedimiento (según los parámetros del procedimiento).

#### Ejemplos básicos

A continuación aparecen algunos ejemplos básicos de la instrucción `ProcCall`.

##### Ejemplo 1

```
weldpipe2 10, lowspeed;
```

Llama al procedimiento `weldpipe2`, con dos argumentos.

##### Ejemplo 2

```
weldpipe3 10 \speed:=20;
```

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.172 ProcCall - Llama a un nuevo procedimiento

RobotWare - OS

Continuación

Llama al procedimiento `weldpipe3`, con un argumento obligatorio y un argumento opcional.

---

### Limitaciones

Los argumentos deben coincidir con sus parámetros:

- Todos los argumentos obligatorios deben incluirse.
- Es necesario situarlos en el mismo orden.
- Deben ser del mismo tipo de datos.
- Deben ser del tipo correcto en cuanto al modo de acceso (entrada, variable o persistente).

Una rutina puede llamar a otra rutina, que a su vez puede llamar a otra. Una rutina también puede llamarse a sí misma, lo que se conoce como llamada recursiva. El número de niveles de rutinas permitido depende del número de parámetros.

Normalmente se permiten más de 10 niveles.

---

### Sintaxis

```
<procedure> [ <argument list> ] ';' 
```

---

### Información relacionada

| Para obtener más información sobre | Consulte   |
|------------------------------------|--|
| Argumentos, parámetros             | <i>Manual de referencia técnica - RAPID Overview</i> |

1.173 ProcerrRecovery - Genera errores de movimiento de proceso y permite la recuperación tras ellos  
RobotWare - OS

## 1.173 ProcerrRecovery - Genera errores de movimiento de proceso y permite la recuperación tras ellos

### Utilización

ProcerrRecovery puede usarse para generar un error de proceso durante un movimiento del robot y ofrece la posibilidad de gestionar el error y reiniciar el proceso y el movimiento desde un gestor de ERROR.

### Ejemplos básicos

Los ejemplos siguientes ilustran la instrucción ProcerrRecovery.

Consulte también [Más ejemplos en la página 544](#).

Los ejemplos siguientes no son realistas, pero se añaden por motivos pedagógicos.

#### Ejemplo 1

```
MoveL p1, v50, z30, tool2;
ProcerrRecovery \SyncOrgMoveInst;
MoveL p2, v50, z30, tool2;
ERROR
  IF ERRNO = ERR_PATH_STOP THEN
    StartMove;
    RETRY;
  ENDIF
```

El movimiento del robot se detiene en su camino hacia p1 y la ejecución del programa se transfiere al gestor de ERROR de la rutina que creó la trayectoria actual en la que se ha producido el error, en este caso la trayectoria hacia MoveL p1. El movimiento se reinicia con StartMove, mientras que la ejecución continúa con RETRY.

#### Ejemplo 2

```
MoveL p1, v50, fine, tool2;
ProcerrRecovery \SyncLastMoveInst;
MoveL p2, v50, z30, tool2;
ERROR
  IF ERRNO = ERR_PATH_STOP THEN
    StartMove;
    RETRY;
  ENDIF
```

El movimiento del robot se detiene inmediatamente en su camino hacia p2. La ejecución del programa se transfiere al gestor de ERROR de la rutina en la que se está ejecutando actualmente el programa o en la que se iba a ejecutar una instrucción de movimiento cuando se produjo el error, en este caso MoveL p2. El movimiento se reinicia con StartMove, mientras que la ejecución continúa con RETRY.

### Argumentos

```
ProcerrRecovery[\SyncOrgMoveInst] | [\SyncLastMoveInst]
[\ProcSignal]
```

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.173 ProcerrRecovery - Genera errores de movimiento de proceso y permite la recuperación tras ellos RobotWare - OS Continuación

[\SyncOrgMoveInst]

Tipo de dato: switch

El error puede ser gestionado en la rutina que creó la trayectoria actual en la que se produjo el error.

[\SyncLastMoveInst]

Tipo de dato: switch

El error puede ser gestionado en la rutina en la que el programa esté ejecutando una instrucción de movimiento en el momento en que se produjo el error.

Si el programa no está ejecutando ninguna instrucción de movimiento en el momento en el que se produce el error, la transferencia de la ejecución al gestor de ERROR se retrasa hasta que el programa ejecuta la siguiente instrucción de movimiento. Esto significa que la transferencia al gestor de ERROR se retrasará si el robot se encuentra en un punto de paro o entre el punto de precaptura del centro de la trayectoria de esquina. El error puede ser gestionado en esa rutina.

[\ProcSignal]

Tipo de dato: signaldo

Un parámetro opcional que permite al usuario activar o desactivar el uso de la instrucción. Si se utiliza este parámetro y el valor de la señal es 0, se genera un error recuperable y no se generará ningún error de proceso.

---

## Ejecución de programas

La ejecución de ProcerrRecovery en el modo continuo da lugar a lo siguiente:

- Instantáneamente, el robot se detiene en su trayectoria.
- Se cambia la variable ERRNO a ERR\_PATH\_STOP.
- La ejecución se transfiere a algún gestor de ERROR, de acuerdo con las reglas para errores elevados asíncronamente.

Esta instrucción no hace nada en el modo de ejecución paso a paso.

Para una descripción de los errores elevados asíncronamente que se generan con ProcerrRecovery, consulte *Technical reference manual - RAPID kernel*.

ProcerrRecovery también puede usarse en los sistemas MultiMove para transferir la ejecución al gestor de ERROR de varias tareas de programa si se utiliza el modo sincronizado.

---

## Más ejemplos

A continuación aparecen más ejemplos de cómo usar la instrucción ProcerrRecovery.

### Ejemplo con ProcerrRecovery\SyncOrgMoveInst

```
MODULE user_module
  VAR intnum proc_sup_int;

  PROC main()
    ...
    MoveL p1, v1000, fine, tool1;
    do_process;
```

*Continúa en la página siguiente*

## 1.173 ProcerrRecovery - Genera errores de movimiento de proceso y permite la recuperación tras ellos

RobotWare - OS

Continuación

```

...
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;
  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

```

En este ejemplo, los errores elevados asincrónicamente generados por ProcerrRecovery con el modificador \SyncOrgMoveInst pueden ser tratados en la rutina do\_process, porque la trayectoria en la que se produjo el error siempre se crea en la rutina o\_process.

El flujo de proceso se inicia cambiando la señal do\_myproc a 1. La señal di\_proc\_sup supervisa el proceso y se genera un error asíncrono si di\_proc\_sup pasa a tener el valor 1. En este ejemplo sencillo, el error se resuelve cambiando de nuevo el valor de do\_myproc a 1 antes de reanudar el movimiento.

## Ejemplo con ProcerrRecovery\SyncLastMoveInst

```

MODULE user_module
PROC main()
  ...
  MoveL p1, v1000, fine, tool1;
  do_process;
  ...
ENDPROC
PROC do_process()
  proc_on;

```

Continúa en la página siguiente

# 1 Instrucciones

---

## 1.173 ProcerrRecovery - Genera errores de movimiento de proceso y permite la recuperación tras ellos

RobotWare - OS

Continuación

```
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;

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 (robtargt 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
  ENDIF
ENDPROC
ENDMODULE
```

*Continúa en la página siguiente*

En este ejemplo, los errores elevados asincrónicamente generados por ProcerrRecovery con el modificador \SyncLastMoveInst pueden ser tratados en la rutina proc\_move debido a que todas las instrucciones de movimiento se crean siempre en la rutina proc\_move. Si el puntero de programa se encuentra en la rutina do\_process, la transferencia al gestor de ERROR se retrasará hasta la ejecución de la siguiente instrucción MoveL en la rutina proc\_move. Recuerde que los movimientos también se detienen inmediatamente.

El flujo de proceso se inicia cambiando la señal do\_myproc a 1. La señal di\_proc\_sup supervisa el proceso y se genera un error asíncrono si di\_proc\_sup pasa a tener el valor 1. En este ejemplo sencillo, el error se resuelve cambiando de nuevo el valor de do\_myproc a 1 antes de reanudar el movimiento.

Si se utiliza la rutina NOSTEPIN predefinida, recomendamos utilizar el parámetro de modificación de opción \SyncLastMoveInst, dado que la rutina predefinida también puede tomar la decisión de gestionar una situación de error dentro de la rutina, mientras que en otros casos deberá ser realizado por el usuario final.

### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema ERRNO cambiará a:

| Nombre             | Causa del error   |
|--------------------|---|
| ERR_NO_ALIASIO_DEF | La variable de señal es una variable declarada en RAPID. No se ha conectado a una señal E/S definida en la configuración de E/S con la instrucción AliasIO.   |
| ERR_NORUNUNIT      | Se ha perdido el contacto con el dispositivo de E/S.  |
| ERR_PATH_STOP      | Ejecución de ProcerrRecovery en modo continuo.  |
| ERR_PROCSIGNAL_OFF | Se utiliza el parámetro opcional \ProcSignal y la señal está desactivada al ejecutar la instrucción.  |
| ERR_SIG_NOT_VALID  | La señal de E/S no está disponible. Los motivos pueden ser que el dispositivo de E/S no está en funcionamiento o que exista un error en la configuración (válido solamente para el bus de campo ICI). |

### Limitaciones

La recuperación de errores desde los errores de proceso elevados asincrónicamente sólo puede hacerse si la tarea de movimiento que contiene la instrucción de movimiento se ejecuta en el nivel de base en el momento en el que se produce el error de proceso. Por tanto, la recuperación de errores no puede hacerse si la tarea de programa que contiene la instrucción de proceso se ejecuta dentro de:

- Cualquier rutina de evento
- Cualquier gestor de rutina (ERROR, BACKWARD o UNDO)
- Nivel de ejecución de usuario (rutina de servicio)

Consulte *Technical reference manual - RAPID kernel, Recuperación en caso de error, Errores elevados asincrónicamente*.

Continúa en la página siguiente

# 1 Instrucciones

---

1.173 ProcerrRecovery - Genera errores de movimiento de proceso y permite la recuperación tras ellos

RobotWare - OS

Continuación

Si no se utiliza ningún gestor de errores con StartMove + RETRY o se utiliza StartMoveRetry, la ejecución del programa se colgará. La única forma de restablecer esta situación es trasladar el PP a Main.

---

## Sintaxis

```
ProcerrRecovery  
[ '\ SyncOrgMoveInst ] | [ '\ SyncLastMoveInst ]  
[ '\ ProcSignal' := ' ] < variable (VAR) of signaldo > ';' 
```

---

## Información relacionada

| Para obtener más información sobre                     | Consulte   |
|--|--|
| Gestores de errores                                    | <i>Manual de referencia técnica - RAPID Overview</i>   |
| Errores elevados asincrónicamente                      | <i>Technical reference manual - RAPID kernel, Recuperar en caso de error</i>                     |
| Propagación de un error al nivel del usuario           | <a href="#">RaiseToUser - Propaga un error al nivel del usuario en la página 574</a>             |
| Reanudación del movimiento y la ejecución de programas | <a href="#">StartMoveRetry - Reanuda el movimiento y la ejecución del robot en la página 799</a> |



## 1.174 PrxActivAndStoreRecord - Activación y almacenamiento de los datos de perfil grabados

*Machine Synchronization*

### 1.174 PrxActivAndStoreRecord - Activación y almacenamiento de los datos de perfil grabados

#### Utilización

PrxActivAndStoreRecord se utiliza para activar los datos de perfil grabados y almacenarlos en un archivo.

Puede usarse en lugar de llamar a PrxActivRecord y PrxStoreRecord.

#### Ejemplo básico

```
PrxActivAndStoreRecord SSYNCl, 1, "profile.log";
```

El perfil de movimiento de sensor activado y almacenado en el archivo *profile.log*.

#### Argumentos

```
PrxActivAndStoreRecord MechUnit Delay File_name
```

MechUnit

Tipo de dato: mechunit

El objeto de unidad mecánica móvil al que está sincronizado el movimiento del robot.

Delay

Tipo de dato: num

Es posible usar el retardo en segundos para desplazar la grabación en el tiempo. Debe estar entre 0,01 y 0,1. Si se indica el valor 0, no se añade ningún retardo. El retardo no se guarda en el perfil; solo se usa para la activación. Si se debe utilizar el retardo junto con un perfil guardado, es necesario especificar el retardo de nuevo en la instrucción PrxUseFileRecord.

File\_name

Tipo de dato: string

Nombre del archivo en el que se almacena el perfil.

#### Ejecución de programas

PrxActivAndStoreRecord debe ejecutarse al menos 0,2 segundos antes del inicio del movimiento del sensor, si se desea usar el registro para la sincronización.

#### Gestión de errores

Pueden generarse los errores recuperables siguientes. Los errores pueden ser gestionados en un gestor de errores. La variable de sistema ERRNO cambia a:

| Nombre         | Causa del error                |
|----------------|--------------------------------|
| ERR_ACTIV_PROF | Error en el perfil activado.   |
| ERR_STORE_PROF | Error en el perfil almacenado. |
| ERR_USE_PROF   | Error en el perfil usado.      |

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.174 PrxActivAndStoreRecord - Activación y almacenamiento de los datos de perfil grabados

### *Machine Synchronization*

#### *Continuación*

#### Sintaxis

```
PrxActivAndStoreRecord
  [ MechUnit ':= ' ] < expression (IN) of mechunit > ','
  [ Delay ':= ' ] < expression (IN) of num > ','
  [ File_name ':= ' ] < expression (IN) of string > ';'

```

#### Información relacionada

| Para obtener más información sobre             | Consulte   |
|--|--|
| Activación de los datos de perfil grabados     | <a href="#">PrxActivRecord - Activación de los datos de perfil grabados en la página 551</a>     |
| Desactivación de un registro                   | <a href="#">PrxDeactRecord - Desactivación de un registro en la página 554</a>                   |
| Almacenamiento de los datos de perfil grabados | <a href="#">PrxStoreRecord - Almacenamiento de los datos de perfil grabados en la página 564</a> |
| <i>Machine Synchronization</i>                 | <i>Application manual - Controller software IRC5</i>   |

## 1.175 PrxActivRecord - Activación de los datos de perfil grabados

### Utilización

PrxActivRecord se utiliza para activar el registro que se acaba de guardar, para poder usarlo sin tener que guardarlo de antemano.

### Ejemplo básico

```
PrxActivRecord SSYNCl, 0;
WaitTime 0.2;
SetDO do_startstop_machine, 1;
!Work synchronized with sensor
...
SetDO do_startstop_machine, 0;
```

La grabación del sensor se activa y se utiliza para la predicción del movimiento del sensor tan pronto como el registro está preparado.

### Argumentos

```
PrxActivRecord MechUnit Delay
```

MechUnit

Tipo de dato: mechunit

El objeto de unidad mecánica móvil al que está sincronizado el movimiento del robot.

Delay

Tipo de dato: num

Es posible usar el retardo en segundos para desplazar la grabación en el tiempo. Debe estar entre 0,01 y 0,1. Si se indica el valor 0, no se añade ningún retardo.

### Ejecución de programas

PrxActivRecord debe ejecutarse al menos 0,2 segundos antes del inicio del movimiento del transportador.

### Gestión de errores

Pueden generarse los errores recuperables siguientes. Los errores pueden ser gestionados en un gestor de errores. La variable de sistema ERRNO cambia a:

| Nombre         | Causa del error               |
|----------------|-------------------------------|
| ERR_ACTIV_PROF | Error en el perfil activado   |
| ERR_STORE_PROF | Error en el perfil almacenado |
| ERR_USE_PROF   | Error en el perfil usado      |

### Sintaxis

```
PrxActivRecord
[ MechUnit ':= ' ] < expression (IN) of mechunit > ', '
[ Delay ':= ' ] < expression (IN) of num > ';'
```

Continúa en la página siguiente

# 1 Instrucciones

---

## 1.175 PrxActivRecord - Activación de los datos de perfil grabados

*Machine Synchronization*

*Continuación*

---

### Información relacionada

| Para obtener más información sobre                          | Consulte  |
|---|---|
| Activación y almacenamiento de los datos de perfil grabados | <a href="#">PrxActivAndStoreRecord - Activación y almacenamiento de los datos de perfil grabados en la página 549</a> |
| Desactivación de un registro                                | <a href="#">PrxDeactRecord - Desactivación de un registro en la página 554</a>  |
| Almacenamiento de los datos de perfil grabados              | <a href="#">PrxStoreRecord - Almacenamiento de los datos de perfil grabados en la página 564</a>                      |
| <i>Machine Synchronization</i>                              | <i>Application manual - Controller software IRC5</i>  |

## 1.176 PrxDbgStoreRecord - Almacenamiento y depuración de los datos de perfil grabados

*Machine Synchronization*

### 1.176 PrxDbgStoreRecord - Almacenamiento y depuración de los datos de perfil grabados

#### Utilización

PrxDbgStoreRecord se utiliza para almacenar un registro no activado, para su depuración.

Puede usarse para comparar grabaciones y verificar la repetibilidad.

#### Ejemplo básico

```
PrxDbgStoreRecord SSYNCl, "debug_profile.log";
```

Guarda la grabación en el archivo *debug\_profile.log*.

#### Argumentos

```
PrxDbgStoreRecord MechUnit Filename
```

MechUnit

Tipo de dato: mechunit

El objeto de unidad mecánica móvil al que está sincronizado el movimiento del robot.

File\_name

Tipo de dato: string

Nombre del archivo en el que se almacena la grabación.

#### Sintaxis

```
PrxDbgStoreRecord
  [ MechUnit ':= ' ] < expression (IN) of mechunit > ','
  [ File_name ':= ' ] < expression (IN) of string > ';'

```

#### Información relacionada

| Para obtener más información sobre                          | Consulte  |
|---|---|
| Activación y almacenamiento de los datos de perfil grabados | <a href="#">PrxActivAndStoreRecord - Activación y almacenamiento de los datos de perfil grabados en la página 549</a> |
| Activación de los datos de perfil grabados                  | <a href="#">PrxActivRecord - Activación de los datos de perfil grabados en la página 551</a>                          |
| Almacenamiento de los datos de perfil grabados              | <a href="#">PrxStoreRecord - Almacenamiento de los datos de perfil grabados en la página 564</a>                      |
| <i>Machine Synchronization</i>                              | <i>Application manual - Controller software IRC5</i>  |

# 1 Instrucciones

---

## 1.177 PrxDeactRecord - Desactivación de un registro *Machine Synchronization*

### 1.177 PrxDeactRecord - Desactivación de un registro

---

#### Utilización

PrxDeactRecord se utiliza para desactivar un registro.

---

#### Ejemplo básico

```
PrxDeactRecord SSYNCl;
```

La grabación del movimiento del sensor se desactiva y ya no se utiliza para la predicción del movimiento del sensor. El registro puede activarse de nuevo.

---

#### Argumentos

```
PrxDeactRecord MechUnit
```

MechUnit

Tipo de dato: mechunit

El objeto de unidad mecánica móvil al que está sincronizado el movimiento del robot.

---

#### Limitaciones

No se deben realizar llamadas PrxDeactRecord durante la sincronización.

---

#### Sintaxis

```
PrxDeactRecord  
[ MechUnit ':=' ] < expression (IN) of mechunit> ';' 
```

---

#### Información relacionada

| Para obtener más información sobre         | Consulte   |
|--|--|
| Activación de los datos de perfil grabados | <a href="#">PrxActivRecord - Activación de los datos de perfil grabados en la página 551</a> |
| <i>Machine Synchronization</i>             | <i>Application manual - Controller software IRC5</i>   |

## 1.178 PrxResetPos - Restablecimiento de la posición cero del sensor

### *Machine Synchronization*

## 1.178 PrxResetPos - Restablecimiento de la posición cero del sensor

### Utilización

PrxResetPos se utiliza para restablecer la posición cero del sensor.

La posición del sensor se restablece para la funcionalidad de sincronización y el archivo grabado, pero el valor de la señal de E/S no se restablece. Esta instrucción se utiliza para el restablecimiento de software de la entrada del sensor cuando no se dispone de ningún interruptor de sincronización para restablecer la señal de E/S.

### Ejemplo básico

```
PrxResetPos SSYNC1;
```

La posición del sensor se cambia a cero.

### Argumentos

```
PrxResetPos MechUnit
```

MechUnit

**Tipo de dato:** mechunit

El objeto de unidad mecánica móvil al que está sincronizado el movimiento del robot.

### Ejecución de programas

La unidad de sensor debe estar parada (en la posición cero deseada) antes de llamar a PrxResetPos.

### Limitaciones

No debe utilizarse junto con la tarjeta DSQC 377A.

Esta instrucción equivale a un interruptor de sincronización. La ventana de movimiento debe indicar 0,0 como una posición de eje adicional tras la ejecución de esta instrucción.

### Sintaxis

```
PrxResetPos
  [ MechUnit ':' ] < expression (IN) of mechunit > ';'

```

### Información relacionada

| Para obtener más información sobre                           | Consulte  |
|--|---|
| Establecimiento de una posición de referencia para el sensor | <a href="#">PrxSetPosOffset - Establecimiento de una posición de referencia para el sensor en la página 557</a> |
| <i>Machine Synchronization</i>                               | <i>Application manual - Controller software IRC5</i>  |

# 1 Instrucciones

---

## 1.179 PrxResetRecords - Restablecimiento y desactivación de todos los registros *Machine Synchronization*

### 1.179 PrxResetRecords - Restablecimiento y desactivación de todos los registros

---

#### Utilización

PrxResetRecords se utiliza para restablecer y desactivar todos los registros.

---

#### Ejemplo básico

```
PrxResetRecords SSYNCl;
```

La grabación del movimiento del sensor se desactiva y ya no se utiliza para la predicción del movimiento del sensor y los datos de registro se eliminan.

---

#### Argumentos

```
PrxResetRecords MechUnit
```

MechUnit

Tipo de dato: mechunit

El objeto de unidad mecánica móvil al que está sincronizado el movimiento del robot.

---

#### Ejecución de programas

PrxResetRecords debe ejecutarse al menos 0,2 segundos antes del inicio del movimiento del transportador.

---

#### Sintaxis

```
PrxResetRecords  
[ MechUnit '[:=' ] < expression (IN) of mechunit> ';' ]
```

---

#### Información relacionada

| Para obtener más información sobre | Consulte   |
|------------------------------------|--|
| <i>Machine Synchronization</i>     | <i>Application manual - Controller software IRC5</i> |



## 1.180 PrxSetPosOffset - Establecimiento de una posición de referencia para el sensor

*Machine Synchronization*

### 1.180 PrxSetPosOffset - Establecimiento de una posición de referencia para el sensor

#### Utilización

`PrxSetPosOffset` se utiliza para establecer una posición de referencia para el sensor.

La posición del sensor se establece como referencia para la funcionalidad de sincronización y el archivo grabado. Esta función se utiliza para un establecimiento de software de la referencia de sensor cuando no se dispone de ningún interruptor de sincronización para restablecer la señal de E/S.

#### Ejemplo básico

```
PrxSetPosOffset SSYNCl, reference;
```

La posición del sensor se cambia al valor de referencia.

#### Argumentos

```
PrxSetPosOffset MechUnit Reference
```

MechUnit

Tipo de dato: `mechunit`

El objeto de unidad mecánica móvil al que está sincronizado el movimiento del robot.

Reference

Tipo de dato: `num`

La referencia en metros (o la unidad de sensor). Debe estar entre -5000 y 5000.

#### Ejecución de programas

La unidad de sensor debe estar parada antes de llamar a `PrxSetPosOffset`.

#### Limitaciones

No debe utilizarse junto con la tarjeta DSQC 377A.

#### Sintaxis

```
PrxSetPosOffset
  [ MechUnit ':' ] < expression (IN) of mechunit > ','
  [ Reference ':' ] < expression (IN) of num > ';'

```

#### Información relacionada

| Para obtener más información sobre              | Consulte   |
|---|--|
| Restablecimiento de la posición cero del sensor | <a href="#">PrxResetPos - Restablecimiento de la posición cero del sensor en la página 555</a> |
| <i>Machine Synchronization</i>                  | <i>Application manual - Controller software IRC5</i>   |

# 1 Instrucciones

---

## 1.181 PrxSetRecordSampleTime - Establecimiento del tiempo de muestreo para la grabación de un perfil

*Machine Synchronization*

## 1.181 PrxSetRecordSampleTime - Establecimiento del tiempo de muestreo para la grabación de un perfil

---

### Utilización

PrxSetRecordSampleTime se utiliza para establecer el tiempo de muestreo, en segundos, para la grabación de un registro.

El tiempo de muestreo predeterminado se toma del parámetro de sistema *Pos Update time*, perteneciente al tipo *CAN interface* del tema *Process*. Recuerde que *Pos Update time* especifica el tiempo de muestreo en milisegundos, mientras que *PrxSetRecordSampleTime* especifica el tiempo de muestreo en segundos.

El número máximo de muestras de un perfil grabado es de 300. Si una grabación tiene más de  $300 * Pos Update time$ , es necesario aumentar el tiempo de muestreo.

---

### Ejemplo básico

Es necesario realizar una grabación de 12 segundos. El tiempo de muestreo no puede ser inferior a  $12/300 = 0,04$ . Por tanto, el tiempo de muestreo es de 0,04 segundos.

```
PrxSetRecordSampleTime SSYNC1, 0.04;
```

---

### Argumentos

```
PrxSetRecordSampleTime MechUnit SampleTime
```

MechUnit

**Tipo de dato:** mechunit

El objeto de unidad mecánica móvil al que está sincronizado el movimiento del robot.

SampleTime

**Tipo de dato:** num

El tiempo de muestreo en segundos. El tiempo de muestreo debe estar entre 0,01 y 0,1.

---

### Sintaxis

```
PrxSetRecordSampleTime  
  [ MechUnit ':= ' ] < expression (IN) of mechunit> ','  
  [ SampleTime ':= ' ] < expression (IN) of num> ';' ;
```

---

### Información relacionada

| Para obtener más información sobre | Consulte   |
|------------------------------------|--|
| <i>Machine Synchronization</i>     | <i>Application manual - Controller software IRC5</i> |

## 1.182 PrxSetSyncalarm - Establecimiento del comportamiento de alarma de sincronización

### Utilización

PrxSetSyncalarm se utiliza para establecer el comportamiento de *sync\_alarm\_signal* a pulsos durante el tiempo especificado.

Si la alarma de sincronización se dispara, el *Sync\_alarm\_signal* funciona en pulsos durante el tiempo especificado por la instrucción PrxSetSyncalarm. También puede configurarse para ningún pulso, es decir, la señal se mantiene alta.

La duración predeterminada del pulso es de 1 segundo.

### Ejemplos básicos

#### Ejemplo 1

```
PrxSetSyncalarm SSYNCl \time:=2;
```

Establece la duración del pulso en el *sync\_alarm\_signal* en 2 segundos.

#### Ejemplo 2

```
PrxSetSyncalarm SSYNCl \NoPulse;
```

Si la alarma de sincronización se dispara, se activa el *sync\_alarm\_signal* (sin pulsos).

### Argumentos

```
PrxSetSyncalarm MechUnit [\Time] | [\NoPulse]
```

MechUnit

**Tipo de dato:** mechunit

El objeto de unidad mecánica móvil al que está sincronizado el movimiento del robot.

[\Time]

**Tipo de dato:** num

La duración de los pulsos en segundos. Debe estar entre 0,1 y 60.

Si *\Time* tiene un valor de más de 60, nos utiliza ningún pulso (mismo efecto que usar *\NoPulse*).

[\NoPulse]

**Tipo de dato:** switch

No se usa ningún pulso. La señal permanece activada hasta que se ejecuta una instrucción *SupSyncSensorOff*:

### Sintaxis

```
PrxSetSyncalarm
  [ MechUnit ':= ' ] < expression (IN) of mechunit >
  [ '\ ' Time ':= ' < expression (IN) of num > ]
  | [ '\ ' NoPulse ] ';' ;
```

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.182 PrxSetSyncalarm - Establecimiento del comportamiento de alarma de sincronización

### *Machine Synchronization*

#### *Continuación*

---

#### Información relacionada

| Para obtener más información sobre                                 | Consulte  |
|--|---|
| SupSyncSensorOff - Parada de la supervisión de sensor sincronizada | <a href="#">SupSyncSensorOff - Parada de la supervisión de sensor sincronizada en la página 845</a> |
| <i>Machine Synchronization</i>                                     | <i>Application manual - Controller software IRC5</i>  |

## 1.183 PrxStartRecord - Grabación de un nuevo perfil

### Utilización

PrxStartRecord se utiliza para restablecer todos los datos de perfil y grabar un nuevo perfil del movimiento de sensor tan pronto como se activa *sensor\_start\_signal*.

Para poder realizar una grabación, es importante establecer en primer lugar una conexión a un sensor (la unidad mecánica cuya velocidad afecta a la velocidad del robot). Esto significa que es necesario ejecutar una instrucción `WaitSensor` antes de iniciar la grabación.

### Ejemplo básico

```
ActUnit SSYNCl;
WaitSensor SSYNCl;
PrxStartRecord SSYNCl, 1, PRX_PROFILE_T1;
WaitTime 0.2;
SetDO do_startstop_machine 1;
```

En este ejemplo, la señal *do\_startstop\_machine* inicia el movimiento de sensor. El perfil del sensor se graba tan pronto como la máquina activa la señal *sensor\_start\_signal*.

### Argumentos

PrxStartRecord MechUnit, Record\_duration, Profile\_type

MechUnit

Tipo de dato: mechunit

El objeto de unidad mecánica móvil al que está sincronizado el movimiento del robot.

Record\_duration

Tipo de dato: num

Especifica la duración del registro en segundos. Debe estar entre 0,1 y *Pos Update time* \* 300. Si se usa el valor 0, es necesario utilizar la instrucción `PrxStopRecord` para detener la grabación.

Profile\_type

Tipo de dato: num

Los valores posibles y su explicación se enumeran a continuación:

| Valor            | Descripción   |
|------------------|---|
| PRX_INDEX_PROF   | Registro iniciado por <i>sensor_start_signal</i> .  |
| PRX_START_ST_PR  | Es posible grabar el inicio y el paro del movimiento. <i>sensor_start_signal</i> se usa para grabar el inicio del movimiento y <i>sensor_stop_signal</i> se usa para grabar la parada del movimiento. |
| PRX_STOP_ST_PROF | Igual para PRX_START_ST_PR, si bien con un orden diferente en las señales. Se usa primero <i>sensor_stop_signal</i> .   |
| PRX_STOP_M_PROF  | La grabación es iniciada por <i>sensor_stop_signal</i> .  |

Continúa en la página siguiente

# 1 Instrucciones

## 1.183 PrxStartRecord - Grabación de un nuevo perfil

### Machine Synchronization

#### Continuación

| Valor           | Descripción   |
|-----------------|---|
| PRX_HPRESS_PROF | Para la grabación de una prensa hidráulica (en la cual la posición cero del sensor corresponde a la posición abierta de la prensa). |
| PRX_PROFILE_T1  | Para la grabación de IMM u otra máquina (en la cual la posición cero del sensor corresponde a la posición cerrada de la prensa).    |

#### Ejecución de programas

PrxStartRecord debe ejecutarse al menos 0,2 segundos antes del inicio del movimiento del sensor.

#### Sintaxis

```
PrxStartRecord
  [ MechUnit ':= ' ] < expression (IN) of mechunit > ','
  [ Record_duration ':= ' ] < expression (IN) of num > ','
  [ Profile_type ':= ' ] < expression (IN) of num > ';'

```

#### Información relacionada

| Para obtener más información sobre  | Consulte   |
|-------------------------------------|--|
| Parada de la grabación de un perfil | <a href="#">PrxStopRecord - Parada de la grabación de un perfil en la página 563</a> |
| Machine Synchronization             | Application manual - Controller software IRC5  |

## 1.184 PrxStopRecord - Parada de la grabación de un perfil

### Utilización

PrxStopRecord se utiliza para detener la grabación de un perfil.

Debe usarse siempre que PrxStartRecord tenga Record\_duration con el valor 0.

### Ejemplo básico

```

ActUnit SSYNCl;
WaitSensor SSYNCl;
PrxStartRecord SSYNCl, 0, PRX_PROFILE_T1;
WaitTime 0.2;
SetDo do_startstop_machine 1;
WaitTime 2;
PrxStopRecord SSYNCl;

```

En este ejemplo, la señal *do\_startstop\_machine* inicia el movimiento de sensor. El perfil del movimiento del sensor se graba tan pronto como se activa *sensor\_start\_signal* y después de dos segundos la grabación se detiene con la instrucción PrxStopRecord.

### Argumentos

PrxStopRecord MechUnit

MechUnit

Tipo de dato: mechunit

El objeto de unidad mecánica móvil al que está sincronizado el movimiento del robot.

### Sintaxis

```

PrxStopRecord
  [ MechUnit ':' ] < expression (IN) of mechunit > ';'

```

### Información relacionada

| Para obtener más información sobre | Consulte   |
|------------------------------------|--|
| Grabación de un nuevo perfil       | <a href="#">PrxStartRecord - Grabación de un nuevo perfil en la página 561</a> |
| <i>Machine Synchronization</i>     | <i>Application manual - Controller software IRC5</i>                           |

## 1 Instrucciones

---

### 1.185 PrxStoreRecord - Almacenamiento de los datos de perfil grabados *Machine Synchronization*

### 1.185 PrxStoreRecord - Almacenamiento de los datos de perfil grabados

---

#### Utilización

PrxStoreRecord se utiliza para guardar en un archivo un registro activado.

---

#### Ejemplo básico

```
ActUnit SSYNCl;  
WaitSensor SSYNCl;  
PrxStartRecord SSYNCl, 0, PRX_PROFILE_T1;  
WaitTime 0.2;  
SetDo do_startstop_machine 1;  
WaitTime 2;  
PrxStopRecord SSYNCl;  
PrxActivRecord SSYNCl;  
SetDo do_startstop_machine 0;  
PrxStoreRecord SSYNCl, 0, "profile.log";
```

El perfil del movimiento del sensor se graba tan pronto como se activa *sensor\_start\_signal* y se almacena en el archivo *profile.log*.

---

#### Argumentos

```
PrxStoreRecord MechUnit Delay Filename
```

MechUnit

**Tipo de dato:** mechunit

El objeto de unidad mecánica móvil al que está sincronizado el movimiento del robot.

Delay

**Tipo de dato:** num

Es posible usar el retardo en segundos para desplazar la grabación en el tiempo. Debe estar entre 0,01 y 0,1. Si se indica el valor 0, no se añade ningún retardo. El retardo no se guarda en el perfil; solo se usa para la activación. Si se debe utilizar el retardo junto con un perfil guardado, es necesario especificar el retardo de nuevo en la instrucción `PrxUseFileRecord`.

File\_name

**Tipo de dato:** string

Nombre del archivo en el que se almacena el perfil.

---

#### Limitaciones

El registro debe estar activado antes de llamar a `PrxStoreRecord`.

---

#### Sintaxis

```
PrxStoreRecord  
[ MechUnit ':' ] < expression (IN) of mechunit > ','  
[ Delay ':' ] < expression (IN) of num > ','  
[ File_name ':' ] < expression (IN) of string > ';'
```

*Continúa en la página siguiente*



## 1.185 PrxStoreRecord - Almacenamiento de los datos de perfil grabados *Machine Synchronization* Continuación

### Información relacionada

| Para obtener más información sobre          | Consulte  |
|---|---|
| Activación de los datos de perfil grabados  | <a href="#">PrxActivRecord - Activación de los datos de perfil grabados en la página 551</a>    |
| Desactivación de un registro                | <a href="#">PrxDeactRecord - Desactivación de un registro en la página 554</a>                  |
| Utilización de los datos de perfil grabados | <a href="#">PrxUseFileRecord - Utilización de los datos de perfil grabados en la página 566</a> |
| <i>Machine Synchronization</i>              | <i>Application manual - Controller software IRC5</i>  |

# 1 Instrucciones

## 1.186 PrxUseFileRecord - Utilización de los datos de perfil grabados *Machine Synchronization*

### 1.186 PrxUseFileRecord - Utilización de los datos de perfil grabados

#### Utilización

PrxUseFileRecord se utiliza para cargar y activar un registro desde un archivo para la sincronización de sensor.

#### Ejemplo básico

```
PrxUseFileRecord SSYNCl, 0, "profile.log";
WaitTime 0.2;
SetDo do_startstop_machine 1;
!Work synchronized with sensorWork synchronized with sensor
...
SetDo do_startstop_machine 0;
```

#### Argumentos

```
PrxUseFileRecord MechUnit Delay Filename
```

MechUnit

Tipo de dato: mechunit

El objeto de unidad mecánica móvil al que está sincronizado el movimiento del robot.

Delay

Tipo de dato: num

Es posible usar el retardo en segundos para desplazar la grabación en el tiempo. Debe estar entre 0,01 y 0,1. Si se indica el valor 0, no se añade ningún retardo.

File\_name

Tipo de dato: string

Nombre del archivo en el que se almacena el perfil.

#### Ejecución de programas

PrxUseFileRecord debe ejecutarse al menos 0,2 segundos antes del inicio del movimiento del transportador.

#### Sintaxis

```
PrxUseFileRecord
  [ MechUnit ':' ] < expression (IN) of mechunit > ','
  [ Delay ':' ] < expression (IN) of num > ','
  [ File_name ':' ] < expression (IN) of string > ';' ;
```

#### Información relacionada

| Para obtener más información sobre         | Consulte   |
|--|--|
| Activación de los datos de perfil grabados | <a href="#">PrxActivRecord - Activación de los datos de perfil grabados en la página 551</a> |
| Desactivación de un registro               | <a href="#">PrxDeactRecord - Desactivación de un registro en la página 554</a>               |

*Continúa en la página siguiente*

## 1.186 PrxUseFileRecord - Utilización de los datos de perfil grabados

### *Machine Synchronization*

#### *Continuación*

| Para obtener más información sobre             | Consulte   |
|--|--|
| Almacenamiento de los datos de perfil grabados | <a href="#">PrxStoreRecord - Almacenamiento de los datos de perfil grabados en la página 564</a> |
| <i>Machine Synchronization</i>                 | <i>Application manual - Controller software IRC5</i>   |

# 1 Instrucciones

---

1.187 PulseDO - Genera un pulso en una señal digital de salida  
RobotWare - OS

## 1.187 PulseDO - Genera un pulso en una señal digital de salida

---

### Utilización

PulseDO se utiliza para generar un pulso en una señal digital de salida.

---

### Ejemplos básicos

Los ejemplos siguientes ilustran la instrucción PulseDO.

#### Ejemplo 1

```
PulseDO do15;
```

Se genera un pulso con una duración de 0,2 s en la señal de salida do15.

#### Ejemplo 2

```
PulseDO \PLength:=1.0, ignition;
```

Se genera un pulso con una duración de 1,0 s en la señal ignition.

#### Ejemplo 3

```
! Program task MAIN  
PulseDO \High, do3;  
! At almost the same time in program task BCK1  
PulseDO \High, do3;
```

Se genera el valor positivo (valor 1) en la señal do3 desde dos tareas de programa casi al mismo tiempo. El resultado es un pulso positivo con una duración más larga que la duración predeterminada de 0,2 s o dos pulsos positivos uno después del otro, con una duración de pulso de 0,2 s.

---

### Argumentos

```
PulseDO [ \High ] [ \PLength ] Signal
```

[ \High ]

#### *High level*

Tipo de dato: switch

Especifica que el valor de la señal debe ser siempre el valor alto (valor 1) cuando se ejecuta la instrucción, independientemente de su estado actual.

[ \PLength ]

#### *Pulse Length*

Tipo de dato: num

La duración del pulso en segundos (de 0,001 a 2.000 s). Si se omite el argumento, se genera un pulso de 0,2 segundos.

Signal

Tipo de dato: signaldo

El nombre de la señal en la que se desea generar el pulso.

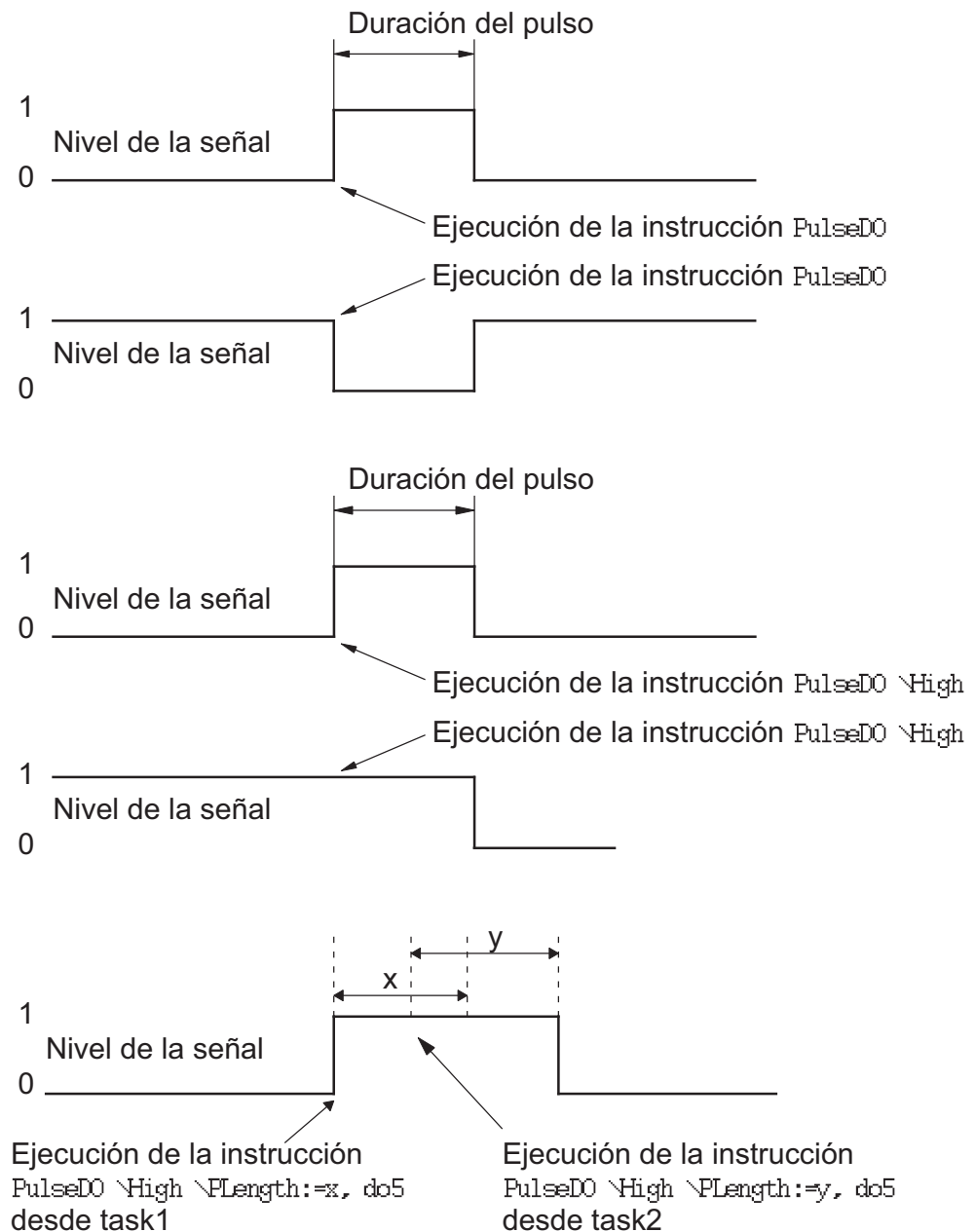
---

*Continúa en la página siguiente*

**Ejecución de programas**

La instrucción siguiente tras `PulseDO` se ejecuta directamente después del inicio del pulso. A continuación, el pulso puede activarse o desactivarse sin afectar al resto de la ejecución del programa.

En la figura siguiente se muestran ejemplos de generación de pulsos en una señal digital de salida.



xx0500002217

La instrucción siguiente se ejecuta directamente después del inicio del pulso. A continuación, el pulso puede activarse o desactivarse sin afectar al resto de la ejecución del programa.

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.187 PulseDO - Genera un pulso en una señal digital de salida

RobotWare - OS

Continuación

### Limitaciones

La duración del pulso tiene una resolución de 0,001 segundos. Los valores programados que difieran de esta resolución se redondean.

### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                          | Causa del error   |
|---------------------------------|---|
| <code>ERR_NO_ALIASIO_DEF</code> | La variable de señal es una variable declarada en RAPID. No se ha conectado a una señal E/S definida en la configuración de E/S con la instrucción <code>AliasIO</code> .                             |
| <code>ERR_NORUNUNIT</code>      | Se ha perdido el contacto con el dispositivo de E/S.  |
| <code>ERR_SIG_NOT_VALID</code>  | La señal de E/S no está disponible. Los motivos pueden ser que el dispositivo de E/S no está en funcionamiento o que exista un error en la configuración (válido solamente para el bus de campo ICI). |

### Sintaxis

```
PulseDO  
  ['\' High]  
  ['\' PLength ':=> <expression (IN) of num>'],''  
  [Signal ':=>] <variable (VAR) of signaldo>';'
```

### Información relacionada

| Para obtener más información sobre         | Consulte   |
|--|--|
| Instrucciones de entrada/salida            | <i>Manual de referencia técnica - RAPID Overview</i>         |
| Funcionalidad de entrada/salida en general | <i>Manual de referencia técnica - RAPID Overview</i>         |
| Configuración de E/S                       | <i>Manual de referencia técnica - Parámetros del sistema</i> |

### 1.188 RAISE - Llamada a un gestor de errores

#### Utilización

RAISE se utiliza para crear un error en el programa y llamar a continuación al gestor de errores de la rutina.

RAISE también puede usarse en el gestor de errores para propagar el error actual hacia el gestor de errores de la rutina desde la que se llamó a la rutina actual.

Por ejemplo, esta instrucción puede usarse para volver a un nivel más alto de la estructura del programa, por ejemplo al gestor de errores de la rutina principal, si se produce un error en el nivel inferior.

#### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción RAISE:

Consulte también [Más ejemplos en la página 572](#).

#### Ejemplo 1

```
MODULE MainModule .
VAR errnum ERR_MY_ERR := -1;

PROC main()
  BookErrNo ERR_MY_ERR;

  IF dil = 0 THEN
    RAISE ERR_MY_ERR;
  ENDIF

ERROR
  IF ERRNO = ERR_MY_ERR THEN
    TPWrite "dil equals 0";
  ENDIF

ENDPROC

ENDMODULE
```

En esta implementación, dil igual a 0 es considerado como un error. RAISE fuerza la ejecución en el gestor de errores. En este ejemplo, el usuario ha creado su propio número de error para gestionar este error en concreto.

#### Argumentos

```
RAISE [ Error no. ]
```

Error no.

**Tipo de dato:** errnum

**Número de error:** Cualquier número de 1 a 90 que pueda usar el gestor de errores para localizar el error que se ha producido (la variable de sistema ERRNO).

También es posible registrar un número de error fuera del rango de 1 a 90 con la instrucción BookErrNo.

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.188 RAISE - Llamada a un gestor de errores

RobotWare-OS

Continuación

El número de error debe especificarse fuera del gestor de errores en una instrucción RAISE, para transferir la ejecución al gestor de errores que se encuentra fuera de la rutina.

Si la instrucción se incluye en el gestor de errores de una rutina, el error se propaga hasta el gestor de errores de la rutina que realiza la llamada. En este caso, no es necesario especificar el número del error.

---

### Más ejemplos

A continuación aparecen más ejemplos de la instrucción RAISE.

#### Ejemplo 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
```

En este ejemplo, la división entre cero dará lugar a un error. En el gestor de ERROR, RAISE propagará el error hasta el gestor de ERROR de la rutina "main" que hace la llamada. El mismo número de error permanece activo. RETRY ejecutará de nuevo la totalidad de la rutina "routine1".

---

### Ejecución de programas

La ejecución del programa continúa en el gestor de errores de la rutina. Después de ejecutar el gestor de errores, la ejecución del programa puede continuar con:

- La rutina que llamó a la rutina actual (RETURN).
- El gestor de errores de la rutina que llamó a la rutina actual (RAISE).

Una instrucción RAISE en el gestor de errores de una rutina también presenta otra característica. Puede usarse para saltos largos (consulte "Recuperación en caso de error con un salto largo"). Con un salto largo, es posible propagar un error

*Continúa en la página siguiente*



desde un gestor de errores que se encuentra a gran profundidad dentro de una cadena de llamadas anidadas, hasta un nivel superior, en un solo paso.

Si la instrucción `RAISE` está presente en una rutina `TRAP`, el error se gestiona en el gestor de errores del sistema.

### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                    | Causa del error  |
|---------------------------|--|
| <code>ERR_ILLRAISE</code> | El número de error de <code>RAISE</code> está fuera de rango |

### Sintaxis

```
RAISE [<error number>] ';' ;'
```

### Información relacionada

| Para obtener más información sobre               | Consulte  |
|--|---|
| Gestión de errores                               | <i>Manual de referencia técnica - Parámetros del sistema</i>                                |
| Recuperación en caso de error con un salto largo | <i>Technical reference manual - RAPID kernel</i>  |
| Registro de números de error                     | <a href="#">BookErrNo - Registra un número de error de sistema de RAPID en la página 50</a> |

## 1 Instrucciones

---

### 1.189 RaiseToUser - Propaga un error al nivel del usuario

RobotWare - OS

### 1.189 RaiseToUser - Propaga un error al nivel del usuario

---

#### Utilización

`RaiseToUser` se utiliza en los gestores de errores de las rutinas `NOSTEPIN` para propagar el error actual o cualquier otro error definido hacia el gestor de errores del nivel del usuario. El nivel del usuario es en este caso la primera rutina de una cadena de llamadas, por encima de la rutina `NOSTEPIN`.

---

#### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción `RaiseToUser`:

##### Ejemplo 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
```

*Continúa en la página siguiente*

La división entre cero que tiene lugar en `routine2` se propaga hacia arriba en el gestor de errores de la rutina `main`, con el valor `ERR_MYDIVZERO` en `errno`. A continuación, la instrucción `TRYNEXT` del gestor de errores hace que la ejecución del programa continúe en la instrucción que sigue a la división entre cero en `routine2`. El modificador `\Continue` controla este comportamiento.

Si se produce cualquier otro error en `routine2`, el modificador `\BreakOff` fuerza la continuación de la ejecución desde el gestor de errores de la rutina `main`. En este caso, el gestor de deshacer de `routine1` se ejecuta mientras se eleva hacia el nivel del usuario. La instrucción `RETRY` del gestor de errores de la rutina `main` ejecuta `routine1`, de nuevo desde el principio.

El gestor de deshacer de `routine1` también se ejecuta en el caso de que se use `\Continue`, si la siguiente instrucción `RAISE` o `RETURN` se realiza en el nivel del usuario.

---

### Argumentos

`RaiseToUser[ \Continue ] | [ \BreakOff ][ \ErrorNumber ]`

`[ \Continue ]`

Tipo de dato: `switch`

Continúa la ejecución en la rutina que provocó el error.

`[ \BreakOff ]`

Tipo de dato: `switch`

Interrumpe la cadena de llamadas y prosigue la ejecución en el nivel del usuario. Cualquier gestor de deshacer de la cadena de llamadas se ejecuta de forma separada del gestor de deshacer de la rutina que elevó el error.

Es necesario programar uno de los argumentos `\Continue` o `\BreakOff`, para evitar un error de ejecución.

`[ \ErrorNumber ]`

Tipo de dato: `errnum`

Cualquier número de 1 a 90 que pueda usar el gestor de errores para localizar el error que se ha producido (la variable de sistema `ERRNO`).

También es posible registrar un número de error fuera del rango de 1 a 90 con la instrucción `BookErrNo`.

Si no se especifica el argumento `\ErrorNumber`, el número de error original se propaga hacia el gestor de errores en la rutina, a nivel de usuario.

---

### Ejecución de programas

`RaiseToUser` can sólo puede usarse en un gestor de errores de una rutina `NOSTEPIN`.

La ejecución del programa continúa en el gestor de errores de la rutina en el nivel del usuario. El mismo número de error permanece activo si no está presente el parámetro opcional `\ErrorNumber`. El gestor de errores del sistema se encarga del error si no hay ningún gestor de errores en el nivel del usuario. La llamada al gestor de errores del sistema se realiza si no se especifica ninguno de los argumentos, `\Continue` ni `\BreakOff`.

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.189 RaiseToUser - Propaga un error al nivel del usuario

RobotWare - OS

Continuación

Existen dos comportamientos diferentes una vez que se ha ejecutado el gestor de errores. La ejecución del programa continúa en la rutina que contiene `RaiseToUser`, si el modificador `\Continue` está activado. La ejecución del programa prosigue en el nivel del usuario si el modificador `\BreakOff` está activado.

La ejecución del programa puede continuar con:

- La instrucción que provocó el error (`RETRY`)
- La instrucción siguiente (`TRYNEXT`)
- El gestor de errores de la rutina que llamó a la rutina en el nivel del usuario (`RAISE`)
- La rutina que llamó a la rutina del nivel del usuario (`RETURN`)

### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                    | Causa del error  |
|---------------------------|--|
| <code>ERR_ILLRAISE</code> | El número de error de <code>RAISE</code> está fuera de rango |

### Sintaxis

```
RaiseToUser  
  [ '\ Continue ]  
  '| [ '\ BreakOff ]  
  [ '\ ErrorNumber ' := ' ] < expression (IN) of errnum > ';' 
```

### Información relacionada

| Para obtener más información sobre | Consulte  |
|------------------------------------|---|
| Gestión de errores                 | <i>Manual de referencia técnica - RAPID Overview</i>  |
| Gestión de deshacer                | <i>Manual de referencia técnica - RAPID Overview</i>  |
| Registro de números de error       | <a href="#">BookErrNo - Registra un número de error de sistema de RAPID en la página 50</a> |

### 1.190 ReadAnyBin - Leer datos de un dispositivo de E/S o un archivo binario

#### Utilización

`ReadAnyBin` (*Read Any Binary*) se utiliza para leer cualquier tipo de dato de un canal o un dispositivo de E/S.

#### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción `ReadAnyBin`:

#### Ejemplo 1

```
VAR iodev file1;  
VAR robtarget next_target;  
...  
Open "HOME:" \File:= "bin_file.txt", file1 \Bin;  
ReadAnyBin file1, next_target;
```

El siguiente objetivo que debe ejecutar el robot, `next_target`, se lee del archivo `file1`.

#### Argumentos

```
ReadAnyBin IODevice Data [\Time]
```

IODevice

Tipo de dato: `iodev`

El nombre (la referencia) del dispositivo de E/S o el archivo binario del que se desea leer.

Data

Tipo de dato: `ANYTYPE`

La variable de tipo `VAR` `o PERS` en la que se desea almacenar la información.

[\Time]

Tipo de dato: `num`

El tiempo máximo para la operación de lectura (tiempo límite) en segundos. Si no se especifica este argumento, el tiempo máximo es de 60 segundos. Para esperar ininterrumpidamente, utilice la constante predefinida `WAIT_MAX`.

Si se agota este tiempo antes de que se complete la operación de lectura, se llama al gestor de errores con el código de error `ERR_DEV_MAXTIME`. Si no hay ningún gestor de errores, se detiene la ejecución.

La función de tiempo límite se utiliza también durante un paro de programa y se notificará en el programa de `RAPID` al poner en marcha el programa.

#### Ejecución de programas

Se lee del dispositivo de E/S o del archivo binario especificado el número de bytes necesario para los datos especificados.

En caso de un reinicio tras una caída de alimentación, todos los archivos o dispositivos de E/S abiertos del sistema se cierran y el descriptor de E/S de la variable del tipo `iodev` se restablece.

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.190 ReadAnyBin - Leer datos de un dispositivo de E/S o un archivo binario

RobotWare - OS

Continuación

### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                       | Causa del error  |
|------------------------------|--|
| <code>ERR_FILEACC</code>     | La ruta apunta a un directorio inexistente o existen demasiados directorios abiertos al mismo tiempo.  |
| <code>ERR_DEV_MAXTIME</code> | Tiempo límite al ejecutar una instrucción <code>ReadBin</code> , <code>ReadNum</code> , <code>ReadStr</code> , <code>ReadStrBin</code> , <code>ReadAnyBin</code> , o <code>ReadRawBytes</code> . |
| <code>ERR_RANYBIN_CHK</code> | Error de suma de comprobación detectado en la transferencia de datos con la instrucción <code>ReadAnyBin</code> .  |
| <code>ERR_RANYBIN_EOF</code> | Se ha detectado el final del archivo antes de que se lean todos los bytes en la instrucción <code>ReadAnyBin</code> .  |

### Limitaciones

Esta instrucción sólo puede usarse con dispositivos de E/S o archivos que hayan sido abiertos para lectura binaria.

Los datos a leer con esta instrucción `ReadAnyBin` deben ser de los tipos de datos `num`, `bool` o `string`. También puede usarse un registro, un componente de registro, una matriz o un elemento de matriz de este tipo de dato de valor. No es posible usar datos enteros o datos parciales con `semivalor` ni tipos de datos sin valor.



#### Nota

La variable `VAR` o `PERS`, que se usa para almacenar los datos leídos, puede actualizarse en varios pasos. Por tanto, espere siempre hasta que se actualice toda la estructura de datos antes de leer la información desde una rutina `TRAP` o desde otra tarea de programa.

Dado que `WriteAnyBin-ReadAnyBin` sólo se han diseñado para gestionar datos binarios internos de controlador con dispositivos de E/S o archivos entre o dentro del controlador del robot, no se hace público ningún protocolo de datos y no es posible interpretar estos datos en ningún PC.

El desarrollo de software de control puede afectar a la compatibilidad, de forma que quizá no sea posible usar `WriteAnyBin-ReadAnyBin` entre versiones de software de RobotWare diferentes.

### Sintaxis

```
ReadAnyBin
  [IODevice ':='] <variable (VAR) of iodev>', '
  [Data ':='] <var or pers (INOUT) of ANYTYPE>
  ['\ ' Time ':='] <expression (IN) of num>'];'
```

Continúa en la página siguiente

### Información relacionada

| Para obtener más información sobre                        | Consulte   |
|---|--|
| Apertura de dispositivos de E/S o archivos                | <i>Manual de referencia técnica - RAPID Overview</i>   |
| Escribir datos en un dispositivo de E/S o archivo binario | <a href="#">WriteAnyBin - Escribe datos en un archivo o dispositivo de E/S binario en la página 1101</a> |
| Gestión de archivos y dispositivos de E/S                 | <i>Application manual - Controller software IRC5</i>   |

# 1 Instrucciones

---

## 1.191 ReadBlock - Lee un bloque de datos de un dispositivo *Sensor Interface*

### 1.191 ReadBlock - Lee un bloque de datos de un dispositivo

---

#### Utilización

ReadBlock se utiliza para leer un bloque de datos de un dispositivo que está conectado a la interfaz de sensores serie.. *The data is stored in a file.*

La interfaz de sensores se comunica con dos sensores a través de canales serie, utilizando el protocolo de transporte RTP1.

Éste es un ejemplo de configuración de un canal de sensor.

COM\_PHY\_CHANNEL:

- Name "COM1:"
- Connector "COM1"
- Baudrate 19200

COM\_TRP:

- Name "sen1:"
- Type "RTP1"
- PhyChannel "COM1"

---

#### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción ReadBlock:

#### Ejemplo 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;
```

---

#### Argumentos

```
ReadBlock device BlockNo FileName [ \TaskName ]
```

device

**Tipo de dato:** string

El nombre del dispositivo de E/S configurado en sio.cfg para el sensor utilizado.

BlockNo

**Tipo de dato:** num

El argumento BlockNo se utiliza para seleccionar el bloque de datos de sensor que se desea leer.

FileName

**Tipo de dato:** string

*Continúa en la página siguiente*



## 1.191 ReadBlock - Lee un bloque de datos de un dispositivo

*Sensor Interface*

*Continuación*

El argumento `FileName` se utiliza para definir un archivo en el que se escriben los datos desde el bloque de datos de sensor seleccionado en el argumento `BlockNo`.

[ `\TaskName` ]

Tipo de dato: `string`

El argumento `TaskName` hace posible el acceso a dispositivos de otras tareas de RAPID.

### Gestión de errores

| Constante de error (valor de <code>ERRNO</code> ) | Descripción                            |
|---|--|
| <code>SEN_NO_MEAS</code>                          | Fallo de medición                      |
| <code>SEN_NOREADY</code>                          | Sensor incapaz de gestionar el comando |
| <code>SEN_GENERRO</code>                          | Error general del sensor               |
| <code>SEN_BUSY</code>                             | Sensor ocupado                         |
| <code>SEN_UNKNOWN</code>                          | Sensor desconocido                     |
| <code>SEN_EXALARM</code>                          | Error de sensor externo                |
| <code>SEN_CAALARM</code>                          | Error de sensor interno                |
| <code>SEN_TEMP</code>                             | Error de temperatura del sensor        |
| <code>SEN_VALUE</code>                            | Valor de comunicación no válido        |
| <code>SEN_CAMCHECK</code>                         | Fallo de comprobación de sensor        |
| <code>SEN_TIMEOUT</code>                          | Error de comunicación                  |

### Sintaxis

```

ReadBlock
  [ device ':= ' ] < expression( IN ) of string > ', '
  [ BlockNo ':= ' ] < expression ( IN ) of num > ', '
  [ FileName ':= ' ] < expression ( IN ) of string > ', '
  [ '\ ' TaskName ':= ' < expression ( IN ) of string > ] ';'
    
```

### Información relacionada

| Para obtener más información sobre                         | Consulte   |
|--|--|
| Establecimiento de una conexión a un dispositivo de sensor | <a href="#">SenDevice - Establece una conexión a un dispositivo de sensor en la página 687</a> |
| Escritura de una variable de sensor                        | <a href="#">WriteVar - Escribir una variable en la página 1116</a>                             |
| Lectura de una variable de sensor                          | <a href="#">ReadVar - Lee una variable de un dispositivo en la página 1451</a>                 |
| Escritura de un bloque de datos de sensor                  | <a href="#">WriteBlock - Escribir un bloque de datos en un dispositivo en la página 1105</a>   |
| Configuración de la comunicación del sensor                | <a href="#">Manual de referencia técnica - Parámetros del sistema</a>                          |

## 1 Instrucciones

---

### 1.192 ReadCfgData - Lee un atributo de un parámetro del sistema *RobotWare - OS*

### 1.192 ReadCfgData - Lee un atributo de un parámetro del sistema

---

#### Utilización

`ReadCfgData` se utiliza para leer un atributo de un parámetro del sistema (de los datos de configuración).

Además de leer parámetros con nombre, también es posible buscar parámetros sin nombre.

---

#### Ejemplos básicos

El ejemplo siguiente ilustra la instrucción `ReadCfgData`. Estos dos ejemplos muestran cómo leer parámetros con nombre.

##### Ejemplo 1

```
VAR num offset1;  
...  
ReadCfgData "/MOC/MOTOR_CALIB/rob1_1", "cal_offset", offset1;
```

Lee el valor del offset de calibración de axis 1 para `rob_1` en la variable de tipo `num offset1`.

##### Ejemplo 2

```
VAR string io_device;  
...  
ReadCfgData "/EIO/EIO_SIGNAL/process_error", "Device", io_device;
```

Lee el nombre del dispositivo de E/S en la que está definida la señal `process_error` y lo almacena en la variable de tipo `string` llamada `io_device`.

---

#### Argumentos

```
ReadCfgData InstancePath Attribute CfgData [\ListNo]
```

`InstancePath`

**Tipo de dato:** `string`

Especifica la ruta hasta el parámetro que debe leerse.

En el caso de los parámetros con nombre, el formato de esta cadena es `/DOMAIN/TYPE/ParameterName`.

En el caso de los parámetros sin nombre, el formato de esta cadena es `/DOMAIN/TYPE/Attribute/AttributeValue`.

`Attribute`

**Tipo de dato:** `string`

El nombre del atributo del parámetro que se desea leer.

`CfgData`

**Tipo de dato:** `any type`

La variable en la que se almacenará el valor del atributo. En función del tipo de atributo, los tipos válidos son `bool`, `num`, `dnum`, o `string`.

`[\ListNo]`

**Tipo de dato:** `num`

---

*Continúa en la página siguiente*

Una variable que contiene el número de instancia de `atributo+AttributeValue` a encontrar.

La primera vez que aparece `Attribute + AttributeValue` tiene el número de instancia 0. Si hay más instancias a buscar, el valor devuelto en `\ListNo` se incrementará en 1. De lo contrario, si no hay más instancias el valor de retorno será -1. La constante predefinida `END_OF_LIST` puede usarse para comprobar si hay más de una instancia a buscar.

### Ejecución de programas

El valor del atributo especificado por el argumento `Attribute` se almacena en la variable especificada por el argumento `CfgData`.

Si se usa el formato `/DOMAIN/TYPE/ParameterName` en `InstancePath`, sólo están disponibles los parámetros con nombre, es decir, los parámetros cuyo primer atributo sea `name, Name` o `NAME`.

En el caso de los parámetros sin nombre, utilice el parámetro opcional `\ListNo` para seleccionar de cuál de las instancias debe leerse el valor del atributo. Se actualiza tras cada lectura exitosa de la siguiente instancia disponible.

### Más ejemplos

A continuación aparecen más ejemplos de la instrucción `ReadCfgdata`. Estos dos ejemplos muestran cómo leer parámetros sin nombre.

#### Ejemplo 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
```

Lee la señal resultante de la señal digital de actuación sin nombre `di_13` y guarda el nombre en la variable de tipo `string` llamada `read_str`.

En este ejemplo, el dominio EIO tiene el código `cfg` siguiente:

**EIO\_CROSS:**

```
-Name "Cross_di_1_do_2" -Res "di_1" -Act1 "do_2"
-Name "Cross_di_2_do_2" -Res "di_2" -Act1 "do_2"
-Name "Cross_di_13_do_13" -Res "di_13" -Act1 "do_13"
```

#### Ejemplo 2

```
VAR num list_index;
VAR string read_str;
...
list_index:=0;
WHILE list_index <> END_OF_LIST DO
  read_str:="";
```

*Continúa en la página siguiente*

## 1 Instrucciones

### 1.192 ReadCfgData - Lee un atributo de un parámetro del sistema

RobotWare - OS

Continuación

```
ReadCfgData "/EIO/EIO_SIGNAL/Device/USERIO", "Name", read_str,
  \ListNo:=list_index;
IF read_str <> "" THEN
  TPWrite "Signal: " + read_str;
ENDIF
ENDWHILE
..
ERROR
  TRYNEXT;
```

Leer los nombres de todas las señales definidas para el dispositivo de E/S USERIO.

En este ejemplo, el dominio EIO tiene el código cfg siguiente:

```
EIO_SIGNAL:
  -Name "USERD01" -SignalType "DO" -Device "USERIO" -DeviceMap "0"
  -Name "USERD02" -SignalType "DO" -Device "USERIO" -DeviceMap "1"
  -Name "USERD03" -SignalType "DO" -Device "USERIO" -DeviceMap "2"
```

#### Ejemplo 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;
```

Leer los nombres de todos los dispositivos DeviceNet.

En este ejemplo, el dominio EIO tiene el código cfg siguiente:

```
DEVICENET_DEVICE:
  -Name PANEL -Network "DeviceNet" -Simulated
  -Name DRV_1 -Network "DeviceNet" -Simulated
  -Name DEVICE1 -Network "DeviceNet" -Simulated
  -Name DEVICE2 -Network "DeviceNet" -Simulated
```

## Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre         | Causa del error   |
|----------------|---|
| ERR_CFG_NOTFND | No fue posible encontrar los datos especificados con "InstancePath + Attribute" en la base de datos de configuración. |

Continúa en la página siguiente

## 1.192 ReadCfgData - Lee un atributo de un parámetro del sistema

RobotWare - OS

Continuación

| Nombre              | Causa del error   |
|---------------------|---|
| ERR_CFG_ILLLTYPE    | El tipo de dato del parámetro CfgData es distinto del tipo de dato real de los datos encontrados y especificados con "InstancePath + Attribute" en la base de datos de configuración. |
| ERR_CFG_INTERNAL    | No se permite leer un parámetro interno   |
| ERR_CFG_OUTOFBOUNDS | La variable del argumento \ListNo tiene un valor que está fuera del rango de instancias disponibles (0 ... n) al ejecutar la instrucción.   |

## Limitaciones

La conversión de las unidades del parámetro del sistema (metros, radianes, segundos, etc.) a las unidades de los programas de RAPID (mm, grados, segundos, etc.), que afecta a CfgData del tipo de datos num y dnum debe ser realizada por el usuario en el programa de RAPID.

Si se usa el formato /DOMAIN/TYPE/ParameterName en InstancePath, sólo están disponibles los parámetros con nombre, es decir, los parámetros cuyo primer atributo sea name, Name o NAME.

Las cadenas de RAPID están limitadas a 80 caracteres. En algunos casos, puede ser en teoría una longitud demasiado reducida para la definición de InstancePath, Attribute o CfgData.

## Datos predefinidos

La constante predefinida END\_OF\_LIST, con valor -1, puede usarse para detener la lectura si no es posible encontrar más instancias.

## Sintaxis

```
ReadCfgData
  [ InstancePath ':' ] < expression (IN) of string > ','
  [ Attribute ':' ] < expression (IN) of string > ','
  [ CfgData ':' ] < variable (VAR) of anytype >
  [ '\' ListNo ':' ] < variable (VAR) of num > ';'

```

## Información relacionada

| Para obtener más información sobre                   | Consulte   |
|--|--|
| Definición de cadena de caracteres                   | <a href="#">string - Cadenas en la página 1778</a>   |
| Escritura de un atributo de un parámetro del sistema | <a href="#">WriteCfgData - Escribe un atributo de un parámetro del sistema en la página 1107</a> |
| Obtención del nombre del robot de la tarea actual    | <a href="#">RobName - Obtiene el nombre del robot del TCP en la página 1458</a>                  |
| Configuración  | <a href="#">Manual de referencia técnica - Parámetros del sistema</a>                            |
| <i>Advanced RAPID</i>                                | <a href="#">Application manual - Controller software IRC5</a>                                    |

# 1 Instrucciones

---

## 1.193 ReadErrData - Obtiene información sobre un error

RobotWare - OS

## 1.193 ReadErrData - Obtiene información sobre un error

---

### Utilización

`ReadErrData` debe utilizarse en rutinas TRAP para obtener información (dominio, tipo, número y cadenas entremezcladas %s) sobre el error, el cambio de estado o el aviso que provocó la ejecución de la rutina TRAP.

---

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción `ReadErrData`:

Consulte el capítulo [Más ejemplos en la página 587](#).

#### Ejemplo 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
```

Quando se detecta un error con la rutina TRAP `trap_err`, el dominio, el número, el tipo y las dos primeras cadenas entremezcladas del mensaje de error se almacenan en las variables adecuadas.

---

### Argumentos

```
ReadErrData TrapEvent ErrorDomain ErrorId ErrorType [\Title] [\Str1]
[\Str2] [\Str3] [\Str4] [\Str5]
```

#### TrapEvent

**Tipo de dato:** trapdata

La variable que contiene la información sobre qué hecho provocó la ejecución de la rutina TRAP.

#### ErrorDomain

**Tipo de dato:** errdomain

La variable que debe almacenar el dominio de error al que pertenece el error, el cambio de estado o la advertencia. Consulte los datos predefinidos del tipo `errdomain`.

#### ErrorId

**Tipo de dato:** num

La variable que debe almacenar el número del error que se ha producido. El número del error se devuelve sin el primer dígito (el dominio del error) y sin los ceros iniciales que sí aparecen en el número de error completo.

---

*Continúa en la página siguiente*

Por ejemplo, el error 10008 Program restarted, se devuelve como 8.

ErrorType

Tipo de dato: `errtype`

La variable que debe almacenar el tipo de evento que se ha producido, por ejemplo un error, un cambio de estado o una advertencia. Consulte los datos predefinidos del tipo `errtype`.

[ \Title ]

Tipo de dato: `string`

Una variable para almacenar el título del mensaje de error. El título tiene el formato UTF8 y es posible que no todos los caracteres se muestren correctamente en todos los idiomas en el FlexPendant.

[ \Str1 ] ... [ \Str5 ]

Tipo de dato: `string`

Actualiza la variable de cadena especificada, con el argumento intercalado en el mensaje de error. Podrían existir hasta cinco argumentos en un mismo mensaje, con los tipos `%s`, `%f`, `%d` o `%ld`, que siempre se convertirán en una cadena al ejecutar esta instrucción. `Str1` contendrá el primer argumento, `Str2` el segundo argumento, etcétera. La información acerca de cuántos argumentos puede contener un mensaje aparece en el *Manual del operador - Solución de problemas*. Los argumentos intercalados se marcan como `arg` en ese documento.

---

### Ejecución de programas

Las variables `ErrorDomain`, `ErrorId`, `ErrorType`, `Title` y `Str1 ... Str5` se actualizan de acuerdo con el contenido de `TrapEvent`.

Si hay varios eventos conectados a una misma rutina TRAP, el programa debe asegurarse de que el evento está relacionado con la monitorización de errores. Esto puede hacerse comprobando que `INTNO` coincide con el número de interrupción utilizado en la instrucción `IError`.

---

### Más ejemplos

A continuación aparecen más ejemplos de la instrucción `ReadErrData`.

#### Ejemplo 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;  
  ...
```

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.193 ReadErrData - Obtiene información sobre un error

RobotWare - OS

Continuación

```
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
```

Cuando se produce un error (sólo en el caso de los errores, no las advertencias ni los cambios de estado), el número de error se obtiene en la rutina TRAP y su valor se utiliza para activar dos grupos de señales digitales de salida.

### Limitación

No es posible obtener información sobre los errores internos.

### Sintaxis

```
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>]';'
```

### Información relacionada

| Para obtener más información sobre                            | Consulte  |
|---|---|
| Resumen de interrupciones                                     | <a href="#">Manual de referencia técnica - RAPID Overview</a>   |
| Más información sobre la gestión de interrupciones            | <a href="#">Manual de referencia técnica - RAPID Overview</a>   |
| Dominios de error, constantes predefinidas                    | <a href="#">errdomain - Dominio del error en la página 1662</a>   |
| Tipos de errores, constantes predefinidas                     | <a href="#">errtype - Tipo de error en la página 1673</a>   |
| Solicitud de una interrupción para errores                    | <a href="#">IError - Solicita una interrupción para errores en la página 257</a>                        |
| Obtención de datos de interrupción para la rutina TRAP actual | <a href="#">GetTrapData - Obtiene datos de interrupción para la rutina TRAP actual en la página 240</a> |
| <i>Advanced RAPID</i>   | <i>Application manual - Controller software IRC5</i>  |



---

## 1.194 ReadRawBytes - Lee datos de tipo rawbytes

---

### Utilización

ReadRawBytes se utiliza para leer datos de tipo rawbytes de un dispositivo abierto con Open\Bin.

---

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción ReadRawBytes:

#### Ejemplo 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;
```

En este ejemplo, raw\_data\_out se deja sin contenido y se empaqueta con el encabezado de DeviceNet, junto con un valor de coma flotante con el valor 0.2. Se abre un dispositivo, "/FC1:/dsqc328\_1", y los datos válidos actuales de raw\_data\_out se escriben en el dispositivo. A continuación, el programa espera al menos 1 segundo antes de leer del dispositivo, almacenando en raw\_data\_in la información leída.

Después de cerrar el dispositivo "/FC1:/dsqc328\_1", los datos leídos se desempaquetan dando lugar a una cadena de caracteres que se almacena en answer.

---

### Argumentos

```
ReadRawBytes IODevice RawData [\Time]
```

IODevice

**Tipo de dato:** iODEV

IODevice es el identificador del dispositivo desde el cual se deben leer los datos.

RawData

**Tipo de dato:** rawbytes

RawData es el contenedor de datos que almacena los datos leídos de IODevice, empezando por el número de índice 1.

---

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.194 ReadRawBytes - Lee datos de tipo rawbytes

RobotWare - OS

Continuación

[ \Time ]

Tipo de dato: num

El tiempo máximo para la operación de lectura (tiempo límite) en segundos (resolución 0,001 s). Si no se especifica este argumento, el tiempo máximo es de 60 segundos. Para esperar ininterrumpidamente, utilice la constante predefinida `WAIT_MAX`.

Si se agota este tiempo antes de que se complete la operación de lectura, se llama al gestor de errores con el código de error `ERR_DEV_MAXTIME`. Si no hay ningún gestor de errores, se detiene la ejecución.

La función de tiempo límite se utiliza también durante un paro de programa y se notificará en el programa de RAPID al poner en marcha el programa.

### Ejecución de programas

Durante la ejecución del programa, los datos se leen del dispositivo indicado por `IODevice`.

Si se utiliza `WriteRawBytes` junto con los comandos de bus de campo, por ejemplo `DeviceNet`, el bus de campo siempre envía una respuesta. La respuesta debe ser manejada en RAPID con la instrucción `ReadRawBytes`.

La longitud actual de los bytes válidos de la variable `RawData` cambia al número de bytes leídos. Los datos comienzan en el número de índice 1 de `RawData`.

En caso de un reinicio tras una caída de alimentación, todos los archivos o dispositivos de E/S abiertos del sistema se cierran y el descriptor de E/S de la variable del tipo `iodev` se restablece.

### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                       | Causa del error   |
|------------------------------|---|
| <code>ERR_FILEACC</code>     | El acceso a un archivo se realiza de forma incorrecta   |
| <code>ERR_DEV_MAXTIME</code> | Tiempo límite al ejecutar la instrucción.   |
| <code>ERR_RANYBIN_EOF</code> | Se ha detectado el final del archivo antes de que se lean todos los bytes en la instrucción <code>ReadRawBytes</code> . |

### Sintaxis

```
ReadRawBytes
  [IODevice ':= ' ] < variable (VAR) of iodev> ' , '
  [RawData ':= ' ] < variable (VAR) of rawbytes> ' , '
  [ '\ ' Time ':= ' < expression (IN) of num> ] ';' ;'
```

### Información relacionada

| Para obtener más información sobre                        | Consulte  |
|---|---|
| <code>rawbytes</code> datos                               | <a href="#">rawbytes - Datos sin formato en la página 1736</a>                                  |
| Obtención de la longitud de un dato <code>rawbytes</code> | <a href="#">RawBytesLen - Obtiene la longitud de un dato de tipo rawbytes en la página 1432</a> |

Continúa en la página siguiente

| Para obtener más información sobre                              | Consulte   |
|---|--|
| Borrado del contenido de un dato de tipo rawbytes               | <a href="#">ClearRawBytes</a> - Borra el contenido de un dato de tipo rawbytes en la página 146        |
| Copiado del contenido de un dato de tipo rawbytes               | <a href="#">CopyRawBytes</a> - Copia el contenido de un dato de tipo rawbytes en la página 171         |
| Empaquetamiento de un encabezado de DeviceNet en datos rawbytes | <a href="#">PackDNHeader</a> - Empaqueta un encabezado de DeviceNet en datos rawbytes en la página 496 |
| Empaquetamiento de datos en datos rawbytes                      | <a href="#">PackRawBytes</a> - Empaqueta datos en un dato de tipo rawbytes en la página 499            |
| Escritura de un dato rawbytes                                   | <a href="#">WriteRawBytes</a> - Escribe un dato de tipo rawbytes en la página 1111                     |
| Desempaquetamiento de datos de un dato rawbytes                 | <a href="#">UnpackRawBytes</a> - Desempaqueta datos de un dato de tipo rawbytes en la página 1024      |
| Gestión de archivos y dispositivos de E/S                       | <a href="#">Application manual</a> - Controller software IRC5  |

## 1 Instrucciones

---

### 1.195 ReadVarArr - Lee múltiples variables de un dispositivo sensor

#### *Sensor Interface*

### 1.195 ReadVarArr - Lee múltiples variables de un dispositivo sensor

---

#### Utilización

**ReadVarArr** Se utiliza para leer hasta seis variables a la vez de un dispositivo sensor. Los resultados se extraen de la misma muestra.

El sensor debe estar configurado y comunicarse a través de la opción *Sensor Interface* de RobotWare.

---

#### Ejemplos básicos

El ejemplo siguiente ilustra la instrucción `ReadVarArr`.

##### Ejemplo 1

```
CONST num xcoord := 8;
CONST num ycoord := 9;
CONST num zcoord := 10;
VAR pos sensorpos;
VAR sensorvardata readdata{4};

! Connect to the sensor device "sen1:" (defined in sio.cfg).
SenDevice "sen1:";

! Read a cartesian position from the sensor.
readdata{1} := [xcoord, 2, false, 1, 0];
readdata{2} := [ycoord, 2, false, 1, 0];
readdata{3} := [zcoord, 2, false, 1, 0];
! A varNumber of -1 will be ignored
readdata{4} := [-1, 2, false, 1, 0];

ReadVarArr "sen1", readdata;
sensorpos.x := DnumToNum(readdata{1}.value);
sensorpos.y := DnumToNum(readdata{2}.value);
sensorpos.z := DnumToNum(readdata{3}.value);
```

El ejemplo muestra una lectura de tres variables al mismo tiempo. La lectura se realiza al mismo tiempo y sobre la misma muestra del sensor.

---

#### Argumentos

```
ReadVarArr Device, Data, [\TaskName]
```

Device

**Tipo de dato:** string

El nombre del dispositivo de E/S configurado en `sio.cfg` para el sensor utilizado.

Data

**Tipo de dato:** sensorvardata

Una variable matricial que hace referencia a una definición de datos de las variables que hay que leer. El valor resultante obtenido se devuelve con su definición.

[ \TaskName ]

**Tipo de dato:** string

*Continúa en la página siguiente*

---

## 1.195 ReadVarArr - Lee múltiples variables de un dispositivo sensor

*Sensor Interface*

*Continuación*

El argumento `TaskName` hace posible el acceso a dispositivos de otras tareas de RAPID.

### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre       | Causa del error                        |
|--------------|--|
| SEN_NO_MEAS  | Fallo de medición                      |
| SEN_NOREADY  | Sensor incapaz de gestionar el comando |
| SEN_GENERRO  | Error general del sensor               |
| SEN_BUSY     | Sensor ocupado                         |
| SEN_UNKNOWN  | Sensor desconocido                     |
| SEN_EXALARM  | Error de sensor externo                |
| SEN_CAALARM  | Error de sensor interno                |
| SEN_TEMP     | Error de temperatura del sensor        |
| SEN_VALUE    | Valor de comunicación no válido        |
| SEN_CAMCHECK | Fallo de comprobación de sensor        |
| SEN_TIMEOUT  | Error de comunicación                  |

### Sintaxis

```
ReadVarArr
  [Device ':='] <expression(IN) of string>', '
  [Data ':='] <array variable {*} (INOUT) of sensorvardata>', '
  ['\ ' TaskName ':=' <expression (IN) of string>'];'
```

### Información relacionada

| Para obtener más información sobre   | Consulte   |
|--|--|
| Establecimiento de una conexión a un dispositivo de sensor                 | <a href="#">SenDevice - Establece una conexión a un dispositivo de sensor en la página 687</a>                               |
| Escribir múltiples variables en un dispositivo                             | <a href="#">WriteVarArr - Escribe múltiples variables en un dispositivo sensor en la página 1119</a>                         |
| Escritura de una variable de sensor  | <a href="#">WriteVar - Escribir una variable en la página 1116</a>   |
| Lectura de una variable de sensor  | <a href="#">ReadVar - Lee una variable de un dispositivo en la página 1451</a>   |
| Escritura de un bloque de datos de sensor                                  | <a href="#">WriteBlock - Escribir un bloque de datos en un dispositivo en la página 1105</a>                                 |
| Lectura de un bloque de datos de sensor                                    | <a href="#">ReadBlock - Lee un bloque de datos de un dispositivo en la página 580</a>  |
| Configuración de múltiples variables de datos para la interfaz de sensores | <a href="#">sensorvardata - Configuración de múltiples variables de datos para la interfaz de sensores en la página 1757</a> |
| Configuración de la comunicación del sensor                                | <a href="#">Manual de referencia técnica - RAPID Overview</a>  |

# 1 Instrucciones

---

1.196 RemoveAllCyclicBool - Eliminar todas las condiciones de Cyclic bool  
RobotWare - OS

## 1.196 RemoveAllCyclicBool - Eliminar todas las condiciones de Cyclic bool

---

### Utilización

RemoveAllCyclicBool se utiliza para eliminar la evaluación cíclica de todas las condiciones de Cyclic bool en la tarea en la que se ejecuta la instrucción.

---

### Ejemplos básicos

El ejemplo siguiente ilustra la instrucción RemoveAllCyclicBool.

#### Ejemplo 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;
  ...
```

En primer lugar se configuran dos evaluaciones cíclicas y posteriormente se eliminan.

#### Ejemplo 2

```
PERS bool cyclicflag1;

PROC main()
  SetupCyclicBool cyclicflag1, di1=1 AND do2=1;
  ...
  RemoveAllCyclicBool;
UNDO
  RemoveAllCyclicBool;
ENDPROC
```

Se eliminan todas las evaluaciones cíclicas cuando el puntero del programa se establece en main.

El mismo comportamiento puede configurarse en los parámetros del sistema sin utilizar un gestor UNDO; consulte *Manual de referencia técnica - Parámetros del sistema*.

---

### Argumentos

```
RemoveAllCyclicBool [\AllTasks]
```

[\AllTasks]

Tipo de dato: switch

Este argumento se utiliza para eliminar la evaluación cíclica para todas las tareas.

*Continúa en la página siguiente*

## 1.196 RemoveAllCyclicBool - Eliminar todas las condiciones de Cyclic bool

RobotWare - OS

Continuación

### Ejecución de programas

El comportamiento de la funcionalidad de Cyclic bool puede configurarse. Para obtener más información, consulte *Application manual - Controller software IRC5* y *Manual de referencia técnica - Parámetros del sistema*.

### Sintaxis

```
RemoveAllCyclicBool  
  ['\AllTasks] ';' 
```

### Información relacionada

| Para obtener más información sobre                             | Consulte   |
|--|--|
| Configurar una condición de Cyclic bool                        | <a href="#">SetupCyclicBool - Configurar una condición de Cyclic bool en la página 716</a> |
| Eliminar una condición de Cyclic bool                          | <a href="#">RemoveCyclicBool - Eliminar una condición de Cyclic bool en la página 596</a>  |
| Condiciones lógicas evaluadas cíclicamente, <i>Cyclic bool</i> | <i>Application manual - Controller software IRC5</i>                                       |
| Configuración de <i>Cyclic bool</i>                            | <i>Manual de referencia técnica - Parámetros del sistema</i>                               |

# 1 Instrucciones

---

## 1.197 RemoveCyclicBool - Eliminar una condición de Cyclic bool RobotWare - OS

### 1.197 RemoveCyclicBool - Eliminar una condición de Cyclic bool

---

#### Utilización

RemoveCyclicBool se utiliza para eliminar la evaluación cíclica de una condición de Cyclic bool.

---

#### Ejemplos básicos

Los siguientes ejemplos ilustran la instrucción RemoveCyclicBool.

##### Ejemplo 1

```
PERS bool cyclicflag1;

PROC main()
  SetupCyclicBool cyclicflag1, dil=1 AND do2=1;
  ...
  RemoveCyclicBool cyclicflag1;
  ...
```

En primer lugar, se configura una evaluación cíclica, y más tarde se elimina.

##### Ejemplo 2

```
PERS bool cyclicflag1;

PROC main()
  SetupCyclicBool cyclicflag1, dil=1 AND do2=1;
  ...
  RemoveCyclicBool "cyclicflag1";
  ...
```

Primeramente, se configura una evaluación cíclica y posteriormente se elimina utilizando el nombre de la variable booleana persistente.

---

#### Argumentos

RemoveCyclicBool Flag | Name

Flag

Tipo de dato: bool

La variable booleana persistente que almacena el valor de la condición lógica.

Name

Tipo de dato: string

El nombre de la variable booleana persistente que almacena el valor de la condición lógica.

---

#### Ejecución de programas

El comportamiento de la funcionalidad de Cyclic bool puede configurarse. Para obtener más información, consulte *Application manual - Controller software IRC5* y *Manual de referencia técnica - Parámetros del sistema*.

*Continúa en la página siguiente*



### Sintaxis

```
RemoveCyclicBool
  [ Flag ':' ] <persistent (PERS) of bool>
  | [ Name ':' ] <expression (IN) of string> ';'

```

### Información relacionada

| Para obtener más información sobre                             | Consulte   |
|--|--|
| Comprobar si una variable persistente es un Cyclic bool        | <a href="#">IsCyclicBool - Comprueba si una variable persistente es un Cyclic bool en la página 1348</a> |
| Configurar una condición de Cyclic bool                        | <a href="#">SetupCyclicBool - Configurar una condición de Cyclic bool en la página 716</a>               |
| Eliminar todas las condiciones de Cyclic bool                  | <a href="#">RemoveAllCyclicBool - Eliminar todas las condiciones de Cyclic bool en la página 594</a>     |
| Condiciones lógicas evaluadas cíclicamente, <i>Cyclic bool</i> | <i>Application manual - Controller software IRC5</i>   |
| Configuración de <i>Cyclic bool</i>                            | <i>Manual de referencia técnica - Parámetros del sistema</i>   |

# 1 Instrucciones

---

## 1.198 RemoveDir - Elimina un directorio

RobotWare - OS

## 1.198 RemoveDir - Elimina un directorio

---

### Utilización

`RemoveDir` se utiliza para eliminar un directorio.

El usuario debe tener permisos de escritura y ejecución en el directorio y éste debe estar vacío.

---

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción `RemoveDir`:

#### Ejemplo 1

```
RemoveDir "HOME:/mydir";
```

En este ejemplo, se elimina el directorio `mydir` que está situado debajo de `HOME:.`

---

### Argumentos

```
RemoveDir Path
```

Path

Tipo de dato: `string`

El nombre del directorio que se desea eliminar, especificado con una ruta completa o relativa.

---

### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                   | Causa del error  |
|--------------------------|--|
| <code>ERR_FILEACC</code> | El directorio no existe o el directorio no está vacío o el usuario no tiene permiso de escritura y ejecución en la biblioteca. |

---

### Sintaxis

```
RemoveDir  
[ Path':=' ] < expression (IN) of string>';'
```

---

### Información relacionada

| Para obtener más información sobre | Consulte  |
|------------------------------------|---|
| Directorio                         | <a href="#">dir - Estructura de directorio de archivos en la página 1659</a>          |
| Apertura de un directorio          | <a href="#">OpenDir - Abre un directorio en la página 494</a>                         |
| Lectura de un directorio           | <a href="#">ReadDir - Lee la siguiente entrada de un directorio en la página 1437</a> |
| Cierre de un directorio            | <a href="#">CloseDir - Cierra un directorio en la página 153</a>                      |
| Creación de un directorio          | <a href="#">MakeDir - Crea un nuevo directorio en la página 358</a>                   |
| Cambio del nombre de un archivo    | <a href="#">RenameFile - Cambia el nombre de un archivo en la página 604</a>          |

---

Continúa en la página siguiente

1.198 RemoveDir - Elimina un directorio  
RobotWare - OS  
Continuación

| Para obtener más información sobre              | Consulte   |
|---|--|
| Eliminación de un archivo                       | <a href="#">RemoveFile - Elimina un archivo en la página 600</a>                       |
| Copia de un archivo                             | <a href="#">CopyFile - Copia un archivo en la página 169</a>                           |
| Comprobación del tipo del archivo               | <a href="#">IsFile - Comprobar el tipo de un archivo en la página 1351</a>             |
| Comprobación del tamaño del archivo             | <a href="#">FileSize - Obtiene el tamaño de un archivo en la página 1285</a>           |
| Comprobación del tamaño del sistema de archivos | <a href="#">FSSize - Obtiene el tamaño de un sistema de archivos en la página 1291</a> |
| Gestión de archivos y dispositivos de E/S       | <a href="#">Application manual - Controller software IRC5</a>                          |

# 1 Instrucciones

---

## 1.199 RemoveFile - Elimina un archivo

RobotWare - OS

## 1.199 RemoveFile - Elimina un archivo

---

### Utilización

`RemoveFile` se utiliza para eliminar un archivo. El usuario debe tener permisos de escritura y ejecución en el directorio en el que se encuentra el archivo, además de permiso de escritura para el archivo en sí.

---

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción `RemoveFile`:

#### Ejemplo 1

```
RemoveFile "HOME:/mydir/myfile.log";
```

En este ejemplo, se elimina el archivo `myfile.log` en el directorio `mydir` del disco `HOME` .

---

### Argumentos

```
RemoveFile Path
```

Path

Tipo de dato: string

El nombre del archivo que se desea eliminar, especificado con una ruta completa o relativa.

---

### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                   | Causa del error  |
|--------------------------|--|
| <code>ERR_FILEACC</code> | El acceso a un archivo se realiza de forma incorrecta. |

---

### Sintaxis

```
RemoveFile  
[ Path':=' ] < expression (IN) of string>';'
```

---

### Información relacionada

| Para obtener más información sobre  | Consulte   |
|-------------------------------------|--|
| Creación de un directorio           | <a href="#">MakeDir - Crea un nuevo directorio en la página 358</a>          |
| Eliminación de un directorio        | <a href="#">RemoveDir - Elimina un directorio en la página 598</a>           |
| Cambio del nombre de un archivo     | <a href="#">RenameFile - Cambia el nombre de un archivo en la página 604</a> |
| Copia de un archivo                 | <a href="#">CopyFile - Copia un archivo en la página 169</a>                 |
| Comprobación del tipo del archivo   | <a href="#">IsFile - Comprobar el tipo de un archivo en la página 1351</a>   |
| Comprobación del tamaño del archivo | <a href="#">FileSize - Obtiene el tamaño de un archivo en la página 1285</a> |

---

*Continúa en la página siguiente*

## 1.199 RemoveFile - Elimina un archivo

*RobotWare - OS*

*Continuación*

| Para obtener más información sobre              | Consulte   |
|---|--|
| Comprobación del tamaño del sistema de archivos | <a href="#">FSSize - Obtiene el tamaño de un sistema de archivos en la página 1291</a> |
| Gestión de archivos y dispositivos de E/S       | <i>Application manual - Controller software IRC5</i>                                   |

# 1 Instrucciones

## 1.200 RemoveSuperv - Eliminar la condición de una señal *Continuous Application Platform (CAP)*

## 1.200 RemoveSuperv - Eliminar la condición de una señal

### Utilización

RemoveSuperv se utiliza para eliminar de la supervisión las condiciones añadidas mediante SetupSuperv.

### Ejemplo básico

```
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
```

Elimina la señal *di\_Arc\_Sup* de la lista START.

### Argumentos

RemoveSuperv Signal Condition Listtype

#### Signal

Tipo de dato: signaldi

La señal digital que se desea eliminar de la lista de supervisión.

#### Condition

Tipo de dato: num

El nombre representa una de las siguientes condiciones disponibles:

|           |   |
|-----------|---|
| ACT:      | Se utiliza para la supervisión del estado. Estado de señal previsto durante la supervisión: activa. Si la señal cambia a pasiva, se dispara la supervisión.   |
| PAS:      | Se utiliza para la supervisión del estado. Estado de señal previsto durante la supervisión: pasiva. Si la señal cambia a activa, se dispara la supervisión.   |
| POS_EDGE: | Se utiliza para la supervisión del intercambio. Estado de señal previsto al final de la supervisión: activa. Si la señal no cambia a activa dentro del tiempo límite seleccionado, se dispara la supervisión. |
| NEG_EDGE: | Se utiliza para la supervisión del intercambio. Estado de señal previsto al final de la supervisión: pasiva. Si la señal no cambia a pasiva dentro del tiempo límite seleccionado, se dispara la supervisión. |

#### Listtype

Tipo de dato: num

El nombre que representa el número de las diferentes listas (por ejemplo, las fases del proceso):

- SUPERV\_PRE
- SUPERV\_PRE\_START
- SUPERV\_END\_PRE
- SUPERV\_START
- SUPERV\_MAIN

*Continúa en la página siguiente*

## 1.200 RemoveSuperv - Eliminar la condición de una señal *Continuous Application Platform (CAP)*

*Continuación*

- SUPERV\_END\_MAIN
- SUPERV\_START\_POST1
- SUPERV\_POST1
- SUPERV\_END\_POST1
- SUPERV\_START\_POST2
- SUPERV\_POST2
- SUPERV\_END\_POST2

### Sintaxis

RemoveSuperv

```
[Signal ':='] < variable (VAR) of signaldi > ','
```

```
[Condition ':='] < variable (IN) of num > ','
```

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

### Información relacionada

| Para obtener más información sobre     | Consulte  |
|--|---|
| <i>Continuous Application Platform</i> | <i>Application manual - Continuous Application Platform</i>   |
| Instrucción InitSuperv                 | <a href="#">InitSuperv - Restablecer toda la supervisión para CAP en la página 286</a>                          |
| Instrucción SetupSuperv                | <a href="#">SetupSuperv - Configurar las condiciones para la supervisión de señales en CAP en la página 719</a> |

# 1 Instrucciones

---

## 1.201 RenameFile - Cambia el nombre de un archivo

RobotWare - OS

## 1.201 RenameFile - Cambia el nombre de un archivo

---

### Utilización

`RenameFile` se utiliza para dar un nuevo nombre a un archivo existente. También puede usarse para trasladar un archivo de un lugar a otro dentro de la estructura de directorios.

---

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción `RenameFile`:

#### Ejemplo 1

```
RenameFile "HOME:/myfile", "HOME:/yourfile";
```

Se da al archivo `myfile` el nombre `yourfile`.

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

Se da al archivo `myfile` el nombre `yourfile`, además de trasladarlo al directorio `mydir`.

---

### Argumentos

```
RenameFile OldPath NewPath
```

OldPath

Tipo de dato:string

La ruta completa del archivo cuyo nombre se desea cambiar.

NewPath

Tipo de dato:string

La ruta completa del archivo cuyo nombre se desea cambiar.

---

### Ejecución de programas

El archivo especificado en `OldPath` recibe el nombre especificado en `NewPath`. Si la ruta indicada en `NewPath` es distinta de la ruta del archivo `OldPath`, el archivo también se traslada a una nueva ubicación.

---

### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                     | Causa del error  |
|----------------------------|--|
| <code>ERR_FILEACC</code>   | El archivo especificado en <code>OldPath</code> no existe. |
| <code>ERR_FILEEXIST</code> | El archivo especificado en <code>NewPath</code> ya existe. |

---

### Sintaxis

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

---

Continúa en la página siguiente



### Información relacionada

| Para obtener más información sobre              | Consulte   |
|---|--|
| Creación de un directorio                       | <a href="#">MakeDir - Crea un nuevo directorio en la página 358</a>                    |
| Eliminación de un directorio                    | <a href="#">RemoveDir - Elimina un directorio en la página 598</a>                     |
| Eliminación de un archivo                       | <a href="#">RemoveFile - Elimina un archivo en la página 600</a>                       |
| Copia de un archivo                             | <a href="#">CopyFile - Copia un archivo en la página 169</a>                           |
| Comprobación del tipo del archivo               | <a href="#">IsFile - Comprobar el tipo de un archivo en la página 1351</a>             |
| Comprobación del tamaño del archivo             | <a href="#">FileSize - Obtiene el tamaño de un archivo en la página 1285</a>           |
| Comprobación del tamaño del sistema de archivos | <a href="#">FSSize - Obtiene el tamaño de un sistema de archivos en la página 1291</a> |
| Gestión de archivos y dispositivos de E/S       | <a href="#">Application manual - Controller software IRC5</a>                          |

# 1 Instrucciones

## 1.202 Reset - Pone a cero una señal digital de salida

RobotWare - OS

## 1.202 Reset - Pone a cero una señal digital de salida

### Utilización

Reset se utiliza para poner a cero el valor de una señal digital de salida.

### Ejemplos básicos

Los ejemplos siguientes ilustran la instrucción `Reset`.

#### Ejemplo 1

```
Reset do15;
```

Se cambia la señal `do15` a 0.

#### Ejemplo 2

```
Reset weld;
```

Se cambia la señal `weld` a 0.

### Argumentos

```
Reset Signal
```

Signal

Tipo de dato: `signaldo`

El nombre de la señal que se desea poner a cero.

### Ejecución de programas

El valor real depende de la configuración de la señal. Si la señal está invertida en los parámetros del sistema, esta instrucción hace que el canal físico cambie a 1.

### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                          | Causa del error   |
|---------------------------------|---|
| <code>ERR_NO_ALIASIO_DEF</code> | La variable de señal es una variable declarada en <code>RAPID</code> . No se ha conectado a una señal E/S definida en la configuración de E/S con la instrucción <code>AliasIO</code> .               |
| <code>ERR_NORUNUNIT</code>      | Se ha perdido el contacto con el dispositivo de E/S.  |
| <code>ERR_SIG_NOT_VALID</code>  | La señal de E/S no está disponible. Los motivos pueden ser que el dispositivo de E/S no está en funcionamiento o que exista un error en la configuración (válido solamente para el bus de campo ICI). |

### Sintaxis

```
Reset  
[Signal '[:=' ] <variable (VAR) of signaldo>'];'
```

### Información relacionada

| Para obtener más información sobre           | Consulte  |
|--|---|
| Cambio de una señal digital de salida a cero | <a href="#">Set - Activa una señal digital de salida en la página 689</a> |

Continúa en la página siguiente

1.202 Reset - Pone a cero una señal digital de salida

*RobotWare - OS*

*Continuación*

| <b>Para obtener más información sobre</b>  | <b>Consulte</b>  |
|--|--|
| Instrucciones de entrada/salida            | <i>Manual de referencia técnica - RAPID Overview</i>         |
| Funcionalidad de entrada/salida en general | <i>Manual de referencia técnica - RAPID Overview</i>         |
| Configuración de E/S                       | <i>Manual de referencia técnica - Parámetros del sistema</i> |

# 1 Instrucciones

---

1.203 ResetAxisDistance - Restablece la información de distancia recorrida para el eje  
RobotWare - OS

## 1.203 ResetAxisDistance - Restablece la información de distancia recorrida para el eje

---

### Utilización

ResetAxisDistance Se utiliza para restablecer la información de distancia recorrida para el eje.

---

### Ejemplos básicos

Los siguientes ejemplos ilustran la instrucción ResetAxisDistance.

#### Ejemplo 1

```
ResetAxisDistance Track,1;
```

Se restablecerá la información de distancia recorrida para el eje 1 en la unidad mecánica Track.

#### Ejemplo 2

```
PERS dnum distanceLimit := 1000;

PROC main()
  IF GetAxisDistance(Track,1) > distanceLimit THEN
    ErrWrite \I, "Distance counter limit reached", "Distance counter
      limit for Track has been reached.";
    DoMaintenance();
  ENDIF
ENDPROC

PROC DoMaintenance()
  ...
  ResetAxisDistance Track, 1;
  ErrWrite \I, "Distance counter reset", "Distance counter for
    Track has been reset.";
ENDPROC
```

El ejemplo describe cómo puede usarse ResetAxisDistance junto con GetAxisDistance para comprobar si ha llegado el momento de realizar el mantenimiento de un eje.

---

### Argumentos

```
ResetAxisDistance MechUnit AxisNo
```

MechUnit

**Mechanical Unit**

Tipo de dato: mecunit

El nombre de la unidad mecánica.

AxisNo

Tipo de dato: num

El número del eje en el que se restablecerá la distancia recorrida.

---

*Continúa en la página siguiente*

## 1.203 ResetAxisDistance - Restablece la información de distancia recorrida para el eje RobotWare - OS Continuación

### Ejecución de programas

Restablece la información de distancia recorrida por el eje seleccionado.

### Sintaxis

```
ResetAxisDistance  
  [MechUnit ':='] < variable (VAR) of mecunit > ','  
  [AxisNo ':='] < variable (VAR) of num > ';' ;
```

### Información relacionada

| Para obtener más información sobre | Consulte   |
|------------------------------------|--|
| ResetAxisMoveTime                  | <a href="#">ResetAxisMoveTime - Restablece el cronómetro de movimiento del eje en la página 610</a>              |
| GetAxisDistance                    | <a href="#">GetAxisDistance - Proporciona la información de distancia recorrida por el eje en la página 1294</a> |
| GetAxisMoveTime                    | <a href="#">GetAxisMoveTime - Obtiene el valor del cronómetro de movimiento del eje en la página 1296</a>        |

# 1 Instrucciones

---

## 1.204 ResetAxisMoveTime - Restablece el cronómetro de movimiento del eje

### 1.204 ResetAxisMoveTime - Restablece el cronómetro de movimiento del eje

---

#### Utilización

`ResetAxisMoveTime` Se utiliza para restablecer la información de tiempo de movimiento del eje.

---

#### Ejemplos básicos

Los siguientes ejemplos ilustran la instrucción `ResetAxisMoveTime`.

##### Ejemplo 1

```
ResetAxisMoveTime Track,1;
```

Se restablecerá la información del tiempo de movimiento del eje 1 en la unidad mecánica `Track`.

##### Ejemplo 2

```
PERS dnum timeLimit := 1000;

PROC main()
  IF GetAxisMoveTime(Track,1) > timeLimit THEN
    ErrWrite \I, "Time counter limit reached", "Time counter limit
      for Track has been reached.";
    DoMaintenance();
  ENDIF
ENDPROC

PROC DoMaintenance()
  ...
  ResetAxisMoveTime Track, 1;
  ErrWrite \I, "Time counter reset", "Time counter for Track has
    been reset.";
ENDPROC
```

El ejemplo describe cómo puede usarse `ResetAxisMoveTime` junto con `GetAxisMoveTime` para comprobar si ha llegado el momento de realizar el mantenimiento de un eje.

---

#### Argumentos

```
ResetAxisMoveTime MechUnit AxisNo
```

`MechUnit`

***Mechanical Unit***

Tipo de dato: `mecunit`

El nombre de la unidad mecánica.

`AxisNo`

Tipo de dato: `num`

El número del eje en el que se restablecerá la información de tiempo de movimiento.

---

#### Ejecución de programas

Restablece la información del tiempo de movimiento del eje seleccionado.

*Continúa en la página siguiente*

---

## 1.204 ResetAxisMoveTime - Restablece el cronómetro de movimiento del eje

*Continuación*

### Sintaxis

```
ResetAxisMoveTime  
  [MechUnit ':='] < variable (VAR) of mecunit > ','  
  [AxisNo ':='] < variable (VAR) of num > ';' ;
```

### Información relacionada

| Para obtener más información sobre | Consulte  |
|------------------------------------|---|
| ResetAxisDistance                  | <a href="#">ResetAxisDistance - Restablece la información de distancia recorrida para el eje en la página 608</a> |
| GetAxisDistance                    | <a href="#">GetAxisDistance - Proporciona la información de distancia recorrida por el eje en la página 1294</a>  |
| GetAxisMoveTime                    | <a href="#">GetAxisMoveTime - Obtiene el valor del cronómetro de movimiento del eje en la página 1296</a>         |

## 1 Instrucciones

---

1.205 ResetPPMoved - Restablecer el estado del puntero de programa movido en el modo manual.  
*RobotWare - OS*

### 1.205 ResetPPMoved - Restablecer el estado del puntero de programa movido en el modo manual.

---

#### Utilización

ResetPPMoved restablece el estado del puntero de programa movido en el modo manual. PPMovedInManMode devuelve TRUE si el usuario ha movido el puntero de programa mientras el controlador se encuentra en el modo manual, es decir, si la llave de operador está en las posiciones de manual a velocidad reducida o manual a máxima velocidad. El estado del puntero de programa movido se pone a cero cuando se sitúa la llave de Auto a Man o cuando se usa la instrucción ResetPPMoved.

---

#### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción ResetPPMoved:

##### Ejemplo 1

```
IF PPMovedInManMode() THEN
  WarnUserOfPPMovement;
  ! DO THIS ONLY ONCE
  ResetPPMoved;
  DoJob;
ELSE
  DoJob;
ENDIF
```

---

#### Ejecución de programas

Restablece el estado del puntero de programa movido en el modo manual para la tarea de programa actual.

---

#### Sintaxis

```
ResetPPMoved';'
```

---

#### Información relacionada

| Para obtener más información sobre                                 | Consulte   |
|--|--|
| Comprobar si el puntero de programa se ha movido en el modo manual | <a href="#">PPMovedInManMode - Comprobar si el puntero de programa se ha movido en el modo manual. en la página 1426</a> |



## 1.206 ResetRetryCount - Restablecer el número de reintentos

### Utilización

ResetRetryCount se usa para restablecer el número de reintentos que se han realizado desde un gestor de errores. El número máximo de reintentos que pueden hacerse se define en la configuración.

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción ResetRetryCount:

#### Ejemplo 1

```
VAR num myretries := 0;
...
ERROR
  IF myretries > 2 THEN
    ResetRetryCount;
    myretries := 0;
    TRYNEXT;
  ENDIF
  myretries:= myretries + 1;
  RETRY;
...
```

El programa reintentará 3 veces la instrucción fallida y a continuación intentará la instrucción siguiente. El contador de reintentos interno del sistema se restablece antes de reintentar la instrucción siguiente (incluso si tal reintento es realizado por el sistema en TRYNEXT).

### Ejecución de programas

Para cada RETRY realizado desde un gestor de errores, un contador interno del sistema comprobará que no se sobrepase el número máximo de reintentos, especificado en la configuración. La ejecución de la instrucción ResetRetryCount pone a cero el contador y hace posible volver a hacer un número máximo de reintentos.

### Sintaxis

```
ResetRetryCount ';' ;'
```

### Información relacionada

| Para obtener más información sobre              | Consulte   |
|---|--|
| Gestores de errores                             | <i>Manual de referencia técnica - RAPID Overview</i>                                     |
| Reanudación de la ejecución después de un error | <a href="#">RETRY - Reanudar la ejecución después de un error en la página 617</a>       |
| Configuración del número máximo de reintentos   | <i>Manual de referencia técnica - Parámetros del sistema</i>                             |
| Número de reintentos restantes                  | <a href="#">RemainingRetries - Reintentos restantes aún pendientes en la página 1455</a> |

# 1 Instrucciones

---

1.207 ResetTorqueMargin - Restablecer el menor margen de par  
RobotWare - OS

## 1.207 ResetTorqueMargin - Restablecer el menor margen de par

---

### Utilización

ResetTorqueMargin se utiliza para restablecer el menor margen de par para permitir el comienzo de una nueva medición.

---

### Ejemplos básicos

El ejemplo siguiente ilustra la instrucción ResetTorqueMargin.

#### Ejemplo 1

```
ResetTorqueMargin \AxisNo:=3;  
! starts a new measurement for axis 3;
```

---

### Argumentos

```
ResetTorqueMargin [ \MecUnit ] [ \AxisNo ]
```

[ \MecUnit ]

**Tipo de dato:** mecunit

El nombre de la unidad mecánica cuyos valores de eje se desea restablecer. Si se omite este argumento, se restablece el valor de un eje del robot conectado.

[ \AxisNo ]

**Tipo de dato:** num

El número del eje cuyo valor se desea restablecer (de 1 a -6).

Si no se especifica ningún eje, se restablecen todos los ejes para la unidad mecánica.

---

### Gestión de errores

Se generan los siguientes errores recuperables que pueden gestionarse con un gestor de errores. La variable del sistema ERRNO se establecerá en:

|              |  |
|--------------|--|
| ERR_AXIS_PAR | Parámetro de eje incorrecto en la función. |
|--------------|--|

---

### Sintaxis

```
ResetTorqueMargin '('  
  [ '\ ' MechUnit ' := ' ] < variable (VAR) of mecunit > ','  
  [ '\ ' AxisNo ' := ' < expression (IN) of num > ] ')'
```

---

### Información relacionada

| Para obtener más información sobre | Consulte  |
|------------------------------------|---|
| Obtener margen de par              | <a href="#">GetTorqueMargin - Lee el menor margen de par en la página 238</a> |

---

## 1.208 RestoPath - Restablece la trayectoria después de una interrupción

### Utilización

RestoPath se utiliza para restablecer una trayectoria almacenada en un momento anterior con la instrucción StorePath.

Esta instrucción sólo puede usarse en la tarea principal T\_ROB1 o, si se cuenta con un sistema MultiMove, en las tareas de movimiento.

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción RestoPath:

Consulte también *Más ejemplos*, a continuación.

#### Ejemplo 1

```
RestoPath;
```

Restablece la trayectoria almacenada anteriormente mediante StorePath.

### Ejecución de programas

La trayectoria de movimiento actual de los ejes del robot y de los ejes externos se elimina y la trayectoria almacenada anteriormente con la instrucción StorePath se restablece. Sin embargo, recuerde que no se produce ningún movimiento hasta que se ejecuta la instrucción StartMove o se reanuda la ejecución mediante RETRY en un gestor de errores.

### Más ejemplos

A continuación aparecen más ejemplos de cómo usar la instrucción RestoPath.

#### Ejemplo 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 robtarg p1;
  StorePath;
  p1 := CRobT();
  MoveL pclean, v100, fine, gun1;
  ...
  MoveL p1, v100, fine, gun1;
  RestoPath;
ENDPROC
```

En caso de un error de soldadura, la ejecución del programa continúa en el gestor de errores de la rutina, lo cual llama a su vez a `gun_cleaning`. A continuación, se almacena la trayectoria de movimiento que se estaba ejecutando y el robot se traslada a la posición `pclean`, en la que se soluciona el error. Una vez hecho

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.208 RestoPath - Restablece la trayectoria después de una interrupción

RobotWare - OS

Continuación

esto, el robot vuelve a la posición en la que se produjo el error,  $p1$ , y almacena de nuevo el movimiento original. A continuación, la soldadura se reanuda automáticamente, lo que significa que el robot invierte su movimiento a lo largo de la trayectoria antes de que comience la soldadura, tras lo cual puede continuar la ejecución normal del programa.

### Limitaciones

Con la instrucción `StorePath` sólo se almacenan los datos de la trayectoria de movimiento. Si el usuario desea solicitar movimientos en un nuevo nivel de trayectoria, es necesario almacenar la posición de paro directamente a continuación de que `StorePath` y de `RestoPath` hacen el movimiento hacia la posición de paro almacenada en la trayectoria.

Si esta instrucción va precedida de una instrucción de movimiento, ésta última debe programarse con un punto de paro (zonedata `fine`), no un punto de paso. De lo contrario, no será posible reanudar la ejecución tras una caída de suministro eléctrico.

`RestoPath` no puede ejecutarse en rutinas de RAPID que estén conectadas a los siguientes eventos especiales del sistema: `PowerOn`, `Stop`, `QStop`, `Restart` o `Step`.

### Sintaxis

```
RestoPath';'
```

### Información relacionada

| Para obtener más información sobre | Consulte   |
|------------------------------------|--|
| Almacenamiento de trayectorias     | <a href="#">StorePath - Almacena la trayectoria cuando se produce una interrupción en la página 835</a>  |
| Más ejemplos                       | <a href="#">StorePath - Almacena la trayectoria cuando se produce una interrupción en la página 835</a><br><a href="#">PathRecStart - Inicia la grabadora de trayectorias en la página 524</a><br><a href="#">SyncMoveSuspend - Activa el movimiento independiente-semicoordinado en la página 862</a> |

## 1.209 RETRY - Reanudar la ejecución después de un error

### Utilización

La instrucción `RETRY` se utiliza para reanudar la ejecución del programa después de un error, empezando por (y ejecutando de nuevo) la instrucción que provocó el error.

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción `RETRY`:

#### Ejemplo 1

```
reg2 := reg3/reg4;
...
ERROR
  IF ERRNO = ERR_DIVZERO THEN
    reg4 :=1;
    RETRY;
  ENDIF
```

Se intenta dividir `reg3` por `reg4`. Si `reg4` es igual a 0 (lo cual da lugar a una división por cero), se salta al gestor de errores, que inicializa `reg4`. A continuación, se usa la instrucción `RETRY` para saltar desde el gestor de errores y se hace otro intento de completar la división.

### Ejecución de programas

La ejecución del programa continúa en (y ejecuta de nuevo) la instrucción que provocó el error.

### Gestión de errores

Si se supera el número máximo de reintentos (4 reintentos), la ejecución del programa se detiene y se genera un mensaje de error. El número máximo de reintentos puede configurarse en los Parámetros del sistema (tipo *General RAPID*).

### Limitaciones

La instrucción sólo puede existir en el gestor de errores de la rutina. Si el error se creó usando una instrucción `RAISE`, no es posible reanudar la ejecución del programa con una instrucción `RETRY`. A continuación debe utilizarse la instrucción `TRYNEXT`.

### Sintaxis

```
RETRY ;'
```

### Información relacionada

| Para obtener más información sobre            | Consulte   |
|---|--|
| Gestores de errores                           | <i>Manual de referencia técnica - RAPID Overview</i>         |
| Configuración del número máximo de reintentos | <i>Manual de referencia técnica - Parámetros del sistema</i> |

*Continúa en la página siguiente*

# 1 Instrucciones

---

1.209 RETRY - Reanudar la ejecución después de un error

RobotWare - OS

Continuación

| Para obtener más información sobre       | Consulte   |
|--|--|
| Continuación en la instrucción siguiente | <a href="#">TRYNEXT - Salta sobre una instrucción que ha provocado un error en la página 992</a> |

### 1.210 RETURN - Finaliza la ejecución de una rutina

#### Utilización

`RETURN` se utiliza para finalizar la ejecución de una rutina. Si la rutina es una función, el valor también se devuelve.

#### Ejemplos básicos

Los ejemplos siguientes ilustran la instrucción `RETURN`.

##### Ejemplo 1

```
errormessage;  
Set do1;  
...  
PROC errormessage()  
  IF dil=1 THEN  
    RETURN;  
  ENDIF  
  TPWrite "Error";  
ENDPROC
```

Se llama al procedimiento `errormessage`. Cuando el procedimiento llega a la instrucción `RETURN`, la ejecución del programa vuelve a la instrucción que sigue a la llamada al procedimiento, `Set do 1`.

##### Ejemplo 2

```
FUNC num abs_value(num value)  
  IF value<0 THEN  
    RETURN -value;  
  ELSE  
    RETURN value;  
  ENDIF  
ENDFUNC
```

La función devuelve el valor absoluto de un número.

#### Argumentos

```
RETURN [ Return value ]
```

Return value

**Tipo de dato:** Según la declaración de la función.

El valor de retorno de una función.

El valor de retorno debe especificarse en una instrucción `RETURN` incluida en una función.

Si la instrucción se incluye en un procedimiento o una rutina `TRAP`, es posible que el número del error no esté especificado.

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.210 RETURN - Finaliza la ejecución de una rutina

RobotWare - OS

Continuación

---

### Ejecución de programas

El resultado de la instrucción `RETURN` puede variar, en función del tipo de rutina en el que se utiliza:

- Rutina principal: Si el programa tiene el modo de ejecución de un solo ciclo, el programa se detiene. De lo contrario, la ejecución del programa continúa con la primera instrucción de la rutina principal.
- Procedimiento: La ejecución del programa continúa con la instrucción que sigue a la llamada al procedimiento.
- Función: Devuelve el valor de la función.
- Rutina TRAP: La ejecución del programa continúa en el punto en el que se produjo la interrupción.
- Gestor de errores en un procedimiento: La ejecución del programa continúa con la rutina que llamó a la rutina que contiene el gestor de errores (en la instrucción que sigue a la llamada al procedimiento).
- Gestor de errores en una función: Se devuelve el valor de la función.

---

### Sintaxis

```
RETURN [ <expression> ]';'
```

---

### Información relacionada

| Para obtener más información sobre | Consulte   |
|------------------------------------|--|
| Funciones y procedimientos         | <i>Manual de referencia técnica - RAPID Overview</i> |
| Rutinas TRAP                       | <i>Manual de referencia técnica - RAPID Overview</i> |
| Gestores de errores                | <i>Manual de referencia técnica - RAPID Overview</i> |



## 1.211 Rewind - Rebobina la posición del archivo

### Utilización

Rewind sitúa la posición del archivo al principio del archivo.

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción `Rewind`:

#### Ejemplo 1

```
Rewind iodev1;
```

En el archivo al que se hace referencia con `iodev1`, la posición del archivo se sitúa en el principio del archivo.

### Argumentos

```
Rewind IODevice
```

IODevice

Tipo de dato: `iodev`

El nombre (la referencia) del archivo que se desea rebobinar.

### Ejecución de programas

El archivo especificado se rebobina hasta el principio.

En caso de un reinicio tras una caída de alimentación, todos los archivos o dispositivos de E/S abiertos del sistema se cierran y el descriptor de E/S de la variable del tipo `iodev` se restablece.

### Limitaciones

Si el archivo utilizado se ha abierto con un modificador `\Bin` o `\Bin \Append`, el uso de `Rewind` antes de cualquier tipo de instrucción `Write` no tendrá ningún efecto. La escritura se realizará al final del archivo.

### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                   | Causa del error  |
|--------------------------|--|
| <code>ERR_FILEACC</code> | El acceso a un archivo se realiza de forma incorrecta. |

### Sintaxis

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

### Información relacionada

| Para obtener más información sobre        | Consulte   |
|---|--|
| Apertura y otras operaciones con archivos | <i>Manual de referencia técnica - RAPID Overview</i> |
| Gestión de archivos y dispositivos de E/S | <i>Application manual - Controller software IRC5</i> |

# 1 Instrucciones

## 1.212 RMQEmptyQueue - Vacía la cola de mensajes de RAPID *FlexPendant Interface, PC Interface, or Multitasking*

### 1.212 RMQEmptyQueue - Vacía la cola de mensajes de RAPID

#### Utilización

`RMQEmptyQueue` vacía la cola de mensajes de RAPID (RMQ) de la tarea en la que se ejecuta la instrucción.

#### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción `RMQEmptyQueue`:

#### Ejemplo

```
RMQEmptyQueue;
```

La instrucción `RMQEmptyQueue` elimina todos los mensajes de la cola RMQ de la tarea en ejecución.

#### Ejecución de programas

Se vacía la cola de mensajes de RAPID de la tarea en la que se ejecuta. Esta instrucción puede usarse en todos los niveles de ejecución.

#### Limitaciones

`RMQEmptyQueue` sólo vacía la cola de mensajes de RAPID de la tarea en la que se ejecuta la instrucción. Todas las demás colas de mensajes de RAPID permanecen sin cambios.

#### Sintaxis

```
RMQEmptyQueue ';' ;
```

#### Información relacionada

| Para obtener más información sobre   | Consulte   |
|--|--|
| Descripción de la funcionalidad de RAPID Message Queue                           | <a href="#">Application manual - Controller software IRC5, sección RAPID Message Queue.</a>                |
| Tipo de dato <code>rmqmessage</code>   | <a href="#">rmqmessage - Mensaje de RAPID Message Queue en la página 1746.</a>                             |
| Enviar datos a la cola de una tarea de RAPID                                     | <a href="#">RMQSendMessage - Enviar un mensaje de datos de RMQ en la página 638.</a>                       |
| Enviar datos a la cola de una tarea de RAPID y esperar una respuesta del cliente | <a href="#">RMQSendWait - Enviar un mensaje de datos de RMQ y esperar una respuesta en la página 642.</a>  |
| Encontrar el número de identidad de una tarea de RAPID Message Queue             | <a href="#">RMQFindSlot - Buscar una identidad de ranura para el nombre de ranura en la página 624.</a>    |
| Extraer los datos de encabezado de un <code>rmqmessage</code>                    | <a href="#">RMQGetMsgHeader - Obtener información de encabezado de un mensaje de RMQ en la página 632.</a> |
| Extraer los datos de un <code>rmqmessage</code>                                  | <a href="#">RMQGetMsgData - Obtener la parte de datos de un mensaje de RMQ en la página 629.</a>           |
| Ordenar y habilitar interrupciones para un tipo de dato en concreto              | <a href="#">IRMQMessage - Ordenar interrupciones de RMQ para un tipo de dato en la página 305.</a>         |

Continúa en la página siguiente

## 1.212 RMQEmptyQueue - Vacía la cola de mensajes de RAPID *FlexPendant Interface, PC Interface, or Multitasking* Continuación

| Para obtener más información sobre                                   | Consulte   |
|--|--|
| Obtener el nombre de ranura de una identidad de ranura especificada. | <a href="#">RMQGetSlotName - Obtener el nombre de un cliente de RMQ en la página 1456.</a> |
| Recibir un mensaje de una cola RMQ                                   | <a href="#">RMQReadWait - Devuelve un mensaje de una cola RMQ en la página 635.</a>        |
| Obtener el primer mensaje de una cola de RAPID Message Queue         | <a href="#">RMQGetMessage - Obtener un mensaje de RMQ en la página 626.</a>                |

## 1 Instrucciones

---

### 1.213 RMQFindSlot - Buscar una identidad de ranura para el nombre de ranura *FlexPendant Interface, PC Interface, or Multitasking*

### 1.213 RMQFindSlot - Buscar una identidad de ranura para el nombre de ranura

---

#### Utilización

RMQFindSlot (*RAPID Message Queue Find Slot*) se utiliza para encontrar la identidad de ranura de un RMQ configurado para una tarea de RAPID, o la identidad de ranura de un cliente de Robot Application Builder.

---

#### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción RMQFindSlot:

##### Ejemplo 1

```
VAR rmqslot myrmqslot;  
RMQFindSlot myrmqslot, "RMQ_T_ROB2";
```

Obtener el número de identidad del RMQ "RMQ\_T\_ROB2" configurado para la tarea de RAPID "T\_ROB2".

---

#### Argumentos

RMQFindSlot Slot Name

Slot

Tipo de dato: `rmqslot`

La variable en la que se devuelve el identificador numérico.

Name

Tipo de dato: `string`

El nombre del cliente cuyo número de identidad se desea encontrar. El nombre debe ser correcto en cuanto al uso de minúsculas y mayúsculas. Si la tarea de RAPID tiene el nombre T\_ROB1 y se utiliza el nombre RMQ\_t\_rob1 para la cola RMQ, dará lugar a un error (consulte el capítulo sobre gestión de errores que aparece a continuación).

---

#### Ejecución de programas

La instrucción RMQFindSlot se utiliza para encontrar la identidad de ranura de un RMQ con nombre o un cliente de Robot Application Builder.

---

#### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre       | Causa del error  |
|--------------|--|
| ERR_RMQ_NAME | El nombre de ranura indicado no es válido o no se encuentra. |

---

#### Sintaxis

```
RMQFindSlot  
[ Slot ':' ] < variable (VAR) of rmqslot > ','  
[ Name ':' ] < expression (IN) of string > ';' 
```

*Continúa en la página siguiente*

## 1.213 RMQFindSlot - Buscar una identidad de ranura para el nombre de ranura FlexPendant Interface, PC Interface, or Multitasking Continuación

### Información relacionada

| Para obtener más información sobre   | Consulte   |
|--|--|
| Descripción de la funcionalidad de RAPID Message Queue                           | <i>Application manual - Controller software IRC5, sección RAPID Message Queue.</i>                               |
| Enviar datos a la cola de una tarea de RAPID                                     | <a href="#"><i>RMQSendMessage - Enviar un mensaje de datos de RMQ en la página 638</i></a>                       |
| Obtener el primer mensaje de una cola de RAPID Message Queue.                    | <a href="#"><i>RMQGetMessage - Obtener un mensaje de RMQ en la página 626</i></a>                                |
| Enviar datos a la cola de una tarea de RAPID y esperar una respuesta del cliente | <a href="#"><i>RMQSendWait - Enviar un mensaje de datos de RMQ y esperar una respuesta en la página 642</i></a>  |
| Extraer los datos de encabezado de un <code>rmqmessage</code>                    | <a href="#"><i>RMQGetMsgHeader - Obtener información de encabezado de un mensaje de RMQ en la página 632</i></a> |
| Ordenar y habilitar interrupciones para un tipo de dato en concreto              | <a href="#"><i>IRMQMessage - Ordenar interrupciones de RMQ para un tipo de dato en la página 305</i></a>         |
| Extraer los datos de un <code>rmqmessage</code>                                  | <a href="#"><i>RMQGetMsgData - Obtener la parte de datos de un mensaje de RMQ en la página 629</i></a>           |
| Obtener el nombre de ranura de una identidad de ranura especificada.             | <a href="#"><i>RMQGetSlotName - Obtener el nombre de un cliente de RMQ en la página 1456</i></a>                 |
| RMQ Slot   | <a href="#"><i>rmqslot - Número de identidad de un cliente de RMQ en la página 1748</i></a>                      |

## 1 Instrucciones

---

### 1.214 RMQGetMessage - Obtener un mensaje de RMQ *FlexPendant Interface, PC Interface, or Multitasking*

### 1.214 RMQGetMessage - Obtener un mensaje de RMQ

---

#### Utilización

RMQGetMessage (*RAPID Message Queue Get Message*) se usa para capturar el primer mensaje de RMQ de la cola que corresponde a la tarea del programa actual.

---

#### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción RMQGetMessage:  
Consulte también [Más ejemplos en la página 626](#).

#### Ejemplo 1

```
TRAP msghandler
  VAR rmqmessage myrmqmsg;
  RMQGetMessage myrmqmsg;
  ...
ENDTRAP
```

En la rutina TRAP , msghandler el rmqmessage es capturado del RMQ y copiado a la variable myrmqmsg.

---

#### Argumentos

RMQGetMessage Message

Message

Tipo de dato: rmqmessage

Variable para el almacenamiento del mensaje de RMQ.

El tamaño máximo de los datos que pueden recibirse en un rmqmessage es de aproximadamente 3.000 bytes.

---

#### Ejecución de programas

La instrucción RMQGetMessage se usa para obtener el primer mensaje de la cola de la tarea que ejecuta la instrucción. Si hay un mensaje, éste se copia a la variable Message y a continuación se elimina de la cola para dejar espacio para nuevos mensajes. Esta instrucción sólo se admite en el nivel TRAP.

---

#### Más ejemplos

A continuación aparecen más ejemplos de cómo usar la instrucción RMQGetMessage.

#### Ejemplo 1

```
RECORD mydatatype
  int x;
  int y;
ENDRECORD

VAR intnum msgreceive;
VAR mydatatype mydata;
```

*Continúa en la página siguiente*

## 1.214 RMQGetMessage - Obtener un mensaje de RMQ FlexPendant Interface, PC Interface, or Multitasking Continuación

```

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

```

Quando se recibe un nuevo mensaje, la rutina TRAP msghandler se ejecuta y el nuevo mensaje se copia a la variable message (instrucción RMQGetMessage). A continuación se copian los datos de encabezado de RMQ (instrucción RMQGetMsgHeader). Si el mensaje es del tipo de dato esperado y tiene las dimensiones correctas, los datos se copian a la variable mydata (instrucción RMQGetMsgData).

### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema ERRNO cambiará a:

| Nombre         | Causa del error  |
|----------------|--|
| ERR_RMQ_NOMSG  | Ningún mensaje de momento en la cola. Si se ejecuta RMQGetMessage dos veces en una rutina TRAP, puede ocurrir esto. El error también puede generarse si se produce un fallo de alimentación entre la orden de ejecutar TRAP y la ejecución de la instrucción RMQGetMessage. Los mensajes de RMQ se pierden en caso de fallo de alimentación. |
| ERR_RMQ_INVMSG | Mensaje no válido, probablemente enviado desde otro cliente diferente de una tarea de RAPID. Esto puede ocurrir por ejemplo si una aplicación de PC envía un mensaje dañado.   |

Continúa en la página siguiente

# 1 Instrucciones

## 1.214 RMQGetMessage - Obtener un mensaje de RMQ

*FlexPendant Interface, PC Interface, or Multitasking*

*Continuación*

### Limitaciones

RMQGetMessage no se admite en el nivel de ejecución de usuarios (es decir, en las rutinas de servicio) ni en el nivel de ejecución normal.

El tamaño máximo de los datos que pueden recibirse en un `rmqmessage` es de aproximadamente 3.000 bytes.

Se recomienda reutilizar una variable del tipo de dato `rmqmessage` todo lo posible, con el fin de ahorrar memoria de RAPID.

### Sintaxis

```
RMQGetMessage  
  [ Message ':=' ] < variable (VAR) of rmqmessage > ';' 
```

### Información relacionada

| Para obtener más información sobre   | Consulte  |
|--|---|
| Descripción de la funcionalidad de RAPID Message Queue                           | <a href="#">Application manual - Controller software IRC5, sección RAPID Message Queue.</a>               |
| Encontrar el número de identidad de una tarea de RAPID Message Queue             | <a href="#">RMQFindSlot - Buscar una identidad de ranura para el nombre de ranura en la página 624</a>    |
| Enviar datos a la cola de una tarea de RAPID                                     | <a href="#">RMQSendMessage - Enviar un mensaje de datos de RMQ en la página 638</a>                       |
| Enviar datos a la cola de una tarea de RAPID y esperar una respuesta del cliente | <a href="#">RMQSendWait - Enviar un mensaje de datos de RMQ y esperar una respuesta en la página 642</a>  |
| Extraer los datos de encabezado de un <code>rmqmessage</code>                    | <a href="#">RMQGetMsgHeader - Obtener información de encabezado de un mensaje de RMQ en la página 632</a> |
| Extraer los datos de un <code>rmqmessage</code>                                  | <a href="#">RMQGetMsgData - Obtener la parte de datos de un mensaje de RMQ en la página 629</a>           |
| Ordenar y habilitar interrupciones para un tipo de dato en concreto              | <a href="#">IRMQMessage - Ordenar interrupciones de RMQ para un tipo de dato en la página 305</a>         |
| Obtener el nombre de ranura de una identidad de ranura especificada.             | <a href="#">RMQGetSlotName - Obtener el nombre de un cliente de RMQ en la página 1456</a>                 |
| RMQ Message  | <a href="#">rmqmessage - Mensaje de RAPID Message Queue en la página 1746</a>                             |



## 1.215 RMQGetMsgData - Obtener la parte de datos de un mensaje de RMQ *FlexPendant Interface, PC Interface, or Multitasking*

### 1.215 RMQGetMsgData - Obtener la parte de datos de un mensaje de RMQ

#### Utilización

RMQGetMsgData (*RAPID Message Queue Get Message Data*) se utiliza para obtener los datos efectivos contenidos en el mensaje de RMQ.

#### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción RMQGetMsgData:  
Consulte también [RMQGetMsgData - Obtener la parte de datos de un mensaje de RMQ en la página 629](#).

#### Ejemplo 1

```
VAR rmqmessage myrmqmsg;  
VAR num data;  
...  
RMQGetMsgData myrmqmsg, data;  
! Handle data
```

Los datos, del tipo de dato num, se capturan de la variable myrmqmsg y se almacenan en la variable data.

#### Argumentos

RMQGetMsgData Message Data

#### Message

Tipo de dato: rmqmessage

Variable que contiene el mensaje de RMQ recibido.

#### Data

Tipo de dato: anytype

Variable del tipo de dato esperado, utilizado para el almacenamiento de los datos recibidos.

#### Ejecución de programas

La instrucción RMQGetMsgData se usa para obtener los datos efectivos contenidos en el mensaje RMQ, convertirlos a datos binarios, compilar los datos para ver si es posible almacenarlos en la variable especificada en la instrucción y a continuación copiarlos a la variable.

#### Más ejemplos

A continuación aparecen más ejemplos de cómo usar la instrucción RMQGetMsgData.

#### Ejemplo 1

```
RECORD mydatatype  
  int x;  
  int y;  
ENDRECORD  
  
VAR intnum msgrceive;
```

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.215 RMQGetMsgData - Obtener la parte de datos de un mensaje de RMQ

*FlexPendant Interface, PC Interface, or Multitasking*

*Continuación*

```
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
```

Quando se recibe un nuevo mensaje, la rutina TRAP msghandler se ejecuta y el nuevo mensaje se copia a la variable message (instrucción RMQGetMessage). A continuación se copian los datos de encabezado de RMQ (instrucción RMQGetMsgHeader). Si el mensaje es del tipo de dato esperado y tiene las dimensiones correctas, los datos se copian a la variable mydata (instrucción RMQGetMsgData).

### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema ERRNO cambiará a:

| Nombre          | Causa del error   |
|-----------------|---|
| ERR_RMQ_VALUE   | El mensaje recibido y el tipo de dato utilizado en el argumento Data no son del mismo tipo de dato.   |
| ERR_RMQ_DIM     | Los tipos de datos son iguales, pero las dimensiones son diferentes en el dato del mensaje y en la variable utilizada en el argumento Data. |
| ERR_RMQ_MSGSIZE | El tamaño de los datos recibidos es mayor que el tamaño máximo configurado para el RMQ en la tarea receptora.                               |

*Continúa en la página siguiente*

## 1.215 RMQGetMsgData - Obtener la parte de datos de un mensaje de RMQ FlexPendant Interface, PC Interface, or Multitasking Continuación

| Nombre         | Causa del error  |
|----------------|--|
| ERR_RMQ_INVMSG | Este error se genera si el mensaje no es válido. Esto puede ocurrir por ejemplo si una aplicación de PC envía un mensaje dañado. |

### Sintaxis

```
RMQGetMsgData
  [ Message ':= ' ] < variable (VAR) of rmqmessage > ', '
  [ Data ':= ' ] < reference (VAR) of anytype > ';'
```

### Información relacionada

| Para obtener más información sobre   | Consulte  |
|--|---|
| Descripción de la funcionalidad de RAPID Message Queue                           | <i>Application manual - Controller software IRC5, sección RAPID Message Queue.</i>                        |
| Encontrar el número de identidad de una tarea de RAPID Message Queue             | <a href="#">RMQFindSlot - Buscar una identidad de ranura para el nombre de ranura en la página 624</a>    |
| Enviar datos a la cola de una tarea de RAPID                                     | <a href="#">RMQSendMessage - Enviar un mensaje de datos de RMQ en la página 638</a>                       |
| Obtener el primer mensaje de una cola de RAPID Message Queue.                    | <a href="#">RMQGetMessage - Obtener un mensaje de RMQ en la página 626</a>                                |
| Enviar datos a la cola de una tarea de RAPID y esperar una respuesta del cliente | <a href="#">RMQSendWait - Enviar un mensaje de datos de RMQ y esperar una respuesta en la página 642</a>  |
| Extraer los datos de encabezado de un rmqmessage                                 | <a href="#">RMQGetMsgHeader - Obtener información de encabezado de un mensaje de RMQ en la página 632</a> |
| Ordenar y habilitar interrupciones para un tipo de dato en concreto              | <a href="#">IRMQMessage - Ordenar interrupciones de RMQ para un tipo de dato en la página 305</a>         |
| Obtener el nombre de ranura de una identidad de ranura especificada.             | <a href="#">RMQGetSlotName - Obtener el nombre de un cliente de RMQ en la página 1456</a>                 |
| RMQ Message  | <a href="#">rmqmessage - Mensaje de RAPID Message Queue en la página 1746</a>                             |

## 1 Instrucciones

---

### 1.216 RMQGetMsgHeader - Obtener información de encabezado de un mensaje de RMQ *FlexPendant Interface, PC Interface, or Multitasking*

### 1.216 RMQGetMsgHeader - Obtener información de encabezado de un mensaje de RMQ

---

#### Utilización

`RMQGetMsgHeader` (*RAPID Message Queue Get Message Header*) obtiene la información de encabezado contenida en el mensaje de RMQ recibida y lo almacena en variables de los tipos `rmqheader`, `rmqslot` o `num`.

---

#### Ejemplos básicos

Los ejemplos siguientes ilustran la instrucción `RMQGetMsgHeader`.

Consulte también [Más ejemplos en la página 633](#).

#### Ejemplo 1

```
VAR rmqmessage myrmqmsg;  
VAR rmqheader myrmqheader;  
...  
RMQGetMsgHeader myrmqmsg, \Header:=myrmqheader;
```

En este ejemplo, la variable `myrmqheader` recibe los datos copiados de la parte `rmqheader` de la variable `myrmqmsg`.

#### Ejemplo 2

```
VAR rmqmessage rmqmessage1;  
VAR rmqheader rmqheader1;  
VAR rmqslot rmqslot1;  
VAR num userdef := 0;  
...  
RRMQGetMsgHeader rmqmessage1 \Header:=rmqheader1 \SenderId:=rmqslot1  
  \UserDef:=userdef;
```

En este ejemplo, las variables `rmqheader1`, `rmqslot1` y `userdef` reciben los datos copiados de la variable `rmqmessage1`.

---

#### Argumentos

```
RMQGetMsgHeader Message [\Header] [\SenderId] [\UserDef]
```

Message

Tipo de dato: `rmqmessage`

Variable que contiene el mensaje de RMQ recibido y desde la que se copia la información acerca del mensaje.

[\Header]

Tipo de dato: `rmqheader`

Variable para el almacenamiento de la información de encabezado de RMQ copiada desde la variable especificada con el parámetro `Message`.

[\SenderId]

Tipo de dato: `rmqslot`

Variable para el almacenamiento de la información de identidad de remitente copiada desde la variable especificada con el parámetro `Message`.

---

*Continúa en la página siguiente*

## 1.216 RMQGetMsgHeader - Obtener información de encabezado de un mensaje de RMQ *FlexPendant Interface, PC Interface, or Multitasking* Continuación

[\UserDef]

**User Defined data**

Tipo de dato: num

Variable para el almacenamiento de datos definidos por el usuario copiada desde la variable especificada con el parámetro `Message`. Para obtener cualquier dato válido en esta variable, el remitente necesita especificar que ésta debe incluirse al enviar un mensaje de RMQ . Si no se utiliza, el valor será cambiado a -1.

**Ejecución de programas**

La instrucción `RMQGetMsgHeader` obtiene la información de encabezado contenida en el mensaje de RMQ y la copia a variables de tipo `rmqheader`, `rmqslot` o `num`, en función de qué argumentos se utilicen.

**Más ejemplos**

A continuación aparecen más ejemplos de cómo usar la instrucción `RMQGetMsgHeader`.

**Ejemplo 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;
  
```

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.216 RMQGetMsgHeader - Obtener información de encabezado de un mensaje de RMQ

*FlexPendant Interface, PC Interface, or Multitasking*

Continuación

```
ELSE
    TPWrite "Received a type not handled or with wrong dimension";
ENDIF
ENDTRAP
```

Quando se recibe un nuevo mensaje, la rutina TRAP `msghandler` se ejecuta y el nuevo mensaje se copia a la variable `message` (instrucción `RMQGetMessage`). A continuación se copian los datos de encabezado de RMQ (instrucción `RMQGetMsgHeader`). Si el mensaje es del tipo de dato esperado y tiene las dimensiones correctas, los datos se copian a la variable `mydata` (instrucción `RMQGetMsgData`).

### Sintaxis

```
RMQGetMsgHeader
[ Message ':' = ] < variable (VAR) of rmqmessage > ','
[ '\ ' Header ':' = < variable (VAR) of rmqheader >
[ '\ ' SenderId ':' = < variable (VAR) of rmqslot >
[ '\ ' UserDef ':' = < variable (VAR) of num > ';'

```

### Información relacionada

| Para obtener más información sobre   | Consulte   |
|--|--|
| Descripción de la funcionalidad de RAPID Message Queue                           | <i>Application manual - Controller software IRC5, sección RAPID Message Queue.</i>                       |
| Encontrar el número de identidad de una tarea de RAPID Message Queue             | <a href="#">RMQFindSlot - Buscar una identidad de ranura para el nombre de ranura en la página 624</a>   |
| Enviar datos a la cola de una tarea de RAPID                                     | <a href="#">RMQSendMessage - Enviar un mensaje de datos de RMQ en la página 638</a>                      |
| Obtener el primer mensaje de una cola de RAPID Message Queue.                    | <a href="#">RMQGetMessage - Obtener un mensaje de RMQ en la página 626</a>                               |
| Enviar datos a la cola de una tarea de RAPID y esperar una respuesta del cliente | <a href="#">RMQSendWait - Enviar un mensaje de datos de RMQ y esperar una respuesta en la página 642</a> |
| Extraer los datos de un <code>rmqmessage</code>                                  | <a href="#">RMQGetMsgData - Obtener la parte de datos de un mensaje de RMQ en la página 629</a>          |
| Ordenar y habilitar interrupciones para un tipo de dato en concreto              | <a href="#">IRMQMessage - Ordenar interrupciones de RMQ para un tipo de dato en la página 305</a>        |
| Obtener el nombre de ranura de una identidad de ranura especificada.             | <a href="#">RMQGetSlotName - Obtener el nombre de un cliente de RMQ en la página 1456</a>                |
| RMQ Slot   | <a href="#">rmqslot - Número de identidad de un cliente de RMQ en la página 1748</a>                     |
| RMQ Header   | <a href="#">rmqmessage - Mensaje de RAPID Message Queue en la página 1746</a>                            |
| RMQ Message  | <a href="#">rmqheader - Encabezado de mensaje de RAPID Message Queue en la página 1744</a>               |

## 1.217 RMQReadWait - Devuelve un mensaje de una cola RMQ *FlexPendant Interface, PC Interface, or Multitasking*

### 1.217 RMQReadWait - Devuelve un mensaje de una cola RMQ

#### Utilización

`RMQReadWait` se utiliza en el modo sincronizado para recibir cualquier tipo de mensaje.

#### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción `RMQReadWait`:  
Consulte también [Más ejemplos en la página 635](#).

#### Ejemplo

```
VAR rmqmessage myrmqmsg;
RMQReadWait myrmqmsg;
```

El primer mensaje de la cola se recibe en la variable `myrmqmsg`.

#### Argumentos

```
RMQReadWait Message [\Timeout]
```

Message

**Tipo de dato:** `rmqmessage`

La variable en la que se coloca el mensaje recibido.

[\Timeout]

**Tipo de dato:** `num`

El tiempo máximo [s] que la ejecución del programa debe esperar un mensaje. Si el tiempo se agota antes de que se cumpla la condición, se llama al gestor de errores si lo hay, con el código de error `ERR_RMQ_TIMEOUT`. Si no hay ningún gestor de errores, se detiene la ejecución. Es posible cambiar el tiempo límite a 0 (cero) segundos, de forma que no se produzca ninguna espera.

Si no se usa el parámetro `\Timeout` el tiempo de espera es de 60 s. Para esperar indefinidamente, utilice la constante predefinida `WAIT_MAX`.

#### Ejecución de programas

Todos los mensajes entrantes se ponen en la cola y `RMQReadWait` los gestiona en orden FIFO, y de uno en uno. El usuario es responsable de evitar que la cola se llene y de estar preparado para gestionar cualquier tipo de mensaje admitido por la cola de mensajes de RAPID.

#### Más ejemplos

A continuación aparecen más ejemplos de cómo usar la instrucción `RMQReadWait`.

#### Ejemplo 1

```
VAR rmqmessage myrmqmsg;
RMQReadWait myrmqmsg \Timeout:=30;
```

El primer mensaje de la cola se recibe en la variable `myrmqmsg`. Si no se recibe ningún mensaje en un intervalo de 30 segundos, se detiene la ejecución del programa.

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.217 RMQReadWait - Devuelve un mensaje de una cola RMQ

*FlexPendant Interface, PC Interface, or Multitasking*

Continuación

### Ejemplo 2

```
PROC main()  
  VAR rmqmessage myrmqmsg;  
  FOR i FROM 1 TO 25 DO  
    RMQReadWait myrmqmsg \TimeOut:=30;  
    ...  
  ENDFOR  
  
  ERROR  
  IF ERRNO = ERR_RMQ_TIMEOUT THEN  
    TPWrite "ERR_RMQ_TIMEOUT error reported";  
    ...  
  ENDIF  
ENDPROC
```

Se reciben los mensajes de la cola, que se almacenan en la variable `myrmqmsg`. Si la recepción de un mensaje requiere más de 30 segundos, se llama al gestor de errores.

### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                       | Causa del error  |
|------------------------------|--|
| <code>ERR_RMQ_TIMEOUT</code> | No se ha recibido ninguna respuesta dentro del tiempo límite.  |
| <code>ERR_RMQ_INVMSG</code>  | Este error se genera si el mensaje no es válido. Esto puede ocurrir por ejemplo si una aplicación de PC envía un mensaje dañado. |

### Limitaciones

`RMQReadWait` sólo se admite en el modo sincronizado. La ejecución de esta instrucción en el modo basado en interrupciones generará un error de tiempo de ejecución no recuperable.

`RMQReadWait` no se admite en el nivel de ejecución de rutina `TRAP` ni en el nivel de ejecución del usuario. La ejecución de esta instrucción en cualquiera de estos niveles generará un error de tiempo de ejecución no recuperable.

### Sintaxis

```
RMQReadWait  
  [ Message ':=' ] < variable (VAR) of rmqmessage >  
  [ '\ TimeOut':=' < expression (IN) of num > ] ';' ;
```

### Información relacionada

| Para obtener más información sobre                     | Consulte  |
|--|---|
| Descripción de la funcionalidad de RAPID Message Queue | <i>Application manual - Controller software IRC5</i> , sección <i>RAPID Message Queue</i> . |
| Descripción de los modos de ejecución de tareas        | <i>Manual de referencia técnica - Parámetros del sistema</i>                                |

Continúa en la página siguiente



## 1.217 RMQReadWait - Devuelve un mensaje de una cola RMQ FlexPendant Interface, PC Interface, or Multitasking Continuación

| Para obtener más información sobre   | Consulte   |
|--|--|
| Tipo de dato <code>rmqmessage</code>   | <a href="#">rmqmessage - Mensaje de RAPID Message Queue en la página 1746.</a>                             |
| Enviar datos a la cola de una tarea de RAPID                                     | <a href="#">RMQSendMessage - Enviar un mensaje de datos de RMQ en la página 638.</a>                       |
| Enviar datos a la cola de una tarea de RAPID y esperar una respuesta del cliente | <a href="#">RMQSendWait - Enviar un mensaje de datos de RMQ y esperar una respuesta en la página 642.</a>  |
| Encontrar el número de identidad de una tarea de RAPID Message Queue             | <a href="#">RMQFindSlot - Buscar una identidad de ranura para el nombre de ranura en la página 624.</a>    |
| Extraer los datos de encabezado de un <code>rmqmessage</code>                    | <a href="#">RMQGetMsgHeader - Obtener información de encabezado de un mensaje de RMQ en la página 632.</a> |
| Extraer los datos de un <code>rmqmessage</code>                                  | <a href="#">RMQGetMsgData - Obtener la parte de datos de un mensaje de RMQ en la página 629.</a>           |
| Ordenar y habilitar interrupciones para un tipo de dato en concreto              | <a href="#">IRMQMessage - Ordenar interrupciones de RMQ para un tipo de dato en la página 305.</a>         |
| Obtener el nombre de ranura de una identidad de ranura especificada.             | <a href="#">RMQGetSlotName - Obtener el nombre de un cliente de RMQ en la página 1456.</a>                 |
| Vaciar la cola de mensajes de RAPID  | <a href="#">RMQEmptyQueue - Vacía la cola de mensajes de RAPID en la página 622.</a>                       |
| Obtener el primer mensaje de una cola de RAPID Message Queue                     | <a href="#">RMQGetMessage - Obtener un mensaje de RMQ en la página 626.</a>                                |

## 1 Instrucciones

---

### 1.218 RMQSendMessage - Enviar un mensaje de datos de RMQ

*FlexPendant Interface, PC Interface, or Multitasking*

### 1.218 RMQSendMessage - Enviar un mensaje de datos de RMQ

---

#### Utilización

RMQSendMessage (*RAPID Message Queue Send Message*) se utiliza para enviar datos a un RMQ configurado para una tarea de RAPID o un cliente de Robot Application Builder.

---

#### Ejemplos básicos

Los ejemplos siguientes ilustran la instrucción RMQSendMessage.

Consulte también [Más ejemplos en la página 639](#).

#### Ejemplo 1

```
VAR rmqslot destination_slot;  
VAR string data:="Hello world";  
..  
RMQFindSlot destination_slot,"RMQ_Task2";  
RMQSendMessage destination_slot,data;
```

Este ejemplo muestra cómo enviar el valor de la variable `data` a la tarea de RAPID "Task2" con el RMQ configurado "RMQ\_Task2".

#### Ejemplo 2

```
VAR rmqslot destination_slot;  
CONST robtarg 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;
```

Este ejemplo muestra cómo enviar el valor de la constante `p5` a la tarea de RAPID "Task2" con el RMQ configurado "RMQ\_Task2". También se envía un número definido por el usuario. Este número puede ser utilizado por el destinatario como identificador.

---

#### Argumentos

```
RMQSendMessage Slot SendData [\UserDef]
```

Slot

**Tipo de dato:** `rmqslot`

El número de ranura de identidad del cliente que debe recibir el mensaje.

SendData

**Tipo de dato:** `anytype`

Referencia a una variable, variable persistente o constante que contiene los datos a enviar al cliente que tiene la identidad indicada en el argumento `Slot`.

[\UserDef]

*User Defined data*

**Tipo de dato:** `num`

*Continúa en la página siguiente*

---

## 1.218 RMQSendMessage - Enviar un mensaje de datos de RMQ *FlexPendant Interface, PC Interface, or Multitasking* Continuación

Datos que especifican la información definida por el usuario al receptor de SendData, es decir, el cliente cuyo número de identidad es el indicado en la variable Slot. El valor debe ser un entero de 0 a 32.767.

### Ejecución de programas

La instrucción `RMQSendMessage` se usa para enviar datos a un cliente especificado. La instrucción empaqueta los datos entrantes de un contenedor de almacenamiento y los envía.

Si el cliente destinatario no está interesado en recibir mensajes, es decir, no ha configurado ninguna interrupción que deba ejecutarse para el tipo de dato especificado en la instrucción `RMQSendMessage`, o bien no está esperando en una instrucción `RMQSendWait`, el mensaje se desecha y se genera un aviso.

No todos los tipos de datos pueden enviarse con esta instrucción (consulte las limitaciones).

### Más ejemplos

A continuación aparecen más ejemplos de cómo usar la instrucción `RMQSendMessage`.

#### Ejemplo 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;
```

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.218 RMQSendMessage - Enviar un mensaje de datos de RMQ

*FlexPendant Interface, PC Interface, or Multitasking*

*Continuación*

```
        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  
ENDMODULE
```

Este ejemplo muestra cómo usar la instrucción `RMQSendMessage` con la gestión de errores aplicada a los errores de tiempo de ejecución que se produzcan. El programa envía los datos definidos por el usuario del tipo `msgrec` a un cliente de Robot Application Builder llamado "My\_RAB\_client".

### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                       | Causa del error  |
|------------------------------|--|
| <code>ERR_RMQ_MSGSIZE</code> | El tamaño del mensaje es excesivo. O bien los datos sobrepasan el tamaño de mensaje máximo permitido o el cliente destinatario no está configurado para recibir el tamaño de datos enviado.  |
| <code>ERR_RMQ_FULL</code>    | La cola de mensajes de destino está llena.   |
| <code>ERR_RMQ_INVALID</code> | La ranura de destino no ha sido conectada o la ranura de destino ya no está disponible. Si no hay conexión, debe realizarse una llamada a <code>RMQFindSlot</code> . Si no está disponible, el motivo es que un cliente remoto se ha desconectado del controlador. |

### Limitaciones

No es posible usar en interrupciones, enviar ni recibir instancias de tipos de datos que no tienen valor, son de semivalor o corresponden al tipo de dato `motsetdata`. El tamaño máximo de los datos que pueden enviarse al cliente de Robot Application Builder es de aproximadamente 5.000 bytes. El tamaño máximo de los datos que pueden ser recibidos por un RMQ y almacenados en un tipo de dato `rmqmessage` es de aproximadamente 3.000 bytes. El tamaño de los datos que pueden ser recibidos por un RMQ es configurable (tamaño predeterminado 400, tamaño máximo 3.000).

### Sintaxis

```
RMQSendMessage  
[ Slot ':' = ' ] < variable (VAR) of rmqslot > ', '  
[ SendData ':' = ' ] < reference (REF) of anytype >  
[ '\ ' UserDef ':' = ' < expression (IN) of num > ] ';' ;
```

*Continúa en la página siguiente*

## 1.218 RMQSendMessage - Enviar un mensaje de datos de RMQ FlexPendant Interface, PC Interface, or Multitasking Continuación

### Información relacionada

| Para obtener más información sobre   | Consulte  |
|--|---|
| Descripción de la funcionalidad de RAPID Message Queue                           | <i>Application manual - Controller software IRC5, sección RAPID Message Queue.</i>                        |
| Encontrar el número de identidad de una tarea de RAPID Message Queue             | <a href="#">RMQFindSlot - Buscar una identidad de ranura para el nombre de ranura en la página 624</a>    |
| Obtener el primer mensaje de una cola de RAPID Message Queue.                    | <a href="#">RMQGetMessage - Obtener un mensaje de RMQ en la página 626</a>                                |
| Enviar datos a la cola de una tarea de RAPID y esperar una respuesta del cliente | <a href="#">RMQSendWait - Enviar un mensaje de datos de RMQ y esperar una respuesta en la página 642</a>  |
| Extraer los datos de encabezado de un <code>rmqmessage</code>                    | <a href="#">RMQGetMsgHeader - Obtener información de encabezado de un mensaje de RMQ en la página 632</a> |
| Extraer los datos de un <code>rmqmessage</code>                                  | <a href="#">RMQGetMsgData - Obtener la parte de datos de un mensaje de RMQ en la página 629</a>           |
| Ordenar y habilitar interrupciones para un tipo de dato en concreto              | <a href="#">IRMQMessage - Ordenar interrupciones de RMQ para un tipo de dato en la página 305</a>         |
| Obtener el nombre de ranura de una identidad de ranura especificada.             | <a href="#">RMQGetSlotName - Obtener el nombre de un cliente de RMQ en la página 1456</a>                 |
| RMQ Slot   | <a href="#">rmqslot - Número de identidad de un cliente de RMQ en la página 1748</a>                      |

## 1 Instrucciones

---

### 1.219 RMQSendWait - Enviar un mensaje de datos de RMQ y esperar una respuesta *FlexPendant Interface, PC Interface, or Multitasking*

### 1.219 RMQSendWait - Enviar un mensaje de datos de RMQ y esperar una respuesta

---

#### Utilización

La instrucción `RMQSendWait` (*RAPID Message Queue Send Wait*) permite enviar datos a un RMQ o a un cliente de Robot Application Builder y esperar una respuesta del cliente especificado. Si se utiliza esta instrucción, el usuario necesita saber qué tipo de dato se enviará en la respuesta proveniente del cliente.

---

#### Ejemplos básicos

Los ejemplos siguientes ilustran la instrucción `RMQSendWait`.

Consulte también [Más ejemplos en la página 644](#).

#### Ejemplo 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;
```

Este ejemplo muestra cómo enviar los datos de la variable `sendstr` a la tarea de RAPID "T\_ROB2" con el RMQ configurado "RMQ\_T\_ROB2". Ahora la instrucción `RMQSendWait` espera una respuesta de la tarea "T\_ROB2". La instrucción "T\_ROB2" necesita enviar datos almacenados en un tipo de dato `num` para dar fin a la instrucción de espera `RMQSendWait`. Una vez recibido el mensaje, los datos se copian a la variable `mynum` desde la variable `receivemsg` con la instrucción `RMQGetMsgData`

#### Ejemplo 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;
```

Este ejemplo muestra cómo enviar los datos de la variable `mysendstr` a la tarea de RAPID "Task2" con el RMQ configurado "RMQ\_Task2". También se envía un número definido por el usuario. Este número puede ser utilizado por el destinatario como un identificador y debe ser devuelto al remitente para dar fin a la instrucción de espera `RMQSendWait`. Otra exigencia para dar fin a la instrucción de espera es que el cliente envíe el tipo de dato correcto. El tipo de dato se especifica en la

*Continúa en la página siguiente*

## 1.219 RMQSendWait - Enviar un mensaje de datos de RMQ y esperar una respuesta *FlexPendant Interface, PC Interface, or Multitasking* Continuación

variable `receivestr` de la instrucción `RMQSendWait`. Una vez recibido el mensaje, los datos efectivos se copian a la variable `receivestr` con la instrucción `RMQGetMsgData`

### Argumentos

```
RMQSendWait Slot SendData [\UserDef] Message ReceiveDataType
[\Timeout]
```

Slot

Tipo de dato: `rmqslot`

El número de identidad del cliente que debe recibir el mensaje.

SendData

Tipo de dato: `anytype`

Referencia a una variable, variable persistente o constante que contiene los datos a enviar al cliente cuyo número de identidad es el indicado en la variable `Slot`.

[\UserDef]

*User Defined data*

Tipo de dato: `num`

Datos que especifican la información definida por el usuario al receptor de `SendData`, es decir, el cliente cuyo número de identidad es el indicado en la variable `Slot`. Si se utiliza este argumento opcional, la instrucción `RMQSendWait` sólo finaliza si los tipos de `ReceiveDataType` y de `UserDef` especificados son iguales a los especificados en la respuesta al mensaje. El valor debe ser un entero de 0 a 32.767.

Message

Tipo de dato: `rmqmessage`

La variable en la que se coloca el mensaje recibido.

ReceiveDataType

Tipo de dato: `anytype`

Una referencia a una variable persistente, una variable o una constante del tipo de dato esperado por la instrucción. Los datos efectivos no se copian a esta variable cuando se ejecuta `RMQSendWait`. Este argumento sólo se usa para especificar el tipo de dato efectivo esperado por la instrucción `RMQSendWait`.

[\Timeout]

Tipo de dato: `num`

El tiempo máximo [s] que debe esperar la ejecución del programa a una respuesta. Si el tiempo se agota antes de que se cumpla la condición, se llama al gestor de errores si lo hay, con el código de error `ERR_RMQ_TIMEOUT`. Si no hay ningún gestor de errores, se detiene la ejecución.

Si no se usa el parámetro `\Timeout` el tiempo de espera es de 60 s. Para esperar ininterrumpidamente, utilice la constante predefinida `WAIT_MAX`.

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.219 RMQSendWait - Enviar un mensaje de datos de RMQ y esperar una respuesta

*FlexPendant Interface, PC Interface, or Multitasking*

*Continuación*

---

### Ejecución de programas

La instrucción `RMQSendWait` envía datos y espera una respuesta del cliente que tiene la identidad de ranura especificada. La respuesta debe ser un `rmqmessage` del cliente que recibió el mensaje, y la respuesta debe ser del mismo tipo de dato especificado en el argumento `ReceiveDataType`. El mensaje se envía de la misma forma que cuando se usa `RMQSendMessage`, es decir, el destinatario recibirá un mensaje normal de RAPID Message Queue. El remitente es responsable de que el destinatario sepa que se necesita una respuesta. Si se usa el argumento opcional `UserDef` en la instrucción `RMQSendWait`, se exige que el cliente destinatario utilice el mismo `UserDef` en la respuesta

Si el cliente destinatario no está interesado en recibir mensajes, es decir, no ha configurado ninguna interrupción que deba ejecutarse para el tipo de dato especificado en la instrucción `RMQSendWait`, el mensaje se desecha y se genera un aviso. La instrucción devuelve un error tras el tiempo utilizado en el argumento `Timeout`, o tras el tiempo límite predeterminado de 60 s. Este error puede ser gestionado en un gestor de errores.

La instrucción `RMQSendWait` tiene la máxima prioridad si se recibe un mensaje y éste se corresponde con la descripción tanto de la respuesta esperada como de un mensaje conectado a una rutina TRAP (consulte la instrucción `IRMQMessage`).

Si se produce una interrupción del servicio eléctrico mientras se espera una respuesta del cliente, la variable utilizada en el argumento `Slot` cambia a 0 y la instrucción se ejecuta de nuevo. En este caso, la instrucción fallará debido a una identidad de ranura no válida y se llamará al gestor de errores si lo hay, con el código de error `ERR_RMQ_INVALID`. La identidad de ranura puede ser reiniciada desde ahí.

No todos los tipos de datos pueden enviarse con esta instrucción (consulte las limitaciones).

---

### Más ejemplos

A continuación aparecen más ejemplos de cómo usar la instrucción `RMQSendWait`.

#### Ejemplo 1

```
MODULE RMQ_Task1_mod
  PROC main()
    VAR rmqslot destination_slot;
    VAR string mysenstr:="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 mysenstr, wait for an answer of type string
      RMQSendWait destination_slot, mysenstr, recmsg, myrecstr;
```

*Continúa en la página siguiente*



## 1.219 RMQSendWait - Enviar un mensaje de datos de RMQ y esperar una respuesta *FlexPendant Interface, PC Interface, or Multitasking* Continuación

```

!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

```

Los datos de la variable `mysendstr` se envían a la tarea de RAPID "Task2" con la cola de RAPID Message Queue configurada "RMQ\_Task2" y la instrucción `RMQSendWait`. La respuesta de la tarea de RAPID "Task2" debe ser una cadena (especificada con el tipo de dato de la variable `myrecstr`). El mensaje de RMQ recibido como respuesta se recibe en la variable `recmsg`. El uso de la variable `myrecstr` en la llamada a `RMQSendWait` es sólo una especificación del tipo de dato esperado por el remitente como respuesta. No se coloca ningún dato válido en la variable de la llamada a `RMQSendWait`

### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                       | Causa del error  |
|------------------------------|--|
| <code>ERR_RMQ_MSGSIZE</code> | El tamaño del mensaje es excesivo. O bien los datos sobrepasan el tamaño de mensaje máximo permitido o el cliente destinatario no está configurado para recibir el tamaño de datos enviado.  |
| <code>ERR_RMQ_FULLL</code>   | La cola de mensajes de destino está llena.   |
| <code>ERR_RMQ_INVALID</code> | La ranura <code>rmqslot</code> no ha sido inicializada o la ranura de destino ya no está disponible. Esto puede ocurrir si la ranura de destino es un cliente remoto y éste ha sido desconectado del controlador. <code>RMQSendWait</code> fue interrumpida por una caída de alimentación y en el momento del reinicio la ranura <code>rmqslot</code> es puesta a 0. |
| <code>ERR_RMQ_TIMEOUT</code> | No se ha recibido ninguna respuesta dentro del tiempo límite.  |
| <code>ERR_RMQ_INVMSG</code>  | Este error se genera si el mensaje no es válido. Esto puede ocurrir por ejemplo si una aplicación de PC envía un mensaje dañado.   |

### Limitaciones

No se permite ejecutar `RMQSendWait` en el modo sincronizado. Provocaría un error de tiempo de ejecución no recuperable.

No es posible usar en interrupciones, enviar ni recibir instancias de tipos de datos que no tienen valor, son de semivalor o corresponden al tipo de dato `motsetdata`.

El tamaño máximo de los datos que pueden enviarse al cliente de Robot Application Builder es de aproximadamente 5.000 bytes. El tamaño máximo de los datos que

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.219 RMQSendWait - Enviar un mensaje de datos de RMQ y esperar una respuesta

*FlexPendant Interface, PC Interface, or Multitasking*

*Continuación*

pueden ser recibidos por un RMQ y almacenados en un tipo de dato `rmqmessage` es de aproximadamente 3.000 bytes. El tamaño de los datos que pueden ser recibidos por un RMQ es configurable (tamaño predeterminado 400, tamaño máximo 3.000).

### Sintaxis

```
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 > ] ';'
```

### Información relacionada

| Para obtener más información sobre                                   | Consulte  |
|--|---|
| Descripción de la funcionalidad de RAPID Message Queue               | <i>Application manual - Controller software IRC5, sección RAPID Message Queue.</i>                        |
| Encontrar el número de identidad de una tarea de RAPID Message Queue | <a href="#">RMQFindSlot - Buscar una identidad de ranura para el nombre de ranura en la página 624</a>    |
| Enviar datos a la cola de una tarea de RAPID                         | <a href="#">RMQSendMessage - Enviar un mensaje de datos de RMQ en la página 638</a>                       |
| Obtener el primer mensaje de una cola de RAPID Message Queue.        | <a href="#">RMQGetMessage - Obtener un mensaje de RMQ en la página 626</a>                                |
| Extraer los datos de encabezado de un <code>rmqmessage</code>        | <a href="#">RMQGetMsgHeader - Obtener información de encabezado de un mensaje de RMQ en la página 632</a> |
| Extraer los datos de un <code>rmqmessage</code>                      | <a href="#">RMQGetMsgData - Obtener la parte de datos de un mensaje de RMQ en la página 629</a>           |
| Ordenar y habilitar interrupciones para un tipo de dato en concreto  | <a href="#">IRMQMessage - Ordenar interrupciones de RMQ para un tipo de dato en la página 305</a>         |
| Obtener el nombre de ranura de una identidad de ranura especificada. | <a href="#">RMQGetSlotName - Obtener el nombre de un cliente de RMQ en la página 1456</a>                 |
| RMQ Slot   | <a href="#">rmqslot - Número de identidad de un cliente de RMQ en la página 1748</a>                      |
| RMQ Message  | <a href="#">rmqmessage - Mensaje de RAPID Message Queue en la página 1746</a>                             |

## 1.220 SafetyControllerSyncRequest - Inicio del procedimiento de sincronización del hardware *SafeMove Basic, SafeMove Pro, PROFIsafe*

### 1.220 SafetyControllerSyncRequest - Inicio del procedimiento de sincronización del hardware

#### Utilización

`SafetyControllerSyncRequest` se utiliza para iniciar el procedimiento de sincronización del hardware.

#### Ejemplos básicos

El ejemplo siguiente ilustra la instrucción `SafetyControllerSyncRequest`.

#### Ejemplo 1

```
SafetyControllerSyncRequest ;
Iniciar el procedimiento de sincronización del hardware.
```

#### Ejecución de programas

Se debe llamar a esta instrucción antes de la activación de la señal de sincronización.

#### Sintaxis

```
SafetyControllerSyncRequest ';' ;
```

#### Información relacionada

| Para obtener más información sobre           | Consulte   |
|--|--|
| <code>SafetyControllerGetChecksum</code>     | <a href="#">SafetyControllerGetChecksum - Obtener la suma de comprobación del archivo de configuración de usuarios en la página 1467</a> |
| <code>SafetyControllerGetSWVersion</code>    | <a href="#">SafetyControllerGetSWVersion - Obtener la versión del firmware del Safety Controller en la página 1469</a>                   |
| <code>SafetyControllerGetUserChecksum</code> | <a href="#">SafetyControllerGetUserChecksum - Obtener la suma de comprobación de los parámetros protegidos en la página 1470</a>         |

# 1 Instrucciones

---

1.221 Save - Guarda un módulo de programa  
RobotWare - OS

## 1.221 Save - Guarda un módulo de programa

---

### Utilización

Save se utiliza para guardar un módulo de programa.

El módulo de programa especificado y presente en la memoria de programa se guarda en la ruta de archivos original (especificada en Load o StartLoad) o en la ruta especificada.

También es posible guardar un módulo de sistema en la ruta de archivos especificada.

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción Save:

Consulte también [Más ejemplos en la página 649](#).

#### Ejemplo 1

```
Load "HOME:/PART_B.MOD" ;  
...  
Save "PART_B";
```

Carga en la memoria de programa el módulo de programa denominado PART\_B.MOD desde HOME:.

Guarda el módulo de programa PART\_B con la ruta de archivo original HOME: y el nombre de archivo original PART\_B.MOD..

### Argumentos

```
Save [\TaskRef][\TaskName] ModuleName [\FilePath] [\File]
```

[\TaskRef]

#### *Task Reference*

Tipo de dato: taskid

La identidad de la tarea de programa en la que debe guardarse el módulo de programa.

Existen variables predefinidas con el tipo de dato taskid para todas las tareas de programa del sistema. La identificación de la variable será "nombre\_tarea"+"ID". Por ejemplo, para la tarea T\_ROB1 la identificación de la tarea es T\_ROB1Id.

[\TaskName]

Tipo de dato: string

El nombre de la tarea de programa en la que debe guardarse el módulo de programa.

Si no se especifica ninguno de los argumentos \TaskRef o \TaskName, se guarda el módulo de programa de la tarea de programa actual (la que se esté ejecutando).

ModuleName

Tipo de dato: string

El módulo de programa que se desea guardar.

*Continúa en la página siguiente*

`[\FilePath]`

Tipo de dato: `string`

La ruta y el nombre de archivo del lugar en el que se desea guardar el módulo de programa. El nombre de archivo se excluye cuando se utiliza el argumento `\File`.

`[\File]`

Tipo de dato: `string`

Cuando se excluye el nombre del archivo en el argumento `\FilePath`, es necesario especificarlo con este argumento.

El argumento `\FilePath\File` sólo puede omitirse en el caso de los módulos de programa cargados con `Load` o `StartLoad-WaitLoad`. Además, el módulo de programa se almacena en el mismo destino que el especificado en estas instrucciones. Para almacenar el módulo de programa en otro destino, también es posible usar el argumento `\FilePath \File`.

Para poder guardar un módulo de programa cargado anteriormente en el FlexPendant, un ordenador externo o en la configuración del sistema, es necesario utilizar el argumento `\FilePath \File`.

---

### Ejecución de programas

La ejecución del programa espera a que el módulo de programa termine de guardarse antes de continuar en la instrucción siguiente.

---

### Más ejemplos

A continuación aparecen más ejemplos de cómo usar la instrucción `Save`.

#### Ejemplo 1

```
Save "PART_A" \FilePath:="HOME:/DOORDIR/PART_A.MOD";
```

Guarda el módulo de programa `PART_A` en `HOME`: en el archivo `PART_A.MOD` y en el directorio `DOORDIR`.

#### Ejemplo 2

```
Save "PART_A" \FilePath:="HOME:" \File:="DOORDIR/PART_A.MOD";
```

Lo mismo que en el ejemplo 1 anterior pero con otra sintaxis.

#### Ejemplo 3

```
Save \TaskRef:=TSK1Id, "PART_A"  
  \FilePath:="HOME:/DOORDIR/PART_A.MOD";
```

Guarda el módulo de programa `PART_A` de la tarea de programa `TSK1` en el destino especificado. Éste es un ejemplo en el que la instrucción `Save` se ejecuta en una tarea de programa y el guardado se hace en otra tarea de programa.

#### Ejemplo 4

```
Save \TaskName:="TSK1", "PART_A"  
  \FilePath:="HOME:/DOORDIR/PART_A.MOD";
```

Guarda el módulo de programa `PART_A` de la tarea de programa `TSK1` en el destino especificado. Éste es otro ejemplo en el que la instrucción `Save` se ejecuta en una tarea de programa y el guardado se hace en otra tarea de programa.

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.221 Save - Guarda un módulo de programa

RobotWare - OS

Continuación

### Limitaciones

Las rutinas TRAP, los eventos de E/S del sistema y otras tareas de programa no pueden ejecutarse durante la operación de guardado. Por tanto, este tipo de operaciones se retardan.

La operación de guardado puede interrumpir la actualización paso a paso de los datos de tipo PERS desde otras tareas de programa. Esto da como resultado datos de tipo PERS completos incoherentes.

Un paro de programa durante la ejecución de la instrucción Save puede dar lugar a un paro protegido con Motors OFF. El mensaje de error "20025 Tiempo excesivo Orden paro" se muestra en el FlexPendant.

Evite tener movimientos en curso durante el guardado.

### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                    | Causa del error  |
|---------------------------|--|
| <code>ERR_IOERROR</code>  | El archivo guardado no puede abrirse por no disponer de los permisos suficientes, si no existe el directorio o no queda espacio libre en el dispositivo. |
| <code>ERR_MODULE</code>   | El módulo del programa no puede guardarse porque no existe ningún nombre de módulo, nombre de módulo desconocido o nombre de módulo ambiguo.             |
| <code>ERR_PATH</code>     | El argumento <code>\FilePath</code> no está especificado para módulos de programa cargados desde FlexPendant, Parámetros del sistema o un PC externo.    |
| <code>ERR_TASKNAME</code> | El nombre de tarea de programa en el argumento <code>\TaskName</code> no puede encontrarse en el sistema.  |

### Sintaxis

```
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>] ';'

```

### Información relacionada

| Para obtener más información sobre | Consulte   |
|------------------------------------|--|
| Tareas de programa                 | <a href="#">taskid - Identificación de tarea en la página 1790</a> |

## 1.222 SaveCfgData - Guardar parámetros del sistema a un archivo

### Utilización

SaveCfgData se utiliza para guardar los parámetros del sistema a un archivo. Puede resultar útil tras actualizar los parámetros del sistema con la instrucción WriteCfgData.

### Ejemplos básicos

Los ejemplos que aparecen a continuación ilustran la instrucción SaveCfgData.

#### Ejemplo 1

```
SaveCfgData "SYSPAR" \File:="MYEIO.cfg", EIO_DOMAIN;
```

Guardado del dominio de la configuración de E/S con el archivo MYEIO.cfg en el directorio SYSPAR.

#### Ejemplo 2

```
SaveCfgData "SYSPAR", ALL_DOMAINS;
```

Guardado de todos los dominios de configuración existentes al directorio SYSPAR. Los archivos reciben los nombres EIO.cfg, MMC.cfg, PROC.cfg, SIO.cfg, SYS.cfg y MOC.cfg.

### Argumentos

```
SaveCfgData FilePath [\File] Domain
```

FilePath

Tipo de dato: string

La ruta y el nombre de archivo del lugar en el que se desea guardar el archivo. El nombre de archivo se excluye cuando se utiliza el argumento \File.

[\File]

Tipo de dato: string

Cuando se excluye el nombre del archivo en el argumento \FilePath, es necesario especificarlo con este argumento.

Domain

Tipo de dato: cfgdomain

El dominio de parámetros de sistema a guardar.

### Ejecución de programas

Guarda los parámetros del sistema en un archivo.

### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema ERRNO cambiará a:

| Nombre             | Causa del error                                       |
|--------------------|---|
| ERR_CFG_ILL_DOMAIN | El cfgdomain utilizado no es válido o no está en uso. |

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.222 SaveCfgData - Guardar parámetros del sistema a un archivo

Continuación

| Nombre            | Causa del error  |
|-------------------|--|
| ERR_CFG_WRITEFILE | El directorio no existe o el FilePath y el File utilizado es un directorio, o bien existe algún otro problema con el guardado del archivo. |

---

### Sintaxis

```
SaveCfgData
  [FilePath ':= ' ] <expression (IN) of string>
  ['\ ' File ':= ' <expression (IN) of string>]
  [Domain ':= ' ] <expression (IN) of cfgdomain> ';' 
```

---

### Información relacionada

| Para obtener más información sobre | Consulte   |
|------------------------------------|--|
| Datos de cfgdomain                 | <a href="#">cfgdomain - Dominio de configuración en la página 1646</a> |
| Parámetros del sistema             | <i>Manual de referencia técnica - Parámetros del sistema</i>           |



### 1.223 SCWrite - Envía los datos de la variable a una aplicación cliente

#### Utilización

SCWrite (*Superior Computer Write*) se utiliza para enviar el nombre, el tipo, el tamaño y el valor de una variable persistente a una aplicación cliente. Es posible enviar tanto variables individuales como matrices de variables.

#### Ejemplos básicos

Los ejemplos siguientes ilustran la instrucción SCWrite.

##### Ejemplo 1

```
PERS num cycle_done;  
  
PERS num numarr{2}:=[1,2];  
  
SCWrite cycle_done;
```

El nombre, el tipo y el valor de la variable persistente `cycle_done` se envía a todas las aplicaciones cliente.

##### Ejemplo 2

```
SCWrite \ToNode := "138.221.228.4", cycle_done;
```

El nombre, el tipo y el valor de la variable persistente `cycle_done` se envía a todas las aplicaciones cliente. El argumento `\ToNode` no se tiene en cuenta.

##### Ejemplo 3

```
SCWrite numarr;
```

El nombre, el tipo, la dimensión y el valor de la variable persistente `numarr` se envía a todas las aplicaciones cliente.

##### Ejemplo 4

```
SCWrite \ToNode := "138.221.228.4", numarr;
```

El nombre, el tipo, la dimensión y el valor de la variable persistente `numarr` se envía a todas las aplicaciones cliente. El argumento `\ToNode` no se tiene en cuenta.

#### Argumentos

```
SCWrite [ \ToNode ] Variable
```

[ \ToNode ]

Tipo de dato: datatype

El argumento no se tiene en cuenta.

Variable

Tipo de dato: anytype

El nombre de una variable persistente.

#### Ejecución de programas

El nombre, el tipo, la dimensión y el valor de la variable persistente se envía a todas las aplicaciones cliente. 'dim' es la dimensión de la variable y sólo se envía si la variable es una matriz.

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.223 SCWrite - Envía los datos de la variable a una aplicación cliente

PC interface/backup

Continuación

---

### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                    | Causa del error                 |
|---------------------------|---------------------------------|
| <code>ERR_SC_WRITE</code> | Error de envío a un PC externo. |

La instrucción `SCWrite` devuelve un error en los casos siguientes:

La variable no pudo ser enviada al cliente. Puede deberse a las causas siguientes:

- Los mensajes de `SCWrite` llegan tan cercanos entre sí que no es posible enviarlos al cliente. Solución: Añada una instrucción `WaitTime` entre las instrucciones `SCWrite` sucesivas.
- El valor de la variable es demasiado grande. Reduzca el tamaño de la `ARRAY` o del `RECORD`.
- El mensaje de error será el siguiente: `41473 System access error, Failed to send variable arg1`, donde `arg1` es el nombre de la variable.

La instrucción `SCWrite` no devolverá ningún error en algunos casos, por ejemplo si la aplicación cliente está cerrada o la comunicación está desactivada. El programa se sigue ejecutando.

---

### `SCWrite` recuperación en caso de error

Para detener el programa cuando se produce un error en una instrucción `SCWrite`, éste debe ser gestionado por un *gestor de errores*. En este caso, el error sólo queda registrado en el registro y el programa se sigue ejecutando.

Recuerde que la gestión de errores dificultará la localización de los errores en la comunicación con la aplicación cliente, dado que el error no se comunica nunca a través de la pantalla del FlexPendant (a pesar de que sí aparece en el registro).

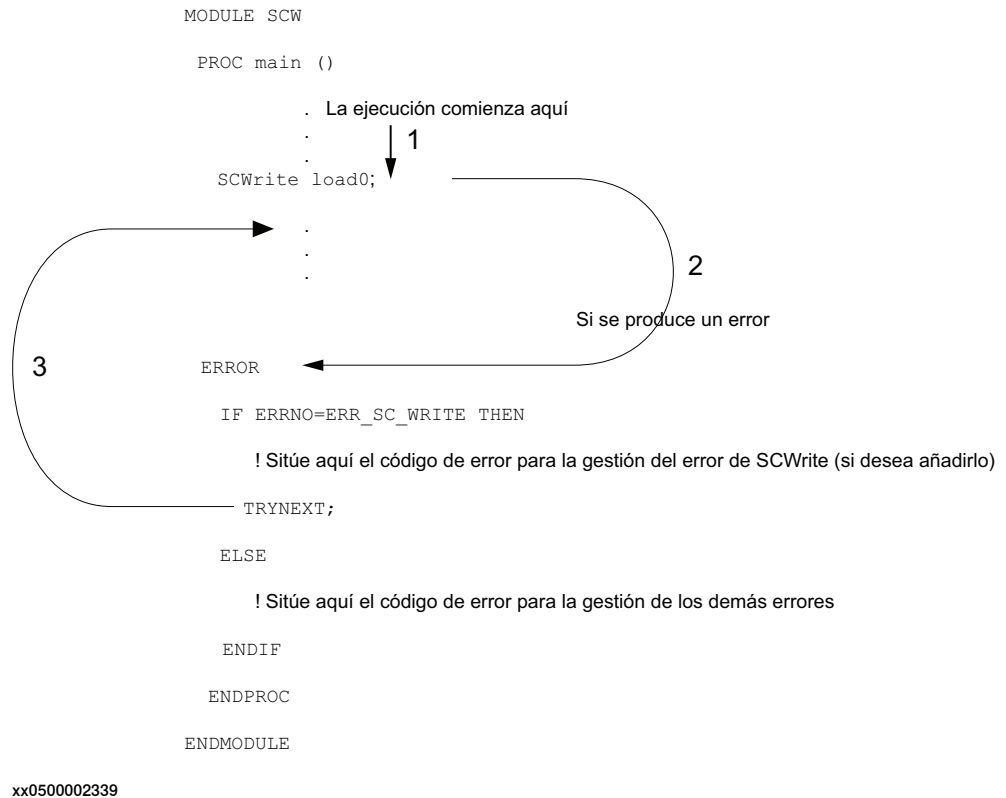
Continúa en la página siguiente

## 1.223 SCWrite - Envía los datos de la variable a una aplicación cliente

*PC interface/backup*

*Continuación*

El programa de RAPID tiene el aspecto siguiente:



# 1 Instrucciones

---

1.224 SearchC - Realiza una búsqueda en círculo usando el robot  
RobotWare - OS

## 1.224 SearchC - Realiza una búsqueda en círculo usando el robot

---

### Utilización

SearchC (*Search Circular* se utiliza para buscar una posición al mover el punto central de la herramienta (TCP) en sentido circular.

Durante el movimiento, el robot supervisa una señal digital de entrada o una variable persistente. Cuando el valor de la señal o de la variable persistente cambia al valor solicitado, el robot lee inmediatamente la posición actual.

Normalmente, esta instrucción puede usarse cuando la herramienta sostenida por el robot es una sonda para detección de superficies. Las coordenadas de contorno de un objeto de trabajo pueden obtenerse con la instrucción SearchC.

Esta instrucción sólo puede usarse en la tarea principal T\_ROB1 o, si se cuenta con un sistema *MultiMove*, en las tareas de movimiento.

Al utilizar instrucciones de búsqueda, resulta importante configurar el sistema de E/S para reducir al mínimo el tiempo que transcurre desde el establecimiento de la señal física hasta que el sistema obtiene información acerca del valor (utilice el dispositivo de E/S con control de interrupciones, no con control de sondeo). La forma de hacerlo puede ser distinta de un bus de campo a otro. Si se utiliza DeviceNet, las unidades de ABB (E/S local) proporcionan tiempos breves, dado que utilizan el *cambio de estado* como tipo de conexión. Si se utilizan otros buses de campo, asegúrese de configurar la red correctamente para conseguir las condiciones correctas.

---

### Ejemplos básicos

Los ejemplos siguientes ilustran la instrucción SearchC.

Consulte también [Más ejemplos en la página 662](#).

#### Ejemplo 1

```
SearchC di1, sp, cirpoint, p10, v100, probe;
```

El TCP de la sonda `probe` se mueve circularmente hacia la posición `p10` a una velocidad de `v100`. Cuando el valor de la señal `di1` cambia a activo, la posición se almacena en `sp`.

#### Ejemplo 2

```
SearchC \Stop, di2, sp, cirpoint, p10, v100, probe;
```

El TCP de la sonda `probe` se mueve circularmente hacia la posición `p10`. Cuando el valor de la señal `di2` cambia a activo, la posición se almacena en `sp` y el robot se detiene inmediatamente.

#### Ejemplo 3

```
PERS bool mypers:=FALSE;  
...  
SearchC \Stop, mypers, sp, cirpoint, p10, v100, probe;
```

El TCP del `probe` se mueve circularmente hacia la posición `p10`. Cuando el valor de la variable persistente `mypers` cambia a TRUE, la posición se almacena en `sp` y el robot se detiene inmediatamente.

*Continúa en la página siguiente*

## Argumentos

```
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**

Tipo de dato: switch

El movimiento del robot se detiene lo antes posible, sin mantener el TCP en la trayectoria (paro rígido) cuando el valor de la señal de búsqueda cambia a activo o el valor de la variable persistente cambia a TRUE. El robot se mueve una distancia corta antes del paro y no regresa a la posición buscada, es decir, a la posición en la que cambió la señal o el valor de la variable persistente.

Pruebe primero con una velocidad baja, por ejemplo <100 mm/s, y a continuación aumente gradualmente la velocidad hasta el valor deseado.

**¡AVISO!**

La detención de la búsqueda con un paro rígido (modificador \Stop) solo está permitida si la velocidad del TCP es inferior a 100 mm/s. En los paros rígidos a una mayor velocidad, algunos ejes pueden moverse en direcciones impredecibles.

**Nota**

Para un robot YuMi, la velocidad máxima para buscar con paro rígido es 1000 mm/s.

[ \PStop ]

**Path Stop**

Tipo de dato: switch

El movimiento del robot se detiene lo antes posible, manteniendo el TCP en la trayectoria (paro blando) cuando el valor de la señal de búsqueda cambia a activo o el valor de la variable persistente cambia a TRUE. El robot se mueve cierta distancia antes del paro y no regresa a la posición buscada, es decir, a la posición en la que cambió la señal o el valor de la variable persistente.

[ \SStop ]

**Soft Stop**

Tipo de dato: switch

El movimiento del robot se detiene lo antes posible, manteniendo el TCP cerca de la trayectoria o en ésta (paro blando) cuando el valor de la señal de búsqueda cambia a activo o el valor de la variable persistente cambia a TRUE. Sin embargo, el robot solo se mueve una distancia corta antes del paro y no regresa a la posición buscada, es decir, a la posición en la que cambió la señal. SStop resulta más rápido que PStop. Cuando el robot funciona a una velocidad superior a los 100 mm/s, se detiene en la dirección de la tangente del movimiento, lo que hace que se deslice marginalmente de la trayectoria.

Continúa en la página siguiente

# 1 Instrucciones

---

## 1.224 SearchC - Realiza una búsqueda en círculo usando el robot

RobotWare - OS

Continuación

[ \Sup ]

### *Supervision*

Tipo de dato: `switch`

La instrucción de búsqueda es sensible a la activación de señales o el cambio de valores de variables persistentes durante el movimiento completo (búsqueda en vuelo), es decir, incluso después de que se informa del primer cambio de señal o cambio de variable persistente. Si se produce más de una coincidencia durante una búsqueda, se genera un error recuperable con el robot en `ToPoint`.

Si se omiten los argumentos `\Stop`, `\PStop`, `\SStop` y `\Sup` (no se utiliza ningún modificador):

- el movimiento continúa (búsqueda en vuelo) hasta la posición especificada en el argumento `ToPoint` (igual que en el argumento `\Sup`)
- Se indica un error cuando no existe ninguna coincidencia de búsqueda, pero no se indica tal error si hay más de una coincidencia de búsqueda (la primera coincidencia de búsqueda se devuelve como `SearchPoint`)

Signal

Tipo de dato: `signal`

El nombre de la señal a supervisar.

PersBool

Tipo de dato: `bool`

La variable persistente que se desea supervisar.

[ \Flanks ]

Tipo de dato: `switch`

Los bordes positivo y negativo de la señal son válidos como coincidencias de búsqueda. Si se utiliza el argumento `PersBool`, el cambio de valor de la variable se considera válido como coincidencia de búsqueda.

Para la señal: si se omite el argumento `\Flanks`, solo el borde positivo de la señal es válido para determinar el éxito de una búsqueda y se activa una supervisión de la señal al comenzar un proceso de búsqueda. Esto significa que si la señal ya tiene el valor positivo al comenzar un proceso de búsqueda o se pierde la comunicación con la señal, el movimiento del robot se detiene lo antes posible, manteniendo el TCP en la trayectoria (paro blando). Se genera un error recuperable por el usuario `ERR_SIGSUPSEARCH`, que puede gestionarse en el gestor de errores.

Para la variable persistente: si se omite el argumento `\Flanks`, solo cuando el valor cambia a `TRUE` se determina el éxito de una búsqueda y se activa una supervisión de la variable al comenzar un proceso de búsqueda. Esto significa que si la variable persistente ya tiene el valor positivo al comenzar un proceso de búsqueda, el movimiento del robot se detiene lo antes posible, manteniendo el TCP en la trayectoria (paro blando). Se genera un error recuperable por el usuario `ERR_PERSSUPSEARCH`, que puede gestionarse en el gestor de errores.

[ \PosFlank ]

Tipo de dato: `switch`

*Continúa en la página siguiente*

## 1.224 SearchC - Realiza una búsqueda en círculo usando el robot

*RobotWare - OS*

*Continuación*

El borde positivo de la señal es válido para determinar el éxito de una búsqueda, o bien el cambio del valor a TRUE si se utiliza una variable persistente.

[ \NegFlank ]

Tipo de dato: `switch`

El borde negativo de la señal es válido para determinar el éxito de una búsqueda, o bien el cambio del valor a FALSE si se utiliza una variable persistente.

[ \HighLevel ]

Tipo de dato: `switch`

La misma funcionalidad que si no usara el modificador `\Flanks`.

Para la señal: el borde positivo de la señal es válido para determinar el éxito de una búsqueda y se activa una supervisión de la señal al comenzar un proceso de búsqueda. Esto significa que si la señal ya tiene el valor positivo al comenzar un proceso de búsqueda o se pierde la comunicación con la señal, el movimiento del robot se detiene lo antes posible, manteniendo el TCP en la trayectoria (paro blando). Se genera un error recuperable por el usuario `ERR_SIGSUPSEARCH`, que puede gestionarse en el gestor de errores.

Para la variable persistente: solo el cambio de valor a TRUE se considera un éxito de búsqueda y se activa una supervisión de la variable al comenzar un proceso de búsqueda. Esto significa que si la variable persistente ya tiene el valor positivo al comenzar un proceso de búsqueda, el movimiento del robot se detiene lo antes posible, manteniendo el TCP en la trayectoria (paro blando). Se genera un error recuperable por el usuario `ERR_PERSSUPSEARCH`, que puede gestionarse en el gestor de errores.

[ \LowLevel ]

Tipo de dato: `switch`

Para la señal: el borde negativo de la señal es válido para determinar el éxito de una búsqueda y se activa una supervisión de la señal al comenzar un proceso de búsqueda. Esto significa que si la señal ya tiene el valor 0 al comenzar un proceso de búsqueda o se pierde la comunicación con la señal, el movimiento del robot se detiene lo antes posible, manteniendo el TCP en la trayectoria (paro blando). Se genera un error recuperable por el usuario `ERR_SIGSUPSEARCH`, que puede gestionarse en el gestor de errores.

Para la variable persistente: solo el cambio de valor a FALSE se considera un éxito de búsqueda y se activa una supervisión de la variable al comenzar un proceso de búsqueda. Esto significa que si la variable persistente ya tiene el valor FALSE al comenzar un proceso de búsqueda, el movimiento del robot se detiene lo antes posible, manteniendo el TCP en la trayectoria (paro blando). Se genera un error recuperable por el usuario `ERR_PERSSUPSEARCH`, que puede gestionarse en el gestor de errores.

SearchPoint

Tipo de dato: `robtarget`

La posición del TCP y de los ejes externos en el momento del disparo de la señal de búsqueda. La posición se especifica en el sistema de coordenadas más externo,

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.224 SearchC - Realiza una búsqueda en círculo usando el robot

RobotWare - OS

Continuación

`ProgDisp/ExtOffs` la herramienta especificada, el objeto de trabajo y el sistema de coordenadas into consideration.

`CirPoint`

Tipo de dato: `robtarget`

El punto de círculo del robot. Consulte la instrucción `MoveC` para obtener una descripción más detallada del movimiento circular. El punto de círculo se define como una posición con nombre o se almacena directamente en la instrucción (marcada con un asterisco \* en la instrucción).

`ToPoint`

Tipo de dato: `robtarget`

El punto de destino de los ejes del robot y de los ejes externos. Se define como una posición con nombre o se almacena directamente en la instrucción (marcada con un asterisco \* en la instrucción). `SearchC` utiliza siempre un punto de paro como dato de zona del destino.

[ `\ID` ]

*Synchronization id*

Tipo de dato: `identno`

El argumento [ `\ID` ] es obligatorio en los sistemas `MultiMove`, si el movimiento es sincronizado o sincronizado coordinado. Este argumento no está permitido en ningún otro caso. El número de ID especificado debe ser el mismo en todas las tareas de programa que cooperan entre sí. Al usar el número de ID los movimientos no se mezclan en tiempo de ejecución.

`Speed`

Tipo de dato: `speeddata`

Los datos de velocidad que se aplican a los movimientos. Los datos de velocidad definen la velocidad del punto central de la herramienta, los ejes externos y la reorientación de la herramienta.

[ `\V` ]

*Velocity*

Tipo de dato: `num`

Este argumento se utiliza para especificar la velocidad del TCP en mm/s directamente en la instrucción. A continuación, se sustituye por la velocidad correspondiente, especificada en los datos de velocidad.

[ `\T` ]

*Time*

Tipo de dato: `num`

Este argumento se utiliza para especificar el tiempo total en segundos que dura el movimiento del robot. A continuación, se sustituye por los datos de velocidad correspondientes.

`Tool`

Tipo de dato: `tooldata`

*Continúa en la página siguiente*



## 1.224 SearchC - Realiza una búsqueda en círculo usando el robot RobotWare - OS Continuación

La herramienta en uso durante el movimiento del robot. El punto central de la herramienta es el punto que se mueve hacia la posición de destino especificada.

[ \WObj ]

### *Work Object*

Tipo de dato: `wobjdata`

El objeto de trabajo (sistema de coordenadas) con el que están relacionadas las posiciones de robot indicadas en la instrucción.

Es posible omitir este argumento. Si se omite, la posición depende del sistema de coordenadas mundo. Si por otro lado se usa un TCP estacionario o ejes externos coordinados, es necesario especificar este argumento para ejecutar un movimiento lineal respecto del objeto de trabajo.

[ \Corr ]

### *Correction*

Tipo de dato: `switch`

Si este argumento está presente, los datos de corrección escritos en una entrada de corrección mediante una instrucción `CorrWrite` se añaden a la trayectoria y a la posición de destino.

Se requiere RobotWare, opción *Path Offset*, cuando se utiliza este argumento.

[ \TLoad ]

### *Total load*

Tipo de dato: `loaddata`

El argumento `\TLoad` describe la carga total usada durante el movimiento. La carga total es la carga de la herramienta más la carga útil transportada por la herramienta. Si se utiliza el argumento `\TLoad`, no se tiene en cuenta el valor de `loaddata` en los `tooldata` actuales.

Si el argumento `\TLoad` tiene el valor `load0`, el argumento `\TLoad` no se tiene en cuenta y se utilizan en su lugar los `loaddata` de los `tooldata`.

Para poder utilizar el argumento `\TLoad`, es necesario cambiar el valor del parámetro de sistema `ModalPayLoadMode` a 0. Si `ModalPayLoadMode` tiene el valor 0, ya no es posible utilizar la instrucción `GripLoad`.

La carga total puede identificarse con la rutina de servicio `LoadIdentify`. Si el parámetro de sistema `ModalPayLoadMode` tiene el valor 0, el operador tiene la posibilidad de copiar los `loaddata` de la herramienta a una variable persistente `loaddata` existente o nueva al ejecutar la rutina de servicio.

Es posible realizar una ejecución de prueba del programa sin ninguna carga útil utilizando una señal de entrada digital conectada a la entrada de sistema `SimMode` (modo simulado). Si la señal de entrada digital tiene el valor 1, los `loaddata` del

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.224 SearchC - Realiza una búsqueda en círculo usando el robot

RobotWare - OS

Continuación

argumento opcional `\TLoad` no se tienen en cuenta y se utilizan en su lugar los `loaddata` de los `tooldata` actuales.



### Nota

La funcionalidad predeterminada de manejo de la carga útil es utilizar la instrucción `GripLoad`. Por tanto, el valor predeterminado del parámetro de sistema `ModalPayLoadMode` es 1.

## Ejecución de programas

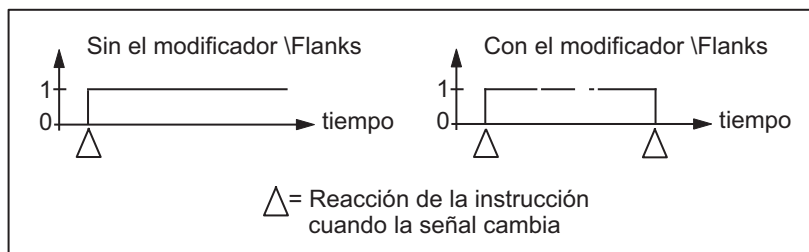
Consulte la instrucción `MoveC` para obtener más información acerca del movimiento circular.

El movimiento termina siempre con un punto de paro, lo que implica que el robot se detiene en el punto de destino.

Cuando se realiza una búsqueda en vuelo, es decir, cuando se utiliza el argumento `\Sup` o no se especifica ningún modificador, el movimiento del robot continúa siempre hacia el punto de destino programado. Cuando se realiza una búsqueda con el modificador `\Stop`, `\PStop` o `\SStop`, el movimiento del robot se detiene cuando se detecta la primera señal.

La instrucción `SearchC` devuelve la posición del TCP cuando el valor de la señal digital o de la variable persistente cambia al valor solicitado, como se representa en la figura siguiente.

La figura muestra cómo se usa la detección de señal disparada por flancos (la posición se almacena sólo cuando la señal cambia por primera vez).



xx0500002237

## Más ejemplos

A continuación aparecen más ejemplos de cómo usar la instrucción `SearchC`.

### Ejemplo 1

```
SearchC \Sup, dil\Flanks, sp, cirpoint, p10, v100, probe;
```

El TCP de la sonda `probe` se mueve circularmente hacia la posición `p10`. Cuando el valor de la señal `dil` cambia a activo o pasivo, la posición se almacena en `sp`. Si el valor de la señal cambia dos veces, el programa genera un error.

## Limitaciones

Limitaciones generales acorde con la instrucción `MoveC`.

Continúa en la página siguiente

## 1.224 SearchC - Realiza una búsqueda en círculo usando el robot

*RobotWare - OS*

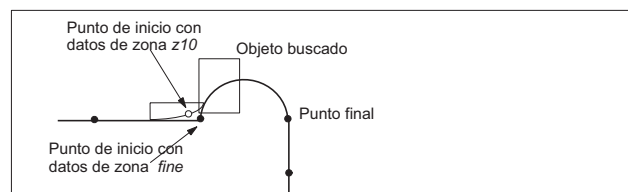
*Continuación*

SearchC no puede ejecutarse en un gestor UNDO o en rutinas de RAPID que estén conectadas a los siguientes eventos especiales del sistema: PowerOn, Stop, QStop, Restart, Reset o Step.

Los datos de zona de la instrucción de posicionamiento que precede a SearchC deben usarse con cuidado. La recomendación consiste en utilizar *z0* o una pequeña zona que aún proporcione un movimiento suave. El comienzo de la búsqueda, es decir, cuando la señal de E/S está preparada para reaccionar, no es en este caso el punto de destino programado de la instrucción de posicionamiento anterior, sino un punto que se encuentra en la trayectoria real del robot. En la figura siguiente se muestra un ejemplo de algo que puede salir mal cuando se usan datos de zona distintos de *fine*.

La instrucción SearchC no debe reanudarse en ningún caso una vez traspasado el punto de círculo. De lo contrario, el robot no toma la trayectoria programada (posicionamiento alrededor de la trayectoria circular en otra dirección, en comparación con la programada).

En la figura siguiente se muestra cómo se encuentra una posición en el lado incorrecto del objeto a causa del uso de datos de zona incorrectos.



xx0500002238

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.224 SearchC - Realiza una búsqueda en círculo usando el robot

RobotWare - OS

Continuación



### ¡AVISO!

Limitaciones de búsqueda con movimiento sincronizado y coordinado:

- Si se utiliza `SearchL`, `SearchC` o `SearchExtJ` en una tarea de programa y en otra instrucción de movimiento de otra tarea de programa, sólo es posible usar la búsqueda en vuelo con el modificador `\Sup`. Además de ello, sólo es posible realizar la recuperación en caso de error con `TRYNEXT`.
- Es posible utilizar toda la funcionalidad de búsqueda si se utilizan instrucciones `SearchL`, `SearchC` o `SearchExtJ` en todas las tareas de programa implicadas con movimiento sincronizado y coordinado y se genera una coincidencia de búsqueda desde la misma señal digital de entrada. Con ello se genera una coincidencia de búsqueda de forma sincronizada en todas las instrucciones de búsqueda. Cualquier recuperación de error debe ser también la misma en todas las tareas de programa implicadas.

Mientras la búsqueda está activa, no es posible almacenar la trayectoria actual con la instrucción `StorePath`.

Exactitud de repetición para la posición de coincidencia de búsqueda con una velocidad de 20 a 1.000 mm/s de 0,1 a 0,3 mm.

Distancia de paro típica con una velocidad de búsqueda de 50 mm/s:

- Sin el TCP en la trayectoria (modificador `\Stop`), de 1 a 3 mm
- Con el TCP en la trayectoria (modificador `\PStop`), de 15 a 25 mm
- Con el TCP cerca de la trayectoria (modificador `\SStop`), de 4 a 8 mm

Limitaciones de la búsqueda en un transportador:

- la búsqueda detiene el robot en caso de coincidencia o si la búsqueda no tiene éxito, de forma que la búsqueda debe hacerse en el mismo sentido que el del movimiento del transportador y después del paro de búsqueda debe ir seguida de un movimiento hasta una posición segura. Utilice la gestión de errores para el movimiento hasta una posición segura si la búsqueda no tiene éxito.
- la exactitud de repetición de la posición de coincidencia de búsqueda será menor al buscar en un transportador y depende de la velocidad del transportador y de hasta qué punto esta velocidad es estable.

## Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                          | Causa del error  |
|---------------------------------|--|
| <code>ERR_NO_ALIASIO_DEF</code> | La variable de señal es una variable declarada en <code>RAPID</code> . No se ha conectado a una señal E/S definida en la configuración de E/S con la instrucción <code>AliasIO</code> .                                    |
| <code>ERR_WHLSEARCH</code>      | <ul style="list-style-type: none"><li>• No se produjo ninguna detección de señal.</li><li>• Se ha producido la detección de más de una señal. Esto solo se produce si se utiliza el argumento <code>\Sup</code>.</li></ul> |

Continúa en la página siguiente

## 1.224 SearchC - Realiza una búsqueda en círculo usando el robot

RobotWare - OS

Continuación

| Nombre            | Causa del error  |
|-------------------|--|
| ERR_SIGSUPSEARCH  | La señal ya tiene un valor positivo al comienzo del proceso de búsqueda o se ha perdido la comunicación con la señal. Esto solo se produce si se omite el argumento \Flanks. |
| ERR_PERSSUPSEARCH | La variable persistente ya es TRUE al comienzo del proceso de búsqueda. Esto solo se produce si se omite el argumento \Flanks.   |

Los errores pueden gestionarse de formas distintas en función del modo de ejecución seleccionado:

- **Ejecución continua hacia delante / Instrucciones hacia adelante**  
/ERR\_WHLSEARCH: No se devuelve ninguna posición y el movimiento continúa siempre hacia el punto de destino programado. La variable de sistema ERRNO cambia a ERR\_WHLSEARCH y es posible gestionar el error en el gestor de errores de la rutina.
- **Ejecución continua hacia adelante / Instrucciones hacia adelante /**  
ERR\_SIGSUPSEARCH y ERR\_PERSSUPSEARCH: No se devuelve ninguna posición y el movimiento se detiene siempre lo antes posible al principio de la trayectoria de búsqueda. La variable de sistema ERRNO cambia a ERR\_SIGSUPSEARCH o ERR\_PERSSUPSEARCH en función del argumento utilizado (señal o variable persistente) y es posible gestionar el error en el gestor de errores de la rutina.
- **Instrucción hacia atrás:** Durante la ejecución hacia atrás, la instrucción realiza el movimiento pero no realiza ninguna supervisión.

## Sintaxis

```
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>';'
```

Continúa en la página siguiente

# 1 Instrucciones

## 1.224 SearchC - Realiza una búsqueda en círculo usando el robot

RobotWare - OS

Continuación

### Información relacionada

| Para obtener más información sobre   | Consulte   |
|--|--|
| Búsquedas lineales   | <a href="#">SearchL - Realiza una búsqueda lineal usando el robot en la página 675</a>   |
| Escritura en una entrada de corrección   | <a href="#">CorrWrite - Escribe en un generador de correcciones en la página 183</a>     |
| Mueve el robot en círculo  | <a href="#">MoveC - Mueve el robot en círculo en la página 389</a>                       |
| Movimiento circular  | <i>Manual de referencia técnica - RAPID Overview</i>                                     |
| Definición de carga  | <a href="#">loaddata - Datos de carga en la página 1694</a>                              |
| Definición de velocidad  | <a href="#">speeddata - Datos de velocidad en la página 1767</a>                         |
| Definición de herramientas   | <a href="#">tooldata - Datos de herramienta en la página 1795</a>                        |
| Definición de objetos de trabajo   | <a href="#">wobjdata - Datos del objeto de trabajo en la página 1823</a>                 |
| Utilización de gestores de errores   | <i>Manual de referencia técnica - RAPID Overview</i>                                     |
| Movimiento en general  | <i>Manual de referencia técnica - RAPID Overview</i>                                     |
| Ejemplo de cómo usar TLoad, carga total.   | <a href="#">MoveL - Mueve el robot siguiendo una trayectoria lineal en la página 447</a> |
| Definición de la carga útil de un robot  | <a href="#">GripLoad - Define la carga útil de un robot en la página 244</a>             |
| LoadIdentify, rutina de servicio de identificación de carga                                    | <i>Manual del operador - IRC5 con FlexPendant</i>  |
| Señal de entrada de sistema SimMode para mover el robot en el modo simulado sin carga útil.    | <i>Manual de referencia técnica - Parámetros del sistema</i>                             |
| Parámetro de sistema ModalPayload-Mode para la activación y la desactivación de la carga útil. | <i>Manual de referencia técnica - Parámetros del sistema</i>                             |
| <i>Path Offset</i>   | <i>Application manual - Controller software IRC5</i>                                     |

## 1.225 SearchExtJ - Busca con una o varias unidades mecánicas sin TCP

### Utilización

`SearchExtJ` (*Search External Joints*) se usa para buscar la posición de ejes externos cuando sólo hay movimiento en ejes externos lineales o de rotación. Estos ejes externos pueden pertenecer a una o varias unidades mecánicas sin TCP.

Durante el movimiento, el robot supervisa una señal digital de entrada o una variable persistente. Cuando el valor de la señal o de la variable persistente cambia al valor solicitado, el robot lee inmediatamente la posición actual.

Esta instrucción sólo puede usarse si:

- La tarea de programa actual está definida como tarea de movimiento
- La tarea controla una o varias unidades mecánicas sin TCP

Al utilizar instrucciones de búsqueda, resulta importante configurar el sistema de E/S para reducir al mínimo el tiempo de retardo que transcurre desde el establecimiento de la señal física hasta que el sistema obtiene información acerca del valor (utilice el dispositivo de E/S con control de interrupciones, no con control de sondeo). La forma de hacerlo puede ser distinta de un bus de campo a otro. Si se utiliza DeviceNet, las unidades de ABB DSQC 651 (AD Combi I/O) y DSQC 652 (Digital I/O) proporcionan retardos breves, dado que utilizan el cambio de estado como tipo de conexión. Si se utilizan otros buses de campo, asegúrese de que la red esté configurada correctamente para obtener las condiciones correctas.

### Ejemplos básicos

Los ejemplos siguientes ilustran la instrucción `SearchExtJ`.

Consulte también [Más ejemplos en la página 671](#).

#### Ejemplo 1

```
SearchExtJ di1, searchp, jpos10, vrot20;
```

La unidad mecánica con ejes de rotación se mueve hacia la posición `jpos10` a la velocidad de `vrot20`. Cuando el valor de la señal `di1` cambia a activo, la posición se almacena en `searchp`.

#### Ejemplo 2

```
SearchExJ \Stop, di2, posx, jpos20, vlin50;
```

La unidad mecánica con eje lineal se mueve hacia la posición `jpos20`. Cuando el valor de la señal `di2` cambia a activo, la posición se almacena en `posx` y el movimiento en curso se detiene inmediatamente.

#### Ejemplo 3

```
PERS bool mypers:=FALSE;
...
SearchExJ \Stop, di2, posx, jpos20, vlin50;
```

La unidad mecánica con eje lineal se mueve hacia la posición `jpos20`. Cuando el valor de la variable persistente `mypers` cambia a TRUE, la posición se almacena en `posx` y el movimiento en curso se detiene inmediatamente.

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.225 SearchExtJ - Busca con una o varias unidades mecánicas sin TCP

RobotWare - OS

Continuación

---

### Argumentos

```
SearchExtJ [\Stop] | [\PStop] | [\SStop] | [\Sup] Signal | PersBool  
[\Flanks] | [\PosFlank] | [\NegFlank] | [\HighLevel] |  
[\LowLevel] SearchJointPos ToJointPos [\ID] [\UseEOffs] Speed  
[\T]
```

[ \Stop ]

#### **Stiff Stop**

Tipo de dato: switch

El movimiento del robot se detiene lo antes posible con un paro rígido cuando el valor de la señal de búsqueda cambia a activo o el valor de la variable persistente cambia a TRUE. Los ejes externos se mueven una distancia pequeña antes de detenerse y no retroceden hasta la posición buscada, es decir, hasta la posición en la que cambió la señal.

[ \PStop ]

#### **Path Stop**

Tipo de dato: switch

El movimiento se detiene con un paro en la trayectoria (paro de programa) cuando el valor de la señal de búsqueda cambia a activo o el valor de la variable persistente cambia a TRUE. Los ejes externos se mueven una distancia bastante grande antes de detenerse y no retroceden hasta la posición buscada, es decir, hasta la posición en la que cambió la señal.

[ \SStop ]

#### **Soft Stop**

Tipo de dato: switch

El movimiento del robot se detiene lo antes posible con un paro blando rápido cuando el valor de la señal de búsqueda cambia a activo o el valor de la variable persistente cambia a TRUE. Los ejes externos solo se mueven una distancia pequeña antes de detenerse y no retroceden hasta la posición buscada, es decir, hasta la posición en la que cambió la señal.

Stop es más rápido en comparación con SStop. SStop, y más rápido en comparación con PStop.

[ \Sup ]

#### **Supervision**

Tipo de dato: switch

La instrucción de búsqueda es sensible a la activación de señales o el cambio de valores de variables persistentes durante el movimiento completo (búsqueda en vuelo), es decir, incluso después de que se informa del primer cambio de señal o cambio de variable persistente. Si se produce más de una coincidencia durante una búsqueda, se genera un error recuperable con el robot en ToPoint.

Si se omiten los argumentos \Stop, \PStop, \SStop y \Sup (no se utiliza ningún modificador):

- El movimiento continúa (búsqueda en vuelo) hasta la posición especificada en el argumento ToJointPos (igual que en el argumento \Sup)

Continúa en la página siguiente

---



- Se indica un error cuando hay una coincidencia de búsqueda, pero no se indica tal error si hay más de una coincidencia de búsqueda (la primera coincidencia de búsqueda se devuelve como `SearchJointPos`)

Signal

Tipo de dato: `signal`

El nombre de la señal a supervisar.

PersBool

Tipo de dato: `bool`

La variable persistente que se desea supervisar.

[ `\Flanks` ]

Tipo de dato: `switch`

Los bordes positivo y negativo de la señal son válidos como coincidencias de búsqueda. Si se utiliza el argumento `PersBool`, el cambio de valor de la variable se considera válido como coincidencia de búsqueda.

Para la señal: si se omite el argumento `\Flanks`, solo el borde positivo de la señal es válido para determinar el éxito de una búsqueda y se activa una supervisión de la señal al comenzar un proceso de búsqueda. Esto significa que si la señal ya tiene el valor positivo al comenzar un proceso de búsqueda o se pierde la comunicación con la señal, el movimiento se detiene lo antes posible. Se genera un error recuperable por el usuario `ERR_SIGSUPSEARCH`, que puede gestionarse en el gestor de errores.

Para la variable persistente: si se omite el argumento `\Flanks`, solo cuando el valor cambia a `TRUE` se determina el éxito de una búsqueda y se activa una supervisión de la variable al comenzar un proceso de búsqueda. Esto significa que si la variable persistente ya tiene el valor positivo al comenzar un proceso de búsqueda, el movimiento se detiene lo antes posible. Se genera un error recuperable por el usuario `ERR_PERSSUPSEARCH`, que puede gestionarse en el gestor de errores.

[ `\PosFlank` ]

Tipo de dato: `switch`

El borde positivo de la señal es válido para determinar el éxito de una búsqueda, o bien el cambio del valor a `TRUE` si se utiliza una variable persistente.

[ `\NegFlank` ]

Tipo de dato: `switch`

El borde negativo de la señal es válido para determinar el éxito de una búsqueda, o bien el cambio del valor a `FALSE` si se utiliza una variable persistente.

[ `\HighLevel` ]

Tipo de dato: `switch`

La misma funcionalidad que si no usara el modificador `\Flanks`.

Para la señal: el borde positivo de la señal es válido para determinar el éxito de una búsqueda y se activa una supervisión de la señal al comenzar un proceso de búsqueda. Esto significa que si la señal ya tiene el valor positivo al comenzar un

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.225 SearchExtJ - Busca con una o varias unidades mecánicas sin TCP

RobotWare - OS

Continuación

proceso de búsqueda o se pierde la comunicación con la señal, el movimiento se detiene lo antes posible. Se genera un error recuperable por el usuario `ERR_SIGSUPSEARCH`, que puede gestionarse en el gestor de errores.

Para la variable persistente: solo cuando el valor cambia a `TRUE` se determina el éxito de una búsqueda y se activa una supervisión de la variable al comenzar un proceso de búsqueda. Esto significa que si la variable persistente ya tiene el valor positivo al comenzar un proceso de búsqueda, el movimiento se detiene lo antes posible. Se genera un error recuperable por el usuario `ERR_PERSSUPSEARCH`, que puede gestionarse en el gestor de errores.

[ \LowLevel ]

Tipo de dato: `switch`

Para la señal: el borde negativo de la señal es válido para determinar el éxito de una búsqueda y se activa una supervisión de la señal al comenzar un proceso de búsqueda. Esto significa que si la señal ya tiene el valor 0 al comenzar un proceso de búsqueda o se pierde la comunicación con la señal, el movimiento se detiene lo antes posible. Se genera un error recuperable por el usuario `ERR_SIGSUPSEARCH`, que puede gestionarse en el gestor de errores.

Para la variable persistente: solo cuando el valor cambia a `FALSE` se determina el éxito de una búsqueda y se activa una supervisión de la variable al comenzar un proceso de búsqueda. Esto significa que si la variable persistente ya tiene el valor `FALSE` al comenzar un proceso de búsqueda, el movimiento se detiene lo antes posible. Se genera un error recuperable por el usuario `ERR_PERSSUPSEARCH`, que puede gestionarse en el gestor de errores.

SearchJointPos

Tipo de dato: `jointtarget`

La posición de los ejes externos en el momento del disparo de la señal de búsqueda. La posición tiene en cuenta cualquier `ExtOffs` activo.

ToJointPos

Tipo de dato: `jointtarget`

El punto de destino de los ejes externos. Se define como una posición con nombre o se almacena directamente en la instrucción (marcada con un asterisco \* en la instrucción). `SearchExtJ` utiliza siempre un punto de paro como dato de zona del destino.

[ \ID ]

*Synchronization id*

Tipo de dato: `identno`

El argumento [ \ID ] es obligatorio en los sistemas `MultiMove`, si el movimiento es sincronizado o sincronizado coordinado. Este argumento no está permitido en ningún otro caso. El número de ID especificado debe ser el mismo en todas las tareas de programa que cooperan entre sí. Al usar el número de ID los movimientos no se mezclan en tiempo de ejecución.

[ \UseEOffs ]

*Use External Offset*

Continúa en la página siguiente

Tipo de dato: `switch`

El offset de los ejes externos, configurado por la instrucción `EOfsSet`, se activa para la instrucción `SearchExtJ` cuando se utiliza el argumento `UseEOffs`. Consulte la instrucción `EOfsSet` para obtener más información acerca del offset externo.

Speed

Tipo de dato: `speeddata`

Los datos de velocidad que se aplican a los movimientos. Los datos de velocidad definen la velocidad del eje externo lineal o de rotación.

[ \T ]

*Time*

Tipo de dato: `num`

Este argumento se utiliza para especificar el tiempo total en segundos que dura el movimiento de las unidades mecánicas. A continuación, se sustituye por los datos de velocidad correspondientes.

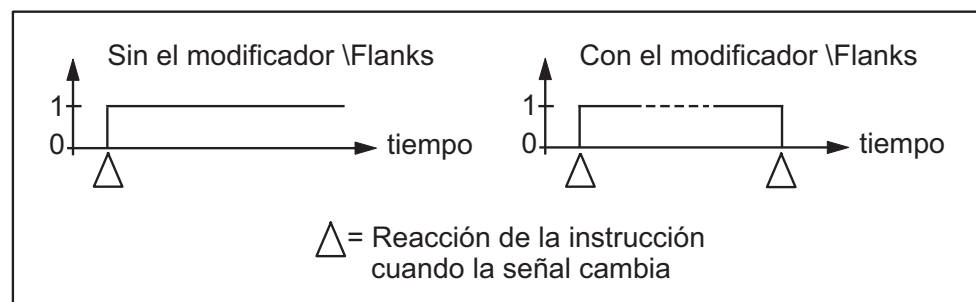
## Ejecución de programas

Consulte la instrucción `MoveExtJ` para obtener más información acerca del movimiento de las unidades mecánicas sin TCP.

El movimiento termina siempre con un punto de paro, lo que implica que los ejes externos se detienen en el punto de destino. Si se realiza una búsqueda en vuelo, es decir, cuando se utiliza el argumento `\Sup` o no se especifica ningún modificador, el movimiento siempre continúa hacia el punto de destino programado. Si se realiza una búsqueda con el modificador `\Stop`, `\Pstop` o `\Sstop`, el movimiento se detiene cuando se detecta la primera señal.

La instrucción `SearchExtJ` almacena la posición de los ejes externos cuando el valor de la señal digital o la variable persistente cambia al valor solicitado, como se representa en la figura siguiente.

La figura muestra cómo se usa la detección de señal disparada por flancos (la posición se almacena sólo cuando la señal cambia por primera vez).



xx0500002243

## Más ejemplos

A continuación aparecen más ejemplos de cómo usar la instrucción `SearchExtJ`.

### Ejemplo 1

```
SearchExtJ \Sup, dil\Flanks, searchp, jpos10, vrot20;
```

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.225 SearchExtJ - Busca con una o varias unidades mecánicas sin TCP

RobotWare - OS

Continuación

La unidad mecánica se mueve hacia la posición `jpos10`. Cuando el valor de la señal `dil` cambia a activo o pasivo, la posición se almacena en `searchp`. Si el valor de la señal cambia dos veces, el programa genera un error una vez que se completa el proceso de búsqueda.

### Ejemplo 2

```
SearchExtJ \Stop, dil, sp, jpos20, vlin50;  
MoveExtJ sp, vlin50, fine \Inpos := inpos50;
```

Al comienzo del proceso de búsqueda, se realiza una comprobación de la señal `dil` y si la señal ya tiene un valor positivo o se pierde la comunicación con el robot, el movimiento se detiene. De lo contrario, la unidad mecánica se mueve hacia la posición `jpos20`. Cuando el valor de la señal `dil` cambia a activo, la posición se almacena en `sp`. La unidad mecánica vuelve a este punto usando un punto de paro definido exactamente.

### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                          | Causa del error  |
|---------------------------------|--|
| <code>ERR_NO_ALIASIO_DEF</code> | La variable de señal es una variable declarada en <code>RAPID</code> . No se ha conectado a una señal E/S definida en la configuración de E/S con la instrucción <code>AliasIO</code> .                                |
| <code>ERR_WHLSEARCH</code>      | <ul style="list-style-type: none"><li>No se produjo ninguna detección de señal.</li><li>Se ha producido la detección de más de una señal. Esto solo se produce si se utiliza el argumento <code>\Sup</code>.</li></ul> |
| <code>ERR_SIGSUPSEARCH</code>   | La señal ya tiene un valor positivo al comienzo del proceso de búsqueda o se ha perdido la comunicación con la señal. Esto solo se produce si se omite el argumento <code>\Flanks</code> .                             |
| <code>ERR_PERSSUPSEARCH</code>  | La variable persistente ya es <code>TRUE</code> al comienzo del proceso de búsqueda. Esto solo se produce si se omite el argumento <code>\Flanks</code> .  |

Los errores pueden gestionarse de formas distintas en función del modo de ejecución seleccionado:

- Ejecución continua hacia adelante / Instrucciones hacia adelante**  
`/ERR_WHLSEARCH`: No se devuelve ninguna posición y el movimiento continúa siempre hacia el punto de destino programado. La variable de sistema `ERRNO` cambia a `ERR_WHLSEARCH` y es posible gestionar el error en el gestor de errores de la rutina.
- Ejecución continua hacia adelante / Instrucciones hacia adelante /**  
`ERR_SIGSUPSEARCH` y `ERR_PERSSUPSEARCH`: No se devuelve ninguna posición y el movimiento se detiene siempre lo antes posible al principio de la trayectoria de búsqueda. La variable de sistema `ERRNO` cambia a `ERR_SIGSUPSEARCH` o `ERR_PERSSUPSEARCH` en función del argumento utilizado (señal o variable persistente) y es posible gestionar el error en el gestor de errores de la rutina.
- Instrucción hacia atrás**: Durante la ejecución hacia atrás, la instrucción realiza el movimiento pero no realiza ninguna supervisión.

Continúa en la página siguiente

## Ejemplo

```

VAR num fk;
...
MoveExtJ jpos10, vrot100, fine;
SearchExtJ \Stop, dil, 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!";
    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
ENDIF

```

Si la señal ya está activa al principio del proceso de búsqueda o se pierde la comunicación con la señal, se activa un cuadro de diálogo (TPReadFK ...;). Si se pone a cero manualmente la señal y se presiona SÍ en la ventana de diálogo del usuario, la unidad mecánica vuelve a la posición jpos10 y vuelve a intentar la operación. De lo contrario, la ejecución del programa se detiene.

Si la señal está desactivada al comienzo del proceso de búsqueda, la unidad mecánica busca desde la posición jpos10 hasta la posición jpos20. Si no se detecta ninguna señal, el robot vuelve a la posición jpos10 y lo intenta de nuevo.

## Limitaciones

SearchExtJ no puede ejecutarse en un gestor UNDO o en rutinas de RAPID que estén conectadas a los siguientes eventos especiales del sistema: PowerOn, Stop, QStop, Restart, Reset o Step.

Limitaciones de búsqueda con movimiento sincronizado y coordinado:

- Si se utiliza SearchL, SearchC o SearchExtJ en una tarea de programa y en otra instrucción de movimiento de otra tarea de programa, sólo es posible usar la búsqueda en vuelo con el modificador \Sup. Además de ello, sólo es posible realizar la recuperación en caso de error con TRYNEXT.

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.225 SearchExtJ - Busca con una o varias unidades mecánicas sin TCP

RobotWare - OS

Continuación

- Es posible utilizar todas las funciones de búsqueda si se utilizan instrucciones SearchL, SearchC o SearchExtJ en todas las tareas de programa implicadas con movimiento sincronizado y coordinado y se generan coincidencias de búsqueda desde la misma señal digital de entrada. Con ello se generan coincidencias de búsqueda de forma sincronizada en todas las instrucciones de búsqueda. Cualquier recuperación de error debe ser también la misma en todas las tareas de programa implicadas.
- Mientras la búsqueda está activa, no es posible almacenar la trayectoria actual con la instrucción StorePath.

### Sintaxis

```
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> ]';'
```

### Información relacionada

| Para obtener más información sobre       | Consulte  |
|--|---|
| Movimiento de unidades mecánicas sin TCP | <a href="#">MoveExtJ - Mueve una o varias unidades mecánicas sin TCP en la página 417</a> |
| Definición de jointtarget                | <a href="#">jointtarget - Datos de posición de eje en la página 1691</a>                  |
| Definición de velocidad                  | <a href="#">speeddata - Datos de velocidad en la página 1767</a>                          |
| Utilización de gestores de errores       | <i>Manual de referencia técnica - RAPID Overview</i>                                      |
| Movimiento en general                    | <i>Manual de referencia técnica - RAPID Overview</i>                                      |

---

## 1.226 SearchL - Realiza una búsqueda lineal usando el robot

---

### Utilización

`SearchL` (*Search Linear*) se utiliza para buscar una posición al mover el punto central de la herramienta (TCP) en sentido lineal.

Durante el movimiento, el robot supervisa una señal digital de entrada o una variable persistente. Cuando el valor de la señal o de la variable persistente cambia al valor solicitado, el robot lee inmediatamente la posición actual.

Normalmente, esta instrucción puede usarse cuando la herramienta sostenida por el robot es una sonda para detección de superficies. La instrucción `SearchL`, permite obtener las coordenadas de contorno de un objeto de trabajo.

Esta instrucción sólo puede usarse en la tarea principal `T_ROB1` o, si se cuenta con un sistema *MultiMove*, en las tareas de movimiento.

Al utilizar instrucciones de búsqueda, resulta importante configurar el sistema de E/S para reducir al mínimo el tiempo que transcurre desde el establecimiento de la señal física hasta que el sistema obtiene información acerca del valor (utilice el dispositivo de E/S con control de interrupciones, no con control de sondeo). La forma de hacerlo puede ser distinta de un bus de campo a otro. Si se utiliza *DeviceNet*, las unidades de ABB (E/S local) proporcionan tiempos breves, dado que utilizan el *cambio de estado* como tipo de conexión. Si se utilizan otros buses de campo, asegúrese de configurar la red correctamente para conseguir las condiciones correctas.

---

### Ejemplos básicos

Los ejemplos siguientes ilustran la instrucción `SearchL`.

Consulte también [Más ejemplos en la página 681](#).

#### Ejemplo 1

```
SearchL di1, sp, p10, v100, probe;
```

El TCP de la sonda `probe` se mueve linealmente hacia la posición `p10` a una velocidad de `v100`. Cuando el valor de la señal `di1` cambia a activo, la posición se almacena en `sp`.

#### Ejemplo 2

```
SearchL \Stop, di2, sp, p10, v100, probe;
```

El TCP de la sonda `probe` se mueve linealmente hacia la posición `p10`. Cuando el valor de la señal `di2` cambia a activo, la posición se almacena en `sp` y el robot se detiene inmediatamente.

#### Ejemplo 3

```
PERS bool mypers:=FALSE;  
...  
SearchL mypers, sp, p10, v100, probe;
```

El TCP del `probe` se mueve linealmente hacia la posición `p10` a una velocidad de `v100`. Cuando el valor de la variable persistente `mypers` cambia a `TRUE`, la posición se almacena en `sp`.

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.226 SearchL - Realiza una búsqueda lineal usando el robot

RobotWare - OS

Continuación

### Argumentos

```
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*

Tipo de dato: switch

El movimiento del robot se detiene lo antes posible, sin mantener el TCP en la trayectoria (paro rígido) cuando el valor de la señal de búsqueda cambia a activo o el valor de la variable persistente cambia a TRUE. El robot se mueve una distancia corta antes del paro y no regresa a la posición buscada, es decir, a la posición en la que cambió la señal o el valor de la variable persistente.

Pruebe primero con una velocidad baja, por ejemplo <100 mm/s, y a continuación aumente gradualmente la velocidad hasta el valor deseado.



#### ¡AVISO!

La detención de la búsqueda con un paro rígido (modificador \Stop) solo está permitida si la velocidad del TCP es inferior a 100 mm/s. En los paros rígidos a una mayor velocidad, algunos ejes pueden moverse en direcciones impredecibles.



#### Nota

Para un robot YuMi, la velocidad máxima para buscar con paro rígido es 1000 mm/s.

[ \PStop ]

#### *Path Stop*

Tipo de dato: switch

El movimiento del robot se detiene lo antes posible, manteniendo el TCP en la trayectoria (paro blando) cuando el valor de la señal de búsqueda cambia a activo o el valor de la variable persistente cambia a TRUE. El robot se mueve cierta distancia antes del paro y no regresa a la posición buscada, es decir, a la posición en la que cambió la señal o el valor de la variable persistente.

[ \SStop ]

#### *Soft Stop*

Tipo de dato: switch

El movimiento del robot se detiene lo antes posible, manteniendo el TCP cerca de la trayectoria o en ésta (paro blando) cuando el valor de la señal de búsqueda cambia a activo o el valor de la variable persistente cambia a TRUE. El robot solo se mueve una distancia corta antes del paro y no regresa a la posición buscada, es decir, a la posición en la que cambió la señal o la variable persistente. SStop resulta más rápido que PStop. Sin embargo, cuando el robot funciona a una velocidad superior a los 100 mm/s, se detiene en la dirección de la tangente del movimiento, lo que hace que se deslice marginalmente de la trayectoria.

Continúa en la página siguiente



## 1.226 SearchL - Realiza una búsqueda lineal usando el robot *RobotWare - OS* *Continuación*

[ \Sup ]

### *Supervision*

Tipo de dato: `switch`

La instrucción de búsqueda es sensible a la activación de señales o el cambio de valores de variables persistentes durante el movimiento completo (búsqueda en vuelo), es decir, incluso después de que se informa del primer cambio de señal o cambio de variable persistente. Si se produce más de una coincidencia durante una búsqueda, se genera un error recuperable con el robot en `ToPoint`.

Si se omiten los argumentos `\Stop`, `\PStop`, `\SStop` y `\Sup` (no se utiliza ningún modificador):

- el movimiento continúa (búsqueda en vuelo) hasta la posición especificada en el argumento `ToPoint` (igual que en el argumento `\Sup`)
- Se indica un error cuando no existe ninguna coincidencia de búsqueda, pero no se indica tal error si hay más de una coincidencia de búsqueda (la primera coincidencia de búsqueda se devuelve como `SearchPoint`)

Signal

Tipo de dato: `signal`

El nombre de la señal a supervisar.

PersBool

Tipo de dato: `bool`

La variable persistente que se desea supervisar.

[ \Flanks ]

Tipo de dato: `switch`

Los bordes positivo y negativo de la señal son válidos como coincidencias de búsqueda. Si se utiliza el argumento `PersBool`, el cambio de valor de la variable se considera válido como coincidencia de búsqueda.

Para la señal: si se omite el argumento `\Flanks`, solo el borde positivo de la señal es válido para determinar el éxito de una búsqueda y se activa una supervisión de la señal al comenzar un proceso de búsqueda. Esto significa que si la señal ya tiene el valor positivo al comenzar un proceso de búsqueda o se pierde la comunicación con la señal, el movimiento del robot se detiene lo antes posible, manteniendo el TCP en la trayectoria (paro blando). Se genera un error recuperable por el usuario `ERR_SIGSUPSEARCH`, que puede gestionarse en el gestor de errores.

Para la variable persistente: si se omite el argumento `\Flanks`, solo cuando el valor cambia a `TRUE` se determina el éxito de una búsqueda y se activa una supervisión de la variable al comenzar un proceso de búsqueda. Esto significa que si la variable persistente ya tiene el valor positivo al comenzar un proceso de búsqueda, el movimiento del robot se detiene lo antes posible, manteniendo el TCP en la trayectoria (paro blando). Se genera un error recuperable por el usuario `ERR_PERSSUPSEARCH`, que puede gestionarse en el gestor de errores.

[ \PosFlank ]

Tipo de dato: `switch`

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.226 SearchL - Realiza una búsqueda lineal usando el robot

RobotWare - OS

Continuación

El borde positivo de la señal es válido para determinar el éxito de una búsqueda, o bien el cambio del valor a TRUE si se utiliza una variable persistente.

[ \NegFlank ]

Tipo de dato: switch

El borde negativo de la señal es válido para determinar el éxito de una búsqueda, o bien el cambio del valor a FALSE si se utiliza una variable persistente.

[ \HighLevel ]

Tipo de dato: switch

La misma funcionalidad que si no usara el modificador \Flanks.

Para la señal: el borde positivo de la señal es válido para determinar el éxito de una búsqueda y se activa una supervisión de la señal al comenzar un proceso de búsqueda. Esto significa que si la señal ya tiene el valor positivo al comenzar un proceso de búsqueda o se pierde la comunicación con la señal, el movimiento del robot se detiene lo antes posible, manteniendo el TCP en la trayectoria (paro blando). Se genera un error recuperable por el usuario ERR\_SIGSUPSEARCH, que puede gestionarse en el gestor de errores.

Para la variable persistente: solo el cambio de valor a TRUE se considera un éxito de búsqueda y se activa una supervisión de la variable al comenzar un proceso de búsqueda. Esto significa que si la variable persistente ya tiene el valor positivo al comenzar un proceso de búsqueda, el movimiento del robot se detiene lo antes posible, manteniendo el TCP en la trayectoria (paro blando). Se genera un error recuperable por el usuario ERR\_PERSSUPSEARCH, que puede gestionarse en el gestor de errores.

[ \LowLevel ]

Tipo de dato: switch

Para la señal: el borde negativo de la señal es válido para determinar el éxito de una búsqueda y se activa una supervisión de la señal al comenzar un proceso de búsqueda. Esto significa que si la señal ya tiene el valor 0 al comenzar un proceso de búsqueda o se pierde la comunicación con la señal, el movimiento del robot se detiene lo antes posible, manteniendo el TCP en la trayectoria (paro blando). Se genera un error recuperable por el usuario ERR\_SIGSUPSEARCH, que puede gestionarse en el gestor de errores.

Para la variable persistente: solo el cambio de valor a FALSE se considera un éxito de búsqueda y se activa una supervisión de la variable al comenzar un proceso de búsqueda. Esto significa que si la variable persistente ya tiene el valor FALSE al comenzar un proceso de búsqueda, el movimiento del robot se detiene lo antes posible, manteniendo el TCP en la trayectoria (paro blando). Se genera un error recuperable por el usuario ERR\_PERSSUPSEARCH, que puede gestionarse en el gestor de errores.

SearchPoint

Tipo de dato: rotarget

*Continúa en la página siguiente*

## 1.226 SearchL - Realiza una búsqueda lineal usando el robot *RobotWare - OS* *Continuación*

La posición del TCP y de los ejes externos en el momento del disparo de la señal de búsqueda. La posición se especifica en el sistema de coordenadas más externo, ProgDisp/ExtOffs **activo**.

ToPoint

Tipo de dato: `robtarget`

El punto de destino de los ejes del robot y de los ejes externos. Se define como una posición con nombre o se almacena directamente en la instrucción (marcada con un asterisco \* en la instrucción). SearchL utiliza siempre un punto de paro como dato de zona del destino.

[ \ID ]

*Synchronization id*

Tipo de dato: `identno`

El argumento [ \ID ] es obligatorio en los sistemas MultiMove, si el movimiento es sincronizado o sincronizado coordinado. Este argumento no está permitido en ningún otro caso. El número de ID especificado debe ser el mismo en todas las tareas de programa que cooperan entre sí. Al usar el número de ID los movimientos no se mezclan en tiempo de ejecución.

Speed

Tipo de dato: `speeddata`

Los datos de velocidad que se aplican a los movimientos. Los datos de velocidad definen la velocidad del punto central de la herramienta, los ejes externos y la reorientación de la herramienta.

[ \V ]

*Velocity*

Tipo de dato: `num`

Este argumento se utiliza para especificar la velocidad del TCP en mm/s directamente en la instrucción. A continuación, se sustituye por la velocidad correspondiente, especificada en los datos de velocidad.

[ \T ]

*Time*

Tipo de dato: `num`

Este argumento se utiliza para especificar el tiempo total en segundos que dura el movimiento del robot. A continuación, se sustituye por los datos de velocidad correspondientes.

Tool

Tipo de dato: `tooldata`

La herramienta en uso durante el movimiento del robot. El punto central de la herramienta es el punto que se mueve hacia la posición de destino especificada.

[ \WObj ]

*Work Object*

Tipo de dato: `wobjdata`

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.226 SearchL - Realiza una búsqueda lineal usando el robot

RobotWare - OS

Continuación

El objeto de trabajo (sistema de coordenadas) con el que está relacionada la posición de robot indicada en la instrucción.

Es posible omitir este argumento. Si se omite, la posición depende del sistema de coordenadas mundo. Si por otro lado se usa un TCP estacionario o ejes externos coordinados, es necesario especificar este argumento para ejecutar un movimiento lineal respecto del objeto de trabajo.

[ \Corr ]

### *Correction*

Tipo de dato: `switch`

Los datos de corrección escritos en una entrada de corrección mediante una instrucción `CorrWrite` se añaden a la trayectoria y a la posición de destino si se utiliza este argumento.

Se requiere RobotWare, opción *Path Offset*, cuando se utiliza este argumento.

[ \TLoad ]

### *Total load*

Tipo de dato: `loaddata`

El argumento `\TLoad` describe la carga total usada durante el movimiento. La carga total es la carga de la herramienta más la carga útil transportada por la herramienta. Si se utiliza el argumento `\TLoad`, no se tiene en cuenta el valor de `loaddata` en los `tooldata` actuales.

Si el argumento `\TLoad` tiene el valor `load0`, el argumento `\TLoad` no se tiene en cuenta y se utilizan en su lugar los `loaddata` de los `tooldata`.

Para poder utilizar el argumento `\TLoad`, es necesario cambiar el valor del parámetro de sistema `ModalPayloadMode` a 0. Si `ModalPayloadMode` tiene el valor 0, ya no es posible utilizar la instrucción `GripLoad`.

La carga total puede identificarse con la rutina de servicio `LoadIdentify`. Si el parámetro de sistema `ModalPayloadMode` tiene el valor 0, el operador tiene la posibilidad de copiar los `loaddata` de la herramienta a una variable persistente `loaddata` existente o nueva al ejecutar la rutina de servicio.

Es posible realizar una ejecución de prueba del programa sin ninguna carga útil utilizando una señal de entrada digital conectada a la entrada de sistema `SimMode` (modo simulado). Si la señal de entrada digital tiene el valor 1, los `loaddata` del argumento opcional `\TLoad` no se tienen en cuenta y se utilizan en su lugar los `loaddata` de los `tooldata` actuales.



### **Nota**

La funcionalidad predeterminada de manejo de la carga útil es utilizar la instrucción `GripLoad`. Por tanto, el valor predeterminado del parámetro de sistema `ModalPayloadMode` es 1.

---

## Ejecución de programas

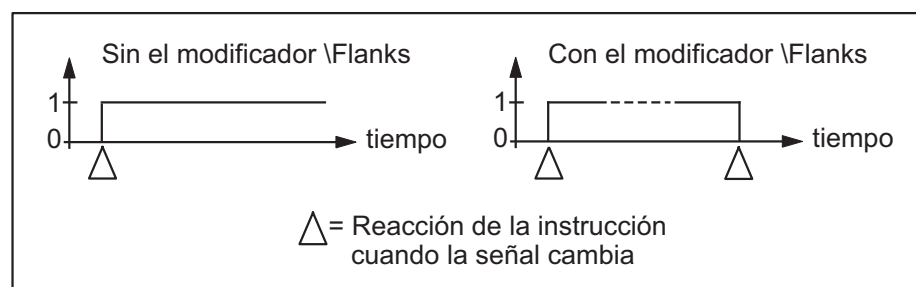
Consulte la instrucción `MoveL` para obtener más información acerca del movimiento lineal.

Continúa en la página siguiente

El movimiento termina siempre con un punto de paro, lo que implica que el robot se detiene en el punto de destino. Si se realiza una búsqueda en vuelo, es decir, cuando se utiliza el argumento `\Sup` o no se especifica ningún modificador, el movimiento del robot continúa siempre hacia el punto de destino programado. Si se realiza una búsqueda con el modificador `\Stop`, `\PStop` o `\SStop`, el movimiento del robot se detiene cuando se detecta la primera señal.

La instrucción `SearchL` almacena la posición del TCP cuando el valor de la señal digital o de la variable persistente cambia al valor solicitado, como se representa en la figura siguiente.

La figura muestra cómo se usa la detección de señal disparada por flancos (la posición se almacena sólo cuando la señal cambia por primera vez).



xx0500002243

## Más ejemplos

A continuación aparecen más ejemplos de cómo usar la instrucción `SearchL`.

### Ejemplo 1

```
SearchL \Sup, di1 \Flanks, sp, p10, v100, probe;
```

El TCP de la sonda `probe` se mueve linealmente hacia la posición `p10`. Cuando el valor de la señal `di1` cambia a activo o pasivo, la posición se almacena en `sp`. Si el valor de la señal cambia dos veces, el programa genera un error una vez que se completa el proceso de búsqueda.

### Ejemplo 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;
```

Al comienzo del proceso de búsqueda, se realiza una comprobación de la señal `di1` y si la señal ya tiene un valor positivo o se pierde la comunicación con el robot, el robot se detiene. De lo contrario, el TCP de `tool1` se mueve linealmente hacia la posición `p10`. Cuando el valor de la señal `di1` cambia a activo, la posición se almacena en `sp`. El robot vuelve a este punto usando un punto de paro definido exactamente. Usando un desplazamiento de programa, el robot se mueve a continuación respecto de la posición buscada, `sp`.

Continúa en la página siguiente

# 1 Instrucciones

## 1.226 SearchL - Realiza una búsqueda lineal usando el robot

RobotWare - OS

Continuación

### Ejemplo 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;
```

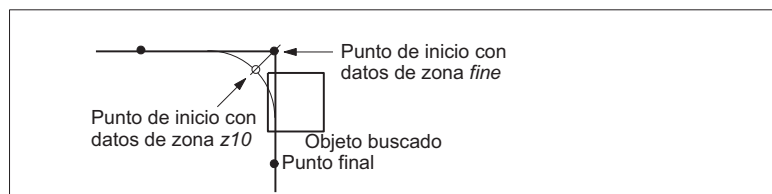
Al comienzo del proceso de búsqueda, se realiza una comprobación de la variable persistente `MyTrigger` y si la variable ya tiene el valor `TRUE`, el robot se detiene. De lo contrario, el TCP de `tool1` se mueve linealmente hacia la posición `p10`. Cuando el valor de la variable persistente `MyTrigger` cambia a `TRUE`, la posición se almacena en `sp`. El robot vuelve a este punto usando un punto de paro definido exactamente. Usando un desplazamiento de programa, el robot se mueve a continuación respecto de la posición buscada, `sp`.

### Limitaciones

`SearchL` no puede ejecutarse en un gestor UNDO o en rutinas de RAPID que estén conectadas a los siguientes eventos especiales del sistema: `PowerOn`, `Stop`, `QStop`, `Restart`, `Reset` o `Step`.

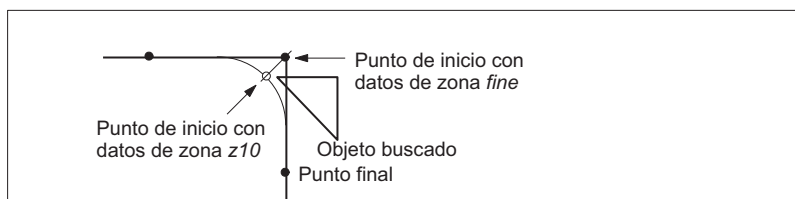
Los datos de zona de la instrucción de posicionamiento que precede a `SearchL` deben usarse con cuidado. El comienzo de la búsqueda, es decir, cuando la señal de E/S está preparada para reaccionar, no es en este caso el punto de destino de la instrucción de posicionamiento anterior, sino un punto que se encuentra en la trayectoria real del robot. En las figuras siguientes se muestran ejemplos de cosas que pueden salir mal cuando se usan datos de zona distintos de `fine`.

En la figura siguiente se muestra cómo se encuentra una posición en el lado incorrecto del objeto a causa del uso de datos de zona incorrectos.



xx0500002244

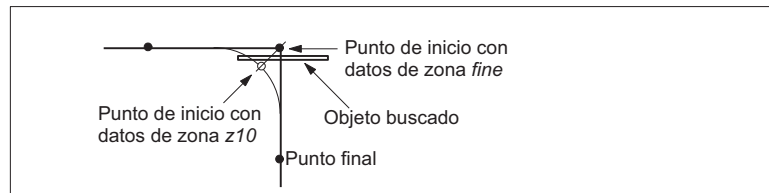
La figura siguiente muestra que no se ha detectado ningún éxito de búsqueda porque se usaron datos de zona erróneos.



xx0500002245

Continúa en la página siguiente

La figura siguiente muestra que no se ha detectado ningún éxito de búsqueda porque se usaron datos de zona erróneos.



xx0500002246

Limitaciones de búsqueda con movimiento sincronizado y coordinado:

- Si se utiliza `SearchL`, `SearchC` o `SearchExtJ` en una tarea de programa y en otra instrucción de movimiento de otra tarea de programa, sólo es posible usar la búsqueda en vuelo con el modificador `\Sup`. Además de ello, sólo es posible realizar la recuperación en caso de error con `TRYNEXT`.
- Es posible utilizar toda la funcionalidad de búsqueda si se utilizan instrucciones `SearchL`, `SearchC` o `SearchExtJ` en todas las tareas de programa implicadas con movimiento sincronizado y coordinado y se genera una coincidencia de búsqueda desde la misma señal digital de entrada. Con ello se genera una coincidencia de búsqueda de forma sincronizada en todas las instrucciones de búsqueda. Cualquier recuperación de error debe ser también la misma en todas las tareas de programa implicadas.

Mientras la búsqueda está activa, no se permite almacenar la trayectoria actual con la instrucción `StorePath`.

Exactitud de repetición para la posición de coincidencia de búsqueda con una velocidad de 20 a 1.000 mm/s de 0,1 a 0,3 mm.

Distancia de paro típica con una velocidad de búsqueda de 50 mm/s:

- Sin el TCP en la trayectoria (modificador `\Stop`), de 1 a 3 mm
- Con el TCP en la trayectoria (modificador `\PStop`), de 15 a 25 mm
- Con el TCP cerca de la trayectoria (modificador `\SStop`), de 4 a 8 mm

Limitaciones de la búsqueda en un transportador:

- la búsqueda detiene el robot en caso de coincidencia o si la búsqueda no tiene éxito, de forma que la búsqueda debe hacerse en el mismo sentido que el del movimiento del transportador y después del paro de búsqueda debe ir seguida de un movimiento hasta una posición segura. Utilice la gestión de errores para el movimiento hasta una posición segura si la búsqueda no tiene éxito.
- la exactitud de repetición de la posición de coincidencia de búsqueda será menor al buscar en un transportador y depende de la velocidad del transportador y de hasta qué punto esta velocidad es estable.

Continúa en la página siguiente

# 1 Instrucciones

## 1.226 SearchL - Realiza una búsqueda lineal usando el robot

RobotWare - OS

Continuación

### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                          | Causa del error  |
|---------------------------------|--|
| <code>ERR_NO_ALIASIO_DEF</code> | La variable de señal es una variable declarada en RAPID. No se ha conectado a una señal E/S definida en la configuración de E/S con la instrucción <code>AliasIO</code> .  |
| <code>ERR_WHLSEARCH</code>      | <ul style="list-style-type: none"><li>No se produjo ninguna detección de señal.</li><li>Se ha producido la detección de más de una señal. Esto solo se produce si se utiliza el argumento <code>\Sup</code>.</li></ul> |
| <code>ERR_SIGSUPSEARCH</code>   | La señal ya tiene un valor positivo al comienzo del proceso de búsqueda o se ha perdido la comunicación con la señal. Esto solo se produce si se omite el argumento <code>\Flanks</code> .                             |
| <code>ERR_PERSSUPSEARCH</code>  | La variable persistente ya es <code>TRUE</code> al comienzo del proceso de búsqueda. Esto solo se produce si se omite el argumento <code>\Flanks</code> .  |

Los errores pueden gestionarse de formas distintas en función del modo de ejecución seleccionado:

- Ejecución continua hacia delante / Instrucciones hacia adelante**  
`/ERR_WHLSEARCH`: No se devuelve ninguna posición y el movimiento continúa siempre hacia el punto de destino programado. La variable de sistema `ERRNO` cambia a `ERR_WHLSEARCH` y es posible gestionar el error en el gestor de errores de la rutina.
- Ejecución continua hacia adelante / Instrucciones hacia adelante /**  
`ERR_SIGSUPSEARCH` y `ERR_PERSSUPSEARCH`: No se devuelve ninguna posición y el movimiento se detiene siempre lo antes posible al principio de la trayectoria de búsqueda. La variable de sistema `ERRNO` cambia a `ERR_SIGSUPSEARCH` o `ERR_PERSSUPSEARCH` en función del argumento utilizado (señal o variable persistente) y es posible gestionar el error en el gestor de errores de la rutina.
- Instrucción hacia atrás**: Durante la ejecución hacia atrás, la instrucción realiza el movimiento pero no realiza ninguna supervisión.

### Ejemplo

```
VAR num fk;  
...  
MoveL p10, v100, fine, tool1;  
SearchL \Stop, dil, sp, p20, v100, tool1;  
...  
ERROR  
  IF ERRNO=ERR_WHLSEARCH THEN  
    StorePath;  
    MoveL p10, v100, fine, tool1;  
    RestoPath;  
    ClearPath;  
    StartMove;  
    RETRY;
```

Continúa en la página siguiente



```

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
ENDIF

```

Si la señal ya está activa al principio del proceso de búsqueda o se pierde la comunicación con la señal, se activa un cuadro de diálogo (TPReadFK ...). Si se pone a cero manualmente la señal y se presiona SÍ en la ventana de diálogo del usuario, el robot vuelve a la posición p10 y vuelve a intentar la operación. De lo contrario, la ejecución del programa se detiene.

Si la señal está desactivada al comienzo del proceso de búsqueda, el robot busca desde la posición p10 hasta la posición p20. Si no se detecta ninguna señal, el robot vuelve a la posición p10 y lo intenta de nuevo.

## Sintaxis

```

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 robtarger> ','
[ToPoint ':=' ] <expression (IN) of robtarger>
['\ ' 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> ] ;'

```

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.226 SearchL - Realiza una búsqueda lineal usando el robot

RobotWare - OS

Continuación

### Información relacionada

| Para obtener más información sobre   | Consulte   |
|--|--|
| Búsquedas circulares   | <a href="#">SearchC - Realiza una búsqueda en círculo usando el robot en la página 656</a> |
| Escritura en una entrada de corrección   | <a href="#">CorrWrite - Escribe en un generador de correcciones en la página 183</a>       |
| Movimiento lineal del robot  | <a href="#">MoveL - Mueve el robot siguiendo una trayectoria lineal en la página 447</a>   |
| Movimiento lineal  | <i>Manual de referencia técnica - RAPID Overview</i>                                       |
| Definición de carga  | <a href="#">loaddata - Datos de carga en la página 1694</a>                                |
| Definición de velocidad  | <a href="#">speeddata - Datos de velocidad en la página 1767</a>                           |
| Definición de herramientas   | <a href="#">tooldata - Datos de herramienta en la página 1795</a>                          |
| Definición de objetos de trabajo   | <a href="#">wobjdata - Datos del objeto de trabajo en la página 1823</a>                   |
| Utilización de gestores de errores   | <i>Manual de referencia técnica - RAPID Overview</i>                                       |
| Movimiento en general  | <i>Manual de referencia técnica - RAPID Overview</i>                                       |
| Ejemplo de cómo usar TLoad, carga total.   | <a href="#">MoveL - Mueve el robot siguiendo una trayectoria lineal en la página 447</a>   |
| Definición de la carga útil de un robot  | <a href="#">GripLoad - Define la carga útil de un robot en la página 244</a>               |
| LoadIdentify, rutina de servicio de identificación de carga                                    | <i>Manual del operador - IRC5 con FlexPendant</i>  |
| Señal de entrada de sistema Sim-Mode para mover el robot en el modo simulado sin carga útil.   | <i>Manual de referencia técnica - Parámetros del sistema</i>                               |
| Parámetro de sistema ModalPay-LoadMode para la activación y la desactivación de la carga útil. | <i>Manual de referencia técnica - Parámetros del sistema</i>                               |
| <i>Path Offset</i>   | <i>Application manual - Controller software IRC5</i>                                       |

## 1.227 SenDevice - Establece una conexión a un dispositivo de sensor

### Utilización

SenDevice se utiliza para conectarse a un dispositivo de sensor conectado a la interfaz de sensores.

La interfaz de sensores se comunica con los sensores a través de dispositivos de E/S.

### Ejemplo de configuración

Éste es un ejemplo de configuración de un canal de sensor.

Estos parámetros corresponden al tipo *Transmission Protocol* del tema *Communication*.

| Name  | Type    | Remote Address  | Remote Port |
|-------|---------|-----------------|-------------|
| sen1: | SOCKDEV | 192.168.125.101 | 6344        |

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción SenDevice:

#### Ejemplo 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 measurements
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;

```

### Argumentos

SenDevice device

device

**Tipo de dato:** string

El nombre del dispositivo de E/S configurado en sio.cfg para el sensor utilizado.

### Sintaxis

```

ReadBlock
[device ':='] <expression(IN) of string>', '
[BlockNo ':='] <expression (IN) of num>', '
[FileName ':='] <expression (IN) of string>';'

```

*Continúa en la página siguiente*

# 1 Instrucciones

---

1.227 SenDevice - Establece una conexión a un dispositivo de sensor

*Sensor Interface*

*Continuación*

---

## Información relacionada

| Para obtener más información sobre          | Consulte   |
|---|--|
| Escritura de una variable de sensor         | <a href="#">WriteVar - Escribir una variable en la página 1116</a>                           |
| Lectura de una variable de sensor           | <a href="#">ReadVar - Lee una variable de un dispositivo en la página 1451</a>               |
| Escritura de un bloque de datos de sensor   | <a href="#">WriteBlock - Escribir un bloque de datos en un dispositivo en la página 1105</a> |
| Configuración de la comunicación del sensor | <a href="#">Manual de referencia técnica - Parámetros del sistema</a>                        |

## 1.228 Set - Activa una señal digital de salida

### Utilización

Set se utiliza para cambiar a 1 una señal digital de salida.

### Ejemplos básicos

Los ejemplos siguientes ilustran la instrucción Set.

#### Ejemplo 1

```
Set do15;
```

Se cambia la señal do15 a 1.

#### Ejemplo 2

```
Set weldon;
```

Se cambia la señal weldon a 1.

### Argumentos

```
Set Signal
```

Signal

Tipo de dato: signaldo

El nombre de la señal que se desea cambiar a uno.

### Ejecución de programas

Existe un pequeño retardo antes de que la señal cambie físicamente al nuevo valor. Si no desea que continúe la ejecución del programa hasta que la señal tenga su nuevo valor, puede usar la instrucción SetDO con el parámetro opcional \Sync.

El valor real depende de la configuración de la señal. Si la señal está invertida en los parámetros del sistema, esta instrucción hace que el canal físico cambie a cero.

### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema ERRNO cambiará a:

| Nombre             | Causa del error   |
|--------------------|---|
| ERR_NO_ALIASIO_DEF | La variable de señal es una variable declarada en RAPID. No se ha conectado a una señal E/S definida en la configuración de E/S con la instrucción AliasIO.   |
| ERR_NORUNUNIT      | Se ha perdido el contacto con el dispositivo de E/S.  |
| ERR_SIG_NOT_VALID  | La señal de E/S no está disponible. Los motivos pueden ser que el dispositivo de E/S no está en funcionamiento o que exista un error en la configuración (válido solamente para el bus de campo ICI). |

### Sintaxis

```
Set  
[ Signal '[:=' ] < variable (VAR) of signaldo > ':]
```

*Continúa en la página siguiente*

# 1 Instrucciones

---

1.228 Set - Activa una señal digital de salida

RobotWare - OS

Continuación

---

## Información relacionada

| Para obtener más información sobre              | Consulte  |
|---|---|
| Cambio de una señal digital de salida a cero    | <a href="#">Reset - Pone a cero una señal digital de salida en la página 606</a>        |
| Cambio del valor de una señal digital de salida | <a href="#">SetDO - Cambia el valor de una señal digital de salida en la página 704</a> |
| Instrucciones de entrada/salida                 | Manual de referencia técnica - <i>RAPID Overview</i>                                    |
| Funcionalidad de entrada/salida en general      | Manual de referencia técnica - <i>RAPID Overview</i>                                    |
| Configuración de E/S                            | Manual de referencia técnica - <i>Parámetros del sistema</i>                            |

## 1.229 SetAllDataVal - Establece un valor en todos los objetos de datos de un conjunto definido

*RobotWare - OS*

### 1.229 SetAllDataVal - Establece un valor en todos los objetos de datos de un conjunto definido

#### Utilización

`SetAllDataVal` (*Set All Data Value* permite establecer un nuevo valor en todos los objetos de datos de un tipo determinado y que coincidan con la gramática determinada.

#### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción `SetAllDataVal`:

```
VAR mydata mydata0:=0;
...
SetAllDataVal "mydata"\TypeMod:="mytypes"\Hidden,mydata0;
```

Esto establecerá todos los objetos de datos del tipo de datos `mydata` en el sistema en el mismo valor que la variable `mydata0` tenga (en el ejemplo en 0). El tipo de datos definido por el usuario `mydata` se define en el módulo `mytypes`.

#### Argumentos

```
SetAllDataVal Type [\TypeMod] [\Object] [\Hidden] Value
```

Type

Tipo de dato: `string`

El nombre del tipo de objetos de datos cuyo valor se desea establecer.

[ \TypeMod ]

*Type Module*

Tipo de dato: `string`

El nombre del módulo en el que está definido el tipo de dato, si se utilizan tipos de datos definidos por el usuario.

El argumento `TypeMode` no puede utilizarse para datos en módulos instalados como `-Shared` o `-Installed`. El nombre de módulo no está disponible para estos datos.

Consulte *Manual de referencia técnica - Parámetros del sistema*, tema *Controller*, tipo *Automatic Loading of Modules*.

[ \Object ]

Tipo de dato: `string`

El comportamiento predeterminado es cambiar el valor de todos los objetos de datos del tipo de datos indicado anteriormente, pero esta opción permite indicar el nombre de uno o varios objetos con una expresión regular. Consulte también la instrucción `SetDataSearch`.

[ \Hidden ]

Tipo de dato: `switch`

También afecta a los objetos de datos que se encuentren en las rutinas (datos o parámetros de rutina) que queden ocultas por otras rutinas en la cadena de llamadas.

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.229 SetAllDataVal - Establece un valor en todos los objetos de datos de un conjunto definido

RobotWare - OS

Continuación

Value

Tipo de dato: anytype

Este argumento contiene el nuevo valor que se desea establecer. El tipo de datos debe ser el mismo que el tipo de datos del objeto cuyo valor se desea establecer.

### Ejecución de programas

La instrucción fallará si la especificación de `Type` o `TypeMod` es incorrecta.

Si el objeto de datos coincidente es una matriz, todos los elementos de la matriz se cambian al valor especificado.

Si el objeto de datos coincidente es un dato de sólo lectura, el valor no se cambia.

Si el sistema no tiene ningún objeto de dato coincidente, la instrucción lo aceptará y retornará sin generar errores.

### Limitaciones

Para un tipo de datos de semivalor, no es posible buscar el tipo de datos de valor asociado. Por ejemplo, si se busca `dionum` no hay ningún resultado de la búsqueda para la señal `signaldi` y si se busca `num` no hay ningún resultado de la búsqueda para las señales `signalgi` o `signalai`.

No es posible asignar un valor a una variable declarada como `LOCAL` en un módulo de RAPID incorporado.

### Gestión de errores

Pueden generarse los errores recuperables enumerados a continuación. Estos errores pueden ser gestionados en un gestor de errores. El valor de la variable de sistema `ERRNO` cambia a:

| Nombre                       | Causa del error   |
|------------------------------|---|
| <code>ERR_SYMBOL_TYPE</code> | El objeto de datos y la variable utilizados en el argumento <code>Value</code> son de tipos diferentes. Si se utilizan los tipos de datos <code>ALIAS</code> , también se producirá un <code>ERROR</code> , incluso si los tipos pudieran tener el mismo tipo de datos básicos. |

### Sintaxis

```
SetAllDataVal
  [Type ':=' ] <expression (IN) of string>
  ['\TypeMod ':='<expression (IN) of string>]
  ['\Object ':='<expression (IN) of string>]
  ['\Hidden ] ','
  [Value ':=' ] <variable (VAR) of anytype>';'
```

### Información relacionada

| Para obtener más información sobre                              | Consulte  |
|---|---|
| Definición de un conjunto de símbolos en una sesión de búsqueda | <a href="#">SetDataSearch - Definir el conjunto de símbolos de una secuencia de búsqueda en la página 696</a> |
| Obtención del siguiente símbolo coincidente                     | <a href="#">GetNextSym - Obtiene el siguiente símbolo coincidente en la página 1309</a>                       |

Continúa en la página siguiente



## 1.229 SetAllDataVal - Establece un valor en todos los objetos de datos de un conjunto definido

RobotWare - OS

Continuación

| Para obtener más información sobre         | Consulte   |
|--|--|
| Obtención del valor de un objeto de datos  | <a href="#">GetDataVal - Obtiene el valor de un objeto de datos en la página 228</a>       |
| Asignación del valor de un objeto de datos | <a href="#">SetDataVal - Establece el valor de un objeto de datos en la página 701</a>     |
| El tipo de datos relacionado data-pos      | <a href="#">datapos - Inclusión de un bloque para un objeto de datos en la página 1657</a> |
| <i>Advanced RAPID</i>                      | <i>Application manual - Controller software IRC5</i>                                       |

## 1 Instrucciones

### 1.230 SetAO - Cambia el valor de una señal analógica de salida

RobotWare - OS

### 1.230 SetAO - Cambia el valor de una señal analógica de salida

#### Utilización

SetAO se utiliza para cambiar el valor de una señal analógica de salida.

#### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción SetAO:

Consulte también [Más ejemplos en la página 695](#).

#### Ejemplo 1

```
SetAO ao2, 5.5;
```

Se cambia la señal ao2 a 5.5.

#### Argumentos

```
SetAO Signal Value
```

Signal

Tipo de dato: signalao

El nombre de la señal analógica de salida que debe cambiar de valor.

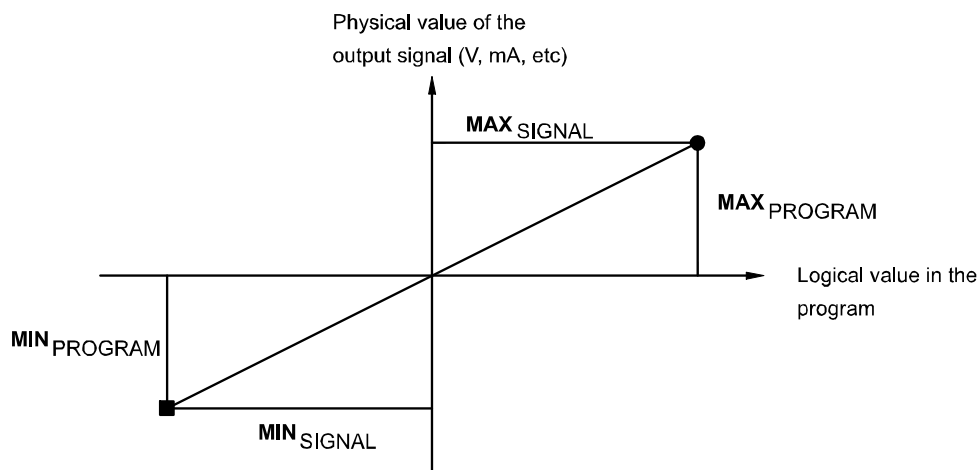
Value

Tipo de dato: num

El valor deseado para la señal.

#### Ejecución de programas

Al valor programado se le aplica una escala (acorde con los parámetros del sistema) antes de enviarlo al canal físico. En la figura siguiente se muestra un diagrama de cómo se ajustan los valores de las señales analógicas a una escala.



xx0500002408

Continúa en la página siguiente

### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                          | Causa del error   |
|---------------------------------|---|
| <code>ERR_AO_LIM</code>         | El argumento programado <code>Value</code> para la señal analógica de salida especificada <code>Signal</code> está fuera de límites.  |
| <code>ERR_NO_ALIASIO_DEF</code> | La variable de señal es una variable declarada en <code>RAPID</code> . No se ha conectado a una señal E/S definida en la configuración de E/S con la instrucción <code>AliasIO</code> .               |
| <code>ERR_NORUNUNIT</code>      | Se ha perdido el contacto con el dispositivo de E/S.  |
| <code>ERR_SIG_NOT_VALID</code>  | La señal de E/S no está disponible. Los motivos pueden ser que el dispositivo de E/S no está en funcionamiento o que exista un error en la configuración (válido solamente para el bus de campo ICI). |

### Más ejemplos

A continuación aparecen más ejemplos de la instrucción `SetAO`.

#### Ejemplo 1

```
SetAO weldcurr, curr_outp;
```

Se asigna a la señal `weldcurr` el mismo valor que el valor actual de la variable `curr_outp`.

### Sintaxis

```
SetAO
  [ Signal ::= ' ] < variable (VAR) of signalao > ','
  [ Value ::= ' ] < expression (IN) of num > ';'

```

### Información relacionada

| Para obtener más información sobre         | Consulte   |
|--|--|
| Instrucciones de entrada/salida            | <i>Manual de referencia técnica - RAPID Overview</i>         |
| Funcionalidad de entrada/salida en general | <i>Manual de referencia técnica - RAPID Overview</i>         |
| Configuración de E/S                       | <i>Manual de referencia técnica - Parámetros del sistema</i> |

## 1 Instrucciones

---

### 1.231 SetDataSearch - Definir el conjunto de símbolos de una secuencia de búsqueda RobotWare - OS

### 1.231 SetDataSearch - Definir el conjunto de símbolos de una secuencia de búsqueda

---

#### Utilización

SetDataSearch se utiliza junto con la función GetNextSym para obtener objetos de datos del sistema.

---

#### Ejemplos básicos

Los siguientes ejemplos ilustran la función SetDataSearch:

##### Ejemplo 1

```
VAR datapos block;  
VAR string name;  
...  
SetDataSearch "robtargt"\InTask;  
WHILE GetNextSym(name,block \Recursive) DO  
...  

```

Esta sesión encontrará todos los objetos robtargt de la tarea.

##### Ejemplo 2

```
RECORD testrecord  
  num value1;  
  num value2;  
ENDRECORD  
  
VAR datapos block;  
VAR string name;  
VAR testrecord mydata1:= [1,2];  
VAR testrecord mydata2:= [3,4];  
...  
SetDataSearch "testrecord" \TypeMod:="MYMODULE" \InTask;  
WHILE GetNextSym(name,block \Recursive) DO  
...  

```

El tipo de datos testrecord es un tipo de datos definido por el usuario, y definido en el módulo con el nombre MYMODULE. Esta sesión encontrará todos los objetos testrecord de la tarea.

---

#### Argumentos

```
SetDataSearch Type [\TypeMod] [\Object] [\PersSym]  
  [\VarSym][\ConstSym] [\InTask] | [\InMod]  
  [\InRout][\GlobalSym] | [\LocalSym]
```

Type

Tipo de dato: string

El nombre del tipo de dato de los objetos de datos que se desea obtener.

[ \TypeMod ]

*Type Module*

Tipo de dato: string

---

Continúa en la página siguiente

## 1.231 SetDataSearch - Definir el conjunto de símbolos de una secuencia de búsqueda

*RobotWare - OS*  
*Continuación*

El nombre del módulo en el que está definido el tipo de dato, si se utilizan tipos de datos definidos por el usuario.

El argumento `TypeMode` no puede utilizarse para datos en módulos instalados como `-Shared` o `-Installed`. El nombre de módulo no está disponible para estos datos.

Consulte *Manual de referencia técnica - Parámetros del sistema*, tema *Controller*, tipo *Automatic Loading of Modules*.

[ \Object ]]

Tipo de dato: `string`

El comportamiento predeterminado es cambiar el valor de todos los objetos de datos del tipo de datos indicado anteriormente, pero esta opción permite indicar el nombre de uno o varios objetos con una expresión regular.

Las expresiones regulares constituyen un potente mecanismo que permite especificar una expresión gramática de coincidencia con los nombres de los objetos de datos. La cadena puede componerse de caracteres normales y metacaracteres. Los metacaracteres son operadores especiales utilizados para representar a uno o varios caracteres normales en la cadena, con el fin de ampliar la búsqueda. Es posible comprobar si una cadena coincide totalmente con un patrón especificado o buscar dentro de una cadena para encontrar una subcadena que coincide con un patrón especificado.

Dentro de una expresión regular, todos los caracteres alfanuméricos coinciden consigo mismos. Esto significa que el patrón "abc" sólo coincidirá con un objeto de datos llamado "abc". Para encontrar todos los nombres de objetos de dato que contienen la secuencia de caracteres "abc", es necesario añadir algunos metacaracteres. La expresión regular para hacerlo es `".*abc.*"`.

El conjunto de metacaracteres disponibles se muestra a continuación.

| Expresión | Significado  |
|-----------|--|
| .         | Cualquier carácter individual.   |
| [s]       | Cualquier carácter individual del conjunto no vacío s, donde s es una secuencia de caracteres. Los rangos pueden especificarse como c-c. |
| [^s]      | Cualquier carácter individual no perteneciente al conjunto s.  |
| r*        | Cero o más casos de la expresión regular r.  |
| r+        | Uno varios casos de la expresión regular r.  |
| r?        | Cero o un caso de la expresión regular r.  |
| (r)       | La expresión regular r. Se utiliza para separar una expresión regular de otra.   |
| r   r'    | Las expresiones regulares r o r'.  |
| .*        | Cualquier secuencia de caracteres (cero, uno o varios caracteres).   |

El comportamiento predeterminado es aceptar todos los símbolos, pero si se especifica uno o varios de los modificadores `PersSym`, `VarSym`, o `ConstSym` sólo se aceptan los símbolos que coincidan con la especificación.

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.231 SetDataSearch - Definir el conjunto de símbolos de una secuencia de búsqueda

RobotWare - OS

Continuación

[ \PersSym ]

### *Persistent Symbols*

Tipo de dato: `switch`

Acepta símbolos de variables persistentes (`PERS`).

[ \VarSym ]

### *Variable Symbols*

Tipo de dato: `switch`

Acepta símbolos de variables (`VAR`).

[ \ConstSym ]

### *Constant Symbols*

Tipo de dato: `switch`

Acepta símbolos de constantes (`CONST`).

Si no se especifica ninguno de los modificadores `\InTask` o `\InMod` la búsqueda comienza en el nivel de sistema. El nivel de sistema es la raíz de todas las demás definiciones de símbolos del árbol de símbolos. En el nivel de sistema se encuentran todos los símbolos incorporados, junto con el manejador del nivel de tareas. En el nivel de tarea se encuentran todos los símbolos globales cargados, junto con el manejador del nivel de módulos.

Si se indica el modificador `\Recursive` en `GetNextSym`, la sesión de búsqueda incluirá todos los módulos y rutinas situadas debajo del nivel del sistema.

[ \InTask ]

### *In Task*

Tipo de dato: `switch`

Inicia la búsqueda en el nivel de tarea. En el nivel de tarea se encuentran todos los símbolos globales cargados, junto con el manejador del nivel de módulos.

Si se indica el modificador `\Recursive` en `GetNextSym`, la sesión de búsqueda incluirá todos los módulos y rutinas situadas debajo del nivel de tarea.

[ \InMod ]

### *In Module*

Tipo de dato: `string`

Inicia la búsqueda en el nivel de módulo especificado. En el nivel de módulo se encuentran todos los símbolos globales y locales cargados y declarados en el módulo especificado, junto con el manejador del nivel de rutinas.

Si se indica el modificador `\Recursive` en `GetNextSym`, la sesión de búsqueda incluirá todas las rutinas cargadas situadas debajo del nivel de módulo especificado (es decir, declaradas en el módulo especificado).

[ \InRout ]

### *In Routine*

Tipo de dato: `string`

Sólo busca en el nivel de rutina especificado.

*Continúa en la página siguiente*

## 1.231 SetDataSearch - Definir el conjunto de símbolos de una secuencia de búsqueda

*RobotWare - OS*  
*Continuación*

El nombre del módulo en el que se encuentra la rutina debe especificarse en el argumento `\InMod`.

El comportamiento predeterminado es encontrar tanto los símbolos de módulo locales como los globales, pero si se especifica uno de los modificadores `\GlobalSym` o `\LocalSym` sólo se aceptan los símbolos que coincidan con la especificación.

[ `\GlobalSym` ]

### *Global Symbols*

Tipo de dato: `switch`

Omite los símbolos locales del módulo.

[ `\LocalSym` ]

### *Local Symbols*

Tipo de dato: `switch`

Omite los símbolos globales del módulo.

---

## Ejecución de programas

La instrucción fallará si la especificación de uno de `Type`, `TypeMod`, `InMod` o `InRout` es incorrecta.

Si el sistema no tiene ningún objeto de dato coincidente, la instrucción lo aceptará y retornará sin generar errores, pero la primera instrucción `GetNextSym` retornará `FALSE`.

---

## Limitaciones

Los objetos de datos de matriz no pueden definirse en el conjunto de búsqueda de símbolos y no pueden encontrarse en una secuencia de búsqueda.

Para un tipo de datos de semivalor, no es posible buscar el tipo de datos de valor asociado. Por ejemplo, si se busca `dionum` no hay ningún resultado de la búsqueda para la señal `signaldi` y si se busca `num` no hay ningún resultado de la búsqueda para las señales `signalgi` o `signalai`.

Los símbolos incorporados instalados que hayan sido declarados como `LOCAL` no se encontrarán en ningún caso, independientemente del uso del argumento `\GlobalSym`, `\LocalSym` o de ninguno de ellos.

Los símbolos incorporados instalados que hayan sido declarados como globales o como `TASK` siempre se encontrarán, independientemente del uso del argumento `\GlobalSym`, `\LocalSym` o de ninguno de ellos.

No es posible usar `SetDataSearch` para buscar datos de algún tipo de dato de `ALIAS` definido con código de `RAPID`. No hay ninguna limitación para el tipo de dato de `ALIAS` predefinido.

---

## Sintaxis

```
SetDataSearch  
  [Type ':='] <expression (IN) of string>  
  ['\TypeMod ':=']<expression (IN) of string>  
  ['\Object ':=']<expression (IN) of string>
```

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.231 SetDataSearch - Definir el conjunto de símbolos de una secuencia de búsqueda

RobotWare - OS

Continuación

```
['\'PersSym]  
['\'VarSym]  
['\'ConstSym]  
['\'InTask]  
|['\'InMod' := '<expression (IN) of string>]  
['\'InRout ' := '<expression (IN) of string>]  
['\'GlobalSym]  
|['\'LocalSym]';'
```

### Información relacionada

| Para obtener más información sobre                | Consulte  |
|---|---|
| Obtención del siguiente símbolo coincidente       | <a href="#">GetNextSym - Obtiene el siguiente símbolo coincidente en la página 1309</a>                                   |
| Obtención del valor de un objeto de datos         | <a href="#">GetDataVal - Obtiene el valor de un objeto de datos en la página 228</a>                                      |
| Asignación del valor de varios objetos de datos   | <a href="#">SetAllDataVal - Establece un valor en todos los objetos de datos de un conjunto definido en la página 691</a> |
| El tipo de datos relacionado <code>datapos</code> | <a href="#">datapos - Inclusión de un bloque para un objeto de datos en la página 1657</a>                                |
| <i>Advanced RAPID</i>                             | <i>Application manual - Controller software IRC5</i>  |



## 1.232 SetDataVal - Establece el valor de un objeto de datos

### Utilización

SetDataVal (*Set Data Value*) permite establecer el valor de un objeto de datos que se especifica mediante una variable de cadena de caracteres.

### Ejemplos básicos

Los ejemplos siguientes ilustran la instrucción SetDataVal.

#### Ejemplo 1

```
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
```

Esta sesión cambia a TRUE todos los objetos locales de tipo bool cuyo nombre comience con my en el módulo mymod.

#### Ejemplo 2

```
VAR string StringArrVar_copy{2};
...
StringArrVar_copy{1} := "test1";
StringArrVar_copy{2} := "test2";
SetDataVal "StringArrVar", StringArrVar_copy;
```

Esta sesión definirá la matriz StringArrVar para que contenga dos cadenas, test1 y test2.

### Argumentos

```
SetDataVal Object [\Block][\TaskRef][\TaskName] Value
```

Object

Tipo de dato: string

El nombre del objeto de datos.

[ \Block ]

Tipo de dato: datapos

El bloque que contiene el objeto de datos. Sólo es posible realizar la obtención con la función GetNextSym.

Si se omite el argumento, se establece el valor del objeto de datos visible en el ámbito de ejecución actual del programa.

[\TaskRef]

*Task Reference*

Tipo de dato: taskid

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.232 SetDataVal - Establece el valor de un objeto de datos

RobotWare - OS

Continuación

La identidad de tarea de programa en la que se buscará el objeto de datos especificado. Con ayuda de este argumento, puede buscar declaraciones PERS o TASKPERS en otras tareas. Cualquier otra declaración dará lugar a un error.

Existen variables predefinidas con el tipo de dato taskid para todas las tareas de programa del sistema. La identificación de la variable será "nombre\_tarea"+ "ID". Por ejemplo, para la tarea T\_ROB1 la identificación de la tarea es T\_ROB1Id.

[ \TaskName ]

Tipo de dato: string

El nombre de la tarea de programa en la que se buscará el objeto de datos especificado. Con ayuda de este argumento, puede buscar declaraciones PERS o TASKPERS en otras tareas. Cualquier otra declaración dará lugar a un error.

Value

Tipo de dato: anytype

Una variable que contiene el nuevo valor que se desea establecer. Su tipo de dato debe ser el mismo que el del objeto de datos a establecer. El valor establecido debe ser capturado de una variable, pero puede almacenarse en una variable o una variable persistente.

### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema ERRNO cambiará a:

| Nombre          | Causa del error   |
|-----------------|---|
| ERR_SYM_ACCESS  | <ul style="list-style-type: none"><li>• El objeto de datos no existe.</li><li>• El objeto de datos es de solo lectura.</li><li>• Si el objeto de datos es un dato o un parámetro de rutina y no está situado en la rutina activa actualmente.</li><li>• Búsqueda en otras tareas para otras declaraciones diferentes de PERS o TASKPERS. <sup>i</sup></li></ul> |
| ERR_INVDIM      | El objeto de datos y la variable utilizados en el argumento Value tienen dimensiones diferentes   |
| ERR_SYMBOL_TYPE | El objeto de datos y la variable utilizados en el argumento Value son de tipos diferentes. Si se utilizan los tipos de datos ALIAS, también se producirá un ERROR, incluso si los tipos pudieran tener el mismo tipo de datos básicos.  |
| ERR_TASKNAME    | El nombre de tarea de programa en el argumento \TaskName no puede encontrarse en el sistema.  |

<sup>i</sup> Al utilizar los argumentos TaskRef o TaskName, puede buscar declaraciones PERS o TASKPERS en otras tareas; cualquier otra declaración provocará un error.  
La búsqueda de PERS declarada como LOCAL en otras tareas también generará un error.

### Limitaciones

En el caso de los tipos de datos de semivalor, no es posible buscar por el tipo de dato asociado al valor. Por ejemplo, si se busca dionum, no se obtendrá ninguna coincidencia de búsqueda con las señales signaldi. Si se busca num, no se obtendrá ninguna coincidencia de búsqueda con las señales signalgi o signalai.

Continúa en la página siguiente

No es posible asignar un valor a una variable declarada como LOCAL en un módulo de RAPID incorporado.

### Sintaxis

```
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>];'
```

### Información relacionada

| Para obtener más información sobre                              | Consulte  |
|---|---|
| Definición de un conjunto de símbolos en una sesión de búsqueda | <a href="#">SetDataSearch - Definir el conjunto de símbolos de una secuencia de búsqueda en la página 696</a>             |
| Obtención del siguiente símbolo coincidente                     | <a href="#">GetNextSym - Obtiene el siguiente símbolo coincidente en la página 1309</a>                                   |
| Obtención del valor de un objeto de datos                       | <a href="#">GetDataVal - Obtiene el valor de un objeto de datos en la página 228</a>                                      |
| Asignación del valor de varios objetos de datos                 | <a href="#">SetAllDataVal - Establece un valor en todos los objetos de datos de un conjunto definido en la página 691</a> |
| El tipo de datos relacionado datapos                            | <a href="#">datapos - Inclusión de un bloque para un objeto de datos en la página 1657</a>                                |
| <i>Advanced RAPID</i>   | <i>Application manual - Controller software IRC5</i>  |

# 1 Instrucciones

---

1.233 SetDO - Cambia el valor de una señal digital de salida  
RobotWare - OS

## 1.233 SetDO - Cambia el valor de una señal digital de salida

---

### Utilización

SetDO se usa para cambiar el valor de una señal digital de salida, con o sin retardo ni sincronización.

---

### Ejemplos básicos

Los siguientes ejemplos ilustran la instrucción SetDO.

#### Ejemplo 1

```
SetDO do15, 1;
```

Se cambia la señal do15 a 1.

#### Ejemplo 2

```
SetDO weld, off;
```

Se cambia la señal weld a off

.

#### Ejemplo 3

```
SetDO \SDelay := 0.2, weld, high;
```

Se cambia el valor de la señal weld a high con un retardo de 0.2 s. La ejecución del programa continúa en la instrucción siguiente.

#### Ejemplo 4

```
SetDO \Sync ,do1, 0;
```

Se cambia la señal do1 a 0 La ejecución del programa espera hasta que la señal cambia físicamente al valor especificado.

---

### Argumentos

```
SetDO [ \SDelay ]|[ \Sync ] Signal Value
```

[ \SDelay ]

#### *Signal Delay*

Tipo de dato: num

Retarda el cambio durante la duración especificada, en segundos (máximo 2.000 s). La ejecución del programa continúa directamente con la instrucción siguiente. Después del tiempo de retardo especificado, la señal cambia sin que el resto de la ejecución del programa se vea afectada.

[ \Sync ]

#### *Synchronization*

Tipo de dato: switch

Si se utiliza este argumento, la ejecución del programa espera hasta que la señal cambie físicamente al valor especificado.

Signal

Tipo de dato: signaldo

El nombre de la señal que debe cambiar.

---

*Continúa en la página siguiente*

Value

Tipo de dato: dionum

El valor deseado para la señal, 0 ó 1.

| Valor especificado        | Cambiar la salida digital a |
|---------------------------|-----------------------------|
| 0                         | 0                           |
| Cualquier valor excepto 0 | 1                           |

**Ejecución de programas**

El valor real depende de la configuración de la señal. Si la señal está invertida en los parámetros del sistema, el valor del canal físico es el opuesto.

Si no se utilizan los argumentos \SDelay ni \Sync, la señal cambia lo antes posible y la siguiente instrucción se ejecuta inmediatamente, sin esperar a que la señal cambie físicamente.

**Limitaciones**

Si una instrucción SetDO con un argumento \SDelay va seguida de una nueva instrucción SetDO para la misma señal, con o sin argumento \SDelay, la primera instrucción SetDO se cancela si la segunda instrucción SetDO se ejecuta antes de que haya transcurrido el tiempo de retardo de la primera instrucción SetDO.

**Gestión de errores**

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema ERRNO cambiará a:

| Nombre             | Causa del error   |
|--------------------|---|
| ERR_ARGVALERR      | El valor del argumento SDelay sobrepasa el valor máximo permitido (2000 s).   |
| ERR_NO_ALIASIO_DEF | La variable de señal es una variable declarada en RAPID. No se ha conectado a una señal E/S definida en la configuración de E/S con la instrucción AliasIO.   |
| ERR_NORUNUNIT      | Se ha perdido el contacto con el dispositivo de E/S.  |
| ERR_SIG_NOT_VALID  | La señal de E/S no está disponible. Los motivos pueden ser que el dispositivo de E/S no está en funcionamiento o que exista un error en la configuración (válido solamente para el bus de campo ICI). |

**Sintaxis**

```
SetDO
  ['\' SDelay :=' <expression (IN) of num>',']
  |['\' Sync',']
  [Signal :='] <variable (VAR) of signaldo>',']
  [Value :='] <expression (IN) of dionum>';'
```

**Información relacionada**

| Para obtener más información sobre | Consulte                                      |
|------------------------------------|---|
| Instrucciones de entrada/salida    | Manual de referencia técnica - RAPID Overview |

Continúa en la página siguiente

# 1 Instrucciones

---

1.233 SetDO - Cambia el valor de una señal digital de salida

RobotWare - OS

Continuación

| Para obtener más información sobre         | Consulte   |
|--|--|
| Funcionalidad de entrada/salida en general | <i>Manual de referencia técnica - RAPID Overview</i>         |
| Configuración de E/S                       | <i>Manual de referencia técnica - Parámetros del sistema</i> |

### 1.234 SetGO - Cambia el valor de un grupo de señales digitales de salida

#### Utilización

SetGO se usa para cambiar el valor de un grupo de señales digitales de salida, con o sin retardo de tiempo.

#### Ejemplos básicos

Los ejemplos siguientes ilustran la instrucción SetGO.

##### Ejemplo 1

```
SetGO go2, 12;
```

La señal go2 se cambia a 12. Si go2 se compone de 4 señales, por ejemplo, las salidas de la 6 a la 9, las salidas 6 y 7 cambian a cero, mientras que las salidas 8 y 9 cambian a uno.

##### Ejemplo 2

```
SetGO \SDelay := 0.4, go2, 10;
```

Se cambia la señal go2 a 10. Si go2 se compone de 4 señales, por ejemplo, las salidas 6-9, las salidas 6 y 8 cambian a cero, mientras que las salidas 7 y 9 cambian a uno con un retardo de 0.4 s. La ejecución del programa continúa con la instrucción siguiente.

##### Ejemplo 3

```
SetGO go32, 4294967295;
```

Se cambia la señal go32 a 4294967295. go32 contiene 32 señales, todas ellas con el valor 1.

#### Argumentos

```
SetGO [ \SDelay ] Signal Value | Dvalue
```

[ \SDelay ]

##### *Signal Delay*

Tipo de dato: num

Retarda el cambio durante el periodo especificado, en segundos (máximo 2.000 s). La ejecución del programa continúa directamente con la instrucción siguiente. Después del tiempo de retardo especificado, el valor de las señales cambia sin que el resto de la ejecución del programa se vea afectada.

Si se omite el argumento, los valores de las señales cambian directamente.

Signal

Tipo de dato: signalgo

El nombre del grupo de señales que debe cambiar.

Value

Tipo de dato: num

El valor deseado para el grupo de señales (un entero positivo) se muestra en la tabla siguiente.

*Continúa en la página siguiente*

## 1 Instrucciones

### 1.234 SetGO - Cambia el valor de un grupo de señales digitales de salida

RobotWare - OS

Continuación

El valor permitido depende del número de señales del grupo. Un tipo de dato `num` puede contener el valor de un grupo de 32 señales o menos.

Dvalue

Tipo de dato: `dnum`

El valor deseado para el grupo de señales (un entero positivo) se muestra en la tabla siguiente.

El valor permitido depende del número de señales del grupo. Un tipo de dato `dnum` puede contener el valor de un grupo de 32 señales o menos.

| Número de señales | Valor permitido | Valor doble permitido |
|-------------------|-----------------|-----------------------|
| 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          |

Continúa en la página siguiente



## 1.234 SetGO - Cambia el valor de un grupo de señales digitales de salida

RobotWare - OS

Continuación

| Número de señales | Valor permitido | Valor doble permitido |
|-------------------|-----------------|-----------------------|
| 31                | *               | 0-2147483647          |
| 32                | *               | 0-4294967295          |

\*) El argumento `Value` del tipo `num` sólo puede contener un máximo de 23 señales en comparación con el argumento `Dvalue` del tipo `dnum`, que puede contener hasta 32 señales.

### Ejecución de programas

El valor programado se convierte en un número binario sin signo. Este número binario se envía al grupo de señales, con el resultado de que se cambian a 0 ó 1 las distintas señales del grupo. Debido a los retardos internos, el valor de la señal puede permanecer sin definir durante un breve periodo de tiempo.

### Limitaciones

El número máximo de señales que pueden usarse para un grupo es de 23 si se usa el argumento `Value` y de 32 si se usa el argumento `Dvalue`. Esta limitación se aplica a todas las instrucciones y funciones que utilicen señales de grupo.

### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                          | Causa del error   |
|---------------------------------|---|
| <code>ERR_ARGVALEERR</code>     | El valor del argumento <code>SDelay</code> sobrepasa el valor máximo permitido (2000 s).  |
| <code>ERR_GO_LIM</code>         | El argumento programado <code>Value</code> o <code>Dvalue</code> para la señal digital de salida de grupo especificada <code>Signal</code> está fuera de límites.                                     |
| <code>ERR_NO_ALIASIO_DEF</code> | La variable de señal es una variable declarada en <code>RAPID</code> . No se ha conectado a una señal E/S definida en la configuración de E/S con la instrucción <code>AliasIO</code> .               |
| <code>ERR_NORUNUNIT</code>      | Se ha perdido el contacto con el dispositivo de E/S.  |
| <code>ERR_SIG_NOT_VALID</code>  | La señal de E/S no está disponible. Los motivos pueden ser que el dispositivo de E/S no está en funcionamiento o que exista un error en la configuración (válido solamente para el bus de campo ICI). |

### Sintaxis

```
SetGO
[ '\ ' SDelay ' := ' < expression (IN) of num > ', ' ]
[ Signal ' := ' ] < variable (VAR) of signalgo > ', '
[ Value ' := ' ] < expression (IN) of num >
| [ Dvalue ' := ' ] < expression (IN) of dnum > ';'

```

### Información relacionada

| Para obtener más información sobre      | Consulte   |
|---|--|
| Otras instrucciones de entrada y salida | <i>Manual de referencia técnica - RAPID Overview</i> |

Continúa en la página siguiente

# 1 Instrucciones

---

1.234 SetGO - Cambia el valor de un grupo de señales digitales de salida

RobotWare - OS

Continuación

| Para obtener más información sobre            | Consulte   |
|---|--|
| Funcionalidad de entrada/salida en general    | <i>Manual de referencia técnica - RAPID Overview</i>         |
| Configuración de E/S (parámetros del sistema) | <i>Manual de referencia técnica - Parámetros del sistema</i> |

## 1.235 SetLeadThrough - Activar y desactivar proceso de guiado

### Utilización

`SetLeadThrough` se utiliza para activar y desactivar el proceso de guiado para un robot TCP.

### Ejemplos básicos

Los siguientes ejemplos ilustran la instrucción `SetLeadThrough`.

#### Ejemplo 1

```
SetLeadThrough \On;
```

Activa el proceso de guiado para el robot TCP `ROB_L` si se ejecuta en la tarea `T_ROB_L RAPID`. De forma predeterminada, se ordena una instrucción `StopMove` cuando se activa el proceso de guiado.

#### Ejemplo 2

```
SetLeadThrough \Off;
```

Desactiva el proceso de guiado para el robot TCP `ROB_L` si se ejecuta en la tarea `T_ROB_L RAPID`. De forma predeterminada, se ordena una instrucción `ClearPath` y también se ejecuta una instrucción `StartMove`.

#### Ejemplo 3

```
SetLeadThrough \On \NoStopMove;
..
StopMove;
..
SetLeadThrough \Off \NoStartMove \NoClearPath;
..
StartMove;
```

Establezca el proceso de guiado para el robot TCP. El proceso de guiado no se activará hasta que se haya ejecutado una instrucción `StopMove` o se haya detenido la ejecución del programa. La desactivación del proceso de guiado se realiza y posteriormente se reinicia el movimiento.

### Argumentos

```
SetLeadThrough [\On] | [\Off] [\NoStopMove] | [\NoStartMove]
[\NoClearPath]
```

`[\On]`

Tipo de dato: switch

Activar el proceso de guiado.

`[\Off]`

Tipo de dato: switch

Desactivar el proceso de guiado.

`[\NoStopMove]`

Tipo de dato: switch

Solo puede utilizarse junto con el modificador `\On`.

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.235 SetLeadThrough - Activar y desactivar proceso de guiado

YuMi

Continuación

Si se utiliza el modificador `\NoStopMove`, no se ejecutará ninguna orden `StopMove`. El proceso de guiado se ha establecido, aunque no activado. El proceso de guiado se activa cuando la ejecución del programa se detiene o cuando se ejecuta una instrucción `StopMove`.

[`\NoStartMove`]

Tipo de dato: `switch`

Solo puede utilizarse junto con el modificador `\Off`.

Si se utiliza este modificador, no se ordenará el reinicio del movimiento del robot TCP. Es necesaria una instrucción `StartMove` para reanudar el movimiento.

[`\NoClearPath`]

Tipo de dato: `switch`

Solo puede utilizarse junto con el modificador `\Off`.

La trayectoria no se borra al desactivar el proceso de guiado y el robot TCP continuará en la trayectoria programada cuando se ejecute la orden `StartMove`.

---

### Ejecución de programas

El estado del proceso de guiado se establece si se utiliza el argumento `\On` (o no argumento), aunque no se activa hasta que se haya ejecutado una instrucción `StopMove` o la ejecución del programa se haya detenido.

De forma predeterminada, una instrucción `StopMove` se ejecuta cuando el proceso de guiado se activa con `SetLeadThrough \On`. Se ejecuta una instrucción `ClearPath` y una instrucción `StartMove` cuando se desactiva el proceso de guiado si no se utiliza el modificador `\NoClearPath` ni `\NoStartMove`.

Si se ejecuta la instrucción `SetLeadThrough` desde una tarea sin movimiento, la activación del proceso de guiado se realizará para el robot TCP en la tarea de movimiento conectada. La orden `StartMove` debe realizarse desde la misma tarea que la orden `StopMove`.

La activación del proceso de guiado es válida hasta que se ejecute una instrucción `SetLeadThrough \Off`.

El valor predeterminado (sin proceso de guiado) se establece automáticamente:

- Cuando se utiliza el modo de reinicio **Restablecer RAPID**.
- al cargar un nuevo programa o un nuevo módulo.
- Cuando se inicia la ejecución del programa desde el principio
- al mover el puntero del programa a `main`.
- al mover el puntero del programa a una rutina.
- al mover el puntero de programa de una forma que se pierde el orden de la ejecución.
- al apagar los motores.

Continúa en la página siguiente

**Gestión de errores**

Se generan los siguientes errores recuperables que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` se establecerá en:

| Nombre           | Causa del error  |
|------------------|--|
| ERR_PATHDIST     | El robot está demasiado alejado de la trayectoria (más de 10 mm o 20 grados) para realizar un reinicio del movimiento interrumpido.<br>Mueva el robot para acercarlo a la trayectoria antes de intentar <code>RETRY</code> . |
| ERR_STARTMOVE    | El robot se encuentra en estado de espera al ejecutar un <code>SetLeadThrough \Off</code> .<br>Espere unos instantes antes de intentar ejecutar <code>RETRY</code> .   |
| ERR_PROGSTOP     | El robot ya está en el estado de programa parado cuando se ejecuta un <code>SetLeadThrough \Off</code> .<br>Espere unos instantes antes de intentar ejecutar <code>RETRY</code> .  |
| ERR_ALRDY_MOVING | El robot ya está en movimiento cuando se ejecuta un <code>SetLeadThrough \Off</code> .<br>Espere unos instantes antes de intentar ejecutar <code>RETRY</code> .  |

**Limitaciones**

- Solo se permite a una de las distintas tareas sin movimiento la realización de `SetLeadThrough` con respecto a alguna tarea de movimiento.
- `SetLeadThrough` solo funciona con los robots TCP.

La instrucción `SetLeadThrough` solo puede usarse con los robots YuMi.

**Sintaxis**

```
SetLeadThrough
  ['\On' | ['\Off]
  ['\NoStopMove' | ['\NoStartMove]
  ['\NoClearPath' ';' ]
```

**Información relacionada**

| Para obtener más información sobre        | Consulte   |
|---|--|
| Comprobar el estado del proceso de guiado | <a href="#">IsLeadThrough - Comprobar el estado del proceso de guiado en la página 1355</a>      |
| Detención del movimiento                  | <a href="#">StopMove - Detiene el movimiento del robot en la página 828</a>                      |
| Continuación de un movimiento             | <a href="#">StartMove - Reanuda el movimiento del robot en la página 796</a>                     |
| Continuación de un movimiento             | <a href="#">StartMoveRetry - Reanuda el movimiento y la ejecución del robot en la página 799</a> |
| Más ejemplos                              | <a href="#">ClearPath - Elimina la trayectoria actual en la página 142</a>                       |

# 1 Instrucciones

---

## 1.236 SetSysData - Establece datos del sistema

RobotWare - OS

## 1.236 SetSysData - Establece datos del sistema

---

### Utilización

SetSysData activa el nombre del sistema especificado para el tipo de dato especificado.

Esta instrucción permite cambiar la herramienta, el objeto de trabajo, la carga útil o la carga total activos actualmente en el robot de la tarea de movimiento actual o conectada.

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción SetSysData:

#### Ejemplo 1

```
SetSysData tool5;
```

Se activa la herramienta tool5.

```
SetSysData tool0 \ObjectName := "tool6";
```

Se activa la herramienta tool6.

```
SetSysData anytool \ObjectName := "tool2";
```

Se activa la herramienta tool2.

### Argumentos

```
SetSysData SourceObject [\ObjectName]
```

SourceObject

Tipo de dato: anytype

La variable persistente, que debe activarse como un dato actual del sistema.

El tipo de dato de este argumento también especifica el tipo de dato de sistema que se desea activar para el robot de la tarea actual o de la tarea de movimiento conectada:

| Tipo de dato | Tipo de dato de sistema |
|--------------|-------------------------|
| tooldata     | Herramienta             |
| wobjdata     | Objeto de trabajo       |
| loaddata     | Carga útil/carga total  |

No es posible usar toda una matriz ni un componente de registro.

[ \ObjectName ]

Tipo de dato: string

Si se especifica este argumento opcional, especifica el nombre del objeto de dato que se desea activar (tiene prioridad sobre el nombre especificado en el argumento SourceObject). El tipo de dato del objeto de datos que se desea activar se captura siempre desde el argumento SourceObject.

### Ejecución de programas

Se establece el objeto de datos del sistema actual de la herramienta, el objeto de trabajo, la carga útil o la carga total, de acuerdo con los argumentos.

*Continúa en la página siguiente*

Recuerde que esta instrucción sólo activa un nuevo objeto de datos (o el mismo que el anterior) y no cambia nunca el valor de ningún objeto de datos.

### Sintaxis

```
SetSysData
  [ SourceObject':=' ] < persistent(PERS) of anytype>
  [ '\ObjectName':=' < expression (IN) of string> ] ';'

```

### Información relacionada

| Para obtener más información sobre   | Consulte   |
|--|--|
| Definición de herramientas   | <a href="#">tooldata - Datos de herramienta en la página 1795</a>                        |
| Definición de objetos de trabajo   | <a href="#">wobjdata - Datos del objeto de trabajo en la página 1823</a>                 |
| Definición de una carga útil   | <a href="#">loaddata - Datos de carga en la página 1694</a>                              |
| Obtención de datos del sistema   | <a href="#">GetSysData - Obtiene datos del sistema en la página 235</a>                  |
| Ejemplo de cómo usar TLoad, carga total.   | <a href="#">MoveL - Mueve el robot siguiendo una trayectoria lineal en la página 447</a> |
| Parámetro de sistema <i>ModalPayloadMode</i> para la activación y la desactivación de la carga útil.<br>(Tema Controller, tipo General RAPID, valores de acción, <i>ModalPayloadMode</i> ) | <i>Manual de referencia técnica - Parámetros del sistema</i>                             |
| <i>Advanced RAPID</i>  | <i>Application manual - Controller software IRC5</i>                                     |

# 1 Instrucciones

---

## 1.237 SetupCyclicBool - Configurar una condición de Cyclic bool RobotWare - OS

### 1.237 SetupCyclicBool - Configurar una condición de Cyclic bool

---

#### Utilización

SetupCyclicBool se utiliza para configurar una condición lógica que se evaluará cíclicamente y se asignará a una variable booleana persistente, un *Cyclic bool*.

---

#### Ejemplos básicos

El ejemplo siguiente ilustra la instrucción SetupCyclicBool.  
Consulte también [Más ejemplos en la página 717](#).

#### Ejemplo 1

```
PERS bool cyclicflag1;  
  
PROC main()  
  SetupCyclicBool cyclicflag1, dil=1 AND do2=1;  
  ...
```

Configura una evaluación cíclica de la condición lógica `dil=1 AND do2=1` y asigna el resultado a la variable booleana persistente `cyclicflag1`.

---

#### Argumentos

```
SetupCyclicBool Flag Cond [\Signal]
```

#### Flag

Tipo de dato: `bool`

La variable booleana persistente que almacena el valor de la condición lógica.

#### Cond

Tipo de dato: `bool`

La expresión lógica que debe evaluarse cíclicamente.

La expresión puede constar de:

- Constantes o variables persistentes de los tipos `bool`, `num` y `dnum` (y alias de `bool`, `num` y `dnum`).
- Señales digitales globales de entrada y salida.
- Operandos: 'NOT' 'AND' 'OR' 'XOR' '=' '(' ')''

#### [\Signal]

Tipo de dato: `signaldo`

El resultado de la condición lógica se establece con la señal de salida digital utilizada en el argumento opcional `Signal` cuando se actualiza la variable booleana persistente.

No se recomienda utilizar la señal resultante como parte de la condición para un *Cyclic bool*.

---

#### Ejecución de programas

Con esta instrucción es posible configurar condiciones más complejas y utilizar en su lugar la indicación cíclica para ver si la condición se cumple o no.

*Continúa en la página siguiente*



## 1.237 SetupCyclicBool - Configurar una condición de Cyclic bool RobotWare - OS Continuación

La evaluación cíclica de la condición lógica y la asignación a la variable booleana persistente se realiza cada 12 ms.

El comportamiento de la funcionalidad de Cyclic bool puede configurarse. Para obtener más información, consulte *Application manual - Controller software IRC5* y *Manual de referencia técnica - Parámetros del sistema*.

### Limitaciones

- La expresión debe evaluarse para un valor booleano `TRUE` o `FALSE`. Todas las partes de la expresión también deben evaluarse para un valor booleano `TRUE` o `FALSE`.
- Cualquier `PERS num` o `dnum`, `CONST num` o `dnum` o literal `num` o `dnum` utilizada en una condición debe ser de tipo entero. Si se utiliza un valor decimal provocará un error no recuperable.

### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                          | Causa del error  |
|---------------------------------|--|
| <code>ERR_NO_ALIASIO_DEF</code> | La variable de señal es una variable declarada en <code>RAPID</code> . No se ha conectado a una señal de E/S definida en la configuración de E/S con la instrucción <code>AliasIO</code> . |
| <code>ERR_NORUNUNIT</code>      | Se ha perdido el contacto con el dispositivo de E/S.   |
| <code>ERR_SIG_NOT_VALID</code>  | La señal de E/S no está disponible (solo válido para el bus de campo ICI).   |

### Más ejemplos

A continuación aparecen más ejemplos de la instrucción `SetupCyclicBool`.

#### Ejemplo 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

```

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.237 SetupCyclicBool - Configurar una condición de Cyclic bool

RobotWare - OS

Continuación

...

En el ejemplo anterior se configura la evaluación cíclica de 2 expresiones. La ejecución espera hasta que se define `cyclicflag1.cyclicflag2` decide en qué posición debe moverse el robot.

### Ejemplo 2

```
!This condition is wrong:
SetupCyclicBool m1, 5;

!This condition is correct:
SetupCyclicBool m1, myNum = 5;
```

La primera condición no es correcta ya que el valor 5 no es booleano. La segunda condición es correcta porque la comparación puede evaluarse como una condición booleana, es decir TRUE o FALSE.

### Sintaxis

```
SetupCyclicBool
[Flag ':='] <persistent (PERS) of bool>', '
[Cond ':='] <expression (IN) of bool>
['\ ' Signal ':=' <variable (VAR) of signaldo>]';'
```

### Información relacionada

| Para obtener más información sobre                             | Consulte   |
|--|--|
| Comprobar si una variable persistente es un Cyclic bool        | <a href="#">IsCyclicBool - Comprueba si una variable persistente es un Cyclic bool en la página 1348</a> |
| Eliminar una condición de Cyclic bool                          | <a href="#">RemoveCyclicBool - Eliminar una condición de Cyclic bool en la página 596</a>                |
| Eliminar todas las condiciones de Cyclic bool                  | <a href="#">RemoveAllCyclicBool - Eliminar todas las condiciones de Cyclic bool en la página 594</a>     |
| Condiciones lógicas evaluadas cíclicamente, <i>Cyclic bool</i> | <i>Application manual - Controller software IRC5</i>   |
| Configuración de <i>Cyclic bool</i>                            | <i>Manual de referencia técnica - Parámetros del sistema</i>   |

## 1.238 SetupSuperv - Configurar las condiciones para la supervisión de señales en CAP *Continuous Application Platform (CAP)*

### 1.238 SetupSuperv - Configurar las condiciones para la supervisión de señales en CAP

#### Utilización

SetupSuperv se utiliza para configurar las condiciones para que se supervisen las señales de E/S. Las condiciones se recopilan en diferentes listas:

- PRE
- PRE\_START
- END\_PRE
- START
- MAIN
- END\_MAIN
- START\_POST1
- POST1
- END\_POST1
- START\_POST2
- POST2
- END\_POST2

Para obtener más información acerca de las listas de supervisión, consulte *Application manual - Continuous Application Platform*.

Como parámetro opcional, puede especificarse una señal de salida. La señal de salida cambia a un valor alto si se incumple la condición dada.

#### Ejemplo básico

```
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 se utiliza para configurar la supervisión en las señales. Si la señal *diWR\_EST* falla durante la fase *SUPERV\_MAIN*, la señal de salida digital *do\_WR\_Sup* cambia a un valor alto.

La instrucción *SetupSuperv* solo debe ejecutarse si cambian los datos de supervisión. Si los datos de supervisión nunca cambian, es una buena idea ponerlos en un módulo que se ejecuta desde el shelf de puesta en marcha.

#### Argumentos

```
SetupSuperv Signal Condition Listtype [\ErrIndSig]
```

#### Signal

Tipo de dato: *signal*di

La señal digital que se va a supervisar.

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.238 SetupSuperv - Configurar las condiciones para la supervisión de señales en CAP

### Continuous Application Platform (CAP)

#### Continuación

#### Condition

Tipo de dato: num

El nombre representa una de las siguientes condiciones disponibles:

|           |   |
|-----------|---|
| ACT:      | Se utiliza para la supervisión del estado. Estado de señal previsto durante la supervisión: activa. Si la señal cambia a pasiva, se dispara la supervisión.   |
| PAS:      | Se utiliza para la supervisión del estado. Estado de señal previsto durante la supervisión: pasiva. Si la señal cambia a activa, se dispara la supervisión.   |
| POS_EDGE: | Se utiliza para la supervisión del intercambio. Estado de señal previsto al final de la supervisión: activa. Si la señal no cambia a activa dentro del tiempo límite seleccionado, se dispara la supervisión. |
| NEG_EDGE: | Se utiliza para la supervisión del intercambio. Estado de señal previsto al final de la supervisión: pasiva. Si la señal no cambia a pasiva dentro del tiempo límite seleccionado, se dispara la supervisión. |

#### Listtype

Tipo de dato: num

El nombre que representa el número de las diferentes listas (por ejemplo, las fases del proceso):

- 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]

Tipo de dato: signaldo

Se utiliza para indicar qué condición se incumple si se ha producido un fallo. Cuando se produce el fallo el valor de esta señal cambia a 1. Es un parámetro opcional.

---

#### Ejecución de programas

La señal dada y su condición se añaden a la lista seleccionada. Si falla una señal, la instrucción `CapI/CapC` informará de que se ha producido un error de supervisión durante la fase especificada y qué señal o señales fallaron.

*Continúa en la página siguiente*

## 1.238 SetupSuperv - Configurar las condiciones para la supervisión de señales en CAP Continuous Application Platform (CAP) Continuación

### Errores

#### CAP\_SPV\_LIM

Se ha excedido el número máximo de supervisiones configuradas.

#### CAP\_SPV\_UNK\_LST

No se conoce el tipo de lista de supervisión.

### Limitaciones

Solo pueden supervisarse las señales de entrada digitales.

La supervisión del estado se aplica para una secuencia completa de instrucciones de CAP (consulte la sección *Supervisión y fases del proceso en Application manual - Continuous Application Platform*).

### Sintaxis

```
SetupSuperv
  [Signal ':='] < variable (VAR) of signaldi > ','
  [Condition ':='] < variable (IN) of num > ','
  [Listtype ':='] < variable (IN) of num >
  [\ErrIndSig ':='] < variable (VAR) of signaldo > ';'

```

### Información relacionada

| Para obtener más información sobre     | Consulte   |
|--|--|
| <i>Continuous Application Platform</i> | <i>Application manual - Continuous Application Platform</i>                            |
| Instrucción <code>InitSuperv</code>    | <a href="#">InitSuperv - Restablecer toda la supervisión para CAP en la página 286</a> |
| Instrucción <code>RemoveSuperv</code>  | <a href="#">RemoveSuperv - Eliminar la condición de una señal en la página 602</a>     |



```

...
! 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

```

Al llamar a la rutina `RRI_Open`, en primer lugar se abre una conexión al dispositivo llamado `AnyDevice`. A continuación, se inicia una transmisión cíclica con una frecuencia de `SampleRate`.

### Ejemplo 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

```

Se establece una conexión al dispositivo llamado `AnyDevice` con el argumento opcional `\NoStop`, lo que impide que el sistema se pare si se interrumpe la conexión a `AnyDevice`. Los estados de error se gestionan en la rutina `TRAP`.

### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                     | Causa del error   |
|----------------------------|---|
| <code>ERR_COMM_INIT</code> | TCP se utiliza como protocolo de comunicación y la operación de conexión falla. |

Si se utiliza UDP como protocolo de comunicación, no existe ninguna garantía en cuanto al éxito de la operación de conexión y por tanto no es posible ninguna gestión de errores en el momento de la conexión.

El interruptor `\NoStop` permite gestionar los errores de comunicación detectados después de una conexión exitosa. `\NoStop` significa que los movimientos en la

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.239 SiConnect - Conexión a Sensor Interface

### Robot Reference Interface

#### Continuación

ejecución de RAPID continúan y que puede usarse la rutina `TRAP` para gestionar errores específicos mediante `IError` o cambios de estado específicos con `IPers`.



#### Nota

Ni `IPers` ni `IError` son interrupciones seguras, de modo que si se detecta un error después de un paro, no se ejecutará ningún `TRAP`. Una forma de gestionar este problema es contar con un `SiConnect \NoStop` en la bandeja de reinicio para asegurarse de que la aplicación intente restablecer la conexión al cliente.

#### Sintaxis

```
SiConnect  
[ Sensor ':=' ] < persistent (PERS) of sensor >  
[ '\ NoStop ] ';' ;
```

#### Información relacionada

| Para obtener más información sobre             | Consulte  |
|--|---|
| Cierre de una conexión a un sistema externo.   | <a href="#">SiClose - Cierre de Sensor Interface en la página 725.</a>                              |
| Registro de datos para transmisión cíclica.    | <a href="#">SiSetCyclic - Establecimiento de modo cíclico de Sensor Interface en la página 733.</a> |
| Suscripción a la transmisión cíclica de datos. | <a href="#">SiGetCyclic - Obtención de modo cíclico de Sensor Interface en la página 727</a>        |
| Descriptor del dispositivo externo.            | <a href="#">sensor - Descriptor de dispositivo externo en la página 1754.</a>                       |
| Estado de comunicación de un dispositivo.      | <a href="#">sensorstate - Estado de comunicación del dispositivo en la página 1756.</a>             |
| <i>Robot Reference Interface</i>               | <i>Application manual - Controller software IRC5</i>  |



## 1.240 SiClose - Cierre de Sensor Interface

### Utilización

SiClose cierra una conexión existente a un dispositivo externo.

### Ejemplos básicos

A continuación aparece un ejemplo básico de la instrucción SiClose.

#### Ejemplo 1

```
PERS sensor AnyDevice;
...
SiClose AnyDevice;
```

Cierre de la conexión al dispositivo denominado AnyDevice.

### Argumentos

SiClose Sensor

Sensor

Tipo de dato: sensor

El descriptor del dispositivo externo que se desea cerrar. El argumento es una variable persistente y su nombre debe ser el mismo al especificado como cliente en el archivo de configuración *Settings.xml*.

### Ejecución de programas

Cierra una conexión existente al dispositivo externo.

### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre        | Causa del error   |
|---------------|---|
| ERR_COMM_INIT | TCP se utiliza como protocolo de comunicación y la operación de conexión falla. |

Si se utiliza UDP como protocolo de comunicación, no existe ninguna garantía en cuanto al éxito de la operación de cierre y por tanto no es posible ninguna gestión de errores.

### Sintaxis

```
SiClose
[ Sensor ':= ' ] < persistent ( PERS ) of sensor > ' ; '
```

### Información relacionada

| Para obtener más información sobre                    | Consulte  |
|---|---|
| Establecimiento de una conexión a un sistema externo. | <a href="#">SiConnect - Conexión a Sensor Interface en la página 722.</a>                           |
| Registro de datos para transmisión cíclica.           | <a href="#">SiSetCyclic - Establecimiento de modo cíclico de Sensor Interface en la página 733.</a> |

Continúa en la página siguiente

# 1 Instrucciones

---

## 1.240 SiClose - Cierre de Sensor Interface

*Continuación*

| <b>Para obtener más información sobre</b>      | <b>Consulte</b>   |
|--|---|
| Suscripción a la transmisión cíclica de datos. | <a href="#">SiGetCyclic - Obtención de modo cíclico de Sensor Interface en la página 727.</a> |
| Descriptor del dispositivo externo.            | <a href="#">sensor - Descriptor de dispositivo externo en la página 1754.</a>                 |
| Estado de comunicación de un dispositivo.      | <a href="#">sensorstate - Estado de comunicación del dispositivo en la página 1756.</a>       |
| <i>Robot Reference Interface</i>               | <i>Application manual - Controller software IRC5</i>  |

### 1.241 SiGetCyclic - Obtención de modo cíclico de Sensor Interface

---

#### Utilización

SiGetCyclic suscribe datos para la transmisión cíclica desde un dispositivo externo.

---

#### Ejemplos básicos

A continuación aparece un ejemplo básico de la instrucción SiGetCyclic. Consulte también [Más ejemplos en la página 727](#).

#### Ejemplo 1

```
SiConnect AnyDevice;  
! Receive data cyclic with 64 ms rate  
SiGetCyclic AnyDevice, DataIn, 64;
```

El ejemplo muestra cómo establecer una conexión a un dispositivo externo y configurar una transmisión cíclica desde el dispositivo AnyDevice.

---

#### Argumentos

SiGetCyclic Sensor Data Rate

##### Sensor

Tipo de dato: sensor

El descriptor del dispositivo externo desde el que se desea recibir datos cíclicos. El argumento es una variable persistente y su nombre debe ser el mismo al especificado como cliente en el archivo de configuración *Settings.xml*.

##### Data

Tipo de dato: anytype

Una referencia a una variable persistente que contiene los datos que se desea recibir del cliente, especificados en el argumento *Sensor*. La variable debe ser definida como *Readable* en el archivo *Configuration.xml*.

##### Rate

Tipo de dato: num

Tasa de transferencia en milisegundos (solo se admiten múltiplos de 4 ms).

---

#### Ejecución de programas

La instrucción SiGetCyclic suscribe datos para la transmisión cíclica desde un dispositivo externo.

En el caso de las instrucciones SiGetCyclic y SiSetCyclic, una tasa de transferencia de 0 detiene (anula el registro/anula la suscripción) la transmisión cíclica del dato o conjunto de datos indicados.

---

#### Más ejemplos

A continuación aparecen más ejemplos de cómo usar la instrucción SiGetCyclic.

#### Ejemplo 1

```
PERS sensor AnyDevice;
```

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.241 SiGetCyclic - Obtención de modo cíclico de Sensor Interface

### Robot Reference Interface

#### Continuación

```
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
```

Al llamar a la rutina `RRI_Open`, en primer lugar se abre una conexión al dispositivo llamado `AnyDevice`. A continuación, se inicia una transmisión cíclica con una frecuencia de `SampleRate`.

#### Sintaxis

```
SiGetCyclic
    [ Sensor ':= ' ] < persistent (PERS) of sensor > ', '
    [ Data ':= ' ] < persistent (PERS) of anytype > ', '
    [ Rate ':= ' ] < expression (IN) of num > ] ';' ;'
```

#### Información relacionada

| Para obtener más información sobre                    | Consulte  |
|---|---|
| Establecimiento de una conexión a un sistema externo. | <a href="#">SiConnect - Conexión a Sensor Interface en la página 722.</a>                           |
| Cierre de una conexión a un sistema externo.          | <a href="#">SiClose - Cierre de Sensor Interface en la página 725.</a>                              |
| Registro de datos para transmisión cíclica.           | <a href="#">SiSetCyclic - Establecimiento de modo cíclico de Sensor Interface en la página 733.</a> |
| Descriptor del dispositivo externo.                   | <a href="#">sensor - Descriptor de dispositivo externo en la página 1754.</a>                       |
| Estado de comunicación de un dispositivo.             | <a href="#">sensorstate - Estado de comunicación del dispositivo en la página 1756.</a>             |
| <i>Robot Reference Interface</i>                      | <i>Application manual - Controller software IRC5</i>  |

## 1.242 SimCollision: simular una colisión

### Utilización

`SimCollision` se utiliza para simular una colisión. La instrucción puede resultar útil cuando se prueba la gestión de errores para colisiones.

### Ejemplos básicos

El ejemplo siguiente ilustra la instrucción `SimCollision`.

#### Ejemplo 1

```
SimCollision;
```

Simular una colisión de movimiento para probar la gestión de errores.

### Ejecución de programas

La ejecución de `SimCollision` simulará una colisión de movimiento. La instrucción puede resultar útil cuando se prueba la gestión de errores para colisiones de movimiento y solo se debe utilizar para fines de comprobación. La instrucción generará un registro de eventos 41910, *Collision simulated with instruction SimCollision*, cuando se utilice.

### Sintaxis

```
SimCollision';'
```

### Información relacionada

| Para obtener más información sobre                                     | Consulte  |
|--|---|
| Configuración para establecer si la gestión de errores debe ejecutarse | <i>Manual de referencia técnica - Parámetros del sistema</i> , tema <i>Controller</i> , tipo <i>General RAPID</i> , <i>CollisionErrorHandling</i> |
| <i>Collision Detection</i>   | <i>Application manual - Controller software IRC5</i>  |

## 1 Instrucciones

---

1.243 SingArea - Define el método de interpolación alrededor de puntos singulares  
RobotWare - OS

### 1.243 SingArea - Define el método de interpolación alrededor de puntos singulares

---

#### Utilización

`SingArea` se utiliza para definir cómo debe moverse el robot en las cercanías de los puntos singulares.

`SingArea` también se usa para definir la interpolación lineal y circular en los robots con menos de seis ejes y un robot de seis ejes se puede programar para funcionar con el eje 4 bloqueado en la posición 0 o en  $\pm 180$  grados.

Esta instrucción sólo puede usarse en la tarea principal `T_ROB1` o, si se cuenta con un sistema *MultiMove*, en las tareas de movimiento.

#### Ejemplos básicos

Los ejemplos siguientes ilustran la instrucción `SingArea`.

##### Ejemplo 1

```
SingArea \Wrist;
```

Es posible cambiar ligeramente la orientación de la herramienta para atravesar un punto singular (ejes 4 y 6 en línea).

Es posible que los robots con menos de seis ejes no puedan alcanzar una orientación de herramienta interpolada. Al utilizar `SingArea \Wrist`, el robot puede conseguir el movimiento, pero la orientación de la herramienta cambiará ligeramente.

##### Ejemplo 2

```
SingArea \LockAxis4;
```

Un robot de seis ejes se puede programar para funcionar con el eje 4 bloqueado en la posición 0 o en  $\pm 180$  grados para evitar problemas de singularidad cuando el eje 5 se aproxime a 0.

La posición programada se alcanza con el eje 4 bloqueado en la posición 0 o en  $\pm 180$  grados. Si la posición no se programó con el eje 4 en la posición 0 o en  $\pm 180$  grados, se alcanza ahora con una orientación de herramienta diferente.

Si la posición inicial del eje 4 se desvía más de 2 grados respecto de la posición bloqueada, el primer movimiento se comportará como si se llamara a `SingArea` con el argumento `\Wrist`.

##### Ejemplo 3

```
SingArea \Off;
```

No se permite que la orientación de la herramienta sea distinta de la orientación programada. Si se atraviesa un punto singular, uno o varios ejes pueden realizar un movimiento de barrido, lo que da lugar a una reducción de velocidad de TCP.

Es posible que los robots con menos de seis ejes no puedan alcanzar una orientación de herramienta programada. Como resultado, el robot se detiene.

#### Argumentos

```
SingArea [\Wrist] | [\LockAxis4] | [\Off]
```

`[\Wrist]`

Tipo de dato: switch

*Continúa en la página siguiente*

## 1.243 SingArea - Define el método de interpolación alrededor de puntos singulares

*RobotWare - OS*

*Continuación*

Se permite que la orientación de la herramienta sea algo diferente, para evitar la singularidad de la muñeca. Se utiliza cuando los ejes 4 y 6 están en paralelo (con el eje 5 a 0 grados). También se usa para la interpolación lineal y circular en los robots con menos de seis ejes, en los que se permite que la orientación de la herramienta sea diferente.

[`\LockAxis4`]

Tipo de dato: `switch`

La posición programada se alcanza con el eje 4 bloqueado en la posición 0 o en  $\pm 180$  grados. Si la posición no se programó con el eje 4 en la posición 0 o en  $\pm 180$  grados, se alcanza ahora con una orientación de herramienta diferente.

Si la posición inicial del eje 4 se desvía más de 2 grados respecto de la posición bloqueada, el primer movimiento se comportará como si se llamara a `SingArea` con el argumento `\Wrist`.

[`\Off`]

Tipo de dato: `switch`

La orientación de la que no puede desviarse la herramienta. Se utiliza cuando no se atraviesa ningún punto singular o cuando no se permite que cambie la orientación.

Si no se especifica ninguno de los argumentos, el sistema pasará a `\Off`.

---

### Ejecución de programas

La interpolación especificada se aplica a la siguiente instrucción de movimiento ejecutada y es válida hasta que se ejecute una nueva instrucción `SingArea`.

Si se especifica el argumento `\Wrist`, la orientación se consigue mediante la interpolación de ejes para evitar los puntos singulares. De esta forma, el TCP sigue la trayectoria correcta, pero la orientación de la herramienta se desvía hasta cierto punto. Esto también se produce cuando no se atraviesa un punto singular.

Si se especifica el argumento `\LockAxis4`, el eje 4 se bloquea en la posición 0 o en  $\pm 180$  grados para evitar puntos singulares. El TCP seguirá en general la trayectoria correcta, pero la orientación de la herramienta se desvía si la posición no se programó con el eje 4 en la posición 0 o en  $\pm 180$  grados. Para rutas con grandes reorientaciones, el TCP puede desviarse de la trayectoria lineal programada.

El movimiento sólo se ve afectado por la ejecución de la interpolación lineal o circular.

De forma predeterminada, la ejecución del programa utiliza automáticamente el argumento `Off` en los robots con seis ejes. Los robots que tienen menos de seis ejes pueden usar el argumento `Off` o el argumento `/Wrist` de forma predeterminada. Esto se establece automáticamente en la rutina de evento `SYS_RESET`.

El valor predeterminado se establece automáticamente

- cuando se utiliza el modo de reinicio **Restablecer RAPID**
- al cargar un nuevo programa o un nuevo módulo
- al iniciar la ejecución del programa desde el principio

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.243 SingArea - Define el método de interpolación alrededor de puntos singulares

RobotWare - OS

Continuación

- al mover el puntero del programa a `main`
- al mover el puntero del programa a una rutina
- al mover el puntero de programa de una forma que se pierde el orden de la ejecución.

---

### Sintaxis

```
SingArea  
  [ '\ ' Wrist ] | [ '\ ' LockAxis4 ] | [ '\ ' Off ] ';' 
```

---

### Información relacionada

| Para obtener más información sobre | Consulte   |
|------------------------------------|--|
| Singularidad                       | <i>Manual de referencia técnica - RAPID Overview</i>                             |
| Interpolación                      | <i>Manual de referencia técnica - RAPID Overview</i>                             |
| Datos de parámetros de movimiento  | <a href="#">motsetdata - Datos de parámetros de movimiento en la página 1705</a> |





# 1 Instrucciones

## 1.244 SiSetCyclic - Establecimiento de modo cíclico de Sensor Interface

### Robot Reference Interface

#### Continuación

#### Más ejemplos

A continuación aparecen más ejemplos de cómo usar la instrucción `SiSetCyclic`.

#### Ejemplo 1

```
PERS sensor AnyDevice;
PERS robdata DataOut := [[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]];
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
```

Al llamar a la rutina `RRI_Open`, en primer lugar se abre una conexión al dispositivo llamado `AnyDevice`. A continuación, se inicia una transmisión cíclica con una frecuencia de `SampleRate`.

#### Sintaxis

```
SiSetCyclic
    [ Sensor ':= ' ] < persistent (PERS) of sensor > ', '
    [ Data ':= ' ] < persistent (PERS) of anytype >
    [ Rate ':= ' ] < expression (IN) of num > ] ';' ;'
```

#### Información relacionada

| Para obtener más información sobre                    | Consulte  |
|---|---|
| Establecimiento de una conexión a un sistema externo. | <a href="#">SiConnect - Conexión a Sensor Interface en la página 722.</a>                     |
| Cierre de una conexión a un sistema externo.          | <a href="#">SiClose - Cierre de Sensor Interface en la página 725.</a>                        |
| Suscripción a la transmisión cíclica de datos.        | <a href="#">SiGetCyclic - Obtención de modo cíclico de Sensor Interface en la página 727.</a> |
| Descriptor del dispositivo externo.                   | <a href="#">sensor - Descriptor de dispositivo externo en la página 1754.</a>                 |
| Estado de comunicación de un dispositivo.             | <a href="#">sensorstate - Estado de comunicación del dispositivo en la página 1756.</a>       |
| <i>Robot Reference Interface</i>                      | <i>Application manual - Controller software IRC5</i>  |

## 1.245 SkipWarn - Omitir el último aviso

### Utilización

SkipWarn(*Skip Warning*) se utiliza para omitir el último mensaje de advertencia generado, con lo que no se almacena en el registro de eventos durante la ejecución en el modo continuo o cíclico (en los modos paso a paso hacia delante o hacia atrás no se omite ninguna advertencia).

SkipWarn hace posible realizar una recuperación de errores repetitiva en RAPID sin llenar el registro de eventos con mensajes que sólo son de aviso.

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción SkipWarn:

#### Ejemplo 1

```

%"notexistingproc"%;
nextinstruction;
ERROR
IF ERRNO = ERR_REFUNKPRC THEN
  SkipWarn;
  TRYNEXT;
ENDIF
ENDPROC

```

El programa ejecuta la instrucción siguiente `nextinstruction` y no se almacena ningún mensaje de aviso en el registro de eventos.

### Sintaxis

```
SkipWarn ';' ;'
```

### Información relacionada

| Para obtener más información sobre | Consulte   |
|------------------------------------|--|
| Recuperación en caso de error      | <i>Manual de referencia técnica - RAPID Overview</i><br><i>Manual de referencia técnica - RAPID Overview</i> |
| Número de error                    | <a href="#">errnum - Número de error en la página 1664</a>   |

# 1 Instrucciones

---

## 1.246 SocketAccept - Aceptar una conexión entrante

### Socket Messaging

## 1.246 SocketAccept - Aceptar una conexión entrante

---

### Utilización

SocketAccept se utiliza para aceptar peticiones de conexión entrantes.  
SocketAccept sólo puede usarse con aplicaciones de servidor.

---

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción SocketAccept:  
Consulte también [Más ejemplos en la página 737](#).

#### Ejemplo 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;
```

Se crea un zócalo de servidor, que se enlaza al puerto 1025 en la dirección de la red del controlador 192.168.0.1. Tras la ejecución de SocketListen, el zócalo del servidor inicia la escucha de conexiones entrantes en este puerto y esta dirección. SocketAccept permanece a la espera de conexiones entrantes, acepta la petición de conexión y devuelve un zócalo de cliente para la conexión establecida.

---

### Argumentos

```
SocketAccept Socket ClientSocket [\ClientAddress] [ \Time ]
```

Socket

Tipo de dato: socketdev

El zócalo de servidor que permanece a la espera de conexiones entrantes. El zócalo debe estar ya creado, enlazado y preparado para la escucha.

ClientSocket

Tipo de dato: socketdev

El nuevo zócalo de cliente devuelto, que se actualizará con la petición de conexión entrante aceptada.

[\ClientAddress]

Tipo de dato: string

La variable que se actualizará con la dirección IP de la petición de conexión entrante aceptada.

[\Time]

Tipo de dato: num

El tiempo máximo [s] que debe esperar la ejecución del programa a que se produzca una conexión entrante. Si el tiempo se agota antes de que se produzca una

*Continúa en la página siguiente*

conexión entrante, se llama al gestor de errores, si lo hay, con el código de error `ERR_SOCK_TIMEOUT`. Si no hay ningún gestor de errores, se detiene la ejecución.

Si no se usa el parámetro `\Time` el tiempo de espera es de 60 s. Para esperar indefinidamente, utilice la constante predefinida `WAIT_MAX`.

### Ejecución de programas

El zócalo de servidor permanece a la espera de peticiones de conexión entrantes. Una vez aceptada la petición de conexión entrante, la instrucción ha finalizado y el zócalo de cliente devuelto está conectado de forma predeterminada y puede usarse en instrucciones `SocketSend` y `SocketReceive`.

### Más ejemplos

A continuación aparecen más ejemplos de la instrucción `SocketAccept`.

#### Ejemplo 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;
```

Se crea un zócalo de servidor, que se enlaza al puerto 1025 en la dirección de la red del controlador 192.168.0.1. Tras la ejecución de `SocketListen`, el zócalo de servidor inicia la escucha de conexiones entrantes en este puerto y dirección. `SocketAccept` aceptará la conexión entrante de algún cliente y guardará la dirección del cliente en la cadena `client_ip`. A continuación, el servidor recibe un mensaje de cadena del cliente y guarda el mensaje en `receive_string`. A continuación, el cliente responde con el mensaje " Hello client with ip-address xxx.xxx.x.x" y cierra la conexión de cliente.

A partir de ese momento, el servidor está preparado para una conexión desde el mismo cliente o desde otro cliente, dentro del bucle `WHILE`. Si se mueve el puntero

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.246 SocketAccept - Aceptar una conexión entrante

### Socket Messaging

#### Continuación

de programa a Main en el programa, se cierran todos los zócalos (`SocketClose` puede ejecutarse siempre, incluso si el zócalo no está creado).

#### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                          | Causa del error   |
|---------------------------------|---|
| <code>ERR_SOCKET_CLOSED</code>  | El zócalo está cerrado (ha sido cerrado o no ha sido creado). Utilice <code>SocketCreate</code> para crear un nuevo zócalo. |
| <code>ERR_SOCKET_TIMEOUT</code> | La conexión no fue establecida antes de alcanzar el tiempo límite   |
| <code>ERR_SOCKET_UNSPEC</code>  | Excepción no especificada de la llamada subyacente al sistema operativo.  |

#### Sintaxis

```
SocketAccept
[Socket ':='] <variable (VAR) of socketdev> ','
[ClientSocket ':='] <variable (VAR) of socketdev>
['\ ' ClientAddress ':=' <variable (VAR) of string>]
['\ ' Time ':=' <expression (IN) of num>'];'
```

#### Información relacionada

| Para obtener más información sobre                         | Consulte   |
|--|--|
| Comunicación con zócalos en general                        | <a href="#">Application manual - Controller software IRC5, sección Socket Messaging</a>        |
| Creación de un nuevo zócalo                                | <a href="#">SocketCreate - Crea un nuevo zócalo en la página 747</a>                           |
| Conexión a un ordenador remoto (sólo cliente)              | <a href="#">SocketConnect - Establece una conexión a un ordenador remoto en la página 744</a>  |
| Envío de datos a un ordenador remoto                       | <a href="#">SocketSend - Envía datos a un ordenador remoto en la página 761</a>                |
| Recepción de datos desde un ordenador remoto               | <a href="#">SocketReceive - Recibe datos de un ordenador remoto en la página 751</a>           |
| Cierre del zócalo  | <a href="#">SocketClose - Cerrar un zócalo en la página 742</a>                                |
| Enlazamiento de un zócalo (sólo servidor)                  | <a href="#">SocketBind - Enlazar un zócalo a mi dirección IP y puerto en la página 739</a>     |
| Cómo permanecer a la escucha de conexiones (sólo servidor) | <a href="#">SocketListen - Permanece a la escucha de conexiones entrantes en la página 749</a> |
| Obtención del estado actual del zócalo                     | <a href="#">SocketGetStatus - Obtiene el estado actual de un zócalo en la página 1473</a>      |
| Aplicación de ejemplo de zócalos de cliente                | <a href="#">SocketSend - Envía datos a un ordenador remoto en la página 761</a>                |
| Ejemplo de aplicación de zócalos de servidor               | <a href="#">SocketReceive - Recibe datos de un ordenador remoto en la página 751</a>           |

### 1.247 SocketBind - Enlazar un zócalo a mi dirección IP y puerto

#### Utilización

`SocketBind` se usa para enlazar un zócalo al número de puerto de servidor y la dirección IP y el número de puerto especificados. `SocketBind` sólo puede usarse con aplicaciones de servidor.

#### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción `SocketBind`:

#### Ejemplo 1

```
VAR socketdev server_socket;  
  
SocketCreate server_socket;  
SocketBind server_socket, "192.168.0.1", 1025;
```

Se crea un zócalo de servidor, que se enlaza al puerto 1025 en la dirección de la red del controlador 192.168.0.1. A partir de este momento, es posible usar el zócalo de servidor en una instrucción `SocketListen` para permanecer a la escucha de conexiones entrantes en este puerto y dirección.

#### Argumentos

```
SocketBind Socket LocalAddress LocalPort
```

#### Socket

Tipo de dato: `socketdev`

El zócalo de servidor a enlazar. El zócalo ya debe estar creado pero no enlazado aún.

#### LocalAddress

Tipo de dato: `string`

La dirección de red de servidor a la que se desea enlazar el zócalo. Las únicas direcciones válidas son las direcciones de WAN públicas o la dirección del puerto de servicio del controlador 192.168.125.1.

#### LocalPort

Tipo de dato: `num`

El número de puerto de servidor al que se desea enlazar el zócalo. Por lo general, los puertos del 1025 al 4999 están libres para su uso.

#### Ejecución de programas

Se enlaza el zócalo de servidor al puerto de servidor y la dirección IP especificados. Si el puerto especificado ya se está utilizando, se genera un error.

Utilice las instrucciones `SocketBind` y `SocketListen` en la puesta en marcha del programa para asociar una dirección local a un zócalo y a continuación permanecer a la espera de conexiones entrantes en el puerto especificado. Se recomienda hacerlo sólo una vez con cada zócalo y puerto utilizado (TCP/IP).

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.247 SocketBind - Enlazar un zócalo a mi dirección IP y puerto

### Socket Messaging

#### Continuación

Utilice la instrucción `SocketBind` al recibir datos con `SocketReceiveFrom` (UDP/IP).

#### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                             | Causa del error   |
|------------------------------------|---|
| <code>ERR_SOCK_ADDR_INVALID</code> | La dirección especificada no es válida.   |
| <code>ERR_SOCK_ADDR_INUSE</code>   | La dirección y el puerto ya se están utilizando y no pueden utilizarse nuevamente. Utilice otro número de puerto.           |
| <code>ERR_SOCK_IS_BOUND</code>     | El zócalo ya ha sido vinculado a una dirección y no puede volver a vincularse.  |
| <code>ERR_SOCK_CLOSED</code>       | El zócalo está cerrado (ha sido cerrado o no ha sido creado). Utilice <code>SocketCreate</code> para crear un nuevo zócalo. |
| <code>ERR_SOCK_UNSPEC</code>       | Excepción no especificada de la llamada subyacente al sistema operativo.  |

#### Sintaxis

```
SocketBind
[Socket ':='] <variable (VAR) of socketdev>','
[LocalAddress ':='] <expression (IN) of string>','
[LocalPort ':='] <expression (IN) of num>;'
```

#### Información relacionada

| Para obtener más información sobre                         | Consulte   |
|--|--|
| Comunicación con zócalos en general                        | <a href="#">Application manual - Controller software IRC5</a>                                  |
| Creación de un nuevo zócalo                                | <a href="#">SocketCreate - Crea un nuevo zócalo en la página 747</a>                           |
| Conexión a un ordenador remoto (sólo cliente)              | <a href="#">SocketConnect - Establece una conexión a un ordenador remoto en la página 744</a>  |
| Envío de datos a un ordenador remoto                       | <a href="#">SocketSend - Envía datos a un ordenador remoto en la página 761</a>                |
| Recepción de datos desde un ordenador remoto               | <a href="#">SocketReceive - Recibe datos de un ordenador remoto en la página 751</a>           |
| Cierre del zócalo  | <a href="#">SocketClose - Cerrar un zócalo en la página 742</a>                                |
| Cómo permanecer a la escucha de conexiones (sólo servidor) | <a href="#">SocketListen - Permanece a la escucha de conexiones entrantes en la página 749</a> |
| Aceptación de conexiones (sólo servidor)                   | <a href="#">SocketAccept - Aceptar una conexión entrante en la página 736</a>                  |
| Obtención del estado actual del zócalo                     | <a href="#">SocketGetStatus - Obtiene el estado actual de un zócalo en la página 1473</a>      |
| Aplicación de ejemplo de zócalos de cliente                | <a href="#">SocketSend - Envía datos a un ordenador remoto en la página 761</a>                |
| Aplicación de ejemplo de zócalos de servidor               | <a href="#">SocketReceive - Recibe datos de un ordenador remoto en la página 751</a>           |

Continúa en la página siguiente



## 1.247 SocketBind - Enlazar un zócalo a mi dirección IP y puerto

*Socket Messaging*

*Continuación*

| Para obtener más información sobre           | Consulte  |
|--|---|
| Recepción de datos desde un ordenador remoto | <a href="#">SocketReceiveFrom - Recepción de datos desde un ordenador remoto en la página 756</a> |

# 1 Instrucciones

---

## 1.248 SocketClose - Cerrar un zócalo

*Socket Messaging*

## 1.248 SocketClose - Cerrar un zócalo

---

### Utilización

`SocketClose` se utiliza cuando ya no se va a utilizar una conexión de zócalo. Una vez cerrado un zócalo, no es posible utilizarlo en ninguna llamada a zócalo, excepto `SocketCreate`.

---

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción `SocketClose`:

#### Ejemplo 1

```
SocketClose socket1;
```

El zócalo se cierra y no es posible seguir utilizándolo.

---

### Argumentos

```
SocketClose Socket
```

Socket

Tipo de dato: `socketdev`

El zócalo que se desea cerrar.

---

### Ejecución de programas

El zócalo se cerrará y se liberarán los recursos que tenga asignados.

Es posible cerrar cualquier zócalo en cualquier momento. Tras su cierre, no es posible usar el zócalo. Puede ser reutilizado para una nueva conexión tras una llamada a `SocketCreate`.

---

### Limitaciones

El cierre de la conexión del zócalo inmediatamente después de enviar los datos con `SocketSend` puede dar lugar a la pérdida de datos de envío. Esto se debe a que el zócalo TCP/IP tiene una funcionalidad incorporada para reenviar los datos si hay cualquier problema de comunicación.

Para evitar estos problemas con pérdida de datos, haga lo siguiente antes de `SocketClose`:

- Intercambie de indicativos de control para el cierre o
- `WaitTime 2`

Evite utilizar bucles rápidos con `SocketCreate ... SocketClose`, dado que el zócalo no queda realmente cerrado hasta un cierto tiempo después (funcionalidad de TCP/IP).

---

### Sintaxis

```
SocketClose  
[Socket ':='] <variable (VAR) of socketdev>;'
```

---

*Continúa en la página siguiente*

## Información relacionada

| Para obtener más información sobre                         | Consulte  |
|--|---|
| Comunicación con zócalos en general                        | <i>Application manual - Controller software IRC5, sección Socket Messaging</i>                    |
| Creación de un nuevo zócalo                                | <a href="#">SocketCreate - Crea un nuevo zócalo en la página 747</a>                              |
| Conexión a un ordenador remoto (sólo cliente)              | <a href="#">SocketConnect - Establece una conexión a un ordenador remoto en la página 744</a>     |
| Envío de datos a un ordenador remoto                       | <a href="#">SocketSend - Envía datos a un ordenador remoto en la página 761</a>                   |
| Recepción de datos desde un ordenador remoto               | <a href="#">SocketReceive - Recibe datos de un ordenador remoto en la página 751</a>              |
| Enlazamiento de un zócalo (sólo servidor)                  | <a href="#">SocketBind - Enlazar un zócalo a mi dirección IP y puerto en la página 739</a>        |
| Cómo permanecer a la escucha de conexiones (sólo servidor) | <a href="#">SocketListen - Permanece a la escucha de conexiones entrantes en la página 749</a>    |
| Aceptación de conexiones (sólo servidor)                   | <a href="#">SocketAccept - Aceptar una conexión entrante en la página 736t</a>                    |
| Obtención del estado actual del zócalo                     | <a href="#">SocketGetStatus - Obtiene el estado actual de un zócalo en la página 1473</a>         |
| Aplicación de ejemplo de zócalos de cliente                | <a href="#">SocketSend - Envía datos a un ordenador remoto en la página 761</a>                   |
| Envío de datos a un ordenador remoto                       | <a href="#">SocketSendTo - Envío de datos a un ordenador remoto en la página 765</a>              |
| Aplicación de ejemplo de zócalos de servidor               | <a href="#">SocketReceive - Recibe datos de un ordenador remoto en la página 751</a>              |
| Recepción de datos desde un ordenador remoto               | <a href="#">SocketReceiveFrom - Recepción de datos desde un ordenador remoto en la página 756</a> |

# 1 Instrucciones

---

## 1.249 SocketConnect - Establece una conexión a un ordenador remoto

### Socket Messaging

## 1.249 SocketConnect - Establece una conexión a un ordenador remoto

---

### Utilización

`SocketConnect` se utiliza para conectar el zócalo a un ordenador remoto en una aplicación cliente.

---

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción `SocketConnect`:  
Consulte también [Más ejemplos en la página 745](#).

#### Ejemplo 1

```
SocketConnect socket1, "192.168.0.1", 1025;
```

Se intenta la conexión a un ordenador remoto en la dirección IP 192.168.0.1 y con el puerto 1025.

---

### Argumentos

```
SocketConnect Socket Address Port [\Time]
```

#### Socket

Tipo de dato: `socketdev`

El zócalo de cliente al que se desea conectar. El zócalo ya debe estar creado pero no conectado aún.

#### Address

Tipo de dato: `string`

La dirección del ordenador remoto. El ordenador remoto debe estar especificado como una dirección IP. No es posible usar el nombre del ordenador remoto.

#### Port

Tipo de dato: `num`

El puerto del ordenador remoto. Por lo general, los puertos del 1025 al 4999 están libres para su uso. Es posible que los puertos con un número inferior al 1025 ya estén ocupados.

#### [ \Time ]

Tipo de dato: `num`

El intervalo máximo [s] que debe esperar la ejecución del programa a que se acepte o deniegue la conexión. Si el tiempo se agota antes de que se cumpla la condición, se llama al gestor de errores si lo hay, con el código de error `ERR_SOCK_TIMEOUT`. Si no hay ningún gestor de errores, se detiene la ejecución.

Si no se usa el parámetro `\Time` el tiempo de espera es de 60 s. Para esperar ininterrumpidamente, utilice la constante predefinida `WAIT_MAX`.

---

### Ejecución de programas

El zócalo intenta establecer una conexión con el ordenador remoto a través de la dirección y el puerto especificados. La ejecución del programa esperará hasta que la conexión sea establecida o falle o hasta que se produzca un error de tiempo límite.

*Continúa en la página siguiente*

---

**Más ejemplos**

A continuación aparecen más ejemplos de la instrucción `SocketConnect`.

**Ejemplo 1**

```

VAR num retry_no := 0;
VAR socketdev my_socket;
...
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

```

Se crea un zócalo y se intenta establecer una conexión a un ordenador remoto. Si la conexión no se establece dentro del tiempo límite predeterminado, es decir 60 segundos, el gestor de errores reintenta la conexión. Se realizan cuatro reintentos, tras lo cual se informa del error al usuario.

**Gestión de errores**

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                             | Causa del error   |
|------------------------------------|---|
| <code>ERR_SOCK_ADDR_INVALID</code> | La dirección especificada no es válida  |
| <code>ERR_SOCK_CLOSED</code>       | El zócalo está cerrado (ha sido cerrado o no ha sido creado). Utilice <code>SocketCreate</code> para crear un nuevo zócalo. |
| <code>ERR_SOCK_IS_CONN</code>      | El zócalo está conectado.   |
| <code>ERR_SOCK_NET_UNREACH</code>  | No se puede acceder a la red o la conexión se pierde después de abrirse un zócalo.  |
| <code>ERR_SOCK_TIMEOUT</code>      | La conexión no fue establecida antes de alcanzar el tiempo límite.  |
| <code>ERR_SOCK_UNSPEC</code>       | Excepción no especificada de la llamada subyacente al sistema operativo.  |

**Sintaxis**

```

SocketConnect
  [Socket ':='] <variable (VAR) of socketdev>', '
  [Address ':='] <expression (IN) of string>', '
  [Port ':='] <expression (IN) of num>
  ['\ ' Time ':='] <expression (IN) of num>';'

```

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.249 SocketConnect - Establece una conexión a un ordenador remoto

### Socket Messaging

#### Continuación

#### Información relacionada

| Para obtener más información sobre                         | Descrito en:   |
|--|--|
| Comunicación con zócalos en general                        | <a href="#">Application manual - Controller software IRC5</a>                                  |
| Creación de un nuevo zócalo                                | <a href="#">SocketCreate - Crea un nuevo zócalo en la página 747</a>                           |
| Envío de datos a un ordenador remoto                       | <a href="#">SocketSend - Envía datos a un ordenador remoto en la página 761</a>                |
| Recepción de datos desde un ordenador remoto               | <a href="#">SocketReceive - Recibe datos de un ordenador remoto en la página 751</a>           |
| Enlazamiento de un zócalo (sólo servidor)                  | <a href="#">SocketBind - Enlazar un zócalo a mi dirección IP y puerto en la página 739</a>     |
| Cómo permanecer a la escucha de conexiones (sólo servidor) | <a href="#">SocketListen - Permanece a la escucha de conexiones entrantes en la página 749</a> |
| Aceptación de conexiones (sólo servidor)                   | <a href="#">SocketAccept - Aceptar una conexión entrante en la página 736</a>                  |
| Obtención del estado actual del zócalo                     | <a href="#">SocketGetStatus - Obtiene el estado actual de un zócalo en la página 1473</a>      |
| Aplicación de ejemplo de zócalos de cliente                | <a href="#">SocketSend - Envía datos a un ordenador remoto en la página 761</a>                |
| Aplicación de ejemplo de zócalos de servidor               | <a href="#">SocketReceive - Recibe datos de un ordenador remoto en la página 751</a>           |

### 1.250 SocketCreate - Crea un nuevo zócalo

#### Utilización

`SocketCreate` se usa para crear un nuevo zócalo para una comunicación basada en conexiones o una comunicación sin conexiones.

Se admiten tanto el intercambio de mensajes de zócalo de tipo canal con protocolo TCP/IP y con garantía de entrega como el protocolo de datagramas UDP/IP. Es posible desarrollar aplicaciones tanto de servidor como de cliente. En el caso del protocolo de datagramas UDP/IP, se admite la difusión.

#### Ejemplos básicos

Los siguientes ejemplos ilustran la función `SocketCreate`:

##### Ejemplo 1

```
VAR socketdev socket1;  
...  
SocketCreate socket1;
```

Se crea un nuevo dispositivo de zócalo utilizando el tipo canal con protocolo TCP/IP que se asigna a la variable `socket1`.

##### Ejemplo 2

```
VAR socketdev udp_sock1;  
...  
SocketCreate udp_sock1 \UDP;
```

Se crea un nuevo dispositivo de zócalo utilizando el protocolo de datagramas UDP/IP y se asigna el zócalo a la variable `udp_sock1`.

#### Argumentos

```
SocketCreate Socket [ \UDP ]
```

Socket

Tipo de dato: `socketdev`

La variable de almacenamiento de los datos de zócalo internos del sistema.

[ \UDP ]

Tipo de dato: `switch`

Especifica que el zócalo debe ser del tipo del protocolo de datagramas UDP/IP.

#### Ejecución de programas

La instrucción crea un nuevo dispositivo de zócalo.

El zócalo no debe estar usándose. El zócalo está en uso entre las instrucciones `SocketCreate` y `SocketClose`.

#### Limitaciones

Es posible declarar tantos zócalos como se desee, pero sólo es posible utilizar 32 de ellos al mismo tiempo.

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.250 SocketCreate - Crea un nuevo zócalo

### Socket Messaging

#### Continuación

Evite utilizar bucles rápidos con `SocketCreate ... SocketClose`, dado que el zócalo no queda realmente cerrado hasta un cierto tiempo después (al usar la funcionalidad de TCP/IP).

### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                       | Causa del error  |
|------------------------------|--|
| <code>ERR_SOCK_UNSPEC</code> | Excepción no especificada de la llamada subyacente al sistema operativo. |

### Sintaxis

```
SocketCreate  
  [Socket ':'='] <variable (VAR) of socketdev>  
  ['\' UDP'] ; ;
```

### Información relacionada

| Para obtener más información sobre                         | Consulte  |
|--|---|
| Comunicación con zócalos en general                        | <a href="#">Application manual - Controller software IRC5, sección Socket Messaging</a>           |
| Conexión a un ordenador remoto (sólo cliente)              | <a href="#">SocketConnect - Establece una conexión a un ordenador remoto en la página 744</a>     |
| Envío de datos a un ordenador remoto                       | <a href="#">SocketSend - Envía datos a un ordenador remoto en la página 761</a>                   |
| Recepción de datos desde un ordenador remoto               | <a href="#">SocketReceive - Recibe datos de un ordenador remoto en la página 751</a>              |
| Cierre del zócalo  | <a href="#">SocketClose - Cerrar un zócalo en la página 742</a>                                   |
| Enlazamiento de un zócalo (sólo servidor)                  | <a href="#">SocketBind - Enlazar un zócalo a mi dirección IP y puerto en la página 739</a>        |
| Cómo permanecer a la escucha de conexiones (sólo servidor) | <a href="#">SocketListen - Permanece a la escucha de conexiones entrantes en la página 749</a>    |
| Aceptación de conexiones (sólo servidor)                   | <a href="#">SocketAccept - Aceptar una conexión entrante en la página 736</a>                     |
| Obtención del estado actual del zócalo                     | <a href="#">SocketGetStatus - Obtiene el estado actual de un zócalo en la página 1473</a>         |
| Aplicación de ejemplo de zócalos de cliente                | <a href="#">SocketSend - Envía datos a un ordenador remoto en la página 761</a>                   |
| Envío de datos a un ordenador remoto                       | <a href="#">SocketSendTo - Envío de datos a un ordenador remoto en la página 765</a>              |
| Aplicación de ejemplo de zócalos de servidor               | <a href="#">SocketReceive - Recibe datos de un ordenador remoto en la página 751</a>              |
| Recepción de datos desde un ordenador remoto               | <a href="#">SocketReceiveFrom - Recepción de datos desde un ordenador remoto en la página 756</a> |



### 1.251 SocketListen - Permanece a la escucha de conexiones entrantes

#### Utilización

`SocketListen` se usa para iniciar la escucha de conexiones entrantes, por ejemplo para empezar a actuar como servidor. `SocketListen` sólo puede usarse para las aplicaciones de servidor.

#### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción `SocketListen`:

#### Ejemplo 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;
```

Se crea un zócalo de servidor, que se enlaza al puerto 1025 en la dirección de la red del controlador 192.168.0.1. Tras la ejecución de `SocketListen`, el zócalo del servidor inicia la escucha de conexiones entrantes en este puerto y esta dirección.

#### Argumentos

`SocketListen` Socket

Socket

Tipo de dato: socketdev

El zócalo de servidor que debe iniciar la escucha de conexiones entrantes. El zócalo ya debe estar creado y enlazado.

#### Ejecución de programas

El zócalo de servidor inicia la escucha de conexiones entrantes. Cuando la instrucción ha finalizado, el zócalo está preparado para aceptar una conexión entrante.

Utilice las instrucciones `SocketBind` y `SocketListen` en la puesta en marcha del programa para asociar una dirección local a un zócalo y a continuación permanecer a la espera de conexiones entrantes en el puerto especificado. Se recomienda hacerlo sólo una vez con cada zócalo y puerto utilizado.

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.251 SocketListen - Permanece a la escucha de conexiones entrantes

### Socket Messaging

#### Continuación

#### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                            | Causa del error   |
|-----------------------------------|---|
| <code>ERR_SOCKET_CLOSED</code>    | El zócalo está cerrado (ha sido cerrado o no ha sido creado). Utilice <code>SocketCreate</code> para crear un nuevo zócalo. |
| <code>ERR_SOCKET_IS_CONN</code>   | El zócalo está conectado.   |
| <code>ERR_SOCKET_NOT_BOUND</code> | El zócalo no ha sido vinculado a una dirección.   |
| <code>ERR_SOCKET_UNSPEC</code>    | Excepción no especificada de la llamada subyacente al sistema operativo.  |

#### Sintaxis

```
SocketListen  
[Socket ':'='] <variable (VAR) of socketdev>;'
```

#### Información relacionada

| Para obtener más información sobre            | Consulte  |
|---|---|
| Comunicación con zócalos en general           | <a href="#">Application manual - Controller software IRC5</a>                                 |
| Creación de un nuevo zócalo                   | <a href="#">SocketCreate - Crea un nuevo zócalo en la página 747</a>                          |
| Conexión a un ordenador remoto (sólo cliente) | <a href="#">SocketConnect - Establece una conexión a un ordenador remoto en la página 744</a> |
| Envío de datos a un ordenador remoto          | <a href="#">SocketSend - Envía datos a un ordenador remoto en la página 761</a>               |
| Recepción de datos desde un ordenador remoto  | <a href="#">SocketReceive - Recibe datos de un ordenador remoto en la página 751</a>          |
| Cierre del zócalo                             | <a href="#">SocketClose - Cerrar un zócalo en la página 742</a>                               |
| Enlazamiento de un zócalo (sólo servidor)     | <a href="#">SocketBind - Enlazar un zócalo a mi dirección IP y puerto en la página 739</a>    |
| Aceptación de conexiones (sólo servidor)      | <a href="#">SocketAccept - Aceptar una conexión entrante en la página 736</a>                 |
| Obtención del estado actual del zócalo        | <a href="#">SocketGetStatus - Obtiene el estado actual de un zócalo en la página 1473</a>     |
| Aplicación de ejemplo de zócalos de cliente   | <a href="#">SocketSend - Envía datos a un ordenador remoto en la página 761</a>               |
| Aplicación de ejemplo de zócalos de servidor  | <a href="#">SocketReceive - Recibe datos de un ordenador remoto en la página 751</a>          |

---

## 1.252 SocketReceive - Recibe datos de un ordenador remoto

---

### Utilización

SocketReceive se usa para recibir datos de un ordenador remoto. SocketReceive puede usarse tanto en aplicaciones de cliente como en aplicaciones de servidor.

---

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción SocketReceive: Consulte también [Más ejemplos en la página 753](#).

#### Ejemplo 1

```
VAR string str_data;
...
SocketReceive socket1 \Str := str_data;
```

Se reciben datos de un ordenador remoto y se guardan en la variable de cadena str\_data.

---

### Argumentos

```
SocketReceive Socket [ \Str ] | [ \RawData ] | [ \Data ]
                [\ReadNoOfBytes] [\NoRecBytes] [\Time]
```

Socket

Tipo de dato: socketdev

En la aplicación cliente en la que el zócalo recibe los datos, el zócalo debe estar ya creado y conectado.

En la aplicación de servidor en la que el zócalo recibe los datos, el zócalo debe estar ya aceptado.

[ \Str ]

Tipo de dato: string

La variable en la que se debe almacenar el dato de tipo string. Es posible gestionar un número máximo de 80 caracteres.

[ \RawData ]

Tipo de dato: rawbytes

La variable en la que se debe almacenar el dato de tipo rawbytes. Es posible gestionar un número máximo de 1.024 rawbytes.

[ \Data ]

Tipo de dato: array of byte

La variable en la que se debe almacenar el dato de tipo byte. Es posible gestionar un número máximo de 1.024 byte.

Sólo es posible usar uno de los parámetros opcionales \Str, \RawData y \Data al mismo tiempo.

[ \ReadNoOfBytes ]

*Read number of Bytes*

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.252 SocketReceive - Recibe datos de un ordenador remoto

### Socket Messaging

#### Continuación

Tipo de dato: num

El número de bytes a leer. El número mínimo de bytes a leer es 1 y el número máximo es el valor del tamaño del tipo de dato utilizado, es decir, 80 bytes si se utiliza una variable del tipo de dato `string`.

Si se está manteniendo una comunicación con un cliente que siempre envía un número fijo de bytes, este parámetro opcional puede usarse para especificar que se lea siempre el mismo número de bytes para cada instrucción `SocketReceive`.

Si el remitente envía datos `RawData`, el destinatario necesita especificar la recepción de 4 bytes por cada `rawbytes` enviado.

[ `\NoRecBytes` ]

#### Number Received Bytes

Tipo de dato: num

Una variable para el almacenamiento del número de bytes necesarios del `socketdev` especificado.

El mismo resultado también puede conseguirse con:

- Función `StrLen` con la variable del argumento `\Str`
- Función `RawBytesLen` con la variable del argumento `\RawData`

[ `\Time` ]

Tipo de dato: num

El intervalo máximo [s] que debe esperar la ejecución del programa a que se reciban los datos. Si el tiempo se agota antes de que se transfieran los datos, se llama al gestor de errores si lo hay, con el código de error `ERR_SOCKET_TIMEOUT`. Si no hay ningún gestor de errores, se detiene la ejecución.

Si no se usa el parámetro `\Time` el tiempo de espera es de 60 s. Para esperar indefinidamente, utilice la constante predefinida `WAIT_MAX`.

---

## Ejecución de programas

La ejecución de `SocketReceive` esperará hasta que los datos estén disponibles o fallen a causa de un error de tiempo límite.

El número de bytes leídos es especificado por el tipo de dato utilizado en la instrucción. Si se utiliza un tipo de dato `string` para recibir datos, se reciben los 80 bytes si en efecto hay 80 bytes a leer. Si se utiliza el argumento opcional `ReadNoOfBytes` el usuario puede especificar cuántos bytes deben recibirse para cada `SocketReceive`.

Los datos transferidos por el cable son siempre bytes, con un máximo de 1.024 bytes por cada mensaje. No se añade ningún encabezado al mensaje de forma predeterminada. El uso de cualquier encabezado está reservado para la aplicación en sí.

| Parámetro             | Datos de entrada | Datos de cable  | Datos de salida |
|-----------------------|------------------|-----------------|-----------------|
| <code>\Str</code>     | 1 carácter       | 1 byte (8 bits) | 1 carácter      |
| <code>\RawData</code> | 1 rawbytes       | 1 byte (8 bits) | 1 rawbytes      |
| <code>\Data</code>    | 1 byte           | 1 byte (8 bits) | 1 byte          |

Continúa en la página siguiente

Es posible combinar los tipos de datos (string, rawbytes, or array of byte) entre SocketSend y SocketReceive.

### Más ejemplos

A continuación aparecen más ejemplos de la instrucción SocketReceive.

#### Ejemplo 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;
  ! Wait for acknowlegde from client
  ...
  SocketClose server_socket;
  SocketClose client_socket;
ERROR
  IF ERRNO=ERR_SOCK_TIMEOUT THEN
    RETRY;
  ELSEIF ERRNO=ERR_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
```

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.252 SocketReceive - Recibe datos de un ordenador remoto

### Socket Messaging

#### Continuación

Éste es un ejemplo de un programa de servidor con creación, enlazamiento, escucha y aceptación de zócalos en los gestores de errores. De esta forma, el programa puede manejar el reinicio tras la caída de alimentación.

En el procedimiento `server_recover`, se crea un zócalo de servidor, que se enlaza al puerto 1025 en la dirección de la red del controlador 192.168.0.1. Tras la ejecución de `SocketListen`, el zócalo de servidor inicia la escucha de conexiones entrantes en este puerto y dirección. `SocketAccept` aceptará la conexión entrante de algún cliente y guardará la dirección del cliente en la cadena `client_ip`.

En el procedimiento de comunicación `server_messaging`, el servidor recibe un mensaje de cadena del cliente y guarda el mensaje en `receive_string`. A continuación, el cliente responde con el mensaje "Hello client with ip-address xxx.xxx.x.x".

---

### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                            | Causa del error  |
|-----------------------------------|--|
| <code>ERR_SOCK_CLOSED</code>      | El zócalo se cierra. Conexión interrumpida.  |
| <code>ERR_SOCK_NET_UNREACH</code> | No se puede acceder a la red o la conexión se pierde después de abrirse un zócalo. |
| <code>ERR_SOCK_NOT_CONN</code>    | El zócalo no está conectado  |
| <code>ERR_SOCK_TIMEOUT</code>     | No se recibieron datos dentro del tiempo límite.                                   |

---

### Limitaciones

La mensajería de zócalos no cuenta con ningún mecanismo de sincronización incorporado para evitar la recepción de mensajes compuestos por varios mensajes de envío. El programador es quien decide si desea gestionar la sincronización con mensajes "Ack" (debe completarse una secuencia de `SocketSend` - `SocketReceive` en el programa cliente o servidor antes de la siguiente secuencia de `SocketSend` - `SocketReceive`).

Todos los zócalos están cerrados tras el reinicio de la caída de alimentación. Este problema puede ser gestionado por la recuperación en caso de error. Consulte el ejemplo anterior.

Evite utilizar bucles rápidos con `SocketCreate ... SocketClose`, dado que el zócalo no queda realmente cerrado hasta un cierto tiempo después (funcionalidad de TCP/IP).

El tamaño máximo de los datos que pueden recibirse en una llamada está limitada a 1.024 bytes.

---

### Sintaxis

```
SocketReceive
[Socket ':='] <variable (VAR) of socketdev>
['\' Str ':='] <variable (VAR) of string>
[['\' RawData ':='] <variable (VAR) of rawbytes>]
```

*Continúa en la página siguiente*

## 1.252 SocketReceive - Recibe datos de un ordenador remoto

Socket Messaging

Continuación

```
[['\ ' Data ' := ' <array {*} (VAR) of byte>]
['\ ' ReadNoOfBytes ' := ' <expression (IN) of num>]
['\ ' NoRecBytes ' := ' <variable (VAR) of num>]
['\ ' Time ' := ' <expression (IN) of num>'];'
```

## Información relacionada

| Para obtener más información sobre                         | Consulte  |
|--|---|
| Comunicación con zócalos en general                        | <i>Application manual - Controller software IRC5</i>  |
| Creación de un nuevo zócalo                                | <a href="#">SocketCreate - Crea un nuevo zócalo en la página 747</a>                                    |
| Conexión a un ordenador remoto (sólo cliente)              | <a href="#">SocketConnect - Establece una conexión a un ordenador remoto en la página 744</a>           |
| Envío de datos a un ordenador remoto                       | <a href="#">SocketSend - Envía datos a un ordenador remoto en la página 761</a>                         |
| Cierre del zócalo  | <a href="#">SocketClose - Cerrar un zócalo en la página 742</a>   |
| Enlazamiento de un zócalo (sólo servidor)                  | <a href="#">SocketBind - Enlazar un zócalo a mi dirección IP y puerto en la página 739</a>              |
| Cómo permanecer a la escucha de conexiones (sólo servidor) | <a href="#">SocketListen - Permanece a la escucha de conexiones entrantes en la página 749</a>          |
| Aceptación de conexiones (sólo servidor)                   | <a href="#">SocketAccept - Aceptar una conexión entrante en la página 736</a>                           |
| Obtención del estado actual del zócalo                     | <a href="#">SocketGetStatus - Obtiene el estado actual de un zócalo en la página 1473</a>               |
| Aplicación de ejemplo de zócalos de cliente                | <a href="#">SocketSend - Envía datos a un ordenador remoto en la página 761</a>                         |
| Prueba para comprobar la presencia de datos en un zócalo.  | <a href="#">SocketPeek - Prueba para comprobar la presencia de datos en un zócalo en la página 1476</a> |

# 1 Instrucciones

---

## 1.253 SocketReceiveFrom - Recepción de datos desde un ordenador remoto

### Socket Messaging

## 1.253 SocketReceiveFrom - Recepción de datos desde un ordenador remoto

---

### Utilización

SocketReceiveFrom se usa para recibir datos de un ordenador remoto. SocketReceiveFrom puede usarse tanto en aplicaciones de cliente como en aplicaciones de servidor.

SocketReceiveFrom se utiliza para la comunicación sin conexiones con el protocolo de datagramas UDP/IP.

---

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción SocketReceiveFrom:

Consulte también [Más ejemplos en la página 753](#).

#### Ejemplo 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;
```

Se reciben datos de un ordenador remoto y se guardan en la variable de cadena str\_data. La dirección del ordenador remoto se almacena en la variable de cadena RemoteAddress y el número de puerto se almacena en la variable de tipo num RemotePort.

---

### Argumentos

```
SocketReceiveFrom Socket [ \Str ] | [ \RawData ] | [ \Data ]
[\ReadNoOfBytes] [\NoRecBytes] RemoteAddress RemotePort
[\Time]
```

Socket

**Tipo de dato:** socketdev

Un dispositivo de zócalo que identifica un zócalo enlazado.

[ \Str ]

**Tipo de dato:** string

La variable en la que se debe almacenar el dato de tipo string. Es posible gestionar un número máximo de 80 caracteres.

[ \RawData ]

**Tipo de dato:** rawbytes

La variable en la que se debe almacenar el dato de tipo rawbytes. Es posible gestionar un número máximo de 1.024 rawbytes.

[ \Data ]

**Tipo de dato:** array of byte

*Continúa en la página siguiente*

---



## 1.253 SocketReceiveFrom - Recepción de datos desde un ordenador remoto

*Socket Messaging*

*Continuación*

La variable en la que se debe almacenar el dato de tipo byte. Es posible gestionar un número máximo de 1.024 byte.

Sólo es posible usar uno de los parámetros opcionales `\Str`, `\RawData` y `\Data` al mismo tiempo.

[ `\ReadNoOfBytes` ]

*Read number of Bytes*

Tipo de dato: `num`

El número de bytes a leer. El número mínimo de bytes a leer es 1 y el número máximo es el valor del tamaño del tipo de dato utilizado, es decir, 80 bytes si se utiliza una variable del tipo de dato `string`.

Si se está manteniendo una comunicación con un cliente que siempre envía un número fijo de bytes, este parámetro opcional puede usarse para especificar que se lea siempre el mismo número de bytes para cada instrucción `SocketReceive`.

Si el remitente envía datos `RawData`, el destinatario necesita especificar la recepción de 4 bytes por cada `rawbytes` enviado.

[ `\NoRecBytes` ]

*Number Received Bytes*

Tipo de dato: `num`

Una variable para el almacenamiento del número de bytes necesarios del `socketdev` especificado.

El mismo resultado también puede conseguirse con:

- Función `StrLen` con la variable del argumento `\Str`
- Función `RawBytesLen` con la variable del argumento `\RawData`

`RemoteAddress`

Tipo de dato: `string`

Una variable de cadena que contiene la dirección de origen del ordenador remoto.

`RemotePort`

Tipo de dato: `num`

Una variable de tipo `num` que contiene el puerto utilizado por el ordenador remoto al enviar el paquete de datagrama.

[ `\Time` ]

Tipo de dato: `num`

El intervalo máximo [s] que debe esperar la ejecución del programa a que se reciban los datos. Si el tiempo se agota antes de que se transfieran los datos, se llama al gestor de errores si lo hay, con el código de error `ERR_SOCKET_TIMEOUT`. Si no hay ningún gestor de errores, se detiene la ejecución.

Si no se usa el parámetro `\Time` el tiempo de espera es de 60 s. Para esperar indefinidamente, utilice la constante predefinida `WAIT_MAX`.

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.253 SocketReceiveFrom - Recepción de datos desde un ordenador remoto

### Socket Messaging

#### Continuación

#### Ejecución de programas

La ejecución de `SocketReceiveFrom` recibe un datagrama y almacena la dirección y el puerto de origen. Esperará hasta que los datos estén disponibles o fallará con un error de tiempo límite.

El número de bytes leídos es especificado por el tipo de dato utilizado en la instrucción. Si se utiliza un tipo de dato `string` para recibir datos, se reciben 80 bytes si en efecto hay 80 bytes a leer.

Los datos transferidos por el cable son siempre bytes, con un máximo de 1.024 bytes por cada mensaje. No se añade ningún encabezado al mensaje de forma predeterminada. El uso de cualquier encabezado está reservado para la aplicación en sí.

| Parámetro             | Datos de entrada | Datos de cable  | Datos de salida |
|-----------------------|------------------|-----------------|-----------------|
| <code>\Str</code>     | 1 carácter       | 1 byte (8 bits) | 1 carácter      |
| <code>\RawData</code> | 1 rawbytes       | 1 byte (8 bits) | 1 rawbytes      |
| <code>\Data</code>    | 1 byte           | 1 byte (8 bits) | 1 byte          |

Es posible combinar los tipos de datos (`string`, `rawbytes`, or `array of byte`) entre `SocketSendTo` y `SocketReceiveFrom`.

#### Más ejemplos

A continuación aparecen más ejemplos de la instrucción `SocketReceiveFrom`.

#### Ejemplo 1

```
VAR socketdev udp_socket;
VAR string client_ip;
VAR num client_port;

PROC server_messaging()
  VAR string receive_string;
  ...
  ! 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
```

Continúa en la página siguiente

## 1.253 SocketReceiveFrom - Recepción de datos desde un ordenador remoto

### Socket Messaging

#### Continuación

```

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

```

Este es un ejemplo de un programa de servidor con creación y enlazamiento de zócalos en los gestores de errores. De esta forma, el programa puede manejar el reinicio tras la caída de alimentación.

En el procedimiento de comunicación `server_messaging`, el servidor recibe un mensaje de cadena del cliente y guarda el mensaje en `receive_string`. A continuación, el cliente responde con el mensaje "Hello client with ip-address xxx.xxx.x.x".

### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                            | Causa del error  |
|-----------------------------------|--|
| <code>ERR_SOCK_CLOSED</code>      | El zócalo se cierra.   |
| <code>ERR_SOCK_NET_UNREACH</code> | No se puede acceder a la red o la conexión se pierde después de abrirse un zócalo. |
| <code>ERR_SOCK_NOT_BOUND</code>   | El zócalo no ha sido vinculado a una dirección.                                    |
| <code>ERR_SOCK_TIMEOUT</code>     | No se recibieron datos dentro del tiempo límite.                                   |

### Limitaciones

Todos los zócalos están cerrados tras el reinicio de la caída de alimentación. Este problema puede ser gestionado por la recuperación en caso de error. Consulte el ejemplo anterior.

El tamaño máximo de los datos que pueden recibirse en una llamada está limitada a 1.024 bytes.

### Sintaxis

```

SocketReceiveFrom
  [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>]
  [RemoteAddress ':='] <variable (VAR) of string>
  [RemotePort ':='] <variable (VAR) of num>
  ['\ Time ':=' <expression (IN) of num>'];'

```

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.253 SocketReceiveFrom - Recepción de datos desde un ordenador remoto

### Socket Messaging

#### Continuación

#### Información relacionada

| Para obtener más información sobre                         | Consulte  |
|--|---|
| Comunicación con zócalos en general                        | <a href="#">Application manual - Controller software IRC5</a>   |
| Creación de un nuevo zócalo                                | <a href="#">SocketCreate - Crea un nuevo zócalo en la página 747</a>                                    |
| Conexión a un ordenador remoto (sólo cliente)              | <a href="#">SocketConnect - Establece una conexión a un ordenador remoto en la página 744</a>           |
| Envío de datos a un ordenador remoto                       | <a href="#">SocketSend - Envía datos a un ordenador remoto en la página 761</a>                         |
| Cierre del zócalo  | <a href="#">SocketClose - Cerrar un zócalo en la página 742</a>   |
| Enlazamiento de un zócalo (sólo servidor)                  | <a href="#">SocketBind - Enlazar un zócalo a mi dirección IP y puerto en la página 739</a>              |
| Cómo permanecer a la escucha de conexiones (sólo servidor) | <a href="#">SocketListen - Permanece a la escucha de conexiones entrantes en la página 749</a>          |
| Aceptación de conexiones (sólo servidor)                   | <a href="#">SocketAccept - Aceptar una conexión entrante en la página 736</a>                           |
| Obtención del estado actual del zócalo                     | <a href="#">SocketGetStatus - Obtiene el estado actual de un zócalo en la página 1473</a>               |
| Aplicación de ejemplo de zócalos de cliente                | <a href="#">SocketSend - Envía datos a un ordenador remoto en la página 761</a>                         |
| Envío de datos a un ordenador remoto                       | <a href="#">SocketSendTo - Envío de datos a un ordenador remoto en la página 765</a>                    |
| Prueba para comprobar la presencia de datos en un zócalo.  | <a href="#">SocketPeek - Prueba para comprobar la presencia de datos en un zócalo en la página 1476</a> |

---

## 1.254 SocketSend - Envía datos a un ordenador remoto

---

### Utilización

SocketSend se usa para enviar datos a un ordenador remoto. SocketSend puede usarse tanto en aplicaciones de cliente como en aplicaciones de servidor.

---

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción SocketSend:  
Consulte también [Más ejemplos en la página 762](#).

#### Ejemplo 1

```
SocketSend socket1 \Str := "Hello world";
```

Envía el mensaje "Hello world" al ordenador remoto.

---

### Argumentos

```
SocketSend Socket [ \Str ] | [ \RawData ] | [ \Data ] [ \NoOfBytes ]
```

Socket

Tipo de dato: socketdev

En la aplicación cliente, el zócalo desde el que se envía debe estar ya creado y conectado.

En la aplicación de servidor, el zócalo al que se envía debe estar ya aceptado.

[ \Str ]

Tipo de dato: string

La cadena de tipo string que se desea enviar al ordenador remoto.

[ \RawData ]

Tipo de dato: rawbytes

El dato de tipo rawbytes que se desea enviar al ordenador remoto.

[ \Data ]

Tipo de dato: array of byte

Los datos de la matriz de tipo byte que se desea enviar al ordenador remoto.

Sólo es posible usar uno de los parámetros opcionales \Str, \RawData, o \Data al mismo tiempo.

[ \NoOfBytes ]

Tipo de dato: num

Si se especifica este argumento, sólo se envía al ordenador remoto este número de bytes. La llamada a SocketSend fallará si el valor de \NoOfBytes es mayor que la cantidad real de bytes de la estructura de datos a enviar.

Si no se especifica este argumento, se envía al ordenador remoto la totalidad de la estructura de datos (la parte válida de rawbytes).

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.254 SocketSend - Envía datos a un ordenador remoto

### Socket Messaging

#### Continuación

---

#### Ejecución de programas

Los datos especificados se envían al ordenador remoto. Si la conexión se interrumpe, se genera un error.

Los datos transferidos por el cable son siempre bytes, con un máximo de 1.024 bytes por cada mensaje. No se añade ningún encabezado al mensaje de forma predeterminada. El uso de cualquier encabezado está reservado para la aplicación en sí.

| Parámetro | Datos de entrada | Datos de cable  | Datos de salida |
|-----------|------------------|-----------------|-----------------|
| \Str      | 1 carácter       | 1 byte (8 bits) | 1 carácter      |
| \RawData  | 1 rawbytes       | 1 byte (8 bits) | 1 rawbytes      |
| \Data     | 1 byte           | 1 byte (8 bits) | 1 byte          |

Es posible combinar los tipos de datos (string, rawbytes o array of byte) entre SocketSend y SocketReceive.

---

#### Más ejemplos

A continuación aparecen más ejemplos de la instrucción SocketSend.

#### Ejemplo 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;
```

*Continúa en la página siguiente*

```

ELSE
    ! No error recovery handling
ENDIF
ENDPROC

```

Éste es un ejemplo de un programa cliente con creación y conexión de zócalos en los gestores de errores. De esta forma, el programa puede manejar el reinicio tras la caída de alimentación.

En el procedimiento `client_recover`, se crea y conecta el zócalo de cliente a un servidor de ordenador remoto con la dirección `192.168.0.2` en el puerto `1025`.

En el procedimiento de comunicación `client_messaging`, el cliente envía "Hello server" al servidor y el servidor responde con "Hello client" al cliente. Este mensaje se almacena en la variable `receive_string`.

### Ejemplo 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

```

Éste es un ejemplo de un programa cliente que envía caracteres no imprimibles (datos binarios) en una cadena. Esto puede resultar útil a la hora de comunicarse con sensores u otros clientes que requieran este tipo de caracteres.

### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                              | Causa del error  |
|-------------------------------------|--|
| <code>ERR_SOCKET_CLOSED</code>      | El zócalo se cierra. Conexión interrumpida.  |
| <code>ERR_SOCKET_NET_UNREACH</code> | No se puede acceder a la red o la conexión se pierde después de abrirse un zócalo. |
| <code>ERR_SOCKET_NOT_CONN</code>    | El zócalo no está conectado  |
| <code>ERR_SOCKET_UNSPEC</code>      | Excepción no especificada de la llamada subyacente al sistema operativo.           |

### Limitaciones

La *mensajería de zócalos* no cuenta con ningún mecanismo de sincronización incorporado para precaver la recepción de mensajes compuestos por varios mensajes de envío. El programador es quien decide si desea gestionar la sincronización con mensajes *Ack* (debe completarse una secuencia de `SocketSend` - `SocketReceive` en el programa cliente o servidor antes de la siguiente secuencia de `SocketSend` - `SocketReceive`).

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.254 SocketSend - Envía datos a un ordenador remoto

### Socket Messaging

#### Continuación

Todos los zócalos están cerrados tras el reinicio de la caída de alimentación. Este problema puede ser gestionado por la recuperación en caso de error. Consulte el ejemplo anterior.

Evite utilizar bucles rápidos con `SocketCreate ... SocketClose`, dado que el zócalo no queda realmente cerrado hasta un cierto tiempo después (funcionalidad de TCP/IP).

El tamaño de los datos a enviar está limitado a 1.024 bytes.

### Sintaxis

```
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>]';'
```

### Información relacionada

| Para obtener más información sobre   | Consulte   |
|--|--|
| Comunicación con zócalos en general  | <i>Application manual - Controller software IRC5</i>   |
| Creación de un nuevo zócalo  | <a href="#">SocketCreate - Crea un nuevo zócalo en la página 747</a>                           |
| Conexión a un ordenador remoto (sólo cliente)  | <a href="#">SocketConnect - Establece una conexión a un ordenador remoto en la página 744</a>  |
| Recepción de datos desde un ordenador remoto   | <a href="#">SocketReceive - Recibe datos de un ordenador remoto en la página 751</a>           |
| Cierre del zócalo  | <a href="#">SocketClose - Cerrar un zócalo en la página 742</a>                                |
| Enlazamiento de un zócalo (sólo servidor)  | <a href="#">SocketBind - Enlazar un zócalo a mi dirección IP y puerto en la página 739</a>     |
| Cómo permanecer a la escucha de conexiones (sólo servidor)                               | <a href="#">SocketListen - Permanece a la escucha de conexiones entrantes en la página 749</a> |
| Aceptación de conexiones (sólo servidor)   | <a href="#">SocketAccept - Aceptar una conexión entrante en la página 736</a>                  |
| Obtención del estado actual del zócalo   | <a href="#">SocketGetStatus - Obtiene el estado actual de un zócalo en la página 1473</a>      |
| Aplicación de ejemplo de zócalos de servidor   | <a href="#">SocketReceive - Recibe datos de un ordenador remoto en la página 751</a>           |
| Uso de caracteres no imprimibles (datos binarios) en literales de cadenas de caracteres. | <i>Technical reference manual - RAPID kernel</i>   |



### 1.255 SocketSendTo - Envío de datos a un ordenador remoto

#### Utilización

`SocketSendTo` se usa para enviar datos a un ordenador remoto. `SocketSendTo` puede usarse tanto en aplicaciones de cliente como en aplicaciones de servidor. `SocketSendTo` se utiliza para la comunicación sin conexiones con el protocolo de datagramas UDP/IP.

#### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción `SocketSendTo`:  
Consulte también [Más ejemplos en la página 762](#).

#### Ejemplo 1

```
VAR socketdev udp_socket;  
  
SocketCreate udp_socket \UDP;  
SocketSendTo udp_socket, Address, Port \Str := "Hello world";
```

Envía el mensaje "Hello world" al ordenador remoto que presenta la dirección IP `Address` y el puerto `Port`.

#### Argumentos

```
SocketSendTo Socket RemoteAddress RemotePort [ \Str ] | [ \RawData  
] | [ \Data ] [ \NoOfBytes ]
```

Socket

Tipo de dato: `socketdev`  
El zócalo debe haberse creado ya.

RemoteAddress

Tipo de dato: `string`  
La dirección del ordenador remoto. El ordenador remoto debe estar especificado como una dirección IP. No es posible usar el nombre del ordenador remoto.

RemotePort

Tipo de dato: `num`  
El puerto del ordenador remoto. Por lo general, los puertos del 1025 al 4999 están libres para su uso. Es posible que los puertos con un número inferior al 1025 ya estén ocupados.

[ \Str ]

Tipo de dato: `string`  
La cadena de tipo `string` que se desea enviar al ordenador remoto.

[ \RawData ]

Tipo de dato: `rawbytes`  
El dato de tipo `rawbytes` que se desea enviar al ordenador remoto.

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.255 SocketSendTo - Envío de datos a un ordenador remoto

### Socket Messaging

#### Continuación

[ \Data ]

**Tipo de dato:** array of byte

Los datos de la matriz de tipo `byte` que se desea enviar al ordenador remoto.

Sólo es posible usar uno de los parámetros opcionales `\Str`, `\RawData`, o `\Data` al mismo tiempo.

[ \NoOfBytes ]

**Tipo de dato:** num

Si se especifica este argumento, sólo se envía al ordenador remoto este número de bytes. La llamada a `SocketSendTo` fallará si el valor de `\NoOfBytes` es mayor que la cantidad real de bytes de la estructura de datos a enviar.

Si no se especifica este argumento, se envía al ordenador remoto la totalidad de la estructura de datos (la parte válida de `rawbytes`).

---

### Ejecución de programas

Los datos especificados se envían al ordenador remoto.

Los datos transferidos por el cable son siempre bytes, con un máximo de 1.024 bytes por cada mensaje. No se añade ningún encabezado al mensaje de forma predeterminada. El uso de cualquier encabezado está reservado para la aplicación en sí.

| Parámetro             | Datos de entrada | Datos de cable  | Datos de salida |
|-----------------------|------------------|-----------------|-----------------|
| <code>\Str</code>     | 1 carácter       | 1 byte (8 bits) | 1 carácter      |
| <code>\RawData</code> | 1 rawbytes       | 1 byte (8 bits) | 1 rawbytes      |
| <code>\Data</code>    | 1 byte           | 1 byte (8 bits) | 1 byte          |

Es posible combinar los tipos de datos (`string`, `rawbytes` o `array of byte`) entre `SocketSendTo` y `SocketReceiveFrom`.

---

### Más ejemplos

A continuación aparecen más ejemplos de la instrucción `SocketSendTo`.

#### Ejemplo 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;
```

*Continúa en la página siguiente*

```
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 \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
```

Este es un ejemplo de un programa cliente con creación y enlazamiento de zócalos en los gestores de errores. De esta forma, el programa puede manejar el reinicio tras la caída de alimentación.

En el procedimiento `client_recover`, se crea y enlaza el zócalo de cliente a un servidor de ordenador remoto que presenta la dirección 192.168.0.2 en el puerto 1025.

En el procedimiento de comunicación `client_messaging`, el cliente envía "Hello server" al servidor y el servidor responde con "Hello client" al cliente. Este mensaje se almacena en la variable `receive_string`.

### Ejemplo 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
```

Este es un ejemplo de un programa que envía caracteres no imprimibles (datos binarios) en una cadena. Esto puede resultar útil a la hora de comunicarse con sensores u otros clientes que requieran este tipo de caracteres.

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.255 SocketSendTo - Envío de datos a un ordenador remoto

### Socket Messaging

#### Continuación

#### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                              | Causa del error  |
|-------------------------------------|--|
| <code>ERR_SOCKET_CLOSED</code>      | El zócalo se cierra.   |
| <code>ERR_SOCKET_NET_UNREACH</code> | No se puede acceder a la red o la conexión se pierde después de abrirse un zócalo. |

#### Limitaciones

Todos los zócalos están cerrados tras el reinicio de la caída de alimentación. Este problema puede ser gestionado por la recuperación en caso de error. Consulte el ejemplo anterior.

El tamaño de los datos a enviar está limitado a 1.024 bytes.

#### Sintaxis

```
SocketSendTo
[Socket ':='] <variable (VAR) of socketdev>
[RemoteAddress ':='] <expression (IN) of string>
[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>'];'
```

#### Información relacionada

| Para obtener más información sobre                         | Consulte  |
|--|---|
| Comunicación con zócalos en general                        | <a href="#">Application manual - Controller software IRC5</a>                                     |
| Creación de un nuevo zócalo                                | <a href="#">SocketCreate - Crea un nuevo zócalo en la página 747</a>                              |
| Conexión a un ordenador remoto (sólo cliente)              | <a href="#">SocketConnect - Establece una conexión a un ordenador remoto en la página 744</a>     |
| Recepción de datos desde un ordenador remoto               | <a href="#">SocketReceive - Recibe datos de un ordenador remoto en la página 751</a>              |
| Cierre del zócalo  | <a href="#">SocketClose - Cerrar un zócalo en la página 742</a>                                   |
| Enlazamiento de un zócalo (sólo servidor)                  | <a href="#">SocketBind - Enlazar un zócalo a mi dirección IP y puerto en la página 739</a>        |
| Cómo permanecer a la escucha de conexiones (sólo servidor) | <a href="#">SocketListen - Permanece a la escucha de conexiones entrantes en la página 749</a>    |
| Aceptación de conexiones (sólo servidor)                   | <a href="#">SocketAccept - Aceptar una conexión entrante en la página 736</a>                     |
| Obtención del estado actual del zócalo                     | <a href="#">SocketGetStatus - Obtiene el estado actual de un zócalo en la página 1473</a>         |
| Aplicación de ejemplo de zócalos de servidor               | <a href="#">SocketReceive - Recibe datos de un ordenador remoto en la página 751</a>              |
| Recepción de datos desde un ordenador remoto               | <a href="#">SocketReceiveFrom - Recepción de datos desde un ordenador remoto en la página 756</a> |

Continúa en la página siguiente

## 1.255 SocketSendTo - Envío de datos a un ordenador remoto

*Socket Messaging*

*Continuación*

| Para obtener más información sobre   | Consulte   |
|--|--|
| Uso de caracteres no imprimibles (datos binarios) en literales de cadenas de caracteres. | <i>Technical reference manual - RAPID kernel</i> |

# 1 Instrucciones

---

1.256 SoftAct - Activa el servo suave  
RobotWare - OS

## 1.256 SoftAct - Activa el servo suave

---

### Utilización

SoftAct (*Soft Servo Activate*) se utiliza para activar el así llamado “servo suave” en cualquier eje del robot o en una unidad mecánica externa.

Esta instrucción sólo puede usarse en la tarea principal T\_ROB1 o, si se cuenta con un sistema *MultiMove*, en las tareas de movimiento.

---

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción SoftAct:

#### Ejemplo 1

```
SoftAct 3, 20;
```

Activación del servo suave en el eje 3 del robot, con un valor de suavidad del 20%.

#### Ejemplo 2

```
SoftAct 1, 90 \Ramp:=150;
```

Activación del servo suave en el eje 1 del robot, con un valor de suavidad del 90% y con un factor de rampa del 150%.

#### Ejemplo 3

```
SoftAct \MechUnit:=orbit1, 1, 40 \Ramp:=120;
```

Activación del servo suave en el eje 1 de la unidad mecánica orbit1 con un valor de suavidad del 40% y un factor de rampa del 120%.

---

### Argumentos

```
SoftAct [\MechUnit] Axis Softness [\Ramp]
```

[ \MechUnit ]

#### *Mechanical Unit*

Tipo de dato: mecunit

El nombre de la unidad mecánica. Si se omite este argumento, significa la activación del servo suave para el eje especificado del robot en la tarea de programa actual.

Axis

Tipo de dato: num

El número del eje del robot o del eje externo que debe funcionar con el servo suave.

Softness

Tipo de dato: num

El valor de suavidad, en porcentaje (del 0% al 100%). El 0% denota la suavidad mínima (la máxima rigidez), mientras que el 100% denota la máxima suavidad.

[ \Ramp ]

Tipo de dato: num

El factor de pendiente, en porcentaje ( $\geq 100\%$ ). El factor de pendiente se utiliza para controlar la aplicación del servo suave. Un factor del 100% denota un valor

*Continúa en la página siguiente*

normal. Cuando se usan valores mayores, el servo suave se aplica más lentamente (con una pendiente mayor). El valor predeterminado para el factor de pendiente es el 100%.

## Ejecución de programas

La suavidad se activa con el valor especificado para el eje actual. El valor de suavidad es válido para todos los movimientos hasta que se programa un nuevo valor de suavidad para el eje actual o hasta que se desactiva mediante la instrucción SoftDeact.

## Limitaciones

El servo suave de cualquier eje del robot o eje externo se desactiva siempre que se produce una caída de alimentación eléctrica. Esta limitación puede gestionarse en el programa del usuario cuando se reanuda la ejecución después de una caída de alimentación.

No se debe activar el mismo eje dos veces, a no ser que exista una instrucción de movimiento entre las dos activaciones. Por tanto, la secuencia de programa siguiente debe evitarse. De lo contrario, se producirán sacudidas en los movimientos del robot:

```
SoftAct n , x ;
SoftAct n , y ;
```

(n = eje n del robot, x e y = valores de suavidad)



**¡AVISO!**

La distancia de frenado de las paradas de categoría 1 será más larga si el servo suave está activado.

## Sintaxis

```
SoftAct
[ '\MechUnit :=' < variable (VAR) of mecunit> ',' ]
[ Axis :=' ] < expression (IN) of num> ',' ]
[ Softness :=' ] < expression (IN) of num> ',' ]
[ '\Ramp :=' < expression (IN) of num> ] ';' ]
```

## Información relacionada

| Para obtener más información sobre         | Consulte   |
|--|--|
| Desactivación del servo suave              | <a href="#">SoftDeact - Desactiva el servo suave en la página 772</a>  |
| Comportamiento con el servo suave activado | <i>Manual de referencia técnica - RAPID Overview</i>                   |
| Configuración de ejes externos             | <i>Application manual - Additional axes and stand alone controller</i> |

# 1 Instrucciones

---

## 1.257 SoftDeact - Desactiva el servo suave

RobotWare - OS

## 1.257 SoftDeact - Desactiva el servo suave

---

### Utilización

SoftDeact (*Soft Servo Deactivate*) se utiliza para desactivar el así llamado “servo suave”.

---

### Ejemplos básicos

Los ejemplos siguientes ilustran la instrucción SoftDeact.

#### Ejemplo 1

```
SoftDeact;
```

Desactiva el servo suave en todos los ejes.

#### Ejemplo 2

```
SoftDeact \Ramp:=150;
```

Desactiva el servo suave en todos los ejes, con un factor de pendiente del 150 %.

---

### Argumentos

```
SoftDeact [ \Ramp ]
```

[ \Ramp ]

Tipo de dato: num

El factor de pendiente, en porcentaje ( $\geq 100\%$ ). El factor de pendiente se utiliza para controlar la desactivación del servo suave. Un factor del 100% es el valor normal. Cuando se usan valores mayores, el servo suave se desactiva más lentamente (con una pendiente mayor). El valor predeterminado para el factor de pendiente es el 100%.

---

### Ejecución de programas

Se desactiva el servo suave para las unidades mecánicas controladas desde la tarea de programa actual. Si SoftDeact se realiza desde una tarea sin movimiento, el servo suave se desactiva para la unidad mecánica controlada por la tarea de movimiento conectada. La ejecución de un SoftDeact durante el modo de movimiento sincronizado, el servo suave se desactiva para todas las unidades mecánicas que estén sincronizadas.

Al desactivar el servo suave con SoftDeact el robot se mueve hasta la posición programada, incluso si el robot ha abandonado su posición durante la activación del servo suave.

---

### Sintaxis

```
SoftDeact  
[ '\Ramp :=' < expression (IN) of num > ]';'
```

---

### Información relacionada

| Para obtener más información sobre | Consulte   |
|------------------------------------|--|
| Activación del servo suave         | <a href="#">SoftAct - Activa el servo suave en la página 770</a> |

---



### 1.258 SoftElbow - Hacer el codo para las fuerzas externas

#### Utilización

`SoftElbow` se utiliza para activar o desactivar un codo suave en un robot de 7 ejes. Cuando está activo, el codo se puede mover de forma que se modifica la posición del codo sin que esto afecte al TCP. El TCP se seguirá moviendo a lo largo de su trayectoria programada.

#### Ejemplo básico

Los siguientes ejemplos ilustran la función `SoftElbow`:

##### Ejemplo 1

```
SoftElbow \On;
```

Tras ejecutar esta instrucción, el codo del robot se puede mover sin afectar al TCP.

#### Argumentos

```
SoftElbow [\On] | [\Off]
```

`[\On]`

Tipo de dato: `switch`

Activa el codo suave.

`[\Off]`

Tipo de dato: `switch`

Desactiva el codo suave.

#### Ejecución de programas

Cuando se activa el codo, este permanece suave hasta que se apaga con una nueva instrucción `SoftElbow`.

Siempre que el codo suave permanezca activo, el codo se doblará para apartarse de cualquier obstáculo o de aquello que le empuje. Si el robot se está desplazando, el TCP seguirá su trayectoria y la ejecución del programa no se verá afectada.

El valor predeterminado (codo suave desactivado) se establece automáticamente:

- cuando se utiliza el modo de reinicio Restablecer RAPID
- al cargar un nuevo programa o un nuevo módulo
- al iniciar la ejecución del programa desde el principio
- al mover el puntero del programa a `main`
- al mover el puntero del programa a una rutina
- al mover el puntero de programa de una forma que se pierde el orden de la ejecución.

#### Limitaciones

- `SoftElbow` solo está disponible para robots de 7 ejes (por ejemplo: YuMi).

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.258 SoftElbow - Hacer el codo para las fuerzas externas

RobotWare - OS

Continuación

- `SoftElbow` no funcionará cuando la herramienta esté en contacto con un objeto fijo o cuando se aplique una fuerza no inapreciable (por ejemplo: cuando se presiona un objeto contra un accesorio).
- Cuando se empuja el brazo, puede conllevar una reducción de la precisión de la trayectoria.
- `SoftElbow` no es compatible con otros modos de cumplimiento (por ejemplo: **Lead Through**, **Force control** o **SoftMove**).
- `SoftElbow` no es compatible con **MultiMove Coordinated**.
- `SoftElbow` solamente funciona con velocidades moderadas, por lo general inferiores a 1000 mm/s.
- La precisión absoluta estará temporalmente desactivada mientras el codo suave esté activo.

---

### Sintaxis

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

## 1.259 SpeedLimAxis - Establecer la limitación de velocidad de un eje

### Utilización

`SpeedLimAxis` se utiliza para establecer el valor límite de velocidad de un eje. La reducción de velocidad se realiza cuando la señal de entrada de sistema `LimitSpeed` cambia a 1. Esta instrucción permite establecer una limitación de velocidad que debe aplicarse más tarde.

Esta instrucción sólo puede usarse en la tarea principal `T_ROB1` o, si se cuenta con un sistema *MultiMove*, en las tareas de movimiento.

### Ejemplos básicos

Los ejemplos siguientes ilustran la instrucción `SpeedLimAxis`.

#### Ejemplo 1

```
SpeedLimAxis STN_1, 1, 20;
```

Así se limita la velocidad a 20 grados/segundo en el eje 1 para la unidad mecánica `STN_1` cuando la entrada del sistema `LimitSpeed` tiene el valor 1.

#### Ejemplo 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;
```

Así se limita la velocidad a 30 grados/segundo en los ejes 2 a 6 y se limita la velocidad a 10 grados/segundo en el eje 1 para la unidad mecánica `ROB_1` cuando la entrada del sistema `LimitSpeed` tiene el valor 1.

### Argumentos

```
SpeedLimAxis MechUnit AxisNo AxisSpeed
```

MechUnit

**Mechanical Unit**

Tipo de dato: `mecunit`

El nombre de la unidad mecánica.

AxisNo

Tipo de dato: `num`

El número del eje actual de la unidad mecánica.

AxisSpeed

Tipo de dato: `num`

La velocidad que debe aplicarse. En el caso del eje de rotación, la velocidad debe indicarse en grados/segundo; en el caso de un eje lineal, debe indicarse en mm/s.

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.259 SpeedLimAxis - Establecer la limitación de velocidad de un eje

RobotWare - OS

Continuación

---

### Ejecución de programas

`SpeedLimAxis` se utiliza para establecer el valor límite de velocidad de un eje para una unidad mecánica específica. La reducción de velocidad no se realiza de inmediato. Los valores se almacenan y se aplican cuando la señal de entrada de sistema `LimitSpeed` cambia a 1.

Si no se utiliza `SpeedLimAxis` para establecer una limitación para un eje, se aplica en su lugar la limitación de velocidad del modo manual. Si no se desea ninguna limitación para un eje específico, debe introducirse un valor elevado. Es más, si no se establece ninguna limitación de la velocidad en los puntos de control mediante la instrucción `SpeedLimCheckPoint`, se utilizarán las limitaciones de velocidad del modo manual para limitar la velocidad en los puntos de control.

Cuando la señal de entrada de sistema `LimitSpeed` cambia a 1, la velocidad se reduce en rampa hasta la velocidad reducida.

Cuando la señal de entrada de sistema `LimitSpeed` cambia a 0, la velocidad aumenta en rampa hasta la velocidad programada que se utiliza en la instrucción de movimiento actual.

La aceleración máxima permitida durante la rampa ascendente se controla mediante el parámetro del sistema *Limit Speed Acc Limitation* en el tipo *Motion Planner*.

La señal de salida de sistema `LimitSpeed` tiene el valor 1, cuando se alcanza la velocidad reducida. La señal de salida de sistema `LimitSpeed` tiene el valor 0, cuando la velocidad comienza a aumentar en rampa.

Los valores predeterminados para la limitación de velocidad se establecen automáticamente.

- cuando se utiliza el modo de reinicio **Restablecer RAPID**
- al cargar un nuevo programa o un nuevo módulo
- al iniciar la ejecución del programa desde el principio
- al mover el puntero del programa a `main`
- al mover el puntero del programa a una rutina
- al mover el puntero de programa de una forma que se pierde el orden de la ejecución.

---

### Más ejemplos

A continuación aparecen más ejemplos de la instrucción `SpeedLimAxis`.

#### Ejemplo 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;
```

*Continúa en la página siguiente*

## 1.259 SpeedLimAxis - Establecer la limitación de velocidad de un eje

RobotWare - OS

Continuación

```

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;
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, sigint1;
  ! Set out signal that is cross connected to system input
  LimitSpeed
  SetDO dolLimitSpeed, 1;
ENDTRAP
TRAP resetlimitspeed
  IDelete sigint2;
  CONNECT sigint2 WITH resetlimitspeed;
  ISignalDI \SingleSafe, mysensorsignal, 0, sigint2;
  ! Reset out signal that is cross connected to system input
  LimitSpeed
  SetDO dolLimitSpeed, 0;
ENDTRAP

```

Durante el movimiento del robot desde la posición  $p_1$  a  $p_{10}$ , se utiliza la limitación de velocidad predeterminada (velocidad en el modo manual). Se añade un nuevo límite de velocidad para los puntos de control del robot del TCP y para los ejes. La rutina TRAP setlimitspeed aplicará la limitación de velocidad si el valor de la señal mysensorsignal cambia a 1.

Continúa en la página siguiente

# 1 Instrucciones

## 1.259 SpeedLimAxis - Establecer la limitación de velocidad de un eje

RobotWare - OS

Continuación

La rutina `TRAP resetlimitspeed` eliminará la limitación de velocidad cuando el valor de la señal `mysensorsignal` cambie a 0.

Los nuevos ajustes para la limitación de velocidad se utilizarán mientras la variable `run_loop` tenga el valor `TRUE` y la señal de entrada de sistema `LimitSpeed` tenga el valor 1. Cuando `run_loop` cambia a `FALSE`, se establece la velocidad máxima predeterminada (velocidad del modo manual).



### Nota

La rutina `TRAP` del ejemplo sólo se utiliza para visualizar la funcionalidad. La señal utilizada para limitar la velocidad también podría conectarse directamente a la señal de entrada de sistema `LimitSpeed` o a través de un PLC de seguridad.

## Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                          | Causa del error  |
|---------------------------------|--|
| <code>ERR_AXIS_PAR</code>       | Parámetro de eje incorrecto en la instrucción                                    |
| <code>ERR_SPEEDLIM_VALUE</code> | La velocidad utilizada en el argumento <code>AxisSpeed</code> es demasiado baja. |

## Limitaciones

`SpeedLimAxis` no puede usarse en la rutina de evento de encendido.

El reducir la velocidad de un eje o punto de control, los otros ejes también deben ser reducidos en el mismo porcentaje para poder moverse a lo largo de la trayectoria programada. La velocidad del proceso a lo largo de la trayectoria programada variará.

Al utilizar `SafeMove` junto con la limitación de velocidad, es necesario configurar `SafeMove` con un margen, dado que los cálculos de `SafeMove` y movimiento son levemente diferentes.

## Sintaxis

```
SpeedLimAxis  
  [MechUnit ':='] <variable (VAR) of mecunit>','  
  [AxisNo ':='] <expression (IN) of num>','  
  [AxisSpeed ':='] <expression (IN) of num>';'
```

## Información relacionada

| Para obtener más información sobre                             | Consulte   |
|--|--|
| Instrucciones de posicionamiento                               | <i>Manual de referencia técnica - RAPID Overview</i>   |
| Establecer la limitación de velocidad de los puntos de control | <a href="#">SpeedLimCheckPoint - Establecer la limitación de velocidad de los puntos de control en la página 779</a> |
| Señales de entrada y salida de sistema                         | <i>Manual de referencia técnica - Parámetros del sistema</i>   |

### 1.260 SpeedLimCheckPoint - Establecer la limitación de velocidad de los puntos de control

#### Utilización

`SpeedLimCheckPoint` se utiliza para establecer el valor límite de velocidad de un robot de TCP. La reducción de velocidad se realiza cuando la señal de entrada de sistema `LimitSpeed` cambia a 1. Esta instrucción permite establecer una limitación de velocidad que debe aplicarse más tarde.

La reducción de velocidad se realiza si cualquiera de los puntos de control se mueven más rápido que el límite establecido mediante `SpeedLimCheckPoint`. (Para obtener más información acerca de los puntos de control, consulte [Más ejemplos en la página 781](#)).

Esta instrucción sólo puede usarse en la tarea principal `T_ROB1` o, si se cuenta con un sistema *MultiMove*, en las tareas de movimiento.

#### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción `SpeedLimCheckPoint`:

#### Ejemplo 1

```
VAR num limit_speed:=200;  
SpeedLimCheckPoint limit_speed;
```

Así se limita la velocidad a 200 mm/s para el robot de TCP cuando la entrada del sistema `LimitSpeed` tiene el valor 1.

#### Argumentos

```
SpeedLimCheckPoint RobSpeed
```

RobSpeed

**Tipo de dato:** num

La limitación de velocidad en mm/s que debe aplicarse.

*Continúa en la página siguiente*

# 1 Instrucciones

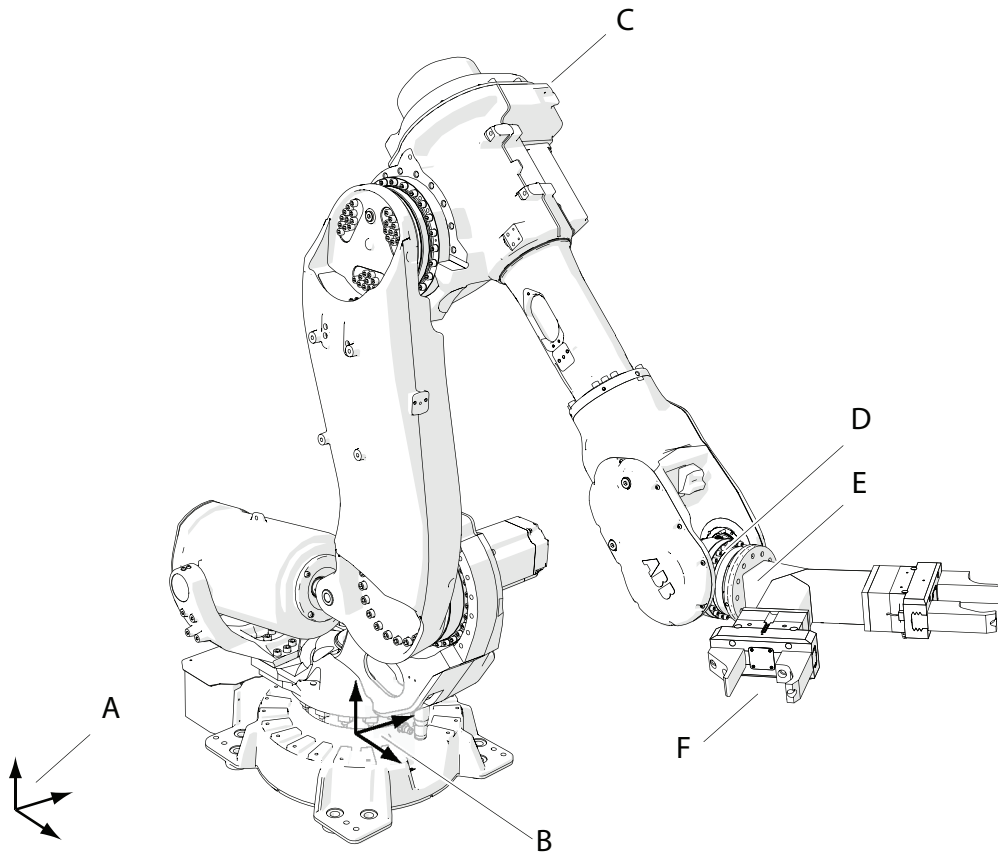
## 1.260 SpeedLimCheckPoint - Establecer la limitación de velocidad de los puntos de control

RobotWare - OS

Continuación

### Ejecución de programas

Definición de puntos de control; consulte la figura que aparece más abajo.



xx120000521

|   |                                     |
|---|-------------------------------------|
| A | Sistema de coordenadas mundo        |
| B | Sistema de coordenadas de la base   |
| C | Punto de control de brazo           |
| D | Punto central de muñeca (WCP)       |
| E | tool0                               |
| F | TCP (acrónimo de Tool Center Point) |

`SpeedLimCheckPoint` se utiliza para establecer el valor límite de velocidad para 4 puntos de control de un robot de TCP. Los puntos de control que se limitarán son el brazo, la muñeca, tool0 y el TCP activo, como se muestra en la imagen que aparece arriba. La reducción de velocidad no se realiza de inmediato. Los valores se almacenan y se aplican cuando la señal de entrada de sistema `LimitSpeed` cambia a 1. La velocidad de los puntos de control está limitada de forma relativa al sistema de coordenadas.

Si no se utiliza la instrucción `SpeedLimCheckPoint` para establecer una limitación, se usa como limitación la limitación de velocidad del modo manual. Si no se desea ninguna limitación para los puntos de control, debe introducirse un valor elevado. Es más, si no se establece ninguna limitación de la velocidad en los ejes mediante

Continúa en la página siguiente



## 1.260 SpeedLimCheckPoint - Establecer la limitación de velocidad de los puntos de control *RobotWare - OS* *Continuación*

la instrucción `SpeedLimAxis`, se utilizarán las limitaciones de velocidad del modo manual para limitar la velocidad en los ejes.

Cuando la señal de entrada de sistema `LimitSpeed` cambia a 1, la velocidad se reduce en rampa hasta la velocidad reducida.

Cuando la señal de entrada de sistema `LimitSpeed` cambia a 0, la velocidad aumenta en rampa hasta la velocidad programada que se utiliza en la instrucción de movimiento actual.

La aceleración máxima permitida durante la rampa ascendente se controla mediante el parámetro del sistema *Limit Speed Acc Limitation* en el tipo *Motion Planner*.

La señal de salida de sistema `LimitSpeed` tiene el valor 1, cuando se alcanza la velocidad reducida. La señal de salida de sistema `LimitSpeed` tiene el valor 0, cuando la velocidad comienza a aumentar en rampa.

Los valores predeterminados para la limitación de velocidad se establecen automáticamente.

- cuando se utiliza el modo de reinicio **Restablecer RAPID**
- al cargar un nuevo programa o un nuevo módulo
- al iniciar la ejecución del programa desde el principio
- al mover el puntero del programa a `main`
- al mover el puntero del programa a una rutina
- al mover el puntero de programa de una forma que se pierde el orden de la ejecución.

### Más ejemplos

A continuación aparecen más ejemplos de la instrucción `SpeedLimCheckPoint`.

#### Ejemplo 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;
```

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.260 SpeedLimCheckPoint - Establecer la limitación de velocidad de los puntos de control

RobotWare - OS

Continuación

```
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;
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, sigint1;
  ! Set out signal that is cross connected to system input
  LimitSpeed
  SetDO doLimitSpeed, 1;
ENDTRAP
TRAP resetlimitspeed
  IDelete sigint2;
  CONNECT sigint2 WITH resetlimitspeed;
  ISignalDI \SingleSafe, mysensorsignal, 0, sigint2;
  ! Reset out signal that is cross connected to system input
  LimitSpeed
  SetDO doLimitSpeed, 0;
ENDTRAP
```

Durante el movimiento del robot desde la posición `p1` a `p10`, se utiliza la limitación de velocidad predeterminada (velocidad en el modo manual). Se añade un nuevo límite de velocidad para los puntos de control del robot del TCP y para los ejes. La rutina `TRAP setlimitspeed` aplicará la limitación de velocidad si el valor de la señal `mysensorsignal` cambia a 1.

La rutina `TRAP resetlimitspeed` eliminará la limitación de velocidad cuando el valor de la señal `mysensorsignal` cambie a 0.

Los nuevos ajustes para la limitación de velocidad se utilizarán mientras la variable `run_loop` tenga el valor `TRUE` y la señal de entrada de sistema `LimitSpeed` tenga

Continúa en la página siguiente

## 1.260 SpeedLimCheckPoint - Establecer la limitación de velocidad de los puntos de control

RobotWare - OS  
Continuación

el valor 1. Cuando `run_loop` cambia a `FALSE`, se establece la velocidad máxima predeterminada (velocidad del modo manual).



### Nota

La rutina `TRAP` del ejemplo sólo se utiliza para visualizar la funcionalidad. La señal utilizada para limitar la velocidad también podría conectarse directamente a la señal de entrada de sistema `LimitSpeed` o a través de un PLC de seguridad.

### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                          | Causa del error   |
|---------------------------------|---|
| <code>ERR_SPEEDLIM_VALUE</code> | La velocidad utilizada en el argumento <code>RobSpeed</code> es demasiado baja. |

### Limitaciones

`SpeedLimCheckPoint` no puede usarse en la rutina de evento de encendido.

Si el robot se encuentra en posición vertical sobre un track móvil, la velocidad del punto de control en la base de coordenadas mundo puede ser mayor que el límite de velocidad de punto de control especificado en la base de coordenadas de la base. La velocidad del punto de control en la base de coordenadas mundo puede ser la suma de la velocidad del track y la velocidad del punto de control en la base de coordenadas de la base. Para limitar también la velocidad del punto de control en la base de coordenadas mundo, asegúrese de que la suma de ambas no rebase el límite.

El reducir la velocidad de un eje o punto de control, los otros ejes también deben ser reducidos en el mismo porcentaje para poder moverse a lo largo de la trayectoria programada. La velocidad del proceso a lo largo de la trayectoria programada variará.

Al utilizar `SafeMove` junto con la limitación de velocidad, es necesario configurar `SafeMove` con un margen y probarlo, dado que los cálculos de `SafeMove` y movimiento son levemente diferentes. Un cambio del TCP de la herramienta sobre la marcha no se sincroniza con `SafeMove`. Por tanto el TCP de la herramienta en `SafeMove` debe ser más corto que las herramientas usadas por el robot, o bien es necesario configurar la velocidad máxima del punto de control de `SafeMove` con un margen adicional y probarla.

### Sintaxis

```
SpeedLimCheckPoint
  [RobSpeed ':='] <expression (IN) of num>;'
```

### Información relacionada

| Para obtener más información sobre | Consulte   |
|------------------------------------|--|
| Instrucciones de posicionamiento   | <i>Manual de referencia técnica - RAPID Overview</i> |

*Continúa en la página siguiente*

## 1 Instrucciones

---

1.260 SpeedLimCheckPoint - Establecer la limitación de velocidad de los puntos de control

RobotWare - OS

Continuación

| Para obtener más información sobre              | Consulte  |
|---|---|
| Establecer la limitación de velocidad de un eje | <a href="#">SpeedLimAxis - Establecer la limitación de velocidad de un eje en la página 775</a> |
| Definición de cargas del brazo                  | <i>Manual de referencia técnica - Parámetros del sistema</i>                                    |
| Señales de entrada y salida de sistema          | <i>Manual de referencia técnica - Parámetros del sistema</i>                                    |

## 1.261 SpeedRefresh - La redefinición de velocidad para el movimiento en curso

### Utilización

`SpeedRefresh` se utiliza para cambiar la velocidad del movimiento del robot en curso en la actual tarea de movimiento de programa. Esta instrucción permite crear algún tipo de adaptación de velocidad aproximada a partir de una entrada de sensor.

Esta instrucción sólo puede usarse en la tarea principal `T_ROB1` o, si se cuenta con un sistema *MultiMove*, en las tareas de movimiento.

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción `SpeedRefresh`:

#### Ejemplo 1

```
VAR num change_speed:=70;
SpeedRefresh change_speed;
```

Esta operación cambia el ajuste de velocidad actual al 70%.

### Argumentos

`SpeedRefresh Override`

Override

Tipo de dato: num

El valor del ajuste de velocidad, dentro del rango del 0% al 100%.

### Ejecución de programas

Se actualiza el valor de redefinición actual de la velocidad para los movimientos en curso del robot y las unidades externas de la tarea de movimiento del programa actual.

Se ven afectados todos los componentes de datos de velocidad de todas las unidades mecánicas de la tarea de movimiento actual.



#### Nota

Una redefinición de velocidad realizada desde `SpeedRefresh` no equivale a un ajuste de velocidad desde el FlexPendant. Existen dos valores diferentes. El producto de estos dos valores y la velocidad programada compondrán la velocidad utilizada en el movimiento.

Si se realiza una operación de PP a Main o se carga un nuevo programa, la velocidad establecida con `SpeedRefresh` se restablece y se aplica la velocidad establecida desde el FlexPendant.

### Más ejemplos

A continuación aparecen más ejemplos de la instrucción `SpeedRefresh`.

#### Ejemplo 1

```
VAR intnum time_int;
VAR num override;
```

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.261 SpeedRefresh - La redefinición de velocidad para el movimiento en curso

RobotWare - OS

Continuación

```
...
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
  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
```

Durante el movimiento del robot desde la posición p1 hasta p2, el valor de ajuste de velocidad se actualiza cada 0.1 s en la rutina TRAPspeed\_refresh. La señal analógica de entrada ai\_sensor se usa para el cálculo del valor de Override para la instrucción SpeedRefresh. No se realiza ninguna ejecución de TRAP ni antes ni después del movimiento de robot entre p1 y p2. Se restablece el ajuste de velocidad manual del FlexPendant. A continuación, el robot debe alcanzar p2.

### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema ERRNO cambiará a:

| Nombre                | Causa del error                                       |
|-----------------------|---|
| ERR_SPEED_REFRESH_LIM | Override tiene un valor fuera del rango de 0 a 100 %. |

### Limitaciones

Recuerde que con SpeedRefresh el ajuste de velocidad no se hará momentáneamente. En su lugar se producirá un retardo de 0,3 a 0,5 segundos entre la orden y su influencia en el robot físico.

El usuario es responsable de restablecer el valor de ajuste de velocidad desde el programa de RAPID a continuación de la secuencia SpeedRefresh.

Continúa en la página siguiente

## 1.261 SpeedRefresh - La redefinición de velocidad para el movimiento en curso

*RobotWare - OS*

*Continuación*

Si se utiliza `SpeedRefresh` en la rutina de evento `START` o `RESET`, la velocidad establecida siempre es la redefinición de velocidad vigente en el FlexPendant.

### Sintaxis

```
SpeedRefresh  
[ Override ':' ] < expression (IN) of num > ';' ;
```

### Información relacionada

| Para obtener más información sobre     | Consulte   |
|--|--|
| Instrucciones de posicionamiento       | <i>Manual de referencia técnica - RAPID Overview</i>                                 |
| Definición de velocidad                | <a href="#">speeddata - Datos de velocidad en la página 1767</a>                     |
| Lectura del ajuste de velocidad actual | <a href="#">CSpeedOverride - Lee el ajuste de velocidad actual en la página 1246</a> |

# 1 Instrucciones

---

1.262 SpyStart - Comienza la grabación de los datos de tiempo de ejecución  
RobotWare - OS

## 1.262 SpyStart - Comienza la grabación de los datos de tiempo de ejecución

---

### Utilización

SpyStart se utiliza para iniciar la grabación de los datos de instrucciones y tiempos durante la ejecución.

Los datos de ejecución se almacenan en un archivo para su análisis posterior.

El uso previsto de esta información es la depuración de los programas de RAPID, específicamente en sistemas multitarea (sólo es necesario incluir SpyStart - SpyStop en una de las tareas de programa).

---

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción SpyStart:

#### Ejemplo 1

```
SpyStart "HOME:/spy.log";
```

Empieza la grabación de los datos de tiempo de ejecución en el archivo `spy.log` en el disco `HOME`.

---

### Argumentos

```
SpyStart File
```

#### Archivo

Tipo de dato: `string`

La ruta y el nombre del archivo que contiene los datos de ejecución.

---

### Ejecución de programas

El archivo especificado se abre para escritura y los datos de tiempo de ejecución empiezan a grabarse en el archivo.

La grabación de los datos de tiempo de ejecución permanece activa hasta los momentos siguientes:

- Ejecución de la instrucción `SpyStop`
- Inicio de la ejecución del programa desde el principio
- Carga de un nuevo programa
- siguiente **Reiniciar**
- cambia de manual a automático y activa **Auto Condition Reset**

---

### Limitaciones

No utilice nunca la función de espía en los programas de producción, porque esta función aumenta el tiempo de ciclo y consume más memoria en el dispositivo de memoria de almacenamiento que utilice.

*Continúa en la página siguiente*



## 1.262 SpyStart - Comienza la grabación de los datos de tiempo de ejecución RobotWare - OS Continuación

### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                    | Causa del error  |
|---------------------------|--|
| <code>ERR_FILEOPEN</code> | El archivo en la instrucción <code>SpyStart</code> no puede abrirse. |

### Formato de archivo

| TASK | INSTR                         | IN  | CODE  | OUT |
|------|-------------------------------|-----|-------|-----|
| MAIN | FOR i FROM 1 TO 3 DO          | 0   | READY | 0   |
| MAIN | mynum:=mynum+i;               | 1   | READY | 1   |
| MAIN | ENDFOR                        | 2   | READY | 2   |
| MAIN | mynum:=mynum+i;               | 2   | READY | 2   |
| MAIN | ENDFOR                        | 2   | READY | 2   |
| MAIN | mynum:=mynum+i;               | 2   | READY | 2   |
| MAIN | ENDFOR                        | 2   | READY | 3   |
| MAIN | SetDo1,1;                     | 3   | READY | 3   |
| MAIN | IF di1=0 THEN                 | 3   | READY | 4   |
| MAIN | MoveL p1, v1000, fine, tool0; | 4   | WAIT  | 14  |
| MAIN | MoveL p1, v1000, fine, tool0; | 111 | READY | 111 |
| MAIN | ENDIF                         | 108 | READY | 108 |
| MAIN | MoveL p2, v1000, fine, tool0; | 111 | WAIT  | 118 |
| MAIN | MoveL p2, v1000, fine, tool0; | 326 | READY | 326 |
| MAIN | SpyStop;                      | 326 | READY |     |

La columna `TASK` muestra la tarea de programa que se ejecuta.

La columna `INSTR` muestra la instrucción ejecutada en la tarea de programa especificada.

La columna `IN` muestra el tiempo en ms en la entrada de la instrucción ejecutada.

La columna `CODE` muestra si la instrucción está preparada (`READY`) o si está en espera (`WAIT`) de completarse en el momento `OUT`.

La columna `OUT` muestra el tiempo en ms en la salida de la instrucción ejecutada.

Todos los tiempos se indican en ms (valores relativos).

`SYSTEM TRAP` significa que el sistema está haciendo algo distinto de la ejecución de instrucciones de `RAPID`.

En el caso de las llamadas a procedimientos (módulos) `NOSTEPIN`, la lista de salida sólo muestra el nombre del procedimiento al que se llama. Esto se repite con cada instrucción ejecutada en la rutina `NOSTEPIN`.

### Sintaxis

```
SpyStart
[File':=']<expression (IN) of string>';'
```

*Continúa en la página siguiente*

# 1 Instrucciones

---

1.262 SpyStart - Comienza la grabación de los datos de tiempo de ejecución

RobotWare - OS

Continuación

---

## Información relacionada

| Para obtener más información sobre                  | Consulte  |
|---|---|
| Detención de la grabación de los datos de ejecución | <a href="#">SpyStop - Detiene la grabación de los datos de tiempo de ejecución en la página 791</a> |
| Condición automática                                | <i>Manual de referencia técnica - Parámetros del sistema</i>  |

## 1.263 SpyStop - Detiene la grabación de los datos de tiempo de ejecución

### Utilización

`SpyStop` se utiliza para detener la grabación de los datos de tiempo durante la ejecución.

Esta información, que puede resultar útil a la hora de optimizar el tiempo de ciclo de ejecución, se almacena en un archivo para su análisis posterior.

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción `SpyStop` :  
Consulte también [Más ejemplos en la página 791](#).

#### Ejemplo 1

```
SpyStop;
```

Detiene la grabación de los datos de tiempo de ejecución en el archivo especificado por la instrucción `SpyStart` anterior.

### Ejecución de programas

La grabación de los datos de ejecución se detiene y el archivo especificado por la instrucción `SpyStart` anterior se cierra. Si no se ha ejecutado anteriormente la instrucción `SpyStart`, la instrucción `SpyStop` no se tiene en cuenta.

### Más ejemplos

A continuación aparecen más ejemplos de la instrucción `SpyStop`.

#### Ejemplo 1

```
IF debug = TRUE SpyStart "HOME:/spy.log";  
produce_sheets;  
IF debug = TRUE SpyStop;
```

Si el indicador de depuración tiene el valor `TRUE`, se inicia la grabación de datos de ejecución en el archivo `spy.log` del disco `HOME`. Realizar la producción en sí, detener la grabación y cerrar el archivo `spy.log`.

### Limitaciones

No utilice nunca la función de espía en los programas de producción, porque esta función aumenta el tiempo de ciclo y consume más memoria en el dispositivo de memoria de almacenamiento que utilice.

### Sintaxis

```
SpyStop' ; '
```

### Información relacionada

| Para obtener más información sobre               | Consulte  |
|--|---|
| Inicio de la grabación de los datos de ejecución | <a href="#">SpyStart - Comienza la grabación de los datos de tiempo de ejecución en la página 788</a> |

# 1 Instrucciones

---

1.264 StartLoad - Carga de programa durante la ejecución  
RobotWare - OS

## 1.264 StartLoad - Carga de programa durante la ejecución

---

### Utilización

StartLoad se utiliza para iniciar la carga de un módulo de programa en la memoria de programa durante la ejecución.

Mientras se está realizando la carga, las otras instrucciones pueden ejecutarse en paralelo. El módulo cargado debe estar conectado a la tarea de programa con la instrucción WaitLoad antes de poder usar cualquier de sus símbolos o rutinas.

El módulo de programa cargado se añade a los módulos que ya existen en la memoria de programa.

Los programas o módulos de sistema pueden cargarse en el modo estático (predeterminado) o en el modo dinámico. En función del modo utilizado, algunas operaciones descargarán el módulo o no afectarán al módulo en ningún sentido.

### Modo estático

En la tabla siguiente se muestra cómo afectan dos operaciones distintas a los programas o módulos de sistema cargados en el modo estático.

|                    | Trasladar el puntero de programa a main desde TP | Abrir un nuevo programa de RAPID |
|--------------------|--|----------------------------------|
| Módulo de programa | No se ve afectado                                | Descargado                       |
| Módulo de sistema  | No se ve afectado                                | No se ve afectado                |

### Modo dinámico

En la tabla siguiente se muestra cómo afectan dos operaciones distintas a los programas o módulos de sistema cargados en el modo dinámico.

|                    | Trasladar el puntero de programa a main desde TP | Abrir un nuevo programa de RAPID |
|--------------------|--|----------------------------------|
| Módulo de programa | Descargado                                       | Descargado                       |
| Módulo de sistema  | Descargado                                       | Descargado                       |

Tanto los módulos cargados en modo estático como los cargados en modo dinámico se descargan al utilizar la instrucción UnLoad.

---

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción StartLoad:

Consulte también [Más ejemplos en la página 794](#).

#### Ejemplo 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"%;
```

Continúa en la página siguiente

## 1.264 StartLoad - Carga de programa durante la ejecución

*RobotWare - OS*

*Continuación*

```
! 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"%;
```

Inicia la carga del módulo de programa PART\_B.MOD desde diskhome en la memoria de programas con la instrucción StartLoad. En paralelo con la carga, el programa ejecuta la rutina routine\_a del módulo PART\_A.MOD. La instrucción WaitLoad espera hasta que la carga y la vinculación han terminado. El módulo se carga en el modo dinámico.

La variable load1 almacena la identidad de la sesión de carga, actualizada por StartLoad y a la que se hace referencia desde WaitLoad.

Para ahorrar tiempo de vinculación, las instrucciones UnLoad y WaitLoad pueden combinarse en la instrucción WaitLoad mediante el uso del argumento \UnLoadPath.

---

### Argumentos

```
StartLoad [\Dynamic] FilePath [\File] LoadNo
```

[\Dynamic]

Tipo de dato: switch

El modificador permite cargar un módulo de programa en modo dinámico. De lo contrario, la carga se realiza en modo estático.

FilePath

Tipo de dato: string

La ruta y el nombre del archivo que se cargará en la memoria de programa. El nombre de archivo se excluye cuando se utiliza el argumento \File .

[\File]

Tipo de dato: string

Cuando se excluye el nombre del archivo en el argumento FilePath, es necesario definirlo con este argumento.

LoadNo

Tipo de dato: loadsession

Existe una referencia en la sesión de carga que debe usarse en la instrucción WaitLoad para conectar el módulo de programa cargado a la tarea de programa.

---

### Ejecución de programas

La ejecución de StartLoad sólo solicita la carga y continúa directamente en la instrucción siguiente, sin esperar a que se complete la carga.

A continuación, la instrucción WaitLoad espera primero a que se complete la carga , si no se completado aún, y después el módulo será vinculado e inicializado.

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.264 StartLoad - Carga de programa durante la ejecución

RobotWare - OS

Continuación

La inicialización del módulo cargado devuelve todas las variables del nivel de módulo a sus valores iniciales.

Las referencias no resueltas se aceptarán de forma predeterminada para esta operación de carga `StartLoad - WaitLoad`, pero será un error de tiempo de ejecución al ejecutar una referencia no resuelta.

Para conseguir una estructura de programa idónea, fácil de comprender y mantener, todas las operaciones de carga y descarga de módulos de programa deben hacerse en el módulo principal ("main") que siempre está presente en la memoria de programa durante la ejecución.

Para la carga de un programa que contiene un procedimiento `main` en un programa principal (que contiene otro procedimiento `main`), consulte la instrucción `Load`, [Load - Carga un módulo de programa durante la ejecución en la página 347](#).

---

### Más ejemplos

A continuación aparecen más ejemplos de cómo usar la instrucción `StartLoad`.

#### Ejemplo 1

```
StartLoad \Dynamic, "HOME:/DOORDIR/DOOR1.MOD", load1;
```

Carga el módulo de programa `DOOR1.MOD` desde `HOME:` en el directorio `DOORDIR` en la memoria de programa. El módulo de programa se carga en el modo dinámico.

#### Ejemplo 2

```
StartLoad \Dynamic, "HOME:" \File:="/DOORDIR/DOOR1.MOD", load1;
```

Lo mismo que en el ejemplo 1, pero con otra sintaxis.

#### Ejemplo 3

```
StartLoad "HOME:" \File:="/DOORDIR/DOOR1.MOD", load1;
```

Lo mismo que en los ejemplos 1 y 2 anteriores, pero el módulo se carga en el modo estático.

#### Ejemplo 4

```
StartLoad \Dynamic, "HOME:" \File:="/DOORDIR/DOOR1.MOD", load1;  
WaitLoad load1;
```

Es lo mismo que:

```
Load \Dynamic, "HOME:" \File:="/DOORDIR/DOOR1.MOD";
```

---

### Limitaciones

Si se produce un fallo de alimentación mientras se está ejecutando una instrucción `StartLoad`, el sistema no podrá efectuar la recuperación al arrancar. El controlador probablemente terminará en estado de fallo de sistema y será necesario restaurar el sistema desde una copia de seguridad.

Para minimizar este riesgo, es mejor cargar varios módulos pequeños en lugar de un módulo grande.

*Continúa en la página siguiente*

**Gestión de errores**

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                        | Causa del error   |
|-------------------------------|---|
| <code>ERR_FILNOTFND</code>    | Archivo no encontrado.  |
| <code>ERR_LOADNO_INUSE</code> | La variable especificada en el argumento <code>LoadNo</code> ya se está utilizando. |

**Sintaxis**

```
StartLoad
  ['\Dynamic ',']
  [FilePath' :='] <expression (IN) of string>
  ['\File :=' <expression (IN) of string> ] ', '
  [LoadNo :='] <variable (VAR) of loadsession>;'
```

**Información relacionada**

| Para obtener más información sobre                              | Consulte  |
|---|---|
| Conexión de un módulo cargado a una tarea                       | <a href="#">WaitLoad - Conectar un módulo cargado a una tarea en la página 1064</a>             |
| Sesión de carga   | <a href="#">loadsession - Sesión de carga de programa en la página 1702</a>                     |
| Carga de un módulo de programa                                  | <a href="#">Load - Carga un módulo de programa durante la ejecución en la página 347</a>        |
| Descarga de un módulo de programa                               | <a href="#">Unload - Descargar un módulo de programa durante la ejecución en la página 1021</a> |
| Cancelación de la carga de un módulo de programa                | <a href="#">CancelLoad - Cancela la carga de un módulo en la página 81</a>                      |
| Llamada a procedimiento con enlazamiento en tiempo de ejecución | <a href="#">Manual de referencia técnica - RAPID Overview</a>                                   |

# 1 Instrucciones

---

## 1.265 StartMove - Reanuda el movimiento del robot

### 1.265 StartMove - Reanuda el movimiento del robot

---

#### Utilización

`StartMove` se utiliza para reanudar el movimiento del robot y los ejes externos y el proceso perteneciente una vez detenido el movimiento.

- La instrucción `StopMove`.
- Tras la ejecución de la secuencia `StorePath ... RestoPath`.
- Los errores de movimiento elevador asincrónicamente, como por ejemplo `ERR_PATH_STOP` o un error de proceso específico tras la gestión en el gestor de `ERROR`.

En el sistema básico, es posible utilizar esta instrucción en el tipo siguiente de tareas de programa:

- Tarea main `T_ROB1`, para el reinicio del movimiento en esa tarea.
- Cualquier otra tarea, para el reinicio del movimiento en la tarea main.

En el sistema `MultiMove`, es posible utilizar esta instrucción en el tipo siguiente de tareas de programa:

- Tarea de movimiento, para el reinicio del movimiento en esa tarea.
- Tarea sin movimiento, para el reinicio del movimiento en la tarea de movimiento conectada. Además, si el movimiento se reinicia en una tarea de movimiento conectada que pertenece a un grupo de tarea coordinada sincronizada, el movimiento se reinicia en todas las tareas cooperativas.

---

#### Ejemplos básicos

Los ejemplos siguientes ilustran la instrucción `StartMove`.

##### Ejemplo 1

```
StopMove;
WaitDI ready_input,1;
StartMove;
```

El robot empieza de nuevo a moverse cuando se activa la entrada `ready_input`.

##### Ejemplo 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;
...
```

Tras volver a la posición del paro `p` (en este ejemplo igual a `p100`), el robot está empezando a moverse de nuevo en el nivel de trayectoria básico.

*Continúa en la página siguiente*



### Argumentos

StartMove [`\AllMotionTasks`]

[`\AllMotionTasks`]

**Tipo de dato:** switch

Reinicia el movimiento de todas las unidades mecánicas del sistema. El modificador [`\AllMotionTasks`] sólo puede usarse desde tareas de programa sin movimiento.

### Ejecución de programas

Cualquier proceso asociado al movimiento detenido se reanuda al mismo tiempo que se reanuda el movimiento.

Para reiniciar una aplicación MultiMove en el modo sincronizado coordinado, StartMove debe ser ejecutado en todas las tareas de movimiento implicadas en la coordinación.

Con el modificador `\AllMotionTasks`, (sólo permitido con las tareas de programa sin movimiento), se reinician los movimientos de todas las unidades mecánicas del sistema.

En un sistema básico sin el modificador `\AllMotionTasks`, se reinician los movimientos de las unidades mecánicas siguientes:

- Siempre las unidades mecánicas de la tarea principal, independientemente de qué tarea ejecute la instrucción StartMove.

En un sistema MultiMove sin el modificador `\AllMotionTasks`, se reinician los movimientos de las unidades mecánicas siguientes:

- Las unidades mecánicas de la tarea de movimiento que ejecute StartMove.
- Las unidades mecánicas de la tarea de movimiento conectada a la tarea sin movimiento en la que se ejecute StartMove. Además, si las unidades mecánicas se reinician en una tarea de movimiento conectada que pertenece a un grupo de tarea coordinada sincronizada, las unidades mecánicas se reinician en todas las tareas cooperativas.

### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema ERRNO cambiará a:

| Nombre        | Causa del error  |
|---------------|--|
| ERR_PATHDIST  | El robot está demasiado alejado de la trayectoria (más de 10 mm o 20 grados) para realizar un reinicio del movimiento interrumpido.<br>Mueva el robot para acercarlo a la trayectoria antes de intentar RETRY. |
| ERR_STARTMOVE | El robot ya está en el estado en espera cuando se ejecuta una instrucción StartMove.<br>Espere unos instantes antes de intentar ejecutar RETRY.  |
| ERR_PROGSTOP  | El robot ya está en el estado de programa parado cuando se ejecuta una instrucción StartMove.<br>Espere unos instantes antes de intentar ejecutar RETRY.   |

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.265 StartMove - Reanuda el movimiento del robot

### Continuación

| Nombre           | Causa del error  |
|------------------|--|
| ERR_ALRDY_MOVING | El robot ya está en movimiento cuando se ejecuta una instrucción <code>StartMove</code> .<br>Espere unos instantes antes de intentar ejecutar <code>RETRY</code> . |

---

### Limitaciones

Sólo se permite a una de las distintas tareas sin movimiento la realización de la secuencia `StopMove` - `StartMove` respecto de alguna tarea de movimiento.

No es posible realizar ninguna recuperación en caso de error si se ejecuta `StartMove` dentro de cualquier gestor de errores.

---

### Sintaxis

```
StartMove  
  ['\AllMotionTasks'];'
```

---

### Información relacionada

| Para obtener más información sobre | Consulte  |
|------------------------------------|---|
| Detención del movimiento           | <a href="#">StopMove - Detiene el movimiento del robot en la página 828</a>   |
| Continuación de un movimiento      | <a href="#">StartMoveRetry - Reanuda el movimiento y la ejecución del robot en la página 799</a>  |
| Más ejemplos                       | <a href="#">StorePath - Almacena la trayectoria cuando se produce una interrupción en la página 835</a><br><a href="#">RestoPath - Restablece la trayectoria después de una interrupción en la página 615</a> |

## 1.266 StartMoveRetry - Reanuda el movimiento y la ejecución del robot

### Utilización

`StartMoveRetry` se utiliza para reanudar el movimiento del robot y de los ejes externos, junto con el proceso correspondiente, y reintenta la ejecución desde un gestor de `ERROR`.

Esta instrucción puede usarse en un gestor de `ERROR` de los tipos siguientes de tareas de programa:

- Tarea principal `T_ROB1` de un sistema básico
- Cualquier tarea de movimiento de un sistema *MultiMove*

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción `StartMoveRetry`:

#### Ejemplo 1

```
VAR robtarg 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
```

Éste es un ejemplo de un sistema *MultiMove* con movimientos sincronizados coordinados (dos robots que funcionan alrededor de un objeto de trabajo en rotación).

Durante el movimiento hasta la posición `p1`, el otro robot cooperativo sufre un error de proceso, de forma que los movimientos sincronizados coordinados se detienen. A continuación, este robot sufre el error `ERR_PATH_STOP` y la ejecución se transfiere al gestor de `ERROR`.

En el gestor de `ERROR`, se hace lo siguiente:

- `StorePath` almacena la trayectoria original, pasa a un nuevo nivel de trayectoria y pone el sistema *MultiMove* en el modo independiente.
- Si se producen problemas con el robot, se inician los movimientos en el nuevo nivel de trayectoria.
- Antes de `RestoPath` es necesario volver a la posición del error.
- `RestoPath` vuelve al nivel de trayectoria original y devuelve el sistema *MultiMove* al modo sincronizado.
- `StartMoveRetry` reinicia el movimiento interrumpido y el proceso que esté en curso. También devuelve la ejecución para reanudar la ejecución normal.

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.266 StartMoveRetry - Reanuda el movimiento y la ejecución del robot

RobotWare - OS

Continuación

### Ejecución de programas

StartMoveRetry realiza la secuencia siguiente:

- Movimiento de recuperación de la trayectoria
- Reinicio de los procesos asociados al movimiento detenido
- Reanudación del movimiento interrumpido
- RETRY de la ejecución del programa

StartMoveRetry hace lo mismo que StartMove y RETRY juntos, pero en una operación indivisible.

Sólo se reanuda el movimiento en las unidades mecánicas de la tarea de programa en la que se ejecute StartMoveRetry.

### Limitaciones

Sólo puede usarse en los gestores de ERROR de las tareas de movimiento.

En los sistemas MultiMove que ejecutan movimientos sincronizados coordinados, es necesario seguir las reglas de programación siguientes en el gestor de ERROR:

- StartMoveRetry debe usarse en todas las tareas de programa cooperativas.
- Si se necesita para un movimiento en cualquier gestor de ERROR, las instrucciones StorePath ... RestoPath deben usarse en todas las tareas de programa cooperativas.
- El programa debe devolver el robot hasta la posición del error antes de ejecutar RestoPath, si el robot fue movido en el nivel de StorePath.

### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema ERRNO cambiará a:

| Nombre           | Causa del error   |
|------------------|---|
| ERR_PATHDIST     | El robot está demasiado alejado de la trayectoria (más de 10 mm o 20 grados) para realizar un reinicio del movimiento interrumpido. |
| ERR_STARTMOVE    | El robot ya está en el estado en espera cuando se ejecuta una instrucción StartMoveRetry.   |
| ERR_PROGSTOP     | El robot ya está en el estado de programa parado cuando se ejecuta una instrucción StartMoveRetry.                                  |
| ERR_ALRDY_MOVING | El robot ya está en movimiento cuando se ejecuta una instrucción StartMoveRetry.  |

### Sintaxis

```
StartMoveRetry ' ; '
```

### Información relacionada

| Para obtener más información sobre | Consulte  |
|------------------------------------|---|
| Detención del movimiento           | <a href="#">StopMove - Detiene el movimiento del robot en la página 828</a> |

Continúa en la página siguiente

## 1.266 StartMoveRetry - Reanuda el movimiento y la ejecución del robot

RobotWare - OS

Continuación

| Para obtener más información sobre              | Consulte  |
|---|---|
| Continuación de un movimiento                   | <a href="#">StartMove</a> - Reanuda el movimiento del robot en la página 796  |
| Reanudación de la ejecución después de un error | <a href="#">RETRY</a> - Reanuda la ejecución después de un error en la página 617   |
| Almacenamiento y restauración de trayectorias   | <a href="#">StorePath</a> - Almacena la trayectoria cuando se produce una interrupción en la página 835<br><a href="#">RestoPath</a> - Restablece la trayectoria después de una interrupción en la página 615 |

# 1 Instrucciones

---

## 1.267 STCalib - Calibra una herramienta servo *Servo Tool Control*

### 1.267 STCalib - Calibra una herramienta servo

---

#### Utilización

STCalib se utiliza para calibrar la distancia entre dos puntas de una herramienta. Resulta necesario después de cambiar las puntas o cambiar de herramienta y se recomienda después de la realización de una rectificación de las puntas o después de utilizar una herramienta durante cierto tiempo.

**¡Atención!** La herramienta realiza dos movimientos de cierre y apertura durante la calibración. El primer movimiento de cierre se usa para detectar la posición de contacto de las puntas.

---

#### Ejemplos básicos

Los ejemplos siguientes ilustran la instrucción STCalib.

##### Ejemplo 1

```
VAR num curr_tip_wear;  
VAR num retval;  
CONST num max_adjustment := 20;  
  
STCalib gun1 \ToolChg;
```

Calibra una pistola servo después de un cambio de herramienta. Se espera hasta que la calibración de la pistola haya finalizado, antes de continuar con la siguiente instrucción de RAPID.

##### Ejemplo 2

```
STCalib gun1 \ToolChg \Conc;
```

Calibra una pistola servo después de un cambio de herramienta. Continuar con la siguiente instrucción de RAPID sin esperar a que finalice la calibración

##### Ejemplo 3

```
STCalib gun1 \TipChg;
```

Calibra una pistola servo después de un cambio de puntas.

##### Ejemplo 4

```
STCalib gun1 \TipWear \RetTipWear := curr_tip_wear;
```

Calibra una pistola servo cuando se han desgastado las puntas. Guarda el desgaste de las puntas en la variable curr\_tip\_wear.

##### Ejemplo 5

```
STCalib gun1 \TipChg \RetPosAdj:=retval;  
IF retval > max_adjustment THEN  
TPWrite "The tips are lost!";  
...
```

Calibra una pistola servo después de un cambio de puntas. Comprueba si faltan las puntas.

##### Ejemplo 6

```
STCalib gun1 \TipChg \PrePos:=10;
```

*Continúa en la página siguiente*

Calibra una pistola servo después de un cambio de puntas. Se mueve rápidamente hasta la posición de 10 mm y empieza a buscar la posición de contacto a una velocidad menor.

## Ejemplo 7

Ejemplo de combinación no válida:

```
STCalib gun1 \TipWear \RetTipWear := curr_tip_wear \Conc;
```

Realizar una calibración de desgaste de puntas. Continuar con la siguiente instrucción de RAPID sin esperar a que finalice la calibración. En este caso, el parámetro `curr_tip_wear` no contendrá ningún valor válido, dado que se usa el modificador `\Conc` (la siguiente instrucción de RAPID se ejecutará antes de que finalice el proceso de calibración).

## Argumentos

```
STCalib ToolName [\ToolChg] | [\TipChg] | [\TipWear] [\RetTipWear]  

[\RetPosAdj] [\PrePos] [\Conc]
```

ToolName

Tipo de dato: string

El nombre de la unidad mecánica.

[\ToolChg]

Tipo de dato: switch

Calibración tras un cambio de herramienta.

[\TipChg]

Tipo de dato: switch

Calibración tras un cambio de puntas.

[\TipWear]

Tipo de dato: switch

Calibración tras el desgaste de las puntas.

[\RetTipWear]

Tipo de dato: num

El desgaste de puntas detectado [en mm].

[\RetPosAdj]

Tipo de dato: num

El ajuste de posición desde la última calibración [mm].

[\PrePos]

Tipo de dato: num

La posición a la que moverse a alta velocidad antes de empezar la búsqueda de la posición de contacto a menor velocidad [en mm].

[\Conc]

Tipo de dato: switch

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.267 STCalib - Calibra una herramienta servo

### *Servo Tool Control*

#### *Continuación*

Distintas instrucciones consecutivas se ejecutan mientras la pistola está en movimiento. El argumento puede utilizarse para acortar el tiempo de ciclo. Esto resulta útil, por ejemplo, si se desea controlar dos pistolas al mismo tiempo.

---

#### Ejecución de programas

##### Modos de calibración

Si la unidad mecánica existe, se ordena la calibración de la herramienta servo. La calibración se realiza de acuerdo con los modificadores. Consulte la información que aparece más abajo. Si se utiliza el parámetro `RetTipWear`, se actualiza el valor de desgaste de las puntas.

##### Calibración tras un cambio de herramienta:

La herramienta se cerrará a baja velocidad en espera de que el contacto de las puntas se abra rápidamente, se cierre rápidamente con una fuerza reducida y se vuelva a abrir en una secuencia. El desgaste de la punta permanece sin cambios.

##### Calibración tras un cambio de puntas:

La herramienta se cerrará a baja velocidad en espera de que el contacto de las puntas se abra rápidamente, se cierre rápidamente con una fuerza reducida y se vuelva a abrir en una secuencia. El desgaste de la punta se restablecerá.

##### Calibración tras el desgaste de las puntas:

La herramienta se cerrará a alta velocidad hasta la posición de contacto, se abre rápidamente y se cierra rápidamente con una fuerza reducida y se vuelve a abrir en una secuencia. El desgaste de la punta se actualiza.

**¡ATENCIÓN!** Si se usa el modificador `Conc`, la instrucción se considerará terminada tan pronto como se inicia y por tanto el valor de retorno `RetTipWear` no estará disponible. En este caso, el valor de `RetTipWear` será devuelto por la función `STIsOpen`. Para obtener más detalles, consulte Funciones de sistema operativo de RobotWare `STIsOpen`.

##### Ajuste de posición

El argumento opcional `RetPosAdj` puede usarse para detectar si, por ejemplo, las puntas se han perdido tras un cambio de puntas. El parámetro conserva el valor del ajuste posicional desde la última calibración. El valor puede ser negativo o positivo.

##### Utilización de una posición previa

Para acelerar la calibración, es posible definir una posición previa. Cuando comienza la calibración, el brazo de la pistola se mueve rápidamente hasta la posición previa, se detiene y continúa lentamente \*) para poder detectar la posición de contacto de la punta. ¡Si utiliza una posición previa, elíjala cuidadosamente! ¡Es importante que las puntas no entren en contacto hasta *después* de alcanzar la posición previa! De lo contrario, la exactitud de la calibración será insuficiente y es posible que se produzcan errores de supervisión del movimiento. La posición previa no se tiene en cuenta si es mayor que la posición actual de la pistola (para no ralentizar la calibración).

\*) El segundo movimiento también será rápido si se usa la opción `\TipWear`.

---

*Continúa en la página siguiente*



## Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre           | Causa del error   |
|------------------|---|
| ERR_NO_SGUN      | El nombre especificado de la herramienta servo no es una herramienta servo configurada.   |
| ERR_SGUN_ESTOP   | Parada de emergencia durante movimiento de herramienta servo.<br>La instrucción es ejecutada desde una tarea en segundo plano y se produce una parada de emergencia; la instrucción será finalizada.<br>Tenga en cuenta que si se ejecuta la instrucción desde la tarea principal, entonces el puntero del programa se detendrá en la instrucción y la instrucción se reiniciará desde el comienzo en el reinicio del programa. |
| ERR_SGUN_MOTOFF  | La instrucción se ejecuta desde una tarea en segundo plano y el sistema se encuentra en estado de motores apagados.   |
| ERR_SGUN_NEGVAL  | El argumento <code>PrePos</code> se especifica con un valor menor que cero.   |
| ERR_SGUN_NOTACT  | La unidad mecánica de la herramienta servo no está activada.<br>Use la instrucción <code>ActUnit</code> para activar la herramienta servo.  |
| ERR_SGUN_NOTINIT | La posición de la herramienta servo no está inicializada.<br>Es necesario inicializar la posición de la herramienta servo la primera vez que se instala la pistola o después de hacer una calibración fina. Utilice la rutina de servicio <code>ManServiceCalib</code> o realice una calibración de cambio de punta. El desgaste de la punta se restablecerá.   |
| ERR_SGUN_NOTOPEN | La pistola no está abierta cuando se ejecuta <code>STCalib</code> .   |
| ERR_SGUN_NOTSYNC | Las puntas de la herramienta servo no están sincronizadas.<br>Las puntas de la herramienta servo deben sincronizarse si el cuentarrevoluciones se ha perdido y/o actualizado. No se perderá ningún dato de proceso como por ejemplo el desgaste de las puntas.  |

## Sintaxis

```
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' ]';'
```

## Información relacionada

| Para obtener más información sobre | Consulte   |
|------------------------------------|--|
| Apertura de una herramienta servo  | <a href="#">STOpen - Abre una herramienta servo en la página 825</a> |

Continúa en la página siguiente

# 1 Instrucciones

---

1.267 STCalib - Calibra una herramienta servo

*Servo Tool Control*

*Continuación*

| Para obtener más información sobre | Consulte  |
|------------------------------------|---|
| Cierre de una herramienta servo    | <a href="#">STClose - Cierra una herramienta servo en la página 807</a> |

## 1.268 STClose - Cierra una herramienta servo

### Utilización

STClose se utiliza para cerrar la herramienta servo.

### Ejemplos básicos

Los ejemplos siguientes ilustran la instrucción STClose.

#### Ejemplo 1

```
VAR num curr_thickness1;
VAR num curr_thickness2;
```

```
STClose gun1, 1000, 5;
```

Cierra la pistola servo con una fuerza de punta de 1000 N y un grosor de plancha de 5 mm. Se espera hasta que la pistola se haya cerrado, antes de continuar con la siguiente instrucción de RAPID.

#### Ejemplo 2

```
STClose gun1, 2000, 3\RetThickness:=curr_thickness;
```

Cierra la pistola servo con una fuerza de punta de 2000 N y un grosor de plancha de 3 mm. Almacena en la variable `curr_thickness` el grosor medido.

#### Ejemplo 3

##### Modo concurrente

```
STClose gun1, 1000, 5 \Conc;
STClose gun2, 2000, 3 \Conc;
```

Se cierra la pistola servo `gun1` con una fuerza de puntas de 1000 N y un espesor de plancha de 5 mm. Se continúa la ejecución del programa sin esperar al cierre de `gun1` y se cierra la pistola servo `gun2` con una fuerza de puntas de 2000 N y un espesor de chapa de 3 mm. La ejecución del programa de RAPID continúa sin esperar a que se cierre la pistola `gun2`.

#### Ejemplo 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
```

Se obtiene el grosor medido en la función `STIsClosed`, variables `curr_thickness1` y `curr_thickness2`.

#### Ejemplo 5

##### Ejemplo de combinación no válida:

```
STClose gun1, 2000, 3\RetThickness:=curr_thickness \Conc;
```

Se cierra la pistola servo y se continúa la ejecución del programa de RAPID. En este caso, el parámetro `curr_thickness` no contiene ningún valor válido, dado que se usa el modificador `\Conc` (la siguiente instrucción de RAPID empezará a ejecutarse antes de que se cierre la pistola).

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.268 STClose - Cierra una herramienta servo

### *Servo Tool Control*

#### *Continuación*

---

#### Argumentos

STClose ToolName TipForce Thickness [`\RetThickness`][`\Conc`]

ToolName

Tipo de dato: string

El nombre de la unidad mecánica.

TipForce

Tipo de dato: num

La fuerza de punta deseada [N].

Thickness

Tipo de dato: num

La posición de contacto esperada para la pistola servo [mm].

[`\RetThickness`]

Tipo de dato: num

El grosor obtenido [mm]. Sólo obtendrá un valor si no se utiliza el modificador `\Conc`.

[`\Conc`]

Tipo de dato: switch

Distintas instrucciones consecutivas se ejecutan mientras la pistola está en movimiento. El argumento puede utilizarse para acortar el tiempo de ciclo. Esto resulta útil, por ejemplo, si se desea controlar dos pistolas al mismo tiempo.

---

#### Ejecución de programas

Si la unidad mecánica existe, se ordena el cierre de la herramienta servo con el grosor y la fuerza esperados.

La operación de cierre empieza a mover el brazo de la herramienta hasta la posición de contacto esperada (el grosor esperado). El movimiento se detiene en esta posición y se realiza un cambio del modo de control de posición al modo de control de fuerza.

El brazo de la herramienta se mueve a la velocidad y la aceleración máximas y está definido en los parámetros del sistema del eje externo correspondiente. En cuanto a los movimientos de los demás ejes, se utiliza la velocidad reducida en el modo manual.

Cuando se consigue la fuerza de punta deseada, la instrucción está preparada y el grosor conseguido se devuelve si se ha especificado el argumento opcional `RetThickness`.

**¡NOTA!** Si se usa el modificador `Conc`, la instrucción se considerará terminada tan pronto como se inicia y por tanto el valor de retorno `RetThickness` no estará disponible. En este caso, el valor de `RetThickness` será devuelto por la función `STIsClosed`. Para obtener más detalles, consulte **Funciones de sistema operativo de RobotWare - STIsClosed**.

*Continúa en la página siguiente*

Es posible cerrar la herramienta durante un movimiento programado del robot, siempre y cuando el movimiento del robot no incluya el movimiento del brazo de la herramienta.

Para obtener más detalles, consulte Control de movimiento de herramientas servo.

### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                        | Causa del error   |
|-------------------------------|---|
| <code>ERR_NO_SGUN</code>      | El nombre especificado de la herramienta servo no es una herramienta servo configurada.   |
| <code>ERR_SGUN_ESTOP</code>   | Parada de emergencia durante movimiento de herramienta servo.<br>La instrucción es ejecutada desde una tarea en segundo plano y se produce una parada de emergencia; la instrucción será finalizada.<br>Tenga en cuenta que si se ejecuta la instrucción desde la tarea principal, entonces el puntero del programa se detendrá en la instrucción y la instrucción se reiniciará desde el comienzo en el reinicio del programa. |
| <code>ERR_SGUN_MOTOFF</code>  | La instrucción se ejecuta desde una tarea en segundo plano y el sistema se encuentra en estado de motores apagados.   |
| <code>ERR_SGUN_NEGVAL</code>  | El argumento <code>PrePos</code> se especifica con un valor menor que cero.   |
| <code>ERR_SGUN_NOTACT</code>  | La unidad mecánica de la herramienta servo no está activada.<br>Use la instrucción <code>ActUnit</code> para activar la herramienta servo.  |
| <code>ERR_SGUN_NOTINIT</code> | La posición de la herramienta servo no está inicializada.<br>Es necesario inicializar la posición de la herramienta servo la primera vez que se instala la pistola o después de hacer una calibración fina. Utilice la rutina de servicio <code>ManServiceCalib</code> o realice una calibración de cambio de punta. El desgaste de la punta se restablecerá.   |
| <code>ERR_SGUN_NOTOPEN</code> | La pistola no está abierta cuando se ejecuta <code>STClose</code> .   |
| <code>ERR_SGUN_NOTSYNC</code> | Las puntas de la herramienta servo no están sincronizadas.<br>Las puntas de la herramienta servo deben sincronizarse si el cuentarrevoluciones se ha perdido y/o actualizado. No se perderá ningún dato de proceso como por ejemplo el desgaste de las puntas.  |

### Sintaxis

```
STClose
[ 'ToolName' := ] < expression (IN) of string > ', '
[ 'Tipforce' := ] < expression (IN) of num > ', '
[ 'Thickness' := ] < expression (IN) of num > ]
[ '\ 'RetThickness' := ] < variable or persistent (INOUT) of num
> ]
[ '\ 'Conc ]
```

*Continúa en la página siguiente*

# 1 Instrucciones

---

1.268 STClose - Cierra una herramienta servo

*Servo Tool Control*

*Continuación*

---

## Información relacionada

| Para obtener más información sobre | Consulte   |
|------------------------------------|--|
| Apertura de una herramienta servo  | <a href="#">STOpen - Abre una herramienta servo en la página 825</a> |

---

## 1.269 StepBwdPath - Retrocede un paso a lo largo de la trayectoria

---

### Utilización

StepBwdPath se utiliza para hacer retroceder el TCP a lo largo de la trayectoria del robot, a partir de una rutina de evento RESTART.

El usuario es el encargado de introducir un indicador de reinicio del proceso, de forma que una instrucción StepBwdPath en una rutina de evento RESTART sólo se ejecute al reiniciar el proceso y no en el reinicio del programa.

Esta instrucción sólo puede usarse en la tarea principal T\_ROB1 o, si se cuenta con un sistema MultiMove, en las tareas de movimiento.

---

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción StepBwdPath:

#### Ejemplo 1

```
StepBwdPath 30, 1;  
StepBwdPath 30, 1;
```

La primera instrucción se mueve hacia atrás 30 mm. La segunda instrucción se mueve hacia atrás 30 mm más.

---

### Argumentos

```
StepBwdPath StepLength StepTime
```

#### StepLength

Tipo de dato: num

Especifica la distancia en milímetros de retroceso en este paso. Este argumento debe ser un valor positivo.

#### StepTime

Tipo de dato: num

Este argumento está obsoleto. Cámbielo a 1.

---

### Ejecución de programas

El robot retrocede la distancia especificada a lo largo de su trayectoria. La trayectoria es exactamente la misma, aunque en orden inverso, que la que existía antes de producirse el paro. En caso de un paro rápido o un paro de emergencia, la llamada a la rutina de evento RESTART se realiza una vez completada la fase de recuperación, de forma que el robot se encuentre ya en su trayectoria cuando se ejecuta esta instrucción.

La velocidad real de este movimiento es la velocidad programada en la orden de movimiento pero limitada a 250 mm/s.

Las propiedades siguientes son válidas en Sistema MultiMove - Movimiento sincronizado y coordinado:

- Todas las unidades mecánicas implicadas retroceden simultáneamente y de forma coordinada.

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.269 StepBwdPath - Retrocede un paso a lo largo de la trayectoria

RobotWare - OS

Continuación

- Cada instrucción `StepBwdPath` ejecutada en cualquier tarea de programa implicada da lugar a un nuevo paso de movimiento de retroceso (sin necesidad de ninguna instrucción `StartMove`).
- Para reiniciar y continuar los movimientos de proceso interrumpidos, debe ejecutarse la instrucción `StartMove` en todas las tareas de programa implicadas.

---

### Limitaciones

Una vez detenido el programa, es posible retroceder a lo largo de la trayectoria con las limitaciones siguientes:

- Los movimientos de `StepBwdPath` están limitados al último punto fino y la longitud del historial de órdenes de movimientos que normalmente es cinco.

---

### Gestión de errores

Si se intenta un movimiento más allá de estos límites, se llama al gestor de errores con la variable de sistema `ERRNO` cambiada a `ERR_BWDLIMIT`.

---

### Sintaxis

```
StepBwdPath
  [ StepLength'::=' ] < expression (IN) of num >', '
  [ StepTime '::=' ] < expression (IN) of num >';'
```

---

### Información relacionada

| Para obtener más información sobre | Consulte   |
|------------------------------------|--|
| Movimiento en general              | <i>Manual de referencia técnica - RAPID Overview</i> |
| Instrucciones de posicionamiento   | <i>Manual de referencia técnica - RAPID Overview</i> |
| <i>Advanced RAPID</i>              | <i>Application manual - Controller software IRC5</i> |



## 1.270 STIndGun: establece la herramienta servo en el modo independiente

### Utilización

STIndGun (*Servo Tool independent gun*) se utiliza para establecer la herramienta servo (p.ej., pistola o pinza) en el modo independiente y, posteriormente, mover la herramienta hasta una posición independiente especificada. La herramienta permanecerá en modo independiente hasta que se ejecute la instrucción STIndGunReset.

Durante el modo independiente, el control de la herramienta se separa del control del robot. Es posible cerrar, abrir y calibrar la herramienta o moverla a una nueva posición independiente, pero no seguirá los movimientos coordinados del robot.

El modo independiente resulta útil si la herramienta realiza una tarea que es independiente de la tarea del robot; por ejemplo, la rectificación de puntas de una pistola estacionaria.

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción STIndGun:

#### Ejemplo 1

Este procedimiento puede ser ejecutado desde una tarea en segundo plano mientras el robot de la tarea principal continúa con otras instrucciones, por ejemplo de movimiento.

```
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
```

Se activa el modo independiente y se mueve la pistola a una posición independiente (30 mm). Durante el modo independiente, las instrucciones StClose, WaitTime y STOpen se ejecutan sin interferir en los movimientos del robot. La instrucción

*Continúa en la página siguiente*

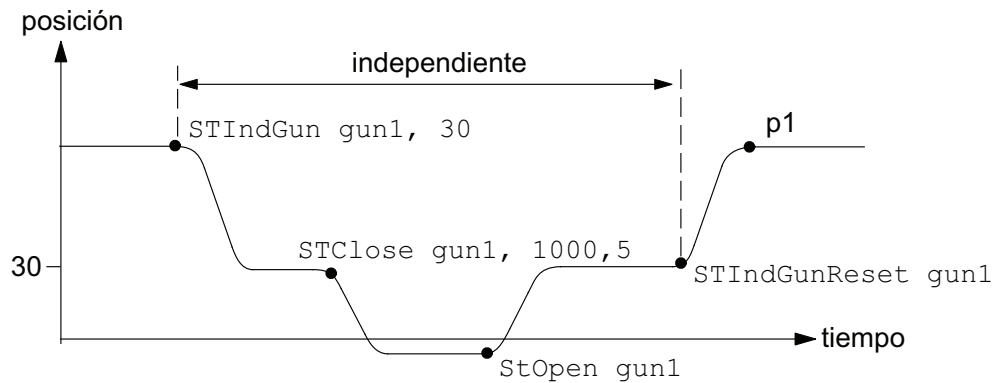
# 1 Instrucciones

## 1.270 STIndGun: establece la herramienta servo en el modo independiente

### Servo Tool Control

#### Continuación

StIndGunReset desactiva el modo independiente de la pistola y la mueve a la posición del objetivo de robot actual.



La posición p1 depende de la posición de la pistola indicada en el objetivo de robot recién ejecutado por el robot.

### Argumentos

STIndGun ToolName GunPos

ToolName

Tipo de dato: string

El nombre de la unidad mecánica.

GunPos

Tipo de dato: num

La posición (recorrido) de la pistola servo en mm.

### Sintaxis

```
STIndGun
  [ ToolName ':=' ] < expression (IN) of string > ','
  [ GunPos ':=' < expression (IN) of num > ]';'
```

1.271 STIndGunReset: restablece la herramienta servo del modo independiente  
*Servo Tool Control*

## 1.271 STIndGunReset: restablece la herramienta servo del modo independiente

---

### Utilización

STIndGunReset (*Servo Tool independent gun reset*) se utiliza para restablecer la herramienta servo desde el modo independiente y, a continuación, mover la pistola a la posición del objetivo de robot actual.

---

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción STIndGunReset:

```
STIndGunReset gun1;
```

---

### Argumentos

```
STIndGunReset ToolName
```

ToolName

Tipo de dato: string

El nombre de la unidad mecánica.

---

### Ejecución de programas

La instrucción restablece el modo independiente de la herramienta y la mueve a la posición del objetivo de robot actual. Durante este movimiento, la velocidad coordinada de la herramienta debe ser cero. De lo contrario, se produce un error. La velocidad coordinada será cero si el robot está en reposo o si el movimiento actual del robot incluye un "movimiento cero" de la pistola.

---

### Sintaxis

```
STIndGunReset  
[ToolName ':=']<expression (IN) of string>;'
```

## 1 Instrucciones

---

1.272 SToolRotCalib - Calibración del TCP y de la rotación de una herramienta estacionaria  
RobotWare - OS

### 1.272 SToolRotCalib - Calibración del TCP y de la rotación de una herramienta estacionaria

---

#### Utilización

SToolRotCalib (*Stationary Tool Rotation Calibration*) se utiliza para calibrar el TCP y la rotación de una herramienta estacionaria.

La posición del robot y sus movimientos dependen siempre de su sistema de coordenadas de herramienta, es decir, del TCP y de la orientación de la herramienta. Para conseguir la máxima exactitud, es importante definir con la mayor corrección posible el sistema de coordenadas de la herramienta.

La calibración también puede realizarse con un método manual, utilizando el FlexPendant, consulte *Manual del operador - IRC5 con FlexPendant*.

#### Descripción

Para definir el TCP y la rotación de una herramienta estacionaria, necesita una herramienta apuntadora móvil montada en el elemento terminal del robot.

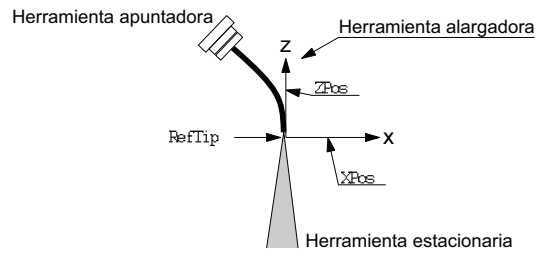
Antes de usar la instrucción SToolRotCalib, es necesario cumplir algunas condiciones previas:

- La herramienta estacionaria que se desea calibrar debe estar montada estacionariamente y definida con el componente robhold (FALSE).
- La herramienta apuntadora (robhold TRUE) debe estar definida y calibrada con los valores de TCP correctos.
- Si se utiliza el robot con una exactitud total, la carga y el centro de gravedad de la herramienta apuntadora deben estar definidos. Es posible usar LoadIdentify para la definición de la carga.
- Es necesario activar la herramienta apuntadora, wobj0 y PDispOff antes de empezar los movimientos del robot.
- Mueva el TCP de la herramienta apuntadora hasta el lugar más cercano posible del TCP de la herramienta estacionaria (origen del sistema de coordenadas de la herramienta) y defina un robtarget para el punto de referencia RefTip.
- Mueva el robot sin cambiar la orientación de la herramienta, de forma que el TCP de la herramienta apuntadora apunte hacia algún punto del eje z positivo del sistema de coordenadas de la herramienta y defina un robtarget para el punto ZPos.
- Mueva el robot sin cambiar la orientación de la herramienta, de forma que el TCP de la herramienta apuntadora apunte hacia algún punto del eje x positivo del sistema de coordenadas de la herramienta y defina un robtarget para el punto XPos.

Como ayuda para apuntar hacia el eje x y el eje x positivos, puede usarse algún tipo de herramienta alargadora.

*Continúa en la página siguiente*

Definición de robtargets RefTip, ZPos y XPos. Consulte la figura siguiente.



xx0500002343

**Nota**

No se recomienda modificar las posiciones de RefTip, ZPos y XPos en la instrucción SToolRotCalib.

**Ejemplos básicos**

El ejemplo que aparece a continuación ilustra la instrucción SToolRotCalib:

**Ejemplo 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;
```

Se calcula y actualiza la posición del TCP (`tframe.trans`) y la orientación de la herramienta (`tframe.rot`) de `tool1` en el sistema de coordenadas mundo.

**Argumentos**

```
SToolRotCalib RefTip ZPos XPos Tool
```

RefTip

Tipo de dato: robtarget

El punto en el que el TCP de la herramienta apuntadora está apuntando hacia el TCP de la herramienta estacionaria a calibrar.

ZPos

Tipo de dato: robtarget

El punto de alargador que define la dirección z positiva.

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.272 SToolRotCalib - Calibración del TCP y de la rotación de una herramienta estacionaria

RobotWare - OS

Continuación

XPos

Tipo de dato: `robtarget`

El punto de alargador que define la dirección x positiva.

Tool

Tipo de dato: `tooldata`

La variable persistente de la herramienta a calibrar.

---

### Ejecución de programas

El sistema calcula y actualiza el TCP (`tframe.trans`) y la orientación (`tframe.rot`) de la herramienta en los datos `tooldata` especificados. El cálculo se basa en el `robtarget 3` especificado. El resto de datos de `tooldata` no se cambian.

---

### Sintaxis

```
SToolRotCalib
  [RefTip ':=' ] <expression (IN) of robtarget>', '
  [ZPos ':=' ] <expression (IN) of robtarget>', '
  [XPos ':=' ] <expression (IN) of robtarget>', '
  [Tool ':=' ] <persistent (PERS) of tooldata>';'
```

---

### Información relacionada

| Para obtener más información sobre                  | Consulte   |
|---|--|
| Calibración del TCP de una herramienta móvil        | <a href="#">MToolTCPCalib - Calibración del TCP de una herramienta móvil en la página 486</a>        |
| Calibración de la rotación de una herramienta móvil | <a href="#">MToolRotCalib - Calibración de la rotación de una herramienta móvil en la página 483</a> |
| Calibración del TCP de una herramienta fija         | <a href="#">MToolTCPCalib - Calibración del TCP de una herramienta móvil en la página 486</a>        |

## 1.273 SToolTCPCalib - Calibración del TCP de una herramienta estacionaria

### Utilización

SToolTCPCalib (*Stationary Tool TCP Calibration*) se usa para calibrar el TCP (punto central de la herramienta) de una herramienta estacionaria.

La posición del robot y sus movimientos dependen siempre de su sistema de coordenadas de herramienta, es decir, del TCP y de la orientación de la herramienta. Para conseguir la máxima exactitud, es importante definir con la mayor corrección posible el sistema de coordenadas de la herramienta.

La calibración también puede realizarse con un método manual, utilizando el FlexPendant (como se describe en *Manual del operador - IRC5 con FlexPendant*).

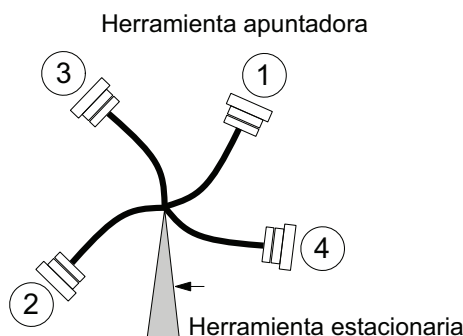
### Descripción

Para definir el TCP de una herramienta estacionaria, necesita una herramienta apuntadora móvil montada en el elemento terminal del robot.

A continuación se enumeran los requisitos previos para el uso de la instrucción SToolTCPCalib:

- La herramienta estacionaria que se desea calibrar debe estar montada estacionariamente y definida con el componente `robhold (FALSE)`.
- La herramienta apuntadora (`robhold TRUE`) debe estar definida y calibrada con los valores de TCP correctos.
- Si se utiliza el robot con una exactitud total, la carga y el centro de gravedad de la herramienta apuntadora deben estar definidos. Es posible usar `LoadIdentify` para la definición de la carga.
- Es necesario activar la herramienta apuntadora, `wobj0` y `PDispOff` antes de empezar los movimientos del robot.
- Mueva el TCP de la herramienta apuntadora lo más cerca posible del TCP de la herramienta estacionaria y defina un `robtarget` para el primer punto `p1`.
- Defina las tres posiciones adicionales, `p2`, `p3` y `p4`, todas con orientaciones diferentes.
- Se recomienda que el TCP esté apuntando en direcciones diferentes para obtener un resultado estadístico fiable.

Definición de 4 `robtargets`, de `p1` a `p4`. Consulte la figura siguiente.



xx0500002344

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.273 SToolTCPCalib - Calibración del TCP de una herramienta estacionaria

RobotWare - OS

Continuación



### Nota

No se recomienda modificar las posiciones de Pos1 a Pos4 en la instrucción SToolTCPCalib.

La reorientación entre las 4 posiciones debe ser la mayor posible, poniendo el robot en configuraciones diferentes. También resulta adecuado comprobar la calidad del TCP antes de una calibración. Esto puede realizarse reorientando la herramienta y comprobando si el TCP permanece en la misma posición.

### Ejemplo básico

El ejemplo que aparece a continuación ilustra la instrucción SToolTCPCalib:

#### Ejemplo 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;
```

Se calibra y actualiza el valor del TCP (tframe.trans) de tool1.max\_err y mean\_err contendrán el error máximo en mm a partir del TCP calculado y el error medio en mm a partir del TCP calculado, respectivamente.

### Argumentos

```
SToolTCPCalib Pos1 Pos2 Pos3 Pos4 Tool MaxErr MeanErr
```

Pos1

Tipo de dato: robtarget

El primer punto de aproximación.

Pos2

Tipo de dato: robtarget

El segundo punto de aproximación.

Pos3

Tipo de dato: robtarget

El tercer punto de aproximación.

Continúa en la página siguiente



## 1.273 SToolTCPCalib - Calibración del TCP de una herramienta estacionaria RobotWare - OS Continuación

Pos4

Tipo de dato: `robtarget`  
El cuarto punto de aproximación.

Tool

Tipo de dato: `tooldata`  
La variable persistente de la herramienta a calibrar.

MaxErr

Tipo de dato: `num`  
El error máximo en mm para un punto de aproximación.

MeanErr

Tipo de dato: `num`  
La distancia media que separa los puntos de aproximación del TCP calculado, es decir, la exactitud con la que el robot se posicionó respecto del TCP estacionario.

### Ejecución de programas

El sistema calcula y actualiza el valor del TCP en el sistema de coordenadas mundo (`tframe.trans`) en el valor `tooldata` especificado. El cálculo se basa en el `robtarget 4` especificado. El resto de datos de `tooldata`, como por ejemplo la orientación de la herramienta (`tframe.rot`) no cambia.

### Sintaxis

```
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 > ';'

```

### Información relacionada

| Para obtener más información sobre                        | Consulte  |
|---|---|
| Calibración del TCP de una herramienta móvil              | <a href="#">SToolTCPCalib - Calibración del TCP de una herramienta estacionaria en la página 819</a>                  |
| Calibración de la rotación de una herramienta móvil       | <a href="#">MToolRotCalib - Calibración de la rotación de una herramienta móvil en la página 483</a>                  |
| Calibración del TCP y la rotación de una herramienta fija | <a href="#">SToolRotCalib - Calibración del TCP y de la rotación de una herramienta estacionaria en la página 816</a> |

# 1 Instrucciones

---

1.274 Stop - Detención de la ejecución del programa  
RobotWare - OS

## 1.274 Stop - Detención de la ejecución del programa

---

### Utilización

Stop se usa para detener la ejecución del programa. Cualquier movimiento realizado en ese momento se finalizará antes de que la instrucción Stop esté lista.

---

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción Stop:  
Consulte también [Más ejemplos en la página 824](#).

#### Ejemplo 1

```
TPWrite "The line to the host computer is broken";  
Stop;
```

La ejecución del programa se detiene después de escribir un mensaje en el FlexPendant.

---

### Argumentos

```
Stop [ \NoRegain ] | [ \AllMoveTasks ]
```

[ \NoRegain ]

Tipo de dato: switch

Especifica si la unidad mecánica afectada debe regresar a la posición de paro la siguiente vez que se reanude el programa.

Si se usa el argumento \NoRegain, ni los ejes del robot ni los ejes externos regresan a la posición de paro (si han sido alejados de esa posición).

Si se omite el argumento y los ejes del robot o los ejes externos han sido alejados de la posición de paro, el robot muestra una pregunta en el FlexPendant. A continuación, el usuario puede decidir si el robot debe regresar a la posición de paro.

[ \AllMoveTasks ]

Tipo de dato: switch

Especifica que deben detenerse los programas de todas las tareas normales en funcionamiento, excepto la tarea actual.

Si se omite el argumento, sólo se detiene el programa de la tarea en la que se ejecuta la instrucción.

---

### Ejecución de programas

La instrucción detiene la ejecución del programa cuando las unidades mecánicas afectadas de la tarea de movimiento actual alcanzan la velocidad cero para el movimiento que esté realizando en cada momento y se detienen. Posteriormente es posible reanudar la ejecución del programa a partir de la instrucción siguiente.

Si la instrucción se usa sin ningún modificador, sólo se verá afectado el programa de dicha tarea.

Si se usa el modificador AllMoveTasks en una tarea (de tipo normal, estática o semiestática), se detendrán tanto el programa de dicha tarea como todas las tareas

*Continúa en la página siguiente*

de tipo normal. Para obtener más información sobre la declaración de tareas, consulte la documentación sobre los parámetros del sistema.

El modificador `NoRegain` sólo puede usarse en las tareas de movimiento, dado que sólo afecta a la trayectoria del movimiento.

Si hay una instrucción `Stop` en una rutina de evento, la ejecución de la rutina se detiene y la ejecución continúa como se describe en [Stop en la página 823](#).

Si hay una instrucción `Stop\AllMoveTasks` en una rutina de evento de un sistema `MultiMove`, la tarea que contiene la instrucción continúa de la forma descrita en [Stop en la página 823](#) y todas las demás tareas de movimiento que estén ejecutando una rutina de evento continúan de la forma descrita en [Stop \AllMoveTasks en la página 823](#) (con el mismo efecto que un paro de programa normal durante la ejecución de la rutina de evento).

Stop

| Rutinas de evento | Efecto de la instrucción <code>Stop</code>  |
|-------------------|---|
| POWER ON          | La ejecución se detiene para todas las tareas. La ejecución continúa en la rutina de eventos en la siguiente orden de inicio. |
| START             | La ejecución se detiene para todas las tareas. La ejecución continúa en la rutina de eventos en la siguiente orden de inicio. |
| RESTART           | La ejecución se detiene para todas las tareas. La ejecución continúa en la rutina de eventos en la siguiente orden de inicio. |
| STOP              | La ejecución se detiene. La ejecución no continúa en la rutina de eventos en la siguiente orden de inicio.                    |
| QSTOP             | La ejecución se detiene. La ejecución no continúa en la rutina de eventos en la siguiente orden de inicio.                    |
| RESET             | La ejecución se detiene. La ejecución no continúa en la rutina de eventos en la siguiente orden de inicio.                    |

Stop \AllMoveTasks

| Rutinas de evento | Efecto de la instrucción <code>Stop \AllMoveTasks</code>   |
|-------------------|--|
| POWER ON          | La ejecución se detiene para todas las tareas. La ejecución continúa en la rutina de eventos en la siguiente orden de inicio.    |
| START             | La ejecución se detiene para todas las tareas. La ejecución continúa en la rutina de eventos en la siguiente orden de inicio.    |
| RESTART           | La ejecución se detiene para todas las tareas. La ejecución continúa en la rutina de eventos en la siguiente orden de inicio.    |
| STOP              | La ejecución se detiene para todas las tareas. La ejecución no continúa en la rutina de eventos en la siguiente orden de inicio. |

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.274 Stop - Detención de la ejecución del programa

RobotWare - OS

Continuación

| Rutinas de evento | Efecto de la instrucción Stop<br>\AllMoveTasks   |
|-------------------|--|
| QSTOP             | La ejecución se detiene para todas las tareas. La ejecución no continúa en la rutina de eventos en la siguiente orden de inicio. |
| RESET             | La ejecución se detiene. La ejecución no continúa en la rutina de eventos en la siguiente orden de inicio.                       |

### Más ejemplos

A continuación aparecen más ejemplos de cómo usar la instrucción Stop.

#### Ejemplo 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;
```

La ejecución del programa se detiene cuando el robot se encuentra en el punto p1. El operador mueve el robot hasta p1\_read. En el siguiente inicio de programa, el robot no vuelve a p1, de forma que la posición p1\_read pueda almacenarse en el programa.

### Sintaxis

```
Stop  
[ '\ NoRegain ] '|'  
[ '\ AllMoveTasks ]';'
```

### Información relacionada

| Para obtener más información sobre           | Consulte   |
|--|--|
| Finalización de la ejecución del programa    | <a href="#">EXIT - Finaliza la ejecución del programa en la página 212</a>   |
| Paro de los movimientos del robot únicamente | <a href="#">StopMove - Detiene el movimiento del robot en la página 828</a>  |
| Detención del programa para depuración       | <a href="#">Break - Interrumpe la ejecución del programa en la página 52</a> |

### 1.275 STOpen - Abre una herramienta servo

---

#### Utilización

STOpen se utiliza para abrir la herramienta servo.

---

#### Ejemplos básicos

Los ejemplos siguientes ilustran la instrucción STOpen.

##### Ejemplo 1

```
STOpen gun1;
```

Abre la herramienta servo gun1. Se espera hasta que la pistola se haya abierto, antes de continuar con la siguiente instrucción de RAPID.

##### Ejemplo 2

```
STOpen gun1 \Conc;
```

Abre la herramienta servo gun1. Se continúa con la siguiente instrucción de RAPID sin esperar a que se abra la pistola.

##### Ejemplo 3

```
STOpen "SERVOGUN"\WaitZeroSpeed;
```

Se detiene la herramienta servo SERVOGUN, se espera hasta que finalice el movimiento coordinado y se abre a continuación la herramienta servo SERVOGUN.

---

#### Argumentos

```
STOpen ToolName [\WaitZeroSpeed] [\Conc]
```

ToolName

Tipo de dato: string

El nombre de la unidad mecánica.

[\WaitZeroSpeed]

Tipo de dato: switch

Se detiene la herramienta servo, se espera hasta que finalice el movimiento coordinado y se abre a continuación la herramienta servo.

[\Conc]

Tipo de dato: switch

Distintas instrucciones consecutivas se ejecutan mientras la pistola está en movimiento. El argumento puede utilizarse para acortar el tiempo de ciclo. Esto resulta útil, por ejemplo, si se desea controlar dos pistolas al mismo tiempo.

---

#### Ejecución de programas

Si la unidad mecánica existe, se ordena la apertura de la herramienta servo. La fuerza de la punta se reduce a cero y el brazo de la herramienta es devuelto a la posición anterior al cierre.

El brazo de la herramienta se mueve a la velocidad y la aceleración máximas y está definido en los parámetros del sistema del eje externo correspondiente. En

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.275 STOpen - Abre una herramienta servo

### Servo Tool Control

#### Continuación

cuanto a los movimientos de los demás ejes, se utiliza la velocidad reducida en el modo manual.

Es posible abrir la herramienta durante un movimiento programado del robot, siempre y cuando el movimiento del robot no incluya el movimiento del brazo de la herramienta. Si la herramienta se abre durante este movimiento, aparecerá el error 50251 Tool opening failed. Puede usarse el modificador `WaitZeroSpeed` para reducir el riesgo de que este error se produzca.

Si se usa el modificador `Conc`, la instrucción se considerará como ejecutada antes de que se abra la herramienta servo. Se recomienda usar la función `STIsOpen` a continuación de `STOpen` para evitar problemas en el modo concurrente.

Para obtener más detalles, consulte Control de movimiento de herramientas servo.

### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                        | Causa del error   |
|-------------------------------|---|
| <code>ERR_NO_SGUN</code>      | El nombre especificado de la herramienta servo no es una herramienta servo configurada.   |
| <code>ERR_SGUN_NOTACT</code>  | La unidad mecánica de la herramienta servo no está activada.<br>Use la instrucción <code>ActUnit</code> para activar la herramienta servo.  |
| <code>ERR_SGUN_NOTINIT</code> | La posición de la herramienta servo no está inicializada.<br>Es necesario inicializar la posición de la herramienta servo la primera vez que se instala la pistola o después de hacer una calibración fina. Utilice la rutina de servicio <code>ManServiceCalib</code> o realice una calibración de cambio de punta. El desgaste de la punta se restablecerá. |
| <code>ERR_SGUN_NOTSYNC</code> | Las puntas de la herramienta servo no están sincronizadas.<br>Las puntas de la herramienta servo deben sincronizarse si el cuentarrevoluciones se ha perdido y/o actualizado. No se perderá ningún dato de proceso como por ejemplo el desgaste de las puntas.  |



#### Nota

Si la instrucción es ejecutada desde una tarea en segundo plano y se produce una parada de emergencia, la instrucción será finalizada sin ningún error.

### Sintaxis

```
STOpen  
[ 'ToolName :=' ] < expression (IN) of string > ','  
[ '\WaitZeroSpeed' ] ,'  
[ '\Conc' ]'
```

Continúa en la página siguiente

## Información relacionada

| Para obtener más información sobre | Consulte  |
|------------------------------------|---|
| Cierre de una herramienta servo    | <a href="#">STClose - Cierra una herramienta servo en la página 807</a> |

# 1 Instrucciones

---

## 1.276 StopMove - Detiene el movimiento del robot

RobotWare - OS

## 1.276 StopMove - Detiene el movimiento del robot

---

### Utilización

StopMove se utiliza para detener temporalmente los movimientos de los ejes del robot y los ejes externos y cualquier proceso asociado. Si se ejecuta la instrucción StartMove, el movimiento y el proceso se reanudan.

Por ejemplo, esta instrucción puede usarse en una rutina TRAP para parar temporalmente el robot cuando se produce una interrupción.

En el sistema básico, es posible utilizar esta instrucción en el tipo siguiente de tareas de programa:

- Tarea T\_ROB1, para la detención del movimiento en esa tarea.
- Cualquier otra tarea, para la detención del movimiento en la tarea main.

En los sistemas MultiMove, es posible utilizar esta instrucción en el tipo siguiente de tareas de programa:

- Tarea de movimiento, para la detención del movimiento en esa tarea.
- Tarea sin movimiento, para la detención del movimiento en la tarea de movimiento conectada. Además, si el movimiento se detiene en una tarea de movimiento que pertenece a un grupo de tarea coordinada sincronizada, el movimiento se detiene en todas las tareas cooperativas.

---

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción StopMove:

Consulte también [Más ejemplos en la página 829](#).

#### Ejemplo 1

```
StopMove;  
WaitDI ready_input, 1;  
StartMove;
```

El movimiento del robot se detiene hasta que se activa la entrada ready\_input.

---

### Argumentos

```
StopMove [\Quick] [\AllMotionTasks]
```

[\Quick]

**Tipo de dato:** switch

Detiene el robot lo antes posible, sin abandonar la trayectoria.

Si no se utiliza el parámetro opcional \Quick, el robot se detiene en su trayectoria, pero la distancia de frenado es mayor (igual que con un paro de programa normal).

[\AllMotionTasks]

**Tipo de dato:** switch

Detiene el movimiento de todas las unidades mecánicas del sistema. El modificador [\AllMotionTasks] sólo puede usarse desde tareas de programa sin movimiento.

*Continúa en la página siguiente*



### Ejecución de programas

Los movimientos de los ejes del robot y de los ejes externos se detienen sin aplicar los frenos. Cualquier proceso asociado con el movimiento en curso se detiene al mismo tiempo que se detiene el movimiento.

La ejecución del programa continúa tras esperar a que se detengan los ejes del robot ni los ejes externos (en reposo).

Con el modificador `\AllMotionTasks`, (sólo permitido con las tareas de programa sin movimiento), se detienen los movimientos de todas las unidades mecánicas del sistema.

En un sistema base sin el modificador `\AllMotionTasks`, se detienen los movimientos de las unidades mecánicas siguientes:

- Siempre las unidades mecánicas de la tarea principal, independientemente de qué tarea ejecute la instrucción `StopMove`.

En un sistema `MultiMove` sin el modificador `\AllMotionTasks`, se detienen los movimientos de las unidades mecánicas siguientes:

- Las unidades mecánicas de la tarea de movimiento que ejecute `StopMove`.
- Las unidades mecánicas de la tarea de movimiento conectada a la tarea sin movimiento en la que se ejecute `StopMove`. Además, si las unidades mecánicas se detienen en una tarea de movimiento conectada que pertenece a un grupo de tarea coordinada sincronizada, las unidades mecánicas se detienen en todas las tareas cooperativas.

El estado `StopMove` de la tarea de movimiento, generado por la propia tarea de movimiento, se restablece automáticamente al reiniciar la tarea desde el principio.

El estado `StopMove` de la tarea de movimiento conectado, generado por alguna tarea sin movimiento, se restablece automáticamente:

- En el caso de una tarea normal sin movimiento, al iniciar la tarea desde el principio.
- En el caso de una tarea semiestática sin movimiento, al reiniciar tras una caída de alimentación si la tarea se inicia desde el principio.
- En el caso de una tarea estática sin movimiento, al iniciar la installation si la tarea se inicia desde el principio.

### Más ejemplos

A continuación aparecen más ejemplos de la instrucción `StopMove`.

#### Ejemplo 1

```
VAR intnum intnol;
...
PROC main()
...
CONNECT intnol WITH go_to_home_pos;
ISignalDI di1,1,intnol;
...

TRAP go_to_home_pos
VAR robtarg p10;
```

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.276 StopMove - Detiene el movimiento del robot

RobotWare - OS

Continuación

```
StopMove;
StorePath;
p10:=CRobT(\Tool:=tool1 \WObj:=wobj0);
MoveL home,v500,fine,tool1;
WaitDI dil,0;
Move L p10,v500,fine,tool1;
RestoPath;
StartMove;
ENDTRAP
```

Cuando se asigna el valor 1 a la entrada `dil`, se activa una interrupción que activa a su vez la rutina de interrupción `go_to_home_pos`. El movimiento actual se detiene y el robot pasa a moverse hacia la posición `home`. Cuando se cambia a 0 la señal `dil`, el robot vuelve a la posición en la cual se produjo la interrupción y sigue moviéndose a lo largo de la trayectoria programada.

### Ejemplo 2

```
VAR intnum intnol;
...
PROC main()
...
CONNECT intnol WITH go_to_home_pos;
ISignalDI dil,1,intnol;
...

TRAP go_to_home_pos ()
VAR robtaraget p10;
StorePath;
p10:=CRobT(\Tool:=tool1 \WObj:=wobj0);
MoveL home,v500,fine,tool1;
WaitDI dil,0;
MoveL p10,v500,fine,tool1;
RestoPath;
StartMove;
ENDTRAP
```

Es similar al ejemplo anterior, pero el robot no se mueve hacia la posición `home` hasta que se completa la instrucción de movimiento actual.

---

### Limitaciones

Sólo se permite a una de las distintas tareas sin movimiento la realización de la secuencia `StopMove - StartMove` respecto de alguna tarea de movimiento.

---

### Sintaxis

```
StopMove
['\Quick]
['\AllMotionTasks'];'
```

Continúa en la página siguiente

### Información relacionada

| Para obtener más información sobre            | Consulte  |
|---|---|
| Continuación de un movimiento                 | <a href="#">StartMove</a> - Reanuda el movimiento del robot en la página 796<br><a href="#">StartMoveRetry</a> - Reanuda el movimiento y la ejecución del robot en la página 799                              |
| Almacenamiento y restauración de trayectorias | <a href="#">StorePath</a> - Almacena la trayectoria cuando se produce una interrupción en la página 835<br><a href="#">RestoPath</a> - Restablece la trayectoria después de una interrupción en la página 615 |

# 1 Instrucciones

---

1.277 StopMoveReset - Restablece el estado de movimiento de paro de sistema  
RobotWare - OS

## 1.277 StopMoveReset - Restablece el estado de movimiento de paro de sistema

---

### Utilización

StopMoveReset se usa para restablecer el estado de movimiento de paro de sistema, sin iniciar ningún movimiento.

Los errores de movimiento elevados asincrónicamente, como ERR\_PATH\_STOP o un error de proceso específico durante el movimiento, pueden gestionarse en el gestor de ERROR. Cuando se produce un error de este tipo, los movimientos se detienen inmediatamente y el indicador de movimiento de paro de sistema se activa para las tareas de programa actuales. Esto significa que el movimiento no se reinicia si se realiza cualquier reinicio de programa mientras el puntero de programa está dentro del gestor de ERROR.

El reinicio de los movimientos tras un error de movimiento de este tipo tiene lugar una vez realizada una de las acciones siguientes:

- Ejecución de StartMove o StartMoveRetry.
- Ejecución de StopMoveReset, con reanudación del movimiento con el siguiente inicio de programa.

---

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción StopMoveReset:

#### Ejemplo 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
    ! stop the program
    ...
    ! Reset the move stop flag, so it's possible to manual restart
    ! the program and the movement after that the program has
    ! stopped
    StopMoveReset;
  ENDIF
ENDPROC
```

Una vez que el gestor de ERROR mostrado anteriormente ha ejecutado ENDPROC, la ejecución del programa se detiene y el puntero se sitúa al comienzo de la instrucción ArcL. El siguiente inicio de programa reinicia el programa y los movimientos desde la posición en la que se produjo el error de movimiento original.

---

### Argumentos

StopMoveReset [\AllMotionTasks]

*Continúa en la página siguiente*

## 1.277 StopMoveReset - Restablece el estado de movimiento de paro de sistema RobotWare - OS Continuación

`[\AllMotionTasks]`

Tipo de dato: `switch`

Restablece el estado de paro de movimiento del sistema en todas las unidades mecánicas del sistema. El modificador `[\AllMotionTasks]` sólo puede usarse desde tareas de programa sin movimiento.

### Ejecución de programas

Para restablecer una aplicación MultiMove en el modo sincronizado coordinado, `StopMoveReset` debe ser ejecutado en todas las tareas de movimiento implicadas en la coordinación.

Con el modificador `\AllMotionTasks` (sólo permitido con las tareas de programa sin movimiento), el restablecimiento se realiza en todas las unidades mecánicas del sistema.

En un sistema básico sin el modificador `\AllMotionTasks`, el restablecimiento siempre se realiza en la tarea main, independientemente de qué tarea ejecute la instrucción `StopMoveReset`.

En el sistema básico, es posible utilizar `StopMoveReset` en el tipo siguiente de tareas de programa:

- La tarea principal `T_ROB1` para restablecer el estado de paro de movimiento en esa tarea.
- Cualquier otra tarea para restablecer el estado de paro de movimiento en la tarea principal.

En el sistema MultiMove, es posible utilizar esta instrucción en el tipo siguiente de tareas de programa:

- Una tarea de movimiento para restablecer el estado de paro de movimiento en esa tarea.
- Una tarea sin movimiento para restablecer el estado de paro de movimiento en la tarea de movimiento conectada. Además, si el restablecimiento del estado de paro de movimiento se realiza en una tarea de movimiento conectada perteneciente a un grupo de tareas coordinadas sincronizadas, el estado de paro de movimiento también se restablece en todas las tareas cooperativas.

### Sintaxis

```
StopMoveReset
  ['\AllMotionTasks'];'
```

### Información relacionada

| Para obtener más información sobre | Consulte   |
|------------------------------------|--|
| Detención del movimiento           | <a href="#">StopMove - Detiene el movimiento del robot en la página 828</a>  |
| Continuación de un movimiento      | <a href="#">StartMove - Reanuda el movimiento del robot en la página 796</a><br><a href="#">StartMoveRetry - Reanuda el movimiento y la ejecución del robot en la página 799</a> |

*Continúa en la página siguiente*

# 1 Instrucciones

---

1.277 StopMoveReset - Restablece el estado de movimiento de paro de sistema

RobotWare - OS

Continuación

| Para obtener más información sobre            | Consulte  |
|---|---|
| Almacenamiento y restauración de trayectorias | <i>StorePath - Almacena la trayectoria cuando se produce una interrupción en la página 835</i><br><i>RestoPath - Restablece la trayectoria después de una interrupción en la página 615</i> |

## 1.278 StorePath - Almacena la trayectoria cuando se produce una interrupción

### Utilización

`StorePath` se utiliza para almacenar la trayectoria de movimiento que se está ejecutando, por ejemplo cuando se produce un error o una interrupción. A continuación, el gestor de errores o la rutina TRAP pueden iniciar un nuevo movimiento temporal y reiniciar por último el movimiento original que se ha almacenado previamente.

Por ejemplo, esta instrucción puede usarse para ir a una posición de servicio o limpiar una pistola cuando se produce un error.

Esta instrucción sólo puede usarse en la tarea principal `T_ROB1` o, si se cuenta con un sistema MultiMove, en las tareas de movimiento.

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción `StorePath`:

Consulte también [Más ejemplos en la página 836](#).

#### Ejemplo 1

```
StorePath;
```

Se almacena para un uso posterior la trayectoria de movimiento actual. Se cambia el sistema al modo de movimiento independiente.

#### Ejemplo 2

```
StorePath \KeepSync;
```

Se almacena para un uso posterior la trayectoria de movimiento actual. Se mantiene el modo de movimiento sincronizado.

### Argumentos

```
StorePath [ \KeepSync ]
```

[ \KeepSync ]

#### **Keep Synchronization**

Tipo de dato: `switch`

Mantiene el modo de movimiento sincronizado tras la instrucción `StorePath \KeepSync`. El modificador `KeepSync` sólo puede usarse si el sistema se encuentra en el modo de movimiento sincronizado antes de la llamada a `StorePath \KeepSync`.

Sin el parámetro opcional `\KeepSync`, en un sistema MultiMove con movimiento sincronizado, el sistema cambia al modo de movimiento independiente semicoordinado. Tras la ejecución de `StorePath` en todas las tareas implicadas, el sistema se encuentra en el modo semicoordinado si se sigue usando el objeto de trabajo coordinado. De lo contrario, se encuentra en el modo independiente.

En el modo semicoordinado, se recomienda empezar siempre con un movimiento en la unidad mecánica que controla la base de coordenadas del usuario antes de usar `WaitSyncTask` en todas las tareas implicadas.

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.278 StorePath - Almacena la trayectoria cuando se produce una interrupción

RobotWare - OS

Continuación

---

### Ejecución de programas

Se guarda la trayectoria actual de movimientos de los ejes del robot y los ejes externos. Después de esta operación, es posible iniciar otro movimiento en una rutina TRAP o un gestor de errores. Tras corregir el motivo del error o de la interrupción, es posible restaurar la trayectoria de movimientos guardada.

---

### Más ejemplos

A continuación aparecen más ejemplos de cómo usar la instrucción `StorePath`.

#### Ejemplo 1

```
TRAP machine_ready
  VAR robtarget p1;
  StorePath;
  p1 := CRobT();
  MoveL p100, v100, fine, tool1;
  ...
  MoveL p1, v100, fine, tool1;
  RestoPath;
  StartMove;
ENDTRAP
```

Cuando se produce una interrupción que activa la rutina `TRAP machine_ready`, la trayectoria de movimiento que está ejecutando el robot en ese momento se detiene al final de la instrucción (`ToPoint`) y se almacena. A continuación, el robot soluciona la interrupción, por ejemplo, sustituyendo una pieza de la máquina. A continuación, se reinicia el movimiento normal.

---

### Limitaciones

Con la instrucción `StorePath` sólo se almacenan los datos de la trayectoria de movimiento.

Si el usuario desea solicitar movimientos en un nuevo nivel de trayectoria, es necesario almacenar la posición de paro directamente a continuación de que `StorePath` y de `RestoPath` hacen el movimiento hacia la posición de paro almacenada en la trayectoria.

Sólo es posible tener almacenada una trayectoria de movimiento cada vez.

---

### Sintaxis

```
StorePath
  ['\KeepSync'];'
```

---

### Información relacionada

| Para obtener más información sobre | Consulte   |
|------------------------------------|--|
| Restauración de una trayectoria    | <a href="#">RestoPath - Restablece la trayectoria después de una interrupción en la página 615</a> |

---

Continúa en la página siguiente



## 1.278 StorePath - Almacena la trayectoria cuando se produce una interrupción

*RobotWare - OS*

*Continuación*

| Para obtener más información sobre | Consulte  |
|------------------------------------|---|
| Más ejemplos                       | <p><i><a href="#">RestoPath</a> - Restablece la trayectoria después de una interrupción en la página 615</i></p> <p><i><a href="#">PathRecStart</a> - Inicia la grabadora de trayectorias en la página 524</i></p> <p><i><a href="#">SyncMoveResume</a> - Activa el modo de movimientos sincronizados coordinados en la página 860</i></p> <p><i><a href="#">SyncMoveSuspend</a> - Activa el movimiento independiente-semicoordinado en la página 862</i></p> |

# 1 Instrucciones

---

## 1.279 StrFormat: dar formato a una cadena

RobotWare - OS

## 1.279 StrFormat: dar formato a una cadena

---

### Utilización

`StrFormat` se utiliza para dar formato a una cadena de texto.

---

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción `StrFormat`:

#### Ejemplo 1

```
VAR string text1 := "Esto es un {1} y esto es {2}";  
...  
TPWrite StrFormat(text1 \Arg1:="robot" \Arg2:="fast");
```

Las cadenas utilizadas en los argumentos opcionales `Arg1` y `Arg2` sustituirán `{1}` y `{2}`. El resultado será:

**This is a robot and this is fast**

---

### Argumentos

```
StrFormat ( Text [\Arg1] [\Arg2] [\Arg3] [\Arg4] [\Arg5] [\Arg6]  
            )
```

Text

Tipo de dato: `string`

La cadena a la que se desea dar formato.

`[\Arg1]`

Tipo de dato: `string`

Si la cadena contiene `{1}`, el `{1}` se sustituye con la cadena utilizada en este argumento.

`[\Arg2]`

Tipo de dato: `string`

Si la cadena contiene `{2}`, el `{2}` se sustituye con la cadena utilizada en este argumento.

`[\Arg3]`

Tipo de dato: `string`

Si la cadena contiene `{3}`, el `{3}` se sustituye con la cadena utilizada en este argumento.

`[\Arg4]`

Tipo de dato: `string`

Si la cadena contiene `{4}`, el `{4}` se sustituye con la cadena utilizada en este argumento.

`[\Arg5]`

Tipo de dato: `string`

Si la cadena contiene `{5}`, el `{5}` se sustituye con la cadena utilizada en este argumento.

*Continúa en la página siguiente*

[\Arg6]

Tipo de dato: string

Si la cadena contiene {6}, el {6} se sustituye con la cadena utilizada en este argumento.

### Limitaciones

La longitud total de la cadena en RAPID es de 80 caracteres. Es la misma para todas las entradas como resultado de la función.

### Sintaxis

```
StrFormat
[Text ':=' ] <expression (IN) of string>
['\ ' Arg1 ':=' <expression (IN) of string>]
['\ ' Arg2 ':=' <expression (IN) of string>]
['\ ' Arg3 ':=' <expression (IN) of string>]
['\ ' Arg4 ':=' <expression (IN) of string>]
['\ ' Arg5 ':=' <expression (IN) of string>]
['\ ' Arg6 ':=' <expression (IN) of string>] )'
```

### Información relacionada

| Para obtener más información sobre   | Consulte   |
|--------------------------------------|--|
| Funciones para cadenas de caracteres | <i>Manual de referencia técnica - RAPID Overview</i> |
| Tipo de dato string                  | <a href="#">string - Cadenas en la página 1778</a>   |
| Valores de cadena de caracteres      | <i>Manual de referencia técnica - RAPID Overview</i> |
| <i>Advanced RAPID</i>                | <i>Application manual - Controller software IRC5</i> |

# 1 Instrucciones

---

## 1.280 STTune - Ajusta una herramienta servo

*Servo Tool Control*

## 1.280 STTune - Ajusta una herramienta servo

---

### Utilización

STTune se utiliza para ajustar/cambiar un parámetro de una herramienta servo. Este parámetro se cambia temporalmente respecto del valor original, que se configura en los parámetros del sistema. El nuevo valor ajustado se activa inmediatamente tras la ejecución de la instrucción.

STTune resulta útil en los procedimientos de ajuste. Estos procedimientos de ajuste suelen utilizarse para encontrar el valor óptimo de un parámetro. Un experimento (por ejemplo la ejecución de un programa con un movimiento de herramienta servo) se repite al utilizar distintos valores de ajuste.

STTune no debe utilizarse durante la calibración o el cierre de la herramienta.

---

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción STTune:

#### Ejemplo 1

```
STTune SEOLO_RG, 0.050, CloseTimeAdjust;
```

El parámetro de herramienta servo `CloseTimeAdjust` se cambia temporalmente a 0.050 segundos.

---

### Argumentos

```
STTune MecUnit TuneValue Type
```

MecUnit

**Tipo de dato:** `mecunit`

El nombre de la unidad mecánica.

TuneValue

**Tipo de dato:** `num`

El nuevo valor de ajuste.

Type

**Tipo de dato:** `tunegtype`

El tipo de parámetro. Los parámetros de herramienta servo disponibles para el ajuste son `RampTorqRefOpen`, `RampTorqRefClose`, `KV`, `SpeedLimit`, `CollAlarmTorq`, `CollContactPos`, `CollisionSpeed`, `CloseTimeAdjust`, `ForceReadyDelayT`, `PostSyncTime`, `CalibTime`, `CalibForceLow`, `CalibForceHigh`. Estos tipos están predefinidos en los parámetros del sistema y constituyen los valores originales.

---

### Descripción

`RampTorqRefOpen`

Ajusta el parámetro del sistema `Ramp when decrease force`, que decide con qué rapidez se libera la fuerza al abrir la herramienta. La unidad es Nm/s y su valor típico es 200.

---

*Continúa en la página siguiente*

Parámetro de sistema correspondiente: Tema *Motion*, tipo *Force master*, parámetro `ramp_torque_ref_opening`.

#### RampTorqRefClose

Ajusta el parámetro del sistema `Ramp when increase force`, que decide con qué rapidez se acumula la fuerza al abrir la herramienta. La unidad es Nm/s y su valor típico es 80.

Parámetro de sistema correspondiente: Tema *Motion*, tipo *Force master*, parámetro `ramp_torque_ref_closing`.

#### KV

Ajusta el parámetro de sistema `KV`, que se utiliza para limitar la velocidad. La unidad es Nms/rad y un valor típico es 1. Para obtener más detalles, consulte la documentación del eje externo.

Parámetro de sistema correspondiente: Tema *Motion*, tipo *Force master*, parámetro `Kv`.

#### SpeedLimit

Ajusta el parámetro de sistema `Speed limit`, que se utiliza para limitar la velocidad. La unidad es rad/s (la velocidad del motor) y un valor típico es 60. Para obtener más detalles, consulte la documentación del eje externo.

Parámetro de sistema correspondiente: Tema *Motion*, tipo *Force master*, parámetro `speed_limit`.

#### CollAlarmTorq

Ajusta el parámetro de sistema `Collision alarm torque`, que se utiliza para la calibración automática de nuevas puntas. La unidad es Nm (par del motor) y un valor típico es 1. Para obtener más detalles, consulte la documentación del eje externo.

Parámetro de sistema correspondiente: Tema *Motion*, tipo *Force master*, parámetro `alarm_torque`.

#### CollContactPos

Ajusta el parámetro de sistema `Collision delta pos`, que se utiliza para la calibración automática de nuevas puntas. La unidad es m y un valor típico es 0,002. Para obtener más detalles, consulte la documentación del eje externo.

Parámetro de sistema correspondiente: Tema *Motion*, tipo *Force master*, parámetro `distance_to_contact_position`.

#### CollisionSpeed

Ajusta el parámetro de sistema `Collision speed`, que se utiliza para la calibración automática de nuevas puntas. La unidad es m/s y un valor típico es 0,02. Para obtener más detalles, consulte la documentación del eje externo.

Parámetro de sistema correspondiente: Tema *Motion*, tipo *Force master*, parámetro `col_speed`.

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.280 STTune - Ajusta una herramienta servo

### *Servo Tool Control*

#### *Continuación*

#### CloseTimeAdjust

El ajuste de tiempo constante (s), positivo o negativo, del momento en el que las puntas de la herramienta establecen contacto durante el cierre de la herramienta. Puede usarse para retrasar ligeramente el cierre cuando se utiliza el precierre sincronizado durante las soldaduras.

Parámetro de sistema correspondiente: Tema *Motion*, tipo *SG process*, parámetro `min_close_time_adjust`.

#### ForceReadyDelayT

Retardo de tiempo constante (s) antes del envío de la señal de soldadura preparada después de alcanzar la fuerza programada.

Parámetro de sistema correspondiente: Tema *Motion*, tipo *SG process*, parámetro `pre_sync_delay_time`.

#### PostSyncTime

Anticipación de tiempo de liberación (s) del siguiente movimiento del robot tras una soldadura. Este tipo de ajuste puede ajustarse para sincronizar la apertura de la pistola con el siguiente movimiento del robot. La sincronización puede fallar si los parámetros tienen valores demasiado elevados.

Parámetro de sistema correspondiente: Tema *Motion*, tipo *SG process*, parámetro `post_sync_time`.

#### CalibTime

El tiempo de espera (s) durante la calibración antes de que se realice la corrección de posición de la punta de la herramienta. Para obtener los mejores resultados, no utilice un valor demasiado bajo, por ejemplo 0,5 s.

Parámetro de sistema correspondiente: Tema *Motion*, tipo *SG process*, parámetro `calib_time`.

#### CalibForceLow

La fuerza mínima de la punta (N) utilizada durante una calibración. Para obtener los mejores resultados de detección de grosor, se recomienda utilizar la mínima fuerza de soldadura programada.

Parámetro de sistema correspondiente: Tema *Motion*, tipo *SG process*, parámetro `calib_force_low`.

#### CalibForceHigh

La fuerza máxima de la punta (N) utilizada durante una calibración. Para obtener los mejores resultados de detección de grosor, se recomienda utilizar la máxima fuerza de soldadura programada.

Parámetro de sistema correspondiente: Tema *Motion*, tipo *SG process*, parámetro `calib_force_high`.

---

## Ejecución de programas

El tipo de ajuste especificado y el valor de ajuste se activan para la unidad mecánica especificada. Este valor se aplica a todos los movimientos hasta que se programa un nuevo valor para la unidad mecánica actual o hasta que se restablecen los tipos y valores de ajuste mediante la instrucción `STTuneReset`.

*Continúa en la página siguiente*

Los valores de ajuste originales pueden cambiarse para siempre en los parámetros del sistema.

Los valores predeterminados de ajuste de la herramienta servo se establecen automáticamente en los casos siguientes:

- Al ejecutar la instrucción `STTuneReset`.
- en un Reinicio.

### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre      | Causa del error   |
|-------------|---|
| ERR_NO_SGUN | El nombre especificado de la herramienta servo no es una herramienta servo configurada. |

### Sintaxis

```
STTune
  [ MecUnit ':= ' ] < variable (VAR) of mecunit > ','
  [ TuneValue ':= ' ] < expression (IN) of num > ','
  [ 'Type ':= ' ] < expression (IN) of tunegtype > '];'
```

### Información relacionada

| Para obtener más información sobre                   | Consulte   |
|--|--|
| Restablecimiento de parámetros de herramientas servo | <a href="#">TuneReset - Restablecimiento del ajuste del servo en la página 993</a> |
| Ajuste de herramientas servo                         | <i>Application manual - Additional axes and stand alone controller</i>             |

# 1 Instrucciones

## 1.281 STTuneReset - Restablece el ajuste de la herramienta servo *Servo Tool Control*

### 1.281 STTuneReset - Restablece el ajuste de la herramienta servo

#### Utilización

STTuneReset se utiliza para restablecer los valores originales de los parámetros de la herramienta servo si han sido cambiados con la instrucción STTune.

#### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción STTuneReset:

#### Ejemplo 1

```
STTuneReset SEOLO_RG;
```

Restablece los *valores originales de los parámetros de la herramienta servo* de la unidad mecánica SEOLO\_RG.

#### Argumentos

```
STTuneReset MecUnit
```

MecUnit

Tipo de dato: mecunit

El nombre de la unidad mecánica.

#### Ejecución de programas

Se restablecen los parámetros originales de la herramienta servo.

También se realiza en un Reinicio.

#### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema ERRNO cambiará a:

| Nombre      | Causa del error   |
|-------------|---|
| ERR_NO_SGUN | El nombre especificado de la herramienta servo no es una herramienta servo configurada. |

#### Sintaxis

```
STTuneReset  
[ MecUnit ':= ' ] < variable (VAR) of mecunit > ','
```

#### Información relacionada

| Para obtener más información sobre         | Consulte   |
|--|--|
| Ajuste de parámetros de herramientas servo | <a href="#">STTune - Ajusta una herramienta servo en la página 840</a> |
| Ajuste de parámetros de herramientas servo | <i>Application manual - Additional axes and stand alone controller</i> |



## 1.282 SupSyncSensorOff - Parada de la supervisión de sensor sincronizada *Machine Synchronization*

### 1.282 SupSyncSensorOff - Parada de la supervisión de sensor sincronizada

#### Utilización

SupSyncSensorOff se utiliza para detener la supervisión del movimiento del robot y el movimiento de sensor sincronizado.

#### Ejemplo básico

A continuación aparece un ejemplo básico de la instrucción SupSyncSensorOff.

#### Ejemplo

```
SupSyncSensorOff SSYNCl;
```

El sensor ya no se supervisa.

#### Argumentos

```
SupSyncSensorOff MechUnit
```

MechUnit

*Mechanical unit*

Tipo de dato: mecunit

El nombre de la unidad mecánica.

#### Sintaxis

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

#### Información relacionada

| Para obtener más información sobre              | Consulte   |
|---|--|
| Inicio de la supervisión de sensor sincronizada | <a href="#">SupSyncSensorOn - Inicio de la supervisión de sensor sincronizada en la página 846</a> |
| Sincronización con un sensor                    | <a href="#">SyncToSensor - Sincronización con un sensor en la página 866</a>                       |
| <i>Machine Synchronization</i>                  | <i>Application manual - Controller software IRC5</i>   |

## 1 Instrucciones

---

1.283 SupSyncSensorOn - Inicio de la supervisión de sensor sincronizada  
*Machine Synchronization*

### 1.283 SupSyncSensorOn - Inicio de la supervisión de sensor sincronizada

---

#### Utilización

SupSyncSensorOn se utiliza para iniciar la supervisión entre el movimiento del robot y el movimiento de sensor sincronizado.

---

#### Ejemplo básico

A continuación aparece un ejemplo básico de la instrucción SupSyncSensorOn.

#### Ejemplo

```
SupSyncSensorOn Ssync1, 150, 100, 50
```

La unidad mecánica Ssync1 se supervisa cuando el sensor está posicionado entre 50 y 150. La supervisión finaliza si la distancia entre el robot y el sensor es menor de 100.

---

#### Argumentos

```
SupSyncSensorOn MechUnit MaxSyncSup SafetyDist MinSyncSup  
[\SafetyDelay]
```

#### MechUnit

***Mechanical unit***

Tipo de dato: mecunit

El nombre de la unidad mecánica.

#### MaxSyncSup

***Maximal Synchronized supervised position***

Tipo de dato: num

El robot supervisa el sensor hasta que este va más allá de la posición de sincronización máxima. Cuando se rebasa este punto, la supervisión se detiene. La unidad se indica en mm.

#### SafetyDist

***Safety distance***

Tipo de dato: num

Safetydist es el límite de la diferencia entre la posición esperada de la máquina y la posición real de la máquina. Debe ser negativa, es decir, el modelo debe moverse siempre con adelanto con respecto a la máquina real. En el caso de las posiciones de máquina decrecientes, el límite debe ser negativo correspondiendo a la diferencia negativa máxima de la posición (y la distancia mínima de avance). En el caso de las posiciones de máquina crecientes, el límite debe ser positivo correspondiendo a la diferencia positiva mínima de la posición (y la distancia mínima de avance).

El robot disparará una alarma si la distancia entre el robot y el sensor es menor que la distancia de seguridad. Cuando se dispara la alarma, la supervisión se detiene.

La unidad son los mm.

*Continúa en la página siguiente*

---

## 1.283 SupSyncSensorOn - Inicio de la supervisión de sensor sincronizada

### *Machine Synchronization*

#### *Continuación*

MinSyncSup

*Minimal synchronized supervised position*

Tipo de dato: num

El robot inicia la supervisión cuando el sensor se encuentra en la ventana definida desde la posición MinSyncSup hasta la posición MaxSyncSup. La unidad son los mm.

[\SafetyDelay]

*Safety delay*

Tipo de dato: num

SafetyDelay se utiliza para ajustar el retardo entre la posición programada el robot y la posición supervisada del sensor. La unidad son los segundos.

### Limitaciones

Si se usa el SupSynSensorOn antes de que finalice la instrucción WaitSensor, el robot se parará.

### Sintaxis

```
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 > ';'

```

### Información relacionada

| Para obtener más información sobre              | Consulte  |
|---|---|
| Parada de la supervisión de sensor sincronizada | <a href="#">SupSyncSensorOff - Parada de la supervisión de sensor sincronizada en la página 845</a> |
| Sincronización con un sensor                    | <a href="#">SyncToSensor - Sincronización con un sensor en la página 866</a>                        |
| <i>Machine Synchronization</i>                  | <i>Application manual - Controller software IRC5</i>  |

## 1 Instrucciones

---

### 1.284 SyncMoveOff - Finaliza los movimientos sincronizados coordinados

*RW-MRS Synchronized*

### 1.284 SyncMoveOff - Finaliza los movimientos sincronizados coordinados

---

#### Utilización

`SyncMoveOff` se utiliza para finalizar una secuencia de movimientos sincronizados y, en muchos casos, movimientos coordinados. En primer lugar, todas las tareas de programa implicadas deben esperar la sincronización en un punto de paro, tras lo cual los planificadores de movimientos de las tareas de programa implicadas cambian al modo independiente.

La instrucción `SyncMoveOff` sólo puede usarse en un sistema *MultiMove* que tenga la opción *Coordinated Robots* y sólo en las tareas de programa definidas como `Motion Task`.



#### ¡AVISO!

Para conseguir un funcionamiento seguro de la sincronización, cada punto de reunión (parámetro `SyncID`) debe tener un nombre exclusivo. El nombre del punto de reunión también debe ser el mismo en todas las tareas de programa que deban coincidir.

---

#### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción `SyncMoveOff`:

Consulte también [Más ejemplos en la página 850](#).

#### Ejemplo 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;
...
```

*Continúa en la página siguiente*

## 1.284 SyncMoveOff - Finaliza los movimientos sincronizados coordinados *RW-MRS Synchronized* *Continuación*

La tarea de programa que llegue en primer lugar a `SyncMoveOff` con la identidad `sync2` espera hasta que la otras tareas lleguen a su `SyncMoveOff` con la misma identidad `sync2`. En ese punto de sincronización `sync2`, los planificadores de movimientos de las tareas de programa implicadas cambian al modo independiente. A continuación, tanto la tarea `T_ROB1` como la tarea `T_ROB2` prosiguen su ejecución.

### Argumentos

`SyncMoveOff SyncID [\TimeOut]`

`SyncID`

#### *Synchronization Identity*

Tipo de dato: `syncident`

Variables que especifican el nombre del punto en el que se anula la sincronización (reunión). El tipo de dato `syncident` es un tipo de dato sin valor. Sólo se utiliza como identificador para denominar el punto de anulación de la sincronización.

La variable debe ser definida y tener un nombre igual en todas las tareas de programa cooperantes. Se recomienda definir siempre la variable global en cada tarea (`VAR syncident ...`).

`[\TimeOut]`

Tipo de dato: `num`

El tiempo máximo que debe esperarse hasta que las demás tareas de programa lleguen hasta el punto de anulación de la sincronización. El tiempo límite se define en segundos (resolución 0,001 s).

Si el tiempo se agota antes de que todas las tareas de programa alcancen el punto de anulación de la sincronización, se llama al gestor de errores si lo hay, con el código de error `ERR_SYNCMOVEOFF`. Si no hay ningún gestor de errores, se detiene la ejecución.

Si se omite este argumento, la tarea de programa esperará de forma indefinida.

### Ejecución de programas

La tarea de programa que llegue en primer lugar a `SyncMoveOff` espera hasta que todas las demás tareas especificadas alcancen `SyncMoveOff` con la misma identidad `SyncID`. En ese punto de anulación de la sincronización `SyncID`, el planificador de movimientos de las tareas de programa implicadas cambia al modo independiente. A continuación, las tareas de programa implicadas prosiguen su ejecución.

Los planificadores de movimientos de las tareas de programa implicadas cambian al modo no sincronizado. Esto significa lo siguiente:

- Todas las tareas de programa de RAPID y todos los movimientos de estas tareas vuelven a funcionar de forma independiente las unas de las otras.
- **Ninguna** instrucción `Move` debe estar marcada con ningún número de ID. Consulte la instrucción `MoveL`.

Es posible excluir las tareas de programa de las funciones de pruebas del panel de selección de tareas del FlexPendant. Las instrucciones `SyncMoveOn` y

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.284 SyncMoveOff - Finaliza los movimientos sincronizados coordinados

*RW-MRS Synchronized*

*Continuación*

SyncMoveOff **seguirán funcionando con el número reducido de tareas de programa, incluso si sólo es una tarea de programa.**

---

### Más ejemplos

A continuación aparecen más ejemplos de cómo usar la instrucción SyncMoveOff.

#### Ejemplo de movimiento sincronizado simple

```
!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;
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;
```

*Continúa en la página siguiente*

### 1.284 SyncMoveOff - Finaliza los movimientos sincronizados coordinados *RW-MRS Synchronized* *Continuación*

```

UNDO
  SyncMoveUndo;
ENDPROC

```

En primer lugar, las tareas de programa T\_ROB1 y T\_ROB2 esperan mutuamente en `WaitSyncTask` con la identidad `sync1`, programada con una trayectoria de esquina para los movimientos precedentes, con el fin de ahorrar tiempo de ciclo. A continuación, las tareas de programa esperan mutuamente en `SyncMoveOn` con la identidad `sync2`, programada con un punto de paro necesario para los movimientos precedentes. Después, el planificador de movimientos de las tareas de programa implicadas cambia al modo sincronizado.

A continuación, T\_ROB2 mueve `obj2` a ID punto 10 y 20 en el sistema de coordenadas mundo mientras T\_ROB1 mueve `tcp1` a ID punto 10 y 20 en el objeto en movimiento `obj2`.

A continuación, las tareas de programa esperan mutuamente en `SyncMoveOff` con la identidad `sync3`, programada con un punto de paro necesario para los movimientos precedentes. Después, el planificador de movimientos de las tareas de programa implicadas cambia al modo independiente.

#### Ejemplo con recuperación de errores

```

!Program example with use of time-out function
VAR syncident sync3;

...
SyncMoveOff sync3 \TimeOut := 60;
...
ERROR
  IF ERRNO = ERR_SYNCMOVEOFF THEN
    RETRY;
  ENDIF

```

La tarea de programa espera en la instrucción `SyncMoveOff` hasta que otra tarea de programa alcance el mismo punto de sincronización `sync3`. Después de esperar en 60 segundos, se llama al gestor de errores con `ERRNO` cambiado al valor `ERR_SYNCMOVEOFF`. A continuación, se llama de nuevo a la instrucción `SyncMoveOff` para una espera adicional en 60 segundos.

#### Ejemplo de movimiento semicoordinado y movimiento coordinado

```

!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;

```

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.284 SyncMoveOff - Finaliza los movimientos sincronizados coordinados

### *RW-MRS Synchronized*

#### *Continuación*

```
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;
!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
```

Al cambiar entre el movimiento semicoordinado y el movimiento sincronizado, se requiere una instrucción `WaitSyncTask` (si se utiliza la identidad `sync1`).

Al cambiar entre el movimiento sincronizado y el movimiento semicoordinado, la tarea que mueve el objeto de trabajo (`rob2_obj`) debe moverse hasta la posición deseada. A continuación se requiere `WaitSyncTask` (identidad `sync3`) para poder realizar el movimiento semicoordinado.

*Continúa en la página siguiente*



## 1.284 SyncMoveOff - Finaliza los movimientos sincronizados coordinados RW-MRS Synchronized Continuación

### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                       | Causa del error                             |
|------------------------------|---|
| <code>ERR_SYNCMOVEOFF</code> | Tiempo límite en <code>SyncMoveOff</code> . |

### Limitaciones

La instrucción `SyncMoveOff` sólo puede ejecutarse si todos los robots implicados están detenidos en un punto de paro.

Si esta instrucción va precedida de una instrucción de movimiento, ésta última debe programarse con un punto de paro (zonedata `fine`), no un punto de paso. De lo contrario, no será posible reanudar la ejecución tras una caída de suministro eléctrico.

`SyncMoveOff` no puede ejecutarse en rutinas de `RAPID` que estén conectadas a los siguientes eventos especiales del sistema: `PowerOn`, `Stop`, `QStop`, `Restart`, `Reset` o `Step`.

### Sintaxis

```
SyncMoveOff
[ SyncID ':= ' ] < variable (VAR) of syncident>
[ '\Timeout' := ' < expression (IN) of num> ] ';'

```

### Información relacionada

| Para obtener más información sobre                            | Consulte   |
|---|--|
| Especificación de tareas de programa cooperativas             | <a href="#">tasks - Tareas de programa RAPID en la página 1791</a>   |
| Identidad para punto de sincronización                        | <a href="#">syncident - Identidad de punto de sincronización en la página 1786</a>                             |
| Inicio de movimientos sincronizados coordinados               | <a href="#">SyncMoveOn - Inicia los movimientos sincronizados coordinados en la página 854</a>                 |
| Definición de movimientos independientes                      | <a href="#">SyncMoveUndo - Activa los movimientos independientes en la página 864</a>                          |
| Comprobación de si está activado el modo sincronizado         | <a href="#">IsSyncMoveOn - Comprueba si el modo de movimiento sincronizado está activado en la página 1364</a> |
| MultiMove que tenga la opción <code>Coordinated robots</code> | <a href="#">Manual de aplicaciones - MultiMove</a>   |

## 1 Instrucciones

---

### 1.285 SyncMoveOn - Inicia los movimientos sincronizados coordinados

*RW-MRS Synchronized*

### 1.285 SyncMoveOn - Inicia los movimientos sincronizados coordinados

---

#### Utilización

`SyncMoveOn` se utiliza para iniciar una secuencia de movimientos sincronizados, en la mayoría de los casos movimientos coordinados. En primer lugar, todas las tareas de programa implicadas deben esperar la sincronización en un punto de paro, tras lo cual el planificador de movimientos de las tareas de programa implicadas cambian al modo sincronizado.

La instrucción `SyncMoveOn` sólo puede usarse en un sistema *MultiMove* que tenga la opción *Coordinated Robots* y sólo en las tareas de programa definidas como `Motion Task`.



#### ¡AVISO!

Para conseguir un funcionamiento seguro de la sincronización, cada punto de reunión (parámetro `SyncID`) debe tener un nombre exclusivo. El nombre del punto de reunión también debe ser el mismo en todas las tareas de programa que deban coincidir en el punto de reunión.

---

#### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción `SyncMoveOn`:

Consulte también [Más ejemplos en la página 856](#).

#### Ejemplo 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;
...
```

*Continúa en la página siguiente*

## 1.285 SyncMoveOn - Inicia los movimientos sincronizados coordinados *RW-MRS Synchronized* *Continuación*

La tarea de programa que llegue en primer lugar a `SyncMoveOn` con la identidad `sync1` espera hasta que la otras tareas lleguen a su `SyncMoveOn` con la misma identidad `sync1`. En ese punto de sincronización `sync1`, el planificador de movimientos de las tareas de programa implicadas cambia al modo sincronizado. A continuación, tanto la tarea `T_ROB1` como la tarea `T_ROB2` prosiguen su ejecución de forma sincronizada hasta que alcanzan la instrucción `SyncMoveOff` con la misma identidad `sync2`.

### Argumentos

```
SyncMoveOn SyncID TaskList [\TimeOut]
```

SyncID

#### *Synchronization Identity*

Tipo de dato: `syncident`

Una variable que especifica el nombre del punto de sincronización (reunión). El tipo de dato `syncident` es de un tipo sin valor y sólo se utiliza como un identificador para asignar un nombre a la posición de sincronización.

La variable debe ser definida y tener un nombre igual en todas las tareas de programa cooperantes. Se recomienda definir siempre la variable global en cada tarea (`VAR syncident ...`).

TaskList

Tipo de dato: `tasks`

Una variable persistente que especifica en una lista de tareas (matriz) el nombre (`string`) de las tareas de programa que deben coincidir en el punto de sincronización cuyo nombre se especifica en el argumento `SyncID`.

La variable persistente debe ser definida y tener un nombre igual y el mismo contenido en todas las tareas de programa cooperantes. Se recomienda definir siempre la variable global en el sistema (`PERS tasks ...`).

[\TimeOut]

Tipo de dato: `num`

El tiempo máximo que debe esperarse hasta que las demás tareas de programa lleguen hasta el punto de sincronización. El tiempo límite se define en segundos (resolución 0,001 s).

Si el tiempo se agota antes de que todas las tareas de programa alcancen el punto de sincronización, se llama al gestor de errores si lo hay, con el código de error `ERR_SYNCMOVEON`. Si no hay ningún gestor de errores, se detiene la ejecución.

Si se omite este argumento, la tarea de programa esperará de forma indefinida.

### Ejecución de programas

La tarea de programa que llegue en primer lugar a `SyncMoveOn` espera hasta que todas las demás tareas especificadas alcancen sus instrucciones `SyncMoveOn` con la misma identidad `SyncID`. En ese punto de sincronización `SyncID`, el planificador de movimientos de las tareas de programa implicadas cambia al modo sincronizado. A continuación, las tareas de programa implicadas prosiguen su ejecución.

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.285 SyncMoveOn - Inicia los movimientos sincronizados coordinados

### *RW-MRS Synchronized*

#### *Continuación*

Los planificadores de movimientos de las tareas de programa implicadas cambian al modo sincronizado. Esto significa lo siguiente:

- Cada instrucción de movimiento de cualquier tarea de programa de `TaskList` funciona de forma sincronizada con instrucciones de movimiento en otras tareas de programa de `TaskList`.
- Todas las instrucciones de movimiento cooperativas se planifican e interpolan con el mismo planificador de movimientos.
- Todos los movimientos comienzan y terminan al mismo tiempo. El movimiento que requiera el tiempo más largo será el que controle la velocidad, con una velocidad reducida respecto del objeto de trabajo de los demás movimientos.
- Todas las instrucciones de movimiento cooperativas deben estar marcadas con el mismo número de ID. Consulte la instrucción `MoveL`.

Es posible excluir las tareas de programa de las funciones de pruebas del panel de selección de tareas del FlexPendant. La instrucción `SyncMoveOn` seguirá funcionando con el número reducido de tareas de programa, incluso si sólo es una tarea de programa.

---

### Más ejemplos

A continuación aparecen más ejemplos de cómo usar la instrucción `SyncMoveOn`.

#### Ejemplo 1

```
!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;
  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;
```

*Continúa en la página siguiente*

## 1.285 SyncMoveOn - Inicia los movimientos sincronizados coordinados

*RW-MRS Synchronized**Continuación*

```

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

```

En primer lugar, las tareas de programa T\_ROB1 y T\_ROB2 se esperan la una a la otra en WaitSyncTask con la identidad sync1. Se programan con la trayectoria de esquina para los movimientos precedentes con el fin de ahorrar tiempo de ciclo.

A continuación, las tareas de programa se esperan la una a la otra en SyncMoveOn con la identidad sync2. Se programan con un punto de paro necesario para los movimientos precedentes. Después, el planificador de movimientos de las tareas de programa implicadas cambia al modo sincronizado.

A continuación, T\_ROB2 mueve obj2 a ID punto 10 y 20 en el sistema de coordenadas mundo mientras que T\_ROB1 mueve tcp1 a ID punto 10 y 20 en el objeto en movimiento obj2.

**Ejemplo 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

```

La tarea de programa espera en la instrucción SyncMoveOn hasta que la tarea de programa T\_ROB2 alcance el mismo punto de sincronización sync3. Después de esperar en 60 segundos, se llama al gestor de errores con ERRNO cambiado al valor ERR\_SYNCMOVEON. A continuación, se llama de nuevo a la instrucción SyncMoveOn para una espera adicional en 60 segundos.

**Ejemplo 3: Ejemplo de programa con tres tareas**

```

!Program example in task T_ROB1
PERS tasks task_list1 {2} := [{"T_ROB1"}, {"T_ROB2"}];

```

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.285 SyncMoveOn - Inicia los movimientos sincronizados coordinados

### *RW-MRS Synchronized*

#### *Continuación*

```
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;
...
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;
...
```

En este ejemplo, inicialmente las tareas de programa T\_ROB1 y T\_ROB2 tienen sincronizados los movimientos y T\_ROB3 se mueve independientemente. Más adelante en el programa, las tres tareas se mueven de forma sincronizada. Para impedir que la instrucción de SyncMoveOn se ejecute en T\_ROB3 antes de que la

*Continúa en la página siguiente*

## 1.285 SyncMoveOn - Inicia los movimientos sincronizados coordinados

*RW-MRS Synchronized*

*Continuación*

primera sincronización de T\_ROB1 y T\_ROB2 haya terminado, se utiliza la instrucción WaitSyncTask.

### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema ERRNO cambiará a:

| Nombre         | Causa del error              |
|----------------|------------------------------|
| ERR_SYNCMOVEON | Tiempo límite en SyncMoveOn. |

### Limitaciones

La instrucción SyncMoveOn sólo puede ejecutarse si todos los robots implicados están detenidos en un punto de paro.

Sólo un grupo de movimientos sincronizados coordinados puede estar activo cada vez.

Si esta instrucción va precedida de una instrucción de movimiento, ésta última debe programarse con un punto de paro (zonedata fine), no un punto de paso. De lo contrario, no será posible reanudar la ejecución tras una caída de suministro eléctrico.

SyncMoveOn no puede ejecutarse en rutinas de RAPID que estén conectadas a los siguientes eventos especiales del sistema: PowerOn, Stop, QStop, Restart, Reset o Step.

### Sintaxis

```
SyncMoveOn
[ SyncID ':= ' ] < variable (VAR) of syncident > ','
[ TaskList ':= ' ] < persistent array {*} (PERS) of tasks > ','
[ '\ ' TimeOut ':= ' < expression (IN) of num > ] ';'

```

### Información relacionada

| Para obtener más información sobre                    | Consulte  |
|---|---|
| Especificación de tareas de programa cooperativas     | <a href="#">tasks - Tareas de programa RAPID en la página 1791</a>  |
| Identidad para punto de sincronización                | <a href="#">syncident - Identidad de punto de sincronización en la página 1786</a>                                  |
| Fin de movimientos sincronizados coordinados          | <a href="#">SyncMoveOff - Finaliza los movimientos sincronizados coordinados en la página 848</a>                   |
| Definición de movimientos independientes              | <a href="#">SyncMoveUndo - Activa los movimientos independientes en la página 864</a>                               |
| Comprobación de si está activado el modo sincronizado | <a href="#">IsSyncMoveOn - Comprueba si el modo de movimiento sincronizado está activado en la página 1364</a>      |
| Sistema MultiMove con la opción Coordinated Robots    | <a href="#">Manual de aplicaciones - MultiMove</a>  |
| Espera a tareas sincronizadas                         | <a href="#">WaitSyncTask - Esperar en un punto de sincronización con otras tareas de programa en la página 1075</a> |

## 1 Instrucciones

---

### 1.286 SyncMoveResume - Activa el modo de movimientos sincronizados coordinados *RW-MRS Synchronized*

### 1.286 SyncMoveResume - Activa el modo de movimientos sincronizados coordinados

---

#### Utilización

`SyncMoveResume` se utiliza para volver al modo de movimiento sincronizado desde el modo de movimiento independiente. Esta instrucción sólo puede usarse en el nivel `StorePath`, por ejemplo, una vez que se ha ejecutado una instrucción `StorePath \KeepSync` y el sistema se encuentra en el modo de movimiento independiente una vez ejecutada la instrucción `SyncMoveSuspend`. Para poder utilizar la instrucción, el sistema debe haber estado en el modo de movimiento sincronizado antes de ejecutar las instrucciones `StorePath` y la instrucción `SyncMoveSuspend`.

La instrucción `SyncMoveResume` sólo puede usarse en un sistema *MultiMove* que tenga las opciones *Coordinated Robots* y *Path Recovery* y sólo en las tareas de programa definidas como *tarea de movimiento*.

#### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción `SyncMoveResume`:

#### Ejemplo 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:=112, v50, fine, tool2;
  RestoPath;
  StartMove;
  RETRY;
```

Se produce algún tipo de error recuperable. El sistema permanece en el modo sincronizado y se realiza un movimiento sincronizado a un punto, por ejemplo retrocediendo por la trayectoria. A continuación, se realiza un movimiento independiente hacia una estación de limpieza. Seguidamente, el robot retrocede hasta el punto en el que se produjo el error y el programa continúa en el punto en el que fue interrumpido por el error.

#### Ejecución de programas

`SyncMoveResume` fuerza la reanudación del modo sincronizado cuando el sistema se encuentra en el modo de movimiento independiente en el nivel `StorePath`.

*Continúa en la página siguiente*



## 1.286 SyncMoveResume - Activa el modo de movimientos sincronizados coordinados RW-MRS Synchronized Continuación

La instrucción `SyncMoveResume` es necesaria en todas las tareas que se estén ejecutando en el modo de movimiento sincronizado antes de activar el modo de movimiento independiente. Si una tarea de movimiento ejecuta una instrucción `SyncMoveResume`, dicha tarea esperará hasta que también todas las tareas que estuvieran anteriormente en el modo de movimiento sincronizado ejecuten una instrucción `SyncMoveResume`. A continuación, las tareas de programa implicadas prosiguen su ejecución.

### Limitaciones

La instrucción `SyncMoveResume` sólo puede usarse para volver al modo de movimiento sincronizado y sólo puede usarse en el nivel `StorePath`.

Si esta instrucción va precedida de una instrucción de movimiento, ésta última debe programarse con un punto de paro (zonedata `fine`), no un punto de paso. De lo contrario, no será posible reanudar la ejecución tras una caída de suministro eléctrico.

`SyncMoveResume` no puede ejecutarse en rutinas de RAPID que estén conectadas a los siguientes eventos especiales del sistema: `PowerOn`, `Stop`, `QStop`, `Restart`, `Reset` o `Step`.

### Sintaxis

```
SyncMoveResume ' ; '
```

### Información relacionada

| Para obtener más información sobre                    | Consulte  |
|---|---|
| Especificación de tareas de programa cooperativas     | <a href="#">tasks - Tareas de programa RAPID en la página 1791</a>                                      |
| Inicio de movimientos sincronizados coordinados       | <a href="#">SyncMoveOn - Inicia los movimientos sincronizados coordinados en la página 854</a>          |
| Fin de movimientos sincronizados coordinados          | <a href="#">SyncMoveOff - Finaliza los movimientos sincronizados coordinados en la página 848</a>       |
| Comprobación de si está activado el modo sincronizado | <a href="#">SyncMoveOn - Inicia los movimientos sincronizados coordinados en la página 854</a>          |
| Almacenamiento de la trayectoria                      | <a href="#">StorePath - Almacena la trayectoria cuando se produce una interrupción en la página 835</a> |
| Restauración de la trayectoria                        | <a href="#">RestoPath - Restablece la trayectoria después de una interrupción en la página 615</a>      |
| Suspensión del movimiento sincronizado                | <a href="#">SyncMoveSuspend - Activa el movimiento independiente-semicoordinado en la página 862</a>    |

## 1 Instrucciones

---

### 1.287 SyncMoveSuspend - Activa el movimiento independiente-semicoordinado *RW-MRS Synchronized*

### 1.287 SyncMoveSuspend - Activa el movimiento independiente-semicoordinado

---

#### Utilización

`SyncMoveSuspend` se usa para suspender el modo de movimiento sincronizado y poner el sistema en el modo de movimiento independiente-semicoordinado. Esta instrucción sólo puede usarse en el nivel `StorePath`, por ejemplo, una vez que se ha ejecutado una instrucción `StorePath` o `StorePath \KeepSync` y el sistema se encuentra en el modo de movimiento sincronizado.

La instrucción `SyncMoveSuspend` sólo puede usarse en un *MultiMove System* que disponga de las opciones *Coordinated Robots* y *Path Recovery* y sólo en tareas de programa definidas como *tarea de movimiento*.

#### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción `SyncMoveSuspend`:

##### Ejemplo 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:=112, v50, fine, tool2;
  RestoPath;
  StartMove;
  RETRY;
```

Se produce algún tipo de error recuperable. El sistema permanece en el modo sincronizado y se realiza un movimiento sincronizado a un punto, por ejemplo retrocediendo por la trayectoria. A continuación, se realiza un movimiento independiente hacia una estación de limpieza. Seguidamente, el robot retrocede hasta el punto en el que se produjo el error y el programa continúa en el punto en el que fue interrumpido por el error.

#### Ejecución de programas

`SyncMoveSuspend` fuerza el restablecimiento de los movimientos sincronizados y pone el sistema en el modo de movimiento independiente-semicoordinado.

La instrucción `SyncMoveSuspend` es necesaria en todas las tareas de movimiento sincronizado, para poner el sistema en el modo de movimiento independiente-semicoordinado. Si una tarea de movimiento ejecuta una instrucción

---

*Continúa en la página siguiente*

## 1.287 SyncMoveSuspend - Activa el movimiento independiente-semicoordinado RW-MRS Synchronized Continuación

SyncMoveSuspend, la tarea espera hasta que las demás tareas hayan ejecutado una instrucción SyncMoveSuspend.

Tras la ejecución de SyncMoveSuspend en todas las tareas implicadas, el sistema se encuentra en el modo semicoordinado si se sigue usando el objeto de trabajo coordinado. De lo contrario, se encuentra en el modo independiente. En el modo semicoordinado, se recomienda empezar siempre con un movimiento en la unidad mecánica que controla la base de coordenadas del usuario antes de usar WaitSyncTask en todas las tareas implicadas.

### Limitaciones

La instrucción SyncMoveSuspend suspende el modo sincronizado sólo en el nivel StorePath. Después de volver del nivel StorePath, el sistema cambia al modo en el que se encontraba antes de StorePath.

Si esta instrucción va precedida de una instrucción de movimiento, ésta última debe programarse con un punto de paro (zonedata fine), no un punto de paso. De lo contrario, no será posible reanudar la ejecución tras una caída de suministro eléctrico.

SyncMoveSuspend no puede ejecutarse en rutinas de RAPID que estén conectadas a los siguientes eventos especiales del sistema: PowerOn, Stop, QStop, Restart, Reset o Step.

### Sintaxis

```
SyncMoveSuspend ' ; '
```

### Información relacionada

| Para obtener más información sobre                    | Consulte   |
|---|--|
| Especificación de tareas de programa cooperativas     | <a href="#">tasks - Tareas de programa RAPID en la página 1791</a>   |
| Inicio de movimientos sincronizados coordinados       | <a href="#">SyncMoveOn - Inicia los movimientos sincronizados coordinados en la página 854</a>                 |
| Fin de movimientos sincronizados coordinados          | <a href="#">SyncMoveOff - Finaliza los movimientos sincronizados coordinados en la página 848</a>              |
| Comprobación de si está activado el modo sincronizado | <a href="#">IsSyncMoveOn - Comprueba si el modo de movimiento sincronizado está activado en la página 1364</a> |
| Almacenamiento de la trayectoria                      | <a href="#">StorePath - Almacena la trayectoria cuando se produce una interrupción en la página 835</a>        |
| Restauración de la trayectoria                        | <a href="#">RestoPath - Restablece la trayectoria después de una interrupción en la página 615</a>             |
| Reanudación del movimiento sincronizado               | <a href="#">SyncMoveResume - Activa el modo de movimientos sincronizados coordinados en la página 860</a>      |

# 1 Instrucciones

---

## 1.288 SyncMoveUndo - Activa los movimientos independientes

RobotWare - OS

## 1.288 SyncMoveUndo - Activa los movimientos independientes

---

### Utilización

`SyncMoveUndo` se utiliza para forzar el restablecimiento de los movimientos sincronizados coordinados y pone el sistema en el modo de movimientos independientes.

La instrucción `SyncMoveUndo` sólo puede usarse en un sistema *MultiMove* que tenga la opción *Coordinated Robots* y sólo en las tareas de programa definidas como `Motion Task`.

---

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción `SyncMoveUndo`:

#### Ejemplo 1

##### Ejemplo de programa de la tarea 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;
MoveL * \ID:=10, v100, z10, tcp1 \WOBJ:= rob2_obj;
MoveL * \ID:=20, v100, fine, tcp1 \WOBJ:= rob2_obj;
SyncMoveOff sync3;
UNDO
SyncMoveUndo;
ENDPROC
```

Si el programa se detiene mientras la ejecución se encuentra dentro del procedimiento `syncmove` y el puntero de programa se mueve hacia el exterior del procedimiento `syncmove`, se ejecutan todas las instrucciones que se encuentren dentro del gestor de deshacer `UNDO`. En este ejemplo se ejecuta la instrucción `SyncMoveUndo` y el sistema cambia al modo de movimientos independientes.

---

### Ejecución de programas

Se fuerza el restablecimiento de los movimientos sincronizados coordinados y pone el sistema en el modo de movimientos independientes.

Basta con ejecutar `SyncMoveUndo` en una tarea de programa para cambiar todo el sistema al modo de movimientos independientes. La instrucción puede ejecutarse

*Continúa en la página siguiente*

varias veces sin que se produzca ningún error si el sistema ya se encuentra en el modo de movimientos independientes.

El sistema cambia también al modo de movimientos independientes en las situaciones siguientes:

- cuando se utiliza el modo de reinicio **Restablecer RAPID**
- al cargar un nuevo programa o un nuevo módulo
- al iniciar la ejecución del programa desde el principio
- al mover el puntero del programa a `main`
- al mover el puntero del programa a una rutina
- al mover el puntero de programa de una forma que se pierde el orden de la ejecución.

### Sintaxis

```
SyncMoveUndo ' ; '
```

### Información relacionada

| Para obtener más información sobre                    | Consulte   |
|---|--|
| Especificación de tareas de programa cooperativas     | <a href="#">tasks - Tareas de programa RAPID en la página 1791</a>   |
| Identidad para punto de sincronización                | <a href="#">syncident - Identidad de punto de sincronización en la página 1786</a>                             |
| Inicio de movimientos sincronizados coordinados       | <a href="#">SyncMoveOn - Inicia los movimientos sincronizados coordinados en la página 854</a>                 |
| Fin de movimientos sincronizados coordinados          | <a href="#">SyncMoveOff - Finaliza los movimientos sincronizados coordinados en la página 848</a>              |
| Comprobación de si está activado el modo sincronizado | <a href="#">IsSyncMoveOn - Comprueba si el modo de movimiento sincronizado está activado en la página 1364</a> |

# 1 Instrucciones

---

## 1.289 SyncToSensor - Sincronización con un sensor *Machine Synchronization*

### 1.289 SyncToSensor - Sincronización con un sensor

---

#### Utilización

SyncToSensor se utiliza para iniciar o detener la sincronización de los movimientos del robot con el movimiento de los sensores.

---

#### Ejemplos básicos

A continuación aparecen algunos ejemplos básicos de la instrucción SyncToSensor.

#### Ejemplo 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;
```

---

#### Argumentos

SyncToSensor MechUnit [*\MaxSync*] [*\On*] | [*\Off*]

MechUnit

#### *Mechanical Unit*

Tipo de dato: mecunit

La unidad mecánica en movimiento con la que está relacionada la posición de robot de la instrucción.

[*\MaxSync*]

Tipo de dato: num

El robot se mueve sincronizado con el sensor hasta que este rebasa la posición *MaxSync*. A continuación, el robot se moverá de forma no sincronizada a la velocidad programada. Si no se define el parámetro opcional *MaxSync*, el robot se mueve de forma sincronizada hasta que se ejecuta la instrucción `SyncToSensor Ssync1\Off`.

[*\On*]

Tipo de dato: switch

El robot se mueve de forma sincronizada con el sensor después de una instrucción que utiliza el argumento `\On`.

[*\Off*]

Tipo de dato: switch

El robot se mueve de forma no sincronizada con el sensor después de una instrucción que utiliza el argumento `\Off`.

*Continúa en la página siguiente*

#### Ejecución de programas

`SyncToSensor Ssync1 \On` significa que el robot comienza a moverse de forma sincronizada con el sensor `Ssync1`. Por tanto, el robot pasa junto al `robtarget` programado al mismo tiempo que el sensor rebasa la posición externa almacenada en el `robtarget`.

`SyncToSensor Ssync1 \Off` significa que el robot deja de moverse de forma sincronizada con el sensor.

#### Limitaciones

Si se ejecuta la instrucción `SyncToSensor Ssync1 \On` mientras el sensor no haya sido conectado con `WaitSensor`, el robot se detendrá.

#### Sintaxis

```
SyncToSensor
  [ MechUnit ':' ] < variable (VAR) of mecunit >
  [ \MaxSync ] [ '\ On' | [ '\ Off' ] ';' ;'
```

#### Información relacionada

| Para obtener más información sobre              | Consulte   |
|---|--|
| Inicio de la supervisión de sensor sincronizada | <a href="#">SupSyncSensorOn - Inicio de la supervisión de sensor sincronizada en la página 846</a> |
| Sincronización con un sensor                    | <a href="#">SyncToSensor - Sincronización con un sensor en la página 866</a>                       |
| Espera a la conexión de un sensor               | <a href="#">WaitSensor - Espera a la conexión de un sensor en la página 1071</a>                   |
| Colocación de un objeto en el sensor            | <a href="#">DropSensor - Colocación de un objeto en el sensor en la página 191</a>                 |
| <i>Machine Synchronization</i>                  | <i>Application manual - Controller software IRC5</i>   |

# 1 Instrucciones

---

## 1.290 SystemStopAction - Para el sistema de robot RobotWare - OS

### 1.290 SystemStopAction - Para el sistema de robot

---

#### Utilización

`SystemStopAction` puede usarse para parar el sistema de robot de distintas formas en función de la gravedad del error o el problema.

---

#### Ejemplos básicos

Los ejemplos siguientes ilustran la instrucción `SystemStopAction`.

##### Ejemplo 1

```
SystemStopAction \Stop;
```

Esto hace que se detenga la ejecución del programa y los movimientos del robot en todas las tareas de movimiento. No se requiere ninguna acción específica antes del reinicio de la ejecución del programa.

##### Ejemplo 2

```
SystemStopAction \StopBlock;
```

Esto hace que se detenga la ejecución del programa y los movimientos del robot en todas las tareas de movimiento. Todos los punteros de programa deben ser movidos antes de que sea posible reanudar la ejecución del programa.

##### Ejemplo 3

```
SystemStopAction \Halt;
```

Esto provoca el estado **Motors OFF**, la detención de la ejecución del programa y los movimientos del robot en todas las tareas de movimiento. Se debe cambiar a **Motors ON** antes de que sea posible reanudar la ejecución del programa.

---

#### Argumentos

```
SystemStopAction [\Stop] [\StopBlock] [\Halt]
```

`[\Stop]`

**Tipo de dato:** switch

`\Stop` se usa para parar la ejecución del programa y los movimientos del robot en todas las tareas de movimiento. No se requiere ninguna acción específica antes del reinicio de la ejecución del programa.

`[\StopBlock]`

**Tipo de dato:** switch

`\StopBlock` se usa para parar la ejecución del programa y los movimientos del robot en todas las tareas de movimiento. Todos los punteros de programa deben ser movidos antes de que sea posible reanudar la ejecución del programa.

`[\Halt]`

**Tipo de dato:** switch

`\Halt` de lugar al estado **Motors OFF**, la detención de la ejecución del programa y los movimientos del robot en todas las tareas de movimiento. Se debe cambiar a **Motors ON** antes de que sea posible reanudar la ejecución del programa.

*Continúa en la página siguiente*



#### Limitaciones

Si el robot realiza un movimiento circular durante el `SystemStopAction` `\StopBlock`, el puntero de programa y el robot deben moverse al principio del movimiento circular antes de reanudar la ejecución del programa.

#### Ejecución de programas

`SystemStopAction` se usa para parar el sistema de robot de distintas formas en función de la gravedad del error o el problema. La ejecución del programa se detiene en la tarea ejecutada si esta es una tarea normal.

Si el `SystemStopAction` se encuentra en una tarea estática o semiestática, la ejecución del programa se detendrá para todas las tareas normales, pero continuará para esa tarea. Para obtener más información sobre la declaración de tareas, consulte la documentación sobre los parámetros del sistema.

#### Sintaxis

```
SystemStopAction
  [ '\Stop ]
  | [ '\StopBlock ]
  | [ '\Halt ]';'
```

#### Información relacionada

| Para obtener más información sobre            | Consulte   |
|---|--|
| Detiene la ejecución del programa             | <a href="#">Stop - Detención de la ejecución del programa en la página 822</a> |
| Finaliza la ejecución del programa            | <a href="#">EXIT - Finaliza la ejecución del programa en la página 212</a>     |
| Paro de los movimientos del robot únicamente. | <a href="#">StopMove - Detiene el movimiento del robot en la página 828</a>    |
| Escribir un mensaje de error                  | <a href="#">ErrLog - Escribe un mensaje de error en la página 201</a>          |

# 1 Instrucciones

---

## 1.291 TEST - En función del valor de una expresión...

RobotWare - OS

## 1.291 TEST - En función del valor de una expresión...

---

### Utilización

TEST se utiliza cuando es necesario ejecutar instrucciones diferentes en función del valor de una expresión o un dato.

Si no hay demasiadas alternativas, también es posible usar la instrucción `IF..ELSE`.

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción TEST:

#### Ejemplo 1

```
TEST reg1
CASE 1,2,3 :
  routine1;
CASE 4 :
  routine2;
DEFAULT :
  TPWrite "Illegal choice";
  Stop;
ENDTEST
```

Se ejecutan instrucciones diferentes en función del valor de `reg1`. Si el valor es 1, 2 ó 3, se ejecuta `routine1`. Si el valor es 4, se ejecuta `routine2`. De lo contrario, se envía un mensaje de error a la unidad de programación y la ejecución se detiene.

### Argumentos

```
TEST Test data {CASE Test value {, Test value} : ...} [ DEFAULT:
... ] ENDTEST
```

Test data

Tipo de dato: All

El dato o la expresión con el que se comparará el valor de prueba.

Test value

Tipo de dato: Del mismo tipo que `test data`

El valor que debe tener el dato de prueba para que se ejecuten las instrucciones asociadas.

### Ejecución de programas

Los datos de prueba se comparan con los valores de prueba de la primera condición CASE. Si la comparación da un resultado positivo, se ejecutan las instrucciones asociadas. Posteriormente, la ejecución del programa continúa con la instrucción que aparece a continuación de `ENDTEST`.

Si no se cumple la primera condición CASE, se comprueban las demás condiciones CASE y de ahí en adelante. Si no se cumple ninguna de las condiciones, se ejecutan las instrucciones asociadas a `DEFAULT` (si se ha incluido en la instrucción).

*Continúa en la página siguiente*

### Sintaxis

```
TEST <expression>
{ CASE <test value> { ',' <test value> } ':'
  <statement list> }
[ DEFAULT ':'
  <statement list> ]
ENDTEST
```

### Información relacionada

| Para obtener más información sobre | Consulte   |
|------------------------------------|--|
| Expresiones                        | <i>Manual de referencia técnica - Instrucciones, funciones y tipos de datos de RAPID</i> |

# 1 Instrucciones

---

1.292 TestSignDefine - Define una señal de prueba  
RobotWare - OS

## 1.292 TestSignDefine - Define una señal de prueba

---

### Utilización

`TestSignDefine` se utiliza para definir una señal de prueba para el sistema de movimiento del robot.

Una señal de prueba refleja continuamente un flujo de datos de movimiento en concreto. Por ejemplo, referencia de par para algún eje especificado. El valor actual en un momento dado puede leerse desde RAPID con la función `TestSignRead`.

Sólo es posible obtener información de las señales de prueba de los ejes externos. Las señales de prueba también están disponibles bajo pedido para los ejes de robot y para las señales de prueba no predefinidas para los ejes externos.

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción `TestSignDefine`:

#### Ejemplo 1

```
TestSignDefine 1, resolver_angle, Orbit, 2, 0.1;
```

La señal de prueba `resolver_angle` conectada al canal 1 proporciona el valor del ángulo del resolver para el eje 2 del manipulador `orbit`, muestreado cada 100 ms.

### Argumentos

```
TestSignDefine Channel SignalId MechUnit Axis SampleTime
```

Channel

Tipo de dato: num

Los números de canal, de 1 a 12, que se desea usar para la señal de prueba. Se debe usar el mismo número en la función `TestSignRead` para leer el valor actual de la señal de prueba.

SignalId

Tipo de dato: testsignal

El nombre o el número de la señal de prueba. Consulte las constantes predefinidas descritas en el tipo de dato `testsignal`.

MechUnit

*Mechanical Unit*

Tipo de dato: mecunit

El nombre de la unidad mecánica.

Axis

Tipo de dato: num

El número de eje de la unidad mecánica.

SampleTime

Tipo de dato: num

El tiempo de muestreo en segundos.

*Continúa en la página siguiente*

Con un tiempo de muestreo  $< 0,004$  s, la función `TestSignRead` devuelve el valor medio de los últimos muestreos internos disponibles, como se muestra en la tabla siguiente.

| Tiempo de muestreo en segundos | Resultado de TestSignRead  |
|--------------------------------|--|
| 0                              | Valor medio de los últimos 8 muestreos generados cada 0,5 ms               |
| 0.001                          | Valor medio de los últimos 4 muestreos generados cada 1 ms                 |
| 0.002                          | Valor medio de los últimos 2 muestreos generados cada 2 ms                 |
| Mayor o igual a 0,004          | Valor momentáneo generado en el tiempo de muestreo especificado            |
| 0.1                            | Valor momentáneo generado en el tiempo de muestreo especificado, de 100 ms |

### Ejecución de programas

La definición de la señal de prueba se activa y el sistema de robot empieza a muestrear la señal de prueba.

El muestreo de la señal de prueba permanece activo en los casos siguientes:

- Hasta que se ejecuta una nueva instrucción `TestSignDefine` para el canal actual.
- Hasta que se desactivan todas las señales de prueba con la ejecución de la instrucción `TestSignReset`.
- Todas las señales se desactivan con el Reinicio del sistema.

### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                    | Causa del error   |
|---------------------------|---|
| <code>ERR_AXIS_PAR</code> | Existe un error en el parámetro <code>Axis</code> .     |
| <code>ERR_UNIT_PAR</code> | Existe un error en el parámetro <code>MechUnit</code> . |

### Sintaxis

```
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 > ' ; '
```

### Información relacionada

| Para obtener más información sobre | Consulte  |
|------------------------------------|---|
| Señal de prueba                    | <a href="#">testsignal - Señal de prueba en la página 1793</a>  |
| Lectura de una señal de prueba     | <a href="#">TestSignRead - Obtiene el valor de una señal de test en la página 1528</a>                  |
| Puesta a cero de señales de prueba | <a href="#">TestSignReset - Restablece todas las definiciones de señales de prueba en la página 874</a> |

# 1 Instrucciones

---

1.293 TestSignReset - Restablece todas las definiciones de señales de prueba  
RobotWare - OS

## 1.293 TestSignReset - Restablece todas las definiciones de señales de prueba

---

### Utilización

`TestSignReset` se utiliza para desactivar todas las señales de prueba definidas anteriormente.

---

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción `TestSignReset`:

#### Ejemplo 1

```
TestSignReset ;
```

Desactiva todas las señales de prueba definidas anteriormente.

---

### Ejecución de programas

Las definiciones de todas las señales de prueba se desactivan y el sistema de robot detiene el muestreo de las señales de prueba.

El muestreo de las señales de prueba definidas permanece activo hasta:

- Un Reinicio del sistema
  - La ejecución de esta instrucción, `TestSignReset`
- 

### Sintaxis

```
TestSignReset ;'
```

---

### Información relacionada

| Para obtener más información sobre | Consulte   |
|------------------------------------|--|
| Señal de prueba                    | <a href="#">testsignal - Señal de prueba en la página 1793</a>                         |
| Definición de una señal de test    | <a href="#">TestSignDefine - Define una señal de prueba en la página 872</a>           |
| Lectura de una señal de prueba     | <a href="#">TestSignRead - Obtiene el valor de una señal de test en la página 1528</a> |

### 1.294 TextTabInstall - Instalación de una tabla de textos

#### Utilización

`TextTabInstall` se usa para instalar una tabla de textos en el sistema.

#### Ejemplos básicos

El ejemplo siguiente ilustra la instrucción `TextTabInstall`.

#### Ejemplo 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.xml";  
  ENDIF  
ENDPROC
```

La primera vez que se ejecuta la rutina de evento `install_text`, la función `TextTabFreeToUse` devuelve `TRUE` y el archivo de texto `text_file.xml` se instala en el sistema. A continuación, es posible obtener con `RAPID` las cadenas de texto instaladas en el sistema, con la ayuda de las funciones `TextTabGet` y `TextGet`.

La próxima vez que se ejecuta la rutina de evento `install_text`, la función `TextTabFreeToUse` devuelve `FALSE` y la instalación no se repite.

#### Argumentos

```
TextTabInstall File
```

File

Tipo de dato: `string`

La ruta y el nombre del archivo que contiene las cadenas de texto que se desea instalar en el sistema.

#### Limitaciones

Limitaciones de la instalación de tablas de textos (recursos de texto) en el sistema:

- No es posible instalar una misma tabla de textos más de una vez en el sistema.
- No es posible desinstalar (liberar) una sola tabla de textos del sistema. La única forma de desinstalar tablas de textos del sistema es reiniciar el controlador usando el modo de reinicio **Restablecer sistema**. De esta forma, se desinstalan todas las tablas de textos (tanto las del sistema como las definidas por el usuario).

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.294 TextTabInstall - Instalación de una tabla de textos

RobotWare - OS

Continuación

### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                    | Causa del error  |
|---------------------------|--|
| <code>ERR_FILEOPEN</code> | El archivo en la instrucción <code>TextTabInstall</code> no puede abrirse. |

### Sintaxis

```
TextTabInstall  
[File ':='] <expression (IN) of string>';'
```

### Información relacionada

| Para obtener más información sobre                      | Consulte   |
|---|--|
| Comprobación de si una tabla de textos está libre       | <a href="#">TextTabFreeToUse - Comprueba si una tabla de textos está libre para su uso en la página 1533</a> |
| Formato de archivos de texto                            | <i>Technical reference manual - RAPID kernel</i>   |
| Obtención de un número de tabla de textos               | <a href="#">TextTabGet - Obtiene el número de una tabla de textos en la página 1535</a>                      |
| Obtención de textos de las tablas de textos del sistema | <a href="#">TextGet - Obtener un texto de las tablas de textos del sistema en la página 1530</a>             |
| Funciones para cadenas de caracteres                    | <i>Manual de referencia técnica - RAPID Overview</i>   |
| Definición de cadena de caracteres                      | <a href="#">string - Cadenas en la página 1778</a>   |
| <i>Advanced RAPID</i>                                   | <i>Application manual - Controller software IRC5</i>   |



### 1.295 TPErase - Borra el texto mostrado en el FlexPendant

#### Utilización

TPErase (*FlexPendant Erase*) se utiliza para borrar la pantalla del FlexPendant.

#### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción TPErase:

##### Ejemplo 1

```
TPErase;  
TPWrite "Execution started";
```

Se borra la pantalla del FlexPendant antes de escribir en ella el mensaje `Execution started`.

#### Ejecución de programas

Se elimina todo el texto de la pantalla del FlexPendant. La próxima vez que se escribe un texto, éste aparece en la línea superior de la pantalla.

#### Sintaxis

```
TPErase;
```

#### Información relacionada

| Para obtener más información sobre | Consulte   |
|------------------------------------|--|
| Escritura en el FlexPendant        | <i>Manual de referencia técnica - RAPID Overview</i> |

# 1 Instrucciones

---

## 1.296 TPReadDnum - Lee un número del FlexPendant

RobotWare - OS

## 1.296 TPReadDnum - Lee un número del FlexPendant

---

### Utilización

TPReadDnum (*FlexPendant Read Numerical*) se usa para leer un número del FlexPendant.

---

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción TPReadDnum:

#### Ejemplo 1

```
VAR dnum value;
```

```
TPReadDnum value, "How many units should be produced?";
```

El texto `How many units should be produced?` se escribe en el FlexPendant. La ejecución del programa espera hasta que se introduzca un número a través del teclado numérico del FlexPendant. El número se almacena en `value`.

---

### Argumentos

```
TPReadDnum TPAnswer TPText [\MaxTime][\DIBreak] [\DIPassive]  
[\DOBreak] [\DOPassive] [\PersBoolBreak] [\PersBoolPassive]  
[\BreakFlag]
```

TPAnswer

Tipo de dato: `dnum`

La variable en la que se almacena el número introducido a través del FlexPendant.

TPText

Tipo de dato: `string`

El texto informativo que debe escribirse en el FlexPendant (con un máximo de 80 caracteres y 40 caracteres por fila).

[\MaxTime]

Tipo de dato: `num`

El periodo máximo que debe esperar el programa para continuar con la ejecución. Si no se introduce ningún número en ese periodo, el programa sigue ejecutándose en el gestor de errores, a no ser que se utilice el indicador `BreakFlag` (que se documenta a continuación). La constante `ERR_TP_MAXTIME` puede usarse para comprobar si ha transcurrido ya el tiempo máximo establecido.

[\DIBreak]

*Digital Input Break*

Tipo de dato: `signal`

La señal digital que puede interrumpir el diálogo con el operador. Si no se introduce ningún número cuando la señal cambia a 1 (o si ya tiene el valor 1), el programa sigue ejecutándose en el gestor de errores, a no ser que se utilice el indicador `BreakFlag` (que se documenta a continuación). La constante `ERR_TP_DIBREAK` puede usarse para comprobar si esto ha ocurrido.

*Continúa en la página siguiente*

[\`DIPassive`]

### *Digital Input Passive*

Tipo de dato: `switch`

Este modificador redefine el comportamiento predeterminado con el argumento opcional `DIBreak`. En lugar de reaccionar cuando la señal cambia a 1 (o ya tiene el valor 1), la instrucción debe continuar en el gestor de errores (si no se utiliza `BreakFlag`) cuando la señal `DIBreak` cambia a 0 (o ya tiene el valor 0). La constante `ERR_TP_DIBREAK` puede usarse para comprobar si esto ha ocurrido.

[\`DOBreak`]

### *Digital Output Break*

Tipo de dato: `signaldo`

La señal digital que soporta la petición de finalización desde otras tareas. Si no se selecciona ningún botón cuando la señal cambia a 1 (o si ya tiene el valor 1), el programa sigue ejecutándose en el gestor de errores, a no ser que se utilice el indicador `BreakFlag` (que se documenta a continuación). La constante `ERR_TP_DOBREAK` puede usarse para comprobar si esto ha ocurrido.

[\`DOPassive`]

### *Digital Output Passive*

Tipo de dato: `switch`

Este modificador redefine el comportamiento predeterminado con el argumento opcional `DOBreak`. En lugar de reaccionar cuando la señal cambia a 1 (o ya tiene el valor 1), la instrucción debe continuar en el gestor de errores (si no se utiliza `BreakFlag`) cuando la señal `DOBreak` cambia a 0 (o ya tiene el valor 0). La constante `ERR_TP_DOBREAK` puede usarse para comprobar si esto ha ocurrido.

[\`PersBoolBreak`]

### *Persistent Boolean Break*

Tipo de dato: `bool`

El booleano persistente que puede interrumpir el diálogo con el operador. Si no se selecciona ningún botón cuando el booleano persistente cambia a `TRUE` (o ya tiene el valor `TRUE`) entonces el programa continuará ejecutándose en el gestor de errores, a no ser que se utilice `BreakFlag` (que se documenta a continuación). La constante `ERR_TP_PERSBOOLBREAK` puede usarse para comprobar si esto ha ocurrido.

[\`PersBoolPassive`]

### *Persistent Boolean Passive*

Tipo de dato: `switch`

Este interruptor redefine el comportamiento con el argumento opcional `PersBoolBreak`. En lugar de reaccionar cuando el booleano persistente cambia a `TRUE` (o ya tiene el valor `TRUE`), la instrucción debe continuar en el gestor de errores (si no se utiliza `BreakFlag`) cuando el booleano persistente `PersBoolBreak` cambia a `FALSE` (o ya tiene el valor `FALSE`). La constante `ERR_TP_PERSBOOLBREAK` puede usarse para comprobar si esto ha ocurrido.

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.296 TPReadDnum - Lee un número del FlexPendant

RobotWare - OS

Continuación

[ \BreakFlag ]

Tipo de dato: errnum

Una variable que contiene el código de error si se utiliza `MaxTime`, `DIBreak`, `DOBREAK`, o `PersBoolBreak`. Si se omite esta variable opcional, se ejecuta el gestor de errores. Las constantes `ERR_TP_MAXTIME`, `ERR_TP_DIBREAK`, `ERR_TP_DOBREAK`, y `ERR_TP_PERSBOOLBREAK` pueden usarse para seleccionar el motivo.

### Ejecución de programas

El texto de información se escribe siempre en una nueva línea. Si la pantalla está llena de texto, el cuerpo de texto se mueve previamente una línea hacia arriba. Puede haber un máximo de 7 líneas por encima del nuevo texto escrito.

La ejecución del programa espera hasta que se escribe un número mediante el teclado numérico (seguido de Intro u OK) o hasta que la instrucción es interrumpida por un tiempo límite agotado o una acción de señal.

Consulte `TPReadFK` para obtener una descripción de la petición concurrente de `TPReadFK` o `TPReadDnum` en el FlexPendant desde la misma tarea de programa o desde tareas de programa diferentes.

### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                            | Causa del error  |
|-----------------------------------|--|
| <code>ERR_NO_ALIASIO_DEF</code>   | La variable de señal es una variable declarada en <code>RAPID</code> . No se ha conectado a una señal E/S definida en la configuración de E/S con la instrucción <code>AliasIO</code> .                            |
| <code>ERR_TP_MAXTIME</code>       | Tiempo límite (parámetro <code>\MaxTime</code> ) antes de que responda el operador.  |
| <code>ERR_TP_DIBREAK</code>       | Una instrucción de lectura de FlexPendant fue interrumpida por una entrada digital.<br>Se estableció una entrada digital (parámetro <code>\DIBreak</code> ) antes de que respondiera el operador.                  |
| <code>ERR_TP_DOBREAK</code>       | Una instrucción de lectura de FlexPendant fue interrumpida por una salida digital.<br>Se estableció una salida digital (parámetro <code>\DOBREAK</code> ) antes de que respondiera el operador.                    |
| <code>ERR_TP_NO_CLIENT</code>     | No hay ningún cliente con el que interactuar al utilizar una instrucción de lectura desde FlexPendant.   |
| <code>ERR_TP_PERSBOOLBREAK</code> | Una instrucción de lectura de FlexPendant fue interrumpida por un booleano persistente.<br>Un booleano persistente ha sido cambiado (parámetro <code>\PersBoolBreak</code> ) antes de que respondiera el operador. |

### Sintaxis

```
TPReadDnum
  [TPAnswer!:='] <var or pers (INOUT) of dnum>', '
  [TPText!:='] <expression (IN) of string>
  [ \MaxTime!:='] <expression (IN) of num>]
```

Continúa en la página siguiente

```
[ '\DIBreak' := <variable (VAR) of signaldi>]
[ '\DIPassive]
[ '\DOBreak' := <variable (VAR) of signaldo>]
[ '\DOPassive]
[ '\PersBoolBreak ' := <persistent (PERS) of bool>]
[ '\PersBoolPassive]
[ '\BreakFlag' := <var or pers (INOUT) of errnum>] ';'

```

### Información relacionada

| Para obtener más información sobre                                | Consulte   |
|---|--|
| Escritura y lectura a través del FlexPendant                      | <i>Manual de referencia técnica - RAPID Overview</i>                                 |
| Introducción de un número en el FlexPendant                       | <i>Manual del operador - IRC5 con FlexPendant</i>                                    |
| Ejemplos de cómo usar los argumentos MaxTime, DIBreak y BreakFlag | <a href="#">TPReadFK - Lee las teclas de función en la página 882</a>                |
| Borrado de la ventana de operador                                 | <a href="#">TPErase - Borra el texto mostrado en el FlexPendant en la página 877</a> |

# 1 Instrucciones

---

1.297 TPreadFK - Lee las teclas de función  
RobotWare - OS

## 1.297 TPreadFK - Lee las teclas de función

---

### Utilización

TPreadFK (*FlexPendant Read Function Key*) se utiliza para escribir un texto en las teclas de función y para determinar qué tecla de función se ha presionado.

---

### Ejemplos básicos

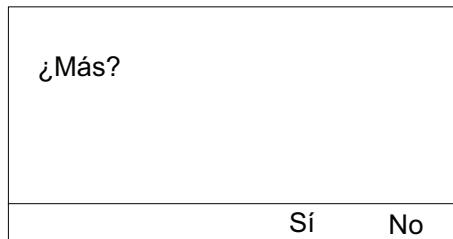
El ejemplo que aparece a continuación ilustra la instrucción TPreadFK:  
Consulte también [Más ejemplos en la página 885](#).

#### Ejemplo 1

```
TPreadFK reg1, "More?", stEmpty, stEmpty, stEmpty, "Yes", "No";
```

Se escribe el texto `More?` en la pantalla del FlexPendant y se activan las teclas de función 4 y 5 usando las cadenas de texto `Yes` y `No` respectivamente (consulte la figura siguiente). La ejecución del programa espera hasta que se presiona una de las teclas de función, la tecla 4 o la 5. En otras palabras, se asigna a `reg1` el valor 4 ó 5 en función de cuál de las teclas se presione.

En la figura se muestra cómo el operador puede introducir información a través de las teclas de función.



xx0500002345

---

### Argumentos

```
TPreadFK TPAnswer TPText TPFK1 TPFK2 TPFK3 TPFK4 TPFK5 [\MaxTime]  
[\DIBreak] [\DIPassive] [\DOBBreak] [\DOPassive]  
[\PersBoolBreak] [\PersBoolPassive] [\BreakFlag]
```

TPAnswer

**Tipo de dato:** num

La variable cuyo valor se devuelve (de 1 a 5) en función de qué tecla se presione. Si se presiona la tecla de función 1, se devuelve 1, etc.

TPText

**Tipo de dato:** string

El texto informativo que debe escribirse en la pantalla (con un máximo de 80 caracteres y 40 caracteres por fila).

TPFKx

**Function key text**

**Tipo de dato:** string

*Continúa en la página siguiente*

El texto que debe escribirse en tecla de función adecuada (con un máximo de 45 caracteres). `TPFK1` es la tecla que se encuentra en el extremo izquierdo.

Para especificar que una tecla de función no debe tener ningún texto, se utiliza la constante de cadena de caracteres predefinida `stEmpty` para cadenas de caracteres vacías ("").

[`\MaxTime`]

Tipo de dato: `num`

El periodo máximo, en segundos, que debe esperar el programa para continuar con la ejecución. Si no se presiona ninguna tecla de función en ese periodo, el programa sigue ejecutándose en el gestor de errores, a no ser que se utilice el indicador `BreakFlag` (que se documenta a continuación). La constante `ERR_TP_MAXTIME` puede usarse para comprobar si ha transcurrido ya el tiempo máximo establecido.

[`\DIBreak`]

**Digital Input Break**

Tipo de dato: `signal di`

La señal digital que puede interrumpir el diálogo con el operador. Si no se presiona ninguna tecla de función cuando la señal cambia a 1 (o si ya tiene el valor 1), el programa sigue ejecutándose en el gestor de errores, a no ser que se utilice el indicador `BreakFlag` (que se documenta a continuación). La constante `ERR_TP_DIBREAK` puede usarse para comprobar si esto ha ocurrido.

[`\DIPassive`]

**Digital Input Passive**

Tipo de dato: `switch`

Este modificador redefine el comportamiento predeterminado con el argumento opcional `DIBreak`. En lugar de reaccionar cuando la señal cambia a 1 (o ya tiene el valor 1), la instrucción debe continuar en el gestor de errores (si no se utiliza `BreakFlag`) cuando la señal `DIBreak` cambia a 0 (o ya tiene el valor 0). La constante `ERR_TP_DIBREAK` puede usarse para comprobar si esto ha ocurrido.

[`\DOBreak`]

**Digital Output Break**

Tipo de dato: `signal do`

La señal digital que soporta la petición de finalización desde otras tareas. Si no se selecciona ningún botón cuando la señal cambia a 1 (o si ya tiene el valor 1), el programa sigue ejecutándose en el gestor de errores, a no ser que se utilice el indicador `BreakFlag` (que se documenta a continuación). La constante `ERR_TP_DOBREAK` puede usarse para comprobar si esto ha ocurrido.

[`\DOPassive`]

**Digital Output Passive**

Tipo de dato: `switch`

Este modificador redefine el comportamiento predeterminado con el argumento opcional `DOBreak`. En lugar de reaccionar cuando la señal cambia a 1 (o ya tiene

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.297 TPreadFK - Lee las teclas de función

RobotWare - OS

Continuación

el valor 1), la instrucción debe continuar en el gestor de errores (si no se utiliza BreakFlag) cuando la señal DOBreak cambia a 0 (o ya tiene el valor 0). La constante ERR\_TP\_DOBREAK puede usarse para comprobar si esto ha ocurrido.

[\PersBoolBreak]

### *Persistent Boolean Break*

Tipo de dato: bool

El booleano persistente que puede interrumpir el diálogo con el operador. Si no se selecciona ningún botón cuando el booleano persistente cambia a TRUE (o ya tiene el valor TRUE) entonces el programa continuará ejecutándose en el gestor de errores, a no ser que se utilice BreakFlag (que se documenta a continuación). La constante ERR\_TP\_PERSBOOLBREAK puede usarse para comprobar si esto ha ocurrido.

[\PersBoolPassive]

### *Persistent Boolean Passive*

Tipo de dato: switch

Este interruptor redefine el comportamiento con el argumento opcional PersBoolBreak. En lugar de reaccionar cuando el booleano persistente cambia a TRUE (o ya tiene el valor TRUE), la instrucción debe continuar en el gestor de errores (si no se utiliza BreakFlag) cuando el booleano persistente PersBoolBreak cambia a FALSE (o ya tiene el valor FALSE). La constante ERR\_TP\_PERSBOOLBREAK puede usarse para comprobar si esto ha ocurrido.

[\BreakFlag]

Tipo de dato: errnum

Una variable que contiene el código de error si se utiliza MaxTime, DIBreak, DOBreak, o PersBoolBreak. Si se omite esta variable opcional, se ejecuta el gestor de errores. Las constantes ERR\_TP\_MAXTIME, ERR\_TP\_DIBREAK, ERR\_TP\_DOBREAK, y ERR\_TP\_PERSBOOLBREAK pueden usarse para seleccionar el motivo.

---

## Ejecución de programas

El texto de información se escribe siempre en una nueva línea. Si la pantalla está llena de texto, el cuerpo de texto se mueve previamente una línea hacia arriba. Puede haber un máximo de 7 líneas por encima del nuevo texto escrito.

El texto se escribe en las teclas de función adecuadas.

La ejecución del programa espera hasta que se presiona una de las teclas de función activadas.

Descripción de la petición concurrente de TPreadFK o TPreadNum en el FlexPendant (petición TP) desde una misma tarea de programa o desde otras tareas de programa:

- Una nueva petición de unidad de programación de otras tareas de programa no recibe el foco de programa (nuevo almacenamiento en la cola)

Continúa en la página siguiente



- Una nueva petición de unidad de programación de una rutina TRAP de la misma tarea de programa recibe el foco de programa (almacenamiento anterior en la cola)
- Un paro de programa recibe el foco de programa (almacenamiento anterior en la cola)
- Una nueva petición de unidad de programación con el programa parado recibe el foco de programa (almacenamiento anterior en la cola)

### Más ejemplos

A continuación aparecen más ejemplos de cómo usar la instrucción TPreadFK.

#### Ejemplo 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;
    
```

El robot se mueve a la posición de servicio si se presiona la cuarta tecla de función ("Yes") o si se activa la señal 5. Si no se da ninguna respuesta en un periodo de 10 minutos, la ejecución finaliza.

### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema ERRNO cambiará a:

| Nombre             | Causa del error   |
|--------------------|---|
| ERR_NO_ALIASIO_DEF | La variable de señal es una variable declarada en RAPID. No se ha conectado a una señal E/S definida en la configuración de E/S con la instrucción AliasIO.                         |
| ERR_TP_MAXTIME     | Tiempo límite (parámetro \MaxTime) antes de que responda el operador.   |
| ERR_TP_DIBREAK     | Una instrucción de lectura de FlexPendant fue interrumpida por una entrada digital.<br>Se estableció una entrada digital (parámetro \DIBreak) antes de que respondiera el operador. |
| ERR_TP_DOBREAK     | Una instrucción de lectura de FlexPendant fue interrumpida por una salida digital.<br>Se estableció una salida digital (parámetro \DOBreak) antes de que respondiera el operador.   |
| ERR_TP_NO_CLIENT   | No hay ningún cliente con el que interactuar al utilizar una instrucción de lectura desde FlexPendant.  |

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.297 TPReadFK - Lee las teclas de función

RobotWare - OS

Continuación

| Nombre               | Causa del error  |
|----------------------|--|
| ERR_TP_PERSBOOLBREAK | Una instrucción de lectura de FlexPendant fue interrumpida por un booleano persistente.<br>Un booleano persistente ha sido cambiado (parámetro \PersBoolBreak) antes de que respondiera el operador. |

### Limitaciones

Evite utilizar valores muy bajos para el parámetro de tiempo límite \MaxTime cuando TPReadFK se ejecuta frecuentemente, por ejemplo, en un bucle. Puede dar como resultado un comportamiento impredecible del rendimiento del sistema, como por ejemplo, ralentizar la velocidad de respuesta de FlexPendant.

### Datos predefinidos

```
CONST string stEmpty := "";
```

La constante predefinida stEmpty puede usarse en el caso de las teclas de función que no tienen textos asignados.

### Sintaxis

```
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]
['\ ' PersBoolBreak ':=' <persistent (PERS) of bool>]
['\ ' PersBoolPassive]
['\ ' BreakFlag ':=' <var or pers (INOUT) of errnum>'];'
```

### Información relacionada

| Para obtener más información sobre           | Consulte   |
|--|--|
| Escritura y lectura a través del FlexPendant | <a href="#">Manual de referencia técnica - RAPID Overview</a>                        |
| Respuestas a través del FlexPendant          | <a href="#">Manual del operador - IRC5 con FlexPendant</a>                           |
| Borrado de la ventana de operador            | <a href="#">TPErase - Borra el texto mostrado en el FlexPendant en la página 877</a> |

## 1.298 TPreadNum - Lee un número del FlexPendant

### Utilización

TPreadNum (*FlexPendant Read Numerical*) se usa para leer un número del FlexPendant.

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción `TPreadNum`. Consulte también [Más ejemplos en la página 889](#).

#### Ejemplo 1

```
TPreadNum reg1, "How many units should be produced?";
```

El texto `How many units should be produced?` se escribe en el FlexPendant. La ejecución del programa espera hasta que se introduzca un número a través del teclado numérico del FlexPendant. El número se almacena en `reg1`.

### Argumentos

```
TPreadNum TPAnswer TPText [\MaxTime][\DIBreak] [\DIPassive]
[\DOBreak] [\DOPassive] [\PersBoolBreak] [\PersBoolPassive]
[\BreakFlag]
```

TPAnswer

Tipo de dato: `num`

La variable en la que se almacena el número introducido a través del FlexPendant.

TPText

Tipo de dato: `string`

El texto informativo que debe escribirse en el FlexPendant (con un máximo de 80 caracteres y 40 caracteres por fila).

[\MaxTime]

Tipo de dato: `num`

El periodo máximo que debe esperar el programa para continuar con la ejecución. Si no se introduce ningún número en ese periodo, el programa sigue ejecutándose en el gestor de errores, a no ser que se utilice el indicador `BreakFlag` (que se documenta a continuación). La constante `ERR_TP_MAXTIME` puede usarse para comprobar si ha transcurrido ya el tiempo máximo establecido.

[\DIBreak]

**Digital Input Break**

Tipo de dato: `signal di`

La señal digital que puede interrumpir el diálogo con el operador. Si no se introduce ningún número cuando la señal cambia a 1 (o si ya tiene el valor 1), el programa sigue ejecutándose en el gestor de errores, a no ser que se utilice el indicador `BreakFlag` (que se documenta a continuación). La constante `ERR_TP_DIBREAK` puede usarse para comprobar si esto ha ocurrido.

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.298 TPReadNum - Lee un número del FlexPendant

RobotWare - OS

Continuación

[\DIPassive]

### *Digital Input Passive*

Tipo de dato: switch

Este modificador redefine el comportamiento predeterminado con el argumento opcional `DIBreak`. En lugar de reaccionar cuando la señal cambia a 1 (o ya tiene el valor 1), la instrucción debe continuar en el gestor de errores (si no se utiliza `BreakFlag`) cuando la señal `DIBreak` cambia a 0 (o ya tiene el valor 0). La constante `ERR_TP_DIBREAK` puede usarse para comprobar si esto ha ocurrido.

[\DOBreak]

### *Digital Output Break*

Tipo de dato: signaldo

La señal digital que soporta la petición de finalización desde otras tareas. Si no se selecciona ningún botón cuando la señal cambia a 1 (o si ya tiene el valor 1), el programa sigue ejecutándose en el gestor de errores, a no ser que se utilice el indicador `BreakFlag` (que se documenta a continuación). La constante `ERR_TP_DOBREAK` puede usarse para comprobar si esto ha ocurrido.

[\DOPassive]

### *Digital Output Passive*

Tipo de dato: switch

Este modificador redefine el comportamiento predeterminado con el argumento opcional `DOBreak`. En lugar de reaccionar cuando la señal cambia a 1 (o ya tiene el valor 1), la instrucción debe continuar en el gestor de errores (si no se utiliza `BreakFlag`) cuando la señal `DOBreak` cambia a 0 (o ya tiene el valor 0). La constante `ERR_TP_DOBREAK` puede usarse para comprobar si esto ha ocurrido.

[\PersBoolBreak]

### *Persistent Boolean Break*

Tipo de dato: bool

El booleano persistente que puede interrumpir el diálogo con el operador. Si no se selecciona ningún botón cuando el booleano persistente cambia a `TRUE` (o ya tiene el valor `TRUE`) entonces el programa continuará ejecutándose en el gestor de errores, a no ser que se utilice `BreakFlag` (que se documenta a continuación). La constante `ERR_TP_PERSBOOLBREAK` puede usarse para comprobar si esto ha ocurrido.

[\PersBoolPassive]

### *Persistent Boolean Passive*

Tipo de dato: switch

Este interruptor redefine el comportamiento con el argumento opcional `PersBoolBreak`. En lugar de reaccionar cuando el booleano persistente cambia a `TRUE` (o ya tiene el valor `TRUE`), la instrucción debe continuar en el gestor de errores (si no se utiliza `BreakFlag`) cuando el booleano persistente `PersBoolBreak` cambia a `FALSE` (o ya tiene el valor `FALSE`). La constante `ERR_TP_PERSBOOLBREAK` puede usarse para comprobar si esto ha ocurrido.

*Continúa en la página siguiente*

[\BreakFlag]

**Tipo de dato:** errnum

Una variable que contiene el código de error si se utiliza `MaxTime`, `DIBreak`, `DOBBreak`, o `PersBoolBreak`. Si se omite esta variable opcional, se ejecuta el gestor de errores. Las constantes `ERR_TP_MAXTIME`, `ERR_TP_DIBREAK`, `ERR_TP_DOBREAK`, y `ERR_TP_PERSBOOLBREAK` pueden usarse para seleccionar el motivo.

### Ejecución de programas

El texto de información se escribe siempre en una nueva línea. Si la pantalla está llena de texto, el cuerpo de texto se mueve previamente una línea hacia arriba. Puede haber un máximo de 7 líneas por encima del nuevo texto escrito.

La ejecución del programa espera hasta que se escribe un número mediante el teclado numérico (seguido de Intro u OK) o hasta que la instrucción es interrumpida por un tiempo límite agotado o una acción de señal.

Consulte `TPreadFK` para obtener una descripción de la petición concurrente de `TPreadFK` o `TPreadNum` en el FlexPendant desde la misma tarea de programa o desde tareas de programa diferentes.

### Más ejemplos

A continuación aparecen más ejemplos de cómo usar la instrucción `TPreadNum`.

#### Ejemplo 1

```
TPreadNum reg1, "How many units should be produced?";
FOR i FROM 1 TO reg1 DO
  produce_part;
ENDFOR
```

Se escribe el texto `How many units should be produced?` en la pantalla del FlexPendant. A continuación, la rutina `produce_part` se repite el número de veces especificado a través del FlexPendant.

### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                          | Causa del error   |
|---------------------------------|---|
| <code>ERR_NO_ALIASIO_DEF</code> | La variable de señal es una variable declarada en RAPID. No se ha conectado a una señal E/S definida en la configuración de E/S con la instrucción <code>AliasIO</code> .                         |
| <code>ERR_TP_MAXTIME</code>     | Tiempo límite (parámetro <code>\MaxTime</code> ) antes de que responda el operador.   |
| <code>ERR_TP_DIBREAK</code>     | Una instrucción de lectura de FlexPendant fue interrumpida por una entrada digital.<br>Se estableció una entrada digital (parámetro <code>\DIBreak</code> ) antes de que respondiera el operador. |
| <code>ERR_TP_DOBREAK</code>     | Una instrucción de lectura de FlexPendant fue interrumpida por una salida digital.<br>Se estableció una salida digital (parámetro <code>\DOBBreak</code> ) antes de que respondiera el operador.  |

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.298 TPreadNum - Lee un número del FlexPendant

RobotWare - OS

Continuación

| Nombre               | Causa del error  |
|----------------------|--|
| ERR_TP_NO_CLIENT     | No hay ningún cliente con el que interactuar al utilizar una instrucción de lectura desde FlexPendant.   |
| ERR_TP_PERSBOOLBREAK | Una instrucción de lectura de FlexPendant fue interrumpida por un booleano persistente.<br>Un booleano persistente ha sido cambiado (parámetro \PersBoolBreak) antes de que respondiera el operador. |

### Sintaxis

```
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]
  ['\'PersBoolBreak ':=' <persistent (PERS) of bool>]
  ['\'PersBoolPassive]
  ['\'BreakFlag':=' <var or pers (INOUT) of errnum>] ';'

```

### Información relacionada

| Para obtener más información sobre                                | Consulte   |
|---|--|
| Escritura y lectura a través del FlexPendant                      | <a href="#">Manual de referencia técnica - RAPID Overview</a>                        |
| Introducción de un número en el FlexPendant                       | <a href="#">Manual del operador - IRC5 con FlexPendant</a>                           |
| Ejemplos de cómo usar los argumentos MaxTime, DIBreak y BreakFlag | <a href="#">TPReadFK - Lee las teclas de función en la página 882</a>                |
| Borrado de la ventana de operador                                 | <a href="#">TPErase - Borra el texto mostrado en el FlexPendant en la página 877</a> |

## 1.299 TPShow - Cambia de ventana en el FlexPendant

### Utilización

TPShow (*FlexPendant Show*) se utiliza para seleccionar desde RAPID una ventana del FlexPendant.

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción TPShow:

#### Ejemplo 1

```
TPShow TP_LATEST;
```

La última ventana usada en el FlexPendant antes de la ventana actual del FlexPendant será la que se active tras la ejecución de esta instrucción.

### Argumentos

```
TPShow Window
```

Window

Tipo de dato: tpnum

La ventana TP\_LATEST mostrará la última ventana usada en el FlexPendant antes de la ventana actual del FlexPendant.

### Datos predefinidos

```
CONST tpnum TP_LATEST := 2;
```

### Ejecución de programas

Se activa la ventana que se seleccione para el FlexPendant.

### Sintaxis

```
TPShow  
[Window' :='] <expression (IN) of tpnum> ';' ;
```

### Información relacionada

| Para obtener más información sobre    | Consulte   |
|---------------------------------------|--|
| Comunicación a través del FlexPendant | <i>Manual de referencia técnica - RAPID Overview</i>                                 |
| Número de ventana del FlexPendant     | <a href="#">tpnum - Número de ventana del FlexPendant en la página 1802</a>          |
| Borrado de la ventana de operador     | <a href="#">TPERase - Borra el texto mostrado en el FlexPendant en la página 877</a> |

# 1 Instrucciones

---

## 1.300 TPWrite - Escribe en el FlexPendant

RobotWare - OS

## 1.300 TPWrite - Escribe en el FlexPendant

---

### Utilización

TPWrite (*FlexPendant Write*) se utiliza para escribir texto en el FlexPendant. Es posible escribir el valor de determinados datos, además de texto.

---

### Ejemplos básicos

Los ejemplos siguientes ilustran la instrucción TPWrite.

#### Ejemplo 1

```
TPWrite "Execution started";
```

El texto `Execution started` se escribe en el FlexPendant.

#### Ejemplo 2

```
TPWrite "No of produced parts="\Num:=reg1;
```

Por ejemplo, si `reg1` contiene el valor 5, se escribe el texto `No of produced parts=5` en el FlexPendant.

#### Ejemplo 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
```

Se escribe en el FlexPendant el nombre del robot de TCP controlado desde esta tarea de programa. Si no se controla ningún robot de TCP, se indica que la tarea no controla ningún robot.

---

### Argumentos

```
TPWrite String [\Num] | [\Bool] | [\Pos] | [\Orient] | [\Dnum]
```

String

Tipo de dato: string

La cadena de texto a escribir (con un máximo de 80 caracteres y 40 caracteres por fila).

[\Num]

*Numeric*

Tipo de dato: num

El dato cuyo valor numérico se desea escribir a continuación de la cadena de texto.

[\Bool]

*Boolean*

Tipo de dato: bool

El dato cuyo valor lógico se desea escribir a continuación de la cadena de texto.

*Continúa en la página siguiente*

---



[ \Pos ]

**Position**

Tipo de dato: pos

El dato cuya posición se desea escribir a continuación de la cadena de texto.

[ \Orient ]

**Orientation**

Tipo de dato: orient

El dato cuya orientación se desea escribir a continuación de la cadena de texto.

[ \Dnum ]

**Numeric**

Tipo de dato: dnum

El dato cuyo valor numérico se desea escribir a continuación de la cadena de texto.

## Ejecución de programas

El texto escrito en el FlexPendant comienza siempre en una nueva línea. Si la pantalla está llena de texto (11 líneas), dicho texto se mueve previamente una línea hacia arriba.

Si se usa uno de los argumentos \Num, \Dnum, \Bool, \Pos u \Orient , su valor se convierte en primer lugar en una cadena de texto, antes de añadirla a la primera cadena. La conversión del valor a una cadena de texto se realiza de la forma siguiente:

| Argumento | Valor                  | Cadena de texto          |
|-----------|------------------------|--------------------------|
| \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"             |

El valor se convierte en una cadena con un formato estándar de RAPID. Esto significa en principio 6 dígitos significativos. Si la parte decimal es menor que 0,000005 o mayor que 0,999995, el número se redondea a un entero.

## Limitaciones

Los argumentos \Num, \Dnum, \Bool, \Pos y \Orient son excluyentes entre sí y por tanto no pueden usarse simultáneamente en una misma instrucción.

## Sintaxis

```
TPWrite
  [TPText':=' ] <expression (IN) of string>
  ['\Num':=' <expression (IN) of num> ]
  | ['\Bool':=' <expression (IN) of bool> ]
  | ['\Pos':=' <expression (IN) of pos> ]
```

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.300 TPWrite - Escribe en el FlexPendant

RobotWare - OS

Continuación

```
| [ '\Orient':=' <expression (IN) of orient> ]  
| [ '\Dnum':=' <expression (IN) of dnum> ]';'
```

---

### Información relacionada

| Para obtener más información sobre         | Consulte  |
|--|---|
| Borrado y lectura a través del FlexPendant | <i>Manual de referencia técnica - RAPID Overview</i>  |
| Borrado de la ventana de operador          | <a href="#"><i>TPErase - Borra el texto mostrado en el FlexPendant en la página 877</i></a> |

## 1.301 TriggC - Movimiento circular del robot con eventos

### Utilización

TriggC (*Trigg Circular*) se utiliza para establecer señales de salida y/o ejecutar rutinas de interrupción en posiciones fijas al mismo tiempo que se mueve el robot a lo largo de una trayectoria circular.

Es posible definir uno o varios eventos (hasta un máximo de 25) mediante las instrucciones TriggIO, TriggEquip, TriggInt, TriggCheckIO, TriggSpeed, o TriggRampAO y hacer referencia posteriormente a estas definiciones en la instrucción TriggC.

Esta instrucción sólo puede usarse en la tarea principal T\_ROB1 o, si se cuenta con un sistema *MultiMove*, en las tareas de movimiento.

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción TriggC:

Consulte también [Más ejemplos en la página 900](#).

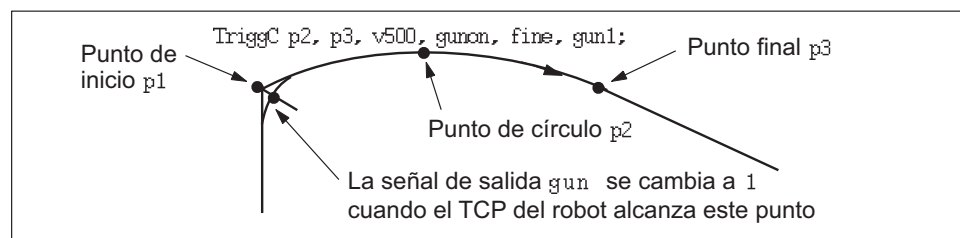
#### Ejemplo 1

```
VAR triggdata gunon;

TriggIO gunon, 0 \Start \DOp:=gun, 1;
MoveL p1, v500, z50, gun1;
TriggC p2, p3, v500, gunon, fine, gun1;
```

La señal digital de salida `gun` se activa cuando el TCP del robot atraviesa el punto central de la trayectoria de esquina del punto `p1`.

En la figura se muestra un ejemplo de evento de E/S basado en una posición fija.



xx0500002267

### Argumentos

```
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

Tipo de dato:switch

Distintas instrucciones consecutivas se ejecutan mientras el robot está en movimiento. Este argumento no se utiliza normalmente, pero puede utilizarse para evitar los paros no deseados, causados por la sobrecarga de la CPU al utilizar

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.301 TriggC - Movimiento circular del robot con eventos

RobotWare - OS

Continuación

puntos de paso. Esto resulta útil cuando los puntos programados están muy cercanos entre sí y se trabaja a velocidades elevadas. Este argumento también resulta útil si, por ejemplo, no se requiere la comunicación con equipos externos ni la sincronización entre los equipos externos y los movimientos del robot. También puede usarse para ajustar la ejecución de la trayectoria del robot, para evitar el aviso 50024 Fallo en punto de paso o el error 50082 Límite de deceleración.

Cuando se utiliza el argumento `\Conc`, el número de instrucciones de movimiento seguidas está limitado a 5. En secciones de programa que incluyen `StorePath-RestoPath`, no se permite el uso de instrucciones con el argumento `\Conc`.

Si se omite este argumento y `ToPoint` no es un punto de paro, la instrucción siguiente se ejecuta algún tiempo antes de que el robot alcance la zona programada.

Este argumento no puede usarse en los movimientos sincronizados coordinados en un sistema `MultiMove`.

`CirPoint`

Tipo de dato: `robtarget`

El punto de círculo del robot. Consulte la instrucción `MoveC` para obtener una descripción más detallada del movimiento circular. El punto de círculo se define como una posición con nombre o se almacena directamente en la instrucción (marcada con un asterisco \* en la instrucción).

`ToPoint`

Tipo de dato: `robtarget`

El punto de destino de los ejes del robot y de los ejes externos. Se define como una posición con nombre o se almacena directamente en la instrucción (marcada con un asterisco \* en la instrucción).

[ `\ID` ]

*Synchronization id*

Tipo de dato: `identno`

El argumento [ `\ID` ] es obligatorio en los sistemas `MultiMove`, si el movimiento es sincronizado o sincronizado coordinado. Este argumento no está permitido en ningún otro caso. El número de ID especificado debe ser el mismo en todas las tareas de programa que cooperan entre sí. Al usar el número de ID los movimientos no se mezclan en tiempo de ejecución.

`Speed`

Tipo de dato: `speeddata`

Los datos de velocidad que se aplican a los movimientos. Los datos de velocidad definen la velocidad del punto central de la herramienta, la reorientación de la misma y los ejes externos.

[ `\T` ]

*Time*

Tipo de dato: `num`

*Continúa en la página siguiente*

Este argumento se utiliza para especificar el tiempo total en segundos que dura el movimiento del robot. A continuación, se sustituye por los datos de velocidad correspondientes.

Trigg\_1

**Tipo de dato:** triggdata

La variable que hace referencia a las condiciones de disparo y la actividad de disparo, definida anteriormente en el programa usando las instrucciones TriggIO, TriggEquip, TriggInt, TriggCheckIO, TriggSpeed o TriggRampAO.

TriggArray

**Trigg Data Array Parameter**

**Tipo de dato:** triggdata

La variable de matriz que hace referencia a las condiciones de disparo y la actividad de disparo, definidas anteriormente en el programa usando las instrucciones TriggIO, TriggEquip, TriggInt, TriggSpeed, TriggCheckIO o TriggRampAO.

La matriz presenta una limitación de 25 elementos y es necesario definir de 1 a 25 condiciones de disparo.

No es posible utilizar los argumentos opcionales T2, T3, T4, T5, T6, T7 o T8 al mismo tiempo que se utiliza el argumento TriggArray.

[ \T2 ]

**Trigg 2**

**Tipo de dato:** triggdata

La variable que hace referencia a las condiciones de disparo y la actividad de disparo, definida anteriormente en el programa usando las instrucciones TriggIO, TriggEquip, TriggInt, TriggCheckIO, TriggSpeed o TriggRampAO.

[ \T3 ]

**Trigg 3**

**Tipo de dato:** triggdata

La variable que hace referencia a las condiciones de disparo y la actividad de disparo, definida anteriormente en el programa usando las instrucciones TriggIO, TriggEquip, TriggInt, TriggCheckIO, TriggSpeed o TriggRampAO.

[ \T4 ]

**Trigg 4**

**Tipo de dato:** triggdata

La variable que hace referencia a las condiciones de disparo y la actividad de disparo, definida anteriormente en el programa usando las instrucciones TriggIO, TriggEquip, TriggInt, TriggCheckIO, TriggSpeed o TriggRampAO.

[ \T5 ]

**Trigg 5**

**Tipo de dato:** triggdata

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.301 TriggC - Movimiento circular del robot con eventos

RobotWare - OS

Continuación

La variable que hace referencia a las condiciones de disparo y la actividad de disparo, definida anteriormente en el programa usando las instrucciones TriggIO, TriggEquip, TriggInt, TriggCheck, TriggSpeed o TriggRampAO.

[ \T6 ]

### *Trigg 6*

Tipo de dato: triggdata

La variable que hace referencia a las condiciones de disparo y la actividad de disparo, definida anteriormente en el programa usando las instrucciones TriggIO, TriggEquip, TriggInt, TriggCheckIO, TriggSpeed o TriggRampAO.

[ \T7 ]

### *Trigg 7*

Tipo de dato: triggdata

La variable que hace referencia a las condiciones de disparo y la actividad de disparo, definida anteriormente en el programa usando las instrucciones TriggIO, TriggEquip, TriggInt, TriggCheckIO, TriggSpeed o TriggRampAO.

[ \T8 ]

### *Trigg 8*

Tipo de dato: triggdata

La variable que hace referencia a las condiciones de disparo y la actividad de disparo, definida anteriormente en el programa usando las instrucciones TriggIO, TriggEquip, TriggInt, TriggCheckIO, TriggSpeed o TriggRampAO.

Zone

Tipo de dato: zonedata

Los datos de zona del movimiento. Los datos de zona describen el tamaño de la trayectoria de esquina generada.

[ \Inpos ]

### *In position*

Tipo de dato: stoppointdata

Este argumento se utiliza para especificar los criterios de convergencia para la posición del TCP del robot en el punto de paro. Los datos de puntos de paro sustituyen a la zona especificada en el parámetro Zone.

Tool

Tipo de dato: tooldata

La herramienta en uso durante el movimiento del robot. El punto central de la herramienta es el punto que se mueve hacia la posición de destino especificada.

[ \WObj ]

### *Work Object*

Tipo de dato: wobjdata

El objeto de trabajo (sistema de coordenadas) con el que está relacionada la posición de robot indicada en la instrucción.

*Continúa en la página siguiente*

Es posible omitir este argumento. Si se omite, la posición depende del sistema de coordenadas mundo. Si por otro lado se usa un TCP estacionario o ejes externos coordinados, es necesario especificar este argumento para ejecutar un movimiento lineal respecto del objeto de trabajo.

[ \Corr ]

#### *Correction*

Tipo de dato: `switch`

Los datos de corrección escritos en una entrada de corrección mediante una instrucción `CorrWrite` se añaden a la trayectoria y a la posición de destino si se utiliza este argumento.

Se requiere RobotWare, opción *Path Offset*, cuando se utiliza este argumento.

[ \TLoad ]

#### *Total load*

Tipo de dato: `loaddata`

El argumento `\TLoad` describe la carga total usada durante el movimiento. La carga total es la carga de la herramienta más la carga útil transportada por la herramienta. Si se utiliza el argumento `\TLoad`, no se tiene en cuenta el valor de `loaddata` en los `tooldata` actuales.

Si el argumento `\TLoad` tiene el valor `load0`, el argumento `\TLoad` no se tiene en cuenta y se utilizan en su lugar los `loaddata` de los `tooldata`.

Para poder utilizar el argumento `\TLoad`, es necesario cambiar el valor del parámetro de sistema `ModalPayLoadMode` a 0. Si `ModalPayLoadMode` tiene el valor 0, ya no es posible utilizar la instrucción `GripLoad`.

La carga total puede identificarse con la rutina de servicio `LoadIdentify`. Si el parámetro de sistema `ModalPayLoadMode` tiene el valor 0, el operador tiene la posibilidad de copiar los `loaddata` de la herramienta a una variable persistente `loaddata` existente o nueva al ejecutar la rutina de servicio.

Es posible realizar una ejecución de prueba del programa sin ninguna carga útil utilizando una señal de entrada digital conectada a la entrada de sistema `SimMode` (modo simulado). Si la señal de entrada digital tiene el valor 1, los `loaddata` del argumento opcional `\TLoad` no se tienen en cuenta y se utilizan en su lugar los `loaddata` de los `tooldata` actuales.



#### **Nota**

La funcionalidad predeterminada de manejo de la carga útil es utilizar la instrucción `GripLoad`. Por tanto, el valor predeterminado del parámetro de sistema `ModalPayLoadMode` es 1.

## Ejecución de programas

Consulte la instrucción `MoveC` para obtener más información acerca del movimiento circular.

A medida que se cumplen las condiciones de disparo cuando el robot se sitúa más y más cerca del punto final, se realizan las actividades de disparo definidas.

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.301 TriggC - Movimiento circular del robot con eventos

RobotWare - OS

Continuación

Las condiciones de disparo se cumplen a una distancia determinada del punto final de la instrucción o a una distancia determinada del punto de inicio de la instrucción, o bien en un momento determinado (limitado a un tiempo breve) antes del punto final de la instrucción.

Durante la ejecución paso a paso hacia delante, las actividades de E/S se realizan pero las rutinas de interrupción no se ejecutan. Durante la ejecución paso a paso hacia atrás, no se realiza ninguna actividad de disparo.

---

### Más ejemplos

A continuación aparecen más ejemplos de cómo usar la instrucción `TriggC`.

#### Ejemplo 1

```
VAR intnum intnol;
VAR triggdata trigg1;
...
PROC main()
...
CONNECT intnol WITH trap1;
TriggInt trigg1, 0.1 \Time, intnol;
...
TriggC p1, p2, v500, trigg1, fine, gun1;
TriggC p3, p4, v500, trigg1, fine, gun1;
...
IDelete intnol;
```

Se ejecuta la rutina de interrupción `trap1` cuando el punto de trabajo se encuentra en la posición a 0,1 s antes del punto `p2` o `p4` respectivamente.

#### Ejemplo 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;
...

```

Las señales de salida digitales de la `do1` a la `do25` se activan durante el movimiento a `p2`. La distancia entre los valores de las señales es de 10 mm.

Continúa en la página siguiente



## Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                      | Causa del error   |
|-----------------------------|---|
| <code>ERR_AO_LIM</code>     | El argumento <code>ScaleValue</code> programado para la señal analógica de salida especificada <code>AOp</code> en algunas de las instrucciones <code>TriggSpeed</code> conectadas da lugar a la salida de límite de la señal analógica junto con la <code>Speed</code> programada en esta instrucción. |
| <code>ERR_DIPLAG_LIM</code> | El argumento <code>DipLag</code> programado en algunas de las instrucciones <code>TriggSpeed</code> conectadas es demasiado grande con respecto al tiempo preestablecido para los eventos en parámetros del sistema.  |
| <code>ERR_NORUNUNIT</code>  | No hay ningún contacto con el dispositivo de E/S al introducir la instrucción, y el <code>triggdata</code> utilizado depende de un dispositivo de E/S en funcionamiento, es decir, que se utiliza una señal en <code>triggdata</code> .   |
| <code>ERR_CONC_MAX</code>   | Se ha superado el número de instrucciones de movimiento seguidas con el argumento <code>\Conc</code> .  |

## Limitaciones

Limitaciones generales acorde con la instrucción `MoveC`.

Si el punto de inicio actual se desvía del punto habitual, de forma que la longitud de posicionamiento total de la instrucción `TriggC` es más corta de lo habitual, es posible que todas las condiciones de disparo o algunas de ellas se satisfagan inmediatamente en la misma posición. En estos casos, la secuencia en la que se realizan las actividades de disparo no estará definida. La lógica de programa del programa del usuario no puede basarse en una secuencia normal de actividades de disparo para un “movimiento incompleto”.



### ¡AVISO!

No se debe iniciar la instrucción `TriggC` desde el principio con la posición del robot más allá del punto de círculo. De lo contrario, el robot no toma la trayectoria programada (posicionamiento alrededor de la trayectoria circular en otra dirección, en comparación con la programada).

`TriggC` no puede ejecutarse en un gestor UNDO o en rutinas de RAPID que estén conectadas a los siguientes eventos especiales del sistema: `PowerOn`, `Stop`, `QStop`, `Restart`, `Reset` o `Step`.

## Sintaxis

```
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 > |
```

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.301 TriggC - Movimiento circular del robot con eventos

RobotWare - OS

Continuación

```
[TriggArray ':=' ] < array variable {*} (VAR) of trigggdata >
[ '\ T2 ':=' < variable (VAR) of trigggdata > ]
[ '\ T3 ':=' < variable (VAR) of trigggdata > ]
[ '\ T4 ':=' < variable (VAR) of trigggdata > ]
[ '\ T5 ':=' < variable (VAR) of trigggdata > ]
[ '\ T6 ':=' < variable (VAR) of trigggdata > ]
[ '\ T7 ':=' < variable (VAR) of trigggdata > ]
[ '\ T8 ':=' < variable (VAR) of trigggdata > ] ','
[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 > ] ';'

```

### Información relacionada

| Para obtener más información sobre     | Consulte   |
|--|--|
| Movimiento lineal con disparadores     | <a href="#">TriggL - Movimiento lineal del robot con eventos en la página 940</a>  |
| Movimiento de ejes con disparadores    | <a href="#">TriggJ - Movimientos de ejes del robot a partir de eventos en la página 932</a>  |
| Movimiento del robot en círculo        | <a href="#">MoveC - Mueve el robot en círculo en la página 389</a>   |
| Definición de disparadores             | <a href="#">TriggIO - Define un evento de E/S de posición o tiempo fijos cerca de un punto de paro en la página 926</a><br><a href="#">TriggEquip - Define un evento de E/S basado en la posición y el tiempo en la trayectoria en la página 914</a><br><a href="#">TriggInt - Define una interrupción dependiente de una posición en la página 921</a><br><a href="#">TriggCheckIO - Define una comprobación de E/S en una posición fija en la página 904</a><br><a href="#">TriggRampAO - Define un evento AO de rampa de posición fija en la trayectoria en la página 965</a><br><a href="#">TriggSpeed - Define la velocidad del TCP en proporción a una salida analógica con un evento de escala fija de posición-tiempo en la página 973</a> |
| Manejo de trigggdata                   | <a href="#">triggdata - Eventos de posicionamiento, triggg en la página 1805</a><br><a href="#">TriggDataReset - Restablecer el contenido en una variable de tipo trigggdata en la página 912</a><br><a href="#">TriggDataCopy - Copiar el contenido de una variable de tipo trigggdata en la página 910</a><br><a href="#">TriggDataValid - Comprobar si el contenido de una variable de tipo trigggdata es válido en la página 1537</a>  |
| Escritura en una entrada de corrección | <a href="#">CorrWrite - Escribe en un generador de correcciones en la página 183</a>   |
| Movimiento circular                    | <a href="#">Manual de referencia técnica - RAPID Overview</a>  |
| Definición de carga                    | <a href="#">loaddata - Datos de carga en la página 1694</a>  |
| Definición de velocidad                | <a href="#">speeddata - Datos de velocidad en la página 1767</a>   |

Continúa en la página siguiente

| Para obtener más información sobre  | Consulte   |
|---|--|
| Definición de datos de punto de paro  | <a href="#">stoppointdata - Datos de punto de paro en la página 1771</a>                 |
| Definición de herramientas  | <a href="#">tooldata - Datos de herramienta en la página 1795</a>                        |
| Definición de objetos de trabajo  | <a href="#">wobjdata - Datos del objeto de trabajo en la página 1823</a>                 |
| Definición de datos de zona   | <a href="#">zonedata - Datos de zonas en la página 1831</a>                              |
| Movimiento en general   | <i>Manual de referencia técnica - RAPID Overview</i>                                     |
| Ejemplo de cómo usar TLoad, carga total.  | <a href="#">MoveL - Mueve el robot siguiendo una trayectoria lineal en la página 447</a> |
| Definición de la carga útil de un robot   | <a href="#">GripLoad - Define la carga útil de un robot en la página 244</a>             |
| LoadIdentify, rutina de servicio de identificación de carga                                   | <i>Manual del operador - IRC5 con FlexPendant</i>  |
| Señal de entrada de sistema SimMode para mover el robot en el modo simulado sin carga útil.   | <i>Manual de referencia técnica - Parámetros del sistema</i>                             |
| Parámetro de sistema ModalPayloadMode para la activación y la desactivación de la carga útil. | <i>Manual de referencia técnica - Parámetros del sistema</i>                             |
| Path Offset   | <i>Application manual - Controller software IRC5</i>                                     |

# 1 Instrucciones

## 1.302 TriggCheckIO - Define una comprobación de E/S en una posición fija RobotWare - OS

### 1.302 TriggCheckIO - Define una comprobación de E/S en una posición fija

#### Utilización

`TriggCheckIO` se utiliza para definir condiciones de prueba del valor de una señal digital, un grupo de señales o una señal analógica de entrada y salida en una posición fija a lo largo de la trayectoria de movimiento del robot. Si la condición se cumple, no tendrá lugar ninguna acción específica. Sin embargo, si no se cumple, se ejecuta una rutina de interrupción después de que el robot se ha detenido (opcionalmente) en la trayectoria en el tiempo más breve posible.

Para conseguir una comprobación de E/S en un punto fijo, `TriggCheckIO` compensa el retardo del sistema de control (el retardo entre el servo y el robot). Los datos definidos se utilizan para la implementación o instrucciones `TriggL`, `TriggC` o `TriggJ` posteriores.

Esta instrucción sólo puede usarse en la tarea principal `T_ROB1` o, si se cuenta con un sistema *MultiMove*, en las tareas de movimiento.

#### Ejemplos básicos

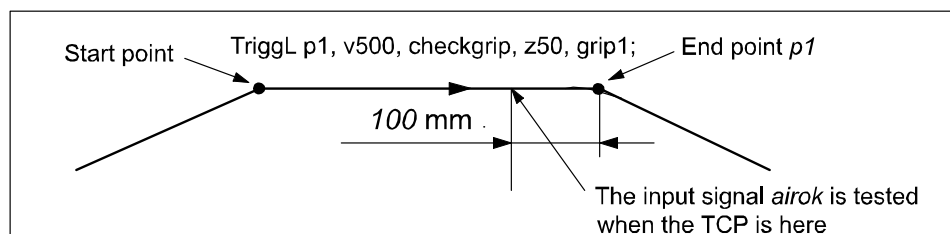
El ejemplo que aparece a continuación ilustra la instrucción `TriggCheckIO`:  
Consulte también [Más ejemplos en la página 907](#).

#### Ejemplo 1

```
VAR triggdata checkgrip;  
VAR intnum intnol;  
  
PROC main()  
  CONNECT intnol WITH trap1;  
  TriggCheckIO checkgrip, 100, airok, EQ, 1, intnol;  
  
  TriggL p1, v500, checkgrip, z50, grip1;
```

La señal digital de entrada `airok` se comprueba para determinar si tiene el valor 1 cuando el TCP se encuentra a 100 mm antes del punto `p1`. Si está activada, la ejecución normal del programa continúa. Si no está activada, se ejecuta la rutina de interrupción `trap1`.

En la figura se muestra un ejemplo de comprobación de E/S basado en una posición fija.



xx0500002254

Continúa en la página siguiente

### Argumentos

```
TriggCheckIO TriggData Distance [\Start] | [Next] | [Time] Signal  
Relation CheckValue | CheckDvalue [\StopMove] Interrupt  
[\Inhib] [\Mode]
```

TriggData

Tipo de dato: `triggdata`

Una variable para almacenar el `triggdata` devuelto por la instrucción. Estos datos `triggdata` se utilizan a continuación en las instrucciones `TriggL`, `TriggC` o `TriggJ` posteriores.

Distance

Tipo de dato: `num`

Define la posición de la trayectoria en la que debe producirse la comprobación de E/S.

Se especifica como la distancia en mm (valor positivo) desde el punto final de la trayectoria de movimiento (aplicable si no se utiliza el argumento `\Start` o `\Time`).

Consulte [Ejecución de programas en la página 906](#) para obtener más detalles.

[ \Start ]

Tipo de dato: `switch`

Se utiliza cuando la distancia del argumento `Distance` comienza en el punto de inicio del movimiento en lugar del punto final.

[ \Next ]

Tipo de dato: `switch`

Se utiliza cuando la distancia para el argumento `Distance` avanza hacia el siguiente punto programado. Si la `Distance` es mayor que la distancia al siguiente punto fino, el evento se ejecutará en el punto fino.

[ \Time ]

Tipo de dato: `switch`

Se utiliza cuando el valor especificado para el argumento `Distance` es en realidad un tiempo en segundos (valor positivo) en lugar de una distancia.

La posición fija de E/S basada en tiempo sólo puede usarse con tiempos breves (< 0,5 s) antes de que el robot alcance el punto final de la instrucción. Consulte la sección *Limitaciones* para obtener más detalles.

Signal

Tipo de dato: `signalxx`

El nombre de la señal que se comprobará. Puede ser cualquier tipo de señal de E/S.

Relation

Tipo de dato: `opnum`

Define cómo comparar el valor actual de la señal con el definido en el argumento `CheckValue`. Consulte el tipo de dato `opnum` para ver la lista de constantes predefinidas que se debe usar.

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.302 TriggCheckIO - Define una comprobación de E/S en una posición fija

RobotWare - OS

Continuación

CheckValue

Tipo de dato: num

El valor con el que debe compararse el valor actual de la señal de entrada o salida (dentro del rango permitido para la señal actual). Si la señal es una señal digital, debe ser un valor entero.

Si la señal es una señal digital de grupo, el valor permitido depende del número de señales del grupo. El valor máximo que puede usarse en el argumento CheckValue es de 8388608, que es el valor que una señal digital de grupo de 23 bits puede tener como valor máximo (consulte los rangos válidos para num).

CheckDvalue

Tipo de dato: dnum

El valor con el que debe compararse el valor actual de la señal de entrada o salida (dentro del rango permitido para la señal actual). Si la señal es una señal digital, debe ser un valor entero.

Si la señal es una señal digital de grupo, el valor permitido depende del número de señales del grupo. El valor máximo de los bits de señales que puede tener una señal digital de grupo es de 32. Con una variable dnum es posible cubrir los valores del 0 al 4294967295, que constituyen el rango que puede tener una señal digital de 32 bits.

[ \StopMove ]

Tipo de dato: switch

Especifica que si no se cumple la señal, el robot se detiene en la trayectoria lo antes posible antes de que la rutina de interrupción se ejecute.

Interrupt

Tipo de dato: intnum

La variable utilizada para identificar la rutina de interrupción a ejecutar.

[ \Inhib ]

*Inhibit*

Tipo de dato: bool

El nombre de un indicador de variable persistente para la inhibición de la ejecución de la rutina de interrupción.

Si se utiliza este argumento opcional y el valor real del indicador especificado es TRUE en la posición y el tiempo de la comprobación de E/S, la comprobación no se realiza.

[ \Mode ]

Tipo de dato: triggmode

Se utiliza para especificar diferentes modos de acción al definir disparadores.

---

## Ejecución de programas

Cuando se ejecuta la instrucción TriggCheckIO, la condición de disparo se almacena en una variable especificada para el argumento TriggData.

Continúa en la página siguiente

## 1.302 TriggCheckIO - Define una comprobación de E/S en una posición fija

RobotWare - OS

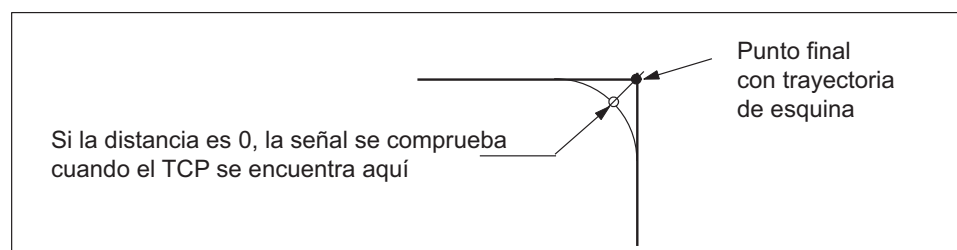
Continuación

A continuación, cuando se ejecuta una de las instrucciones `TriggL`, `TriggC` o `TriggJ`, se aplica lo siguiente en cuanto a las definiciones de `TriggCheckIO`:

En la tabla se describe la distancia especificada en el argumento `Distance`:

|                      |  |
|----------------------|--|
| Movimiento lineal    | La distancia en línea recta  |
| Movimiento circular  | La longitud del arco del círculo   |
| Movimiento no lineal | La longitud de arco aproximada a lo largo de la trayectoria (para conseguir la exactitud adecuada, la distancia no debe superar la mitad de la longitud del arco). |

La figura muestra la comprobación de E/S en una posición fija con una trayectoria de esquina.



xx0500002256

La comprobación de la E/S con una posición fija se realiza cuando se atraviesa el punto de inicio (punto final) si la distancia especificada respecto del punto final (punto de inicio) no se encuentra dentro de la longitud de movimiento de la instrucción actual (`TriggL...`).

Cuando el TCP se encuentra en el punto especificado de la trayectoria, se realiza la comprobación de E/S siguiente en el sistema:

- Lectura del valor actual de la señal de E/S.
- Comparación del valor leído con `CheckValue`, acorde con la relación especificada en `Relation`.
- Si la comparación da como resultado `TRUE`, no se hace nada más
- Si la comparación da como resultado `FALSE`, se hace lo siguiente:
- Si está presente el parámetro opcional `\StopMove`, el robot se detiene en la trayectoria lo antes posible.
- Se genera y se ejecuta la rutina `TRAP` especificada.

### Más ejemplos

A continuación aparecen más ejemplos de cómo usar la instrucción `TriggCheckIO`.

#### Ejemplo 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, gripl;
```

Continúa en la página siguiente

# 1 Instrucciones

## 1.302 TriggCheckIO - Define una comprobación de E/S en una posición fija

RobotWare - OS

Continuación

```
...
TRAP waitgate
! Block movement
StopMove;
! log some information
...
! Wait until signal is set
WaitDI gatedi,1;
! Unlock block, and resume movement
StartMove;
ENDTRAP
```

Se comprueba la puerta de la siguiente operación con una pieza de trabajo para determinar si está abierta (se comprueba la señal digital de entrada `gatedi` para determinar si tiene el valor 1) cuando el TCP se encuentra 150 mm antes del punto `p1`. Si está abierta, el robot se desplaza hasta `p1` y continúa. Si no está abierta, el robot se detiene en la trayectoria y se ejecuta la rutina de interrupción `waitgate`. Esta interrupción bloquea los movimientos posteriores, registra determinada información y suele esperar a que se satisfagan las condiciones para ejecutar una instrucción `StartMove` con el fin de reanudar la trayectoria interrumpida.

### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                          | Causa del error   |
|---------------------------------|---|
| <code>ERR_AO_LIM</code>         | El argumento programado <code>CheckValue</code> o <code>CheckDvalue</code> para la señal analógica de salida especificada <code>Signal</code> está fuera de límites.                    |
| <code>ERR_GO_LIM</code>         | El argumento programado <code>CheckValue</code> o <code>CheckDvalue</code> para la señal digital de salida de grupo especificada <code>Signal</code> está fuera de límites.             |
| <code>ERR_NO_ALIASIO_DEF</code> | La variable de señal es una variable declarada en <code>RAPID</code> . No se ha conectado a una señal E/S definida en la configuración de E/S con la instrucción <code>AliasIO</code> . |

### Limitaciones

La comprobación de E/S basada en una distancia (sin el argumento `\Time`) se ha diseñado para los puntos de paso (trayectorias de esquina). Las comprobaciones de E/S basadas en distancia, con puntos de paro, presentan una exactitud menor que la especificada a continuación.

La comprobación de E/S basada en tiempo (con el argumento `\Time`) se ha diseñado para los puntos de paro. Las comprobaciones de E/S basadas en tiempo, con puntos de paso, presentan una exactitud menor que la especificada a continuación.

Las comprobaciones de E/S basadas en tiempo sólo pueden especificarse a partir del punto final del movimiento. Este tiempo no puede superar el tiempo de frenado actual del robot, que tiene un máximo aproximado de 0,5 s (valores típicos a una velocidad de 500 mm/s: para IRB 2400 es 150 ms, y para IRB 6400 es 250 ms). Si el tiempo especificado es mayor que el tiempo de frenado actual, la comprobación

Continúa en la página siguiente



## 1.302 TriggCheckIO - Define una comprobación de E/S en una posición fija RobotWare - OS Continuación

de E/S se genera en cualquier caso, pero no hasta que se inicia el frenado (más tarde de lo especificado). La totalidad del tiempo de movimiento para el movimiento actual puede utilizarse durante los movimientos pequeños y rápidos.

Los valores absolutos típicos en cuanto a la exactitud de las pruebas de entradas digitales es de +/- 5 ms. Los valores típicos en cuanto a la repetición de las pruebas de entradas digitales es de +/- 2 ms.

TriggCheckIO no puede ejecutarse en un gestor UNDO o en rutinas de RAPID que estén conectadas a los siguientes eventos especiales del sistema: PowerOn, Stop, QStop, Restart, Reset o Step.

### Sintaxis

```
TriggCheckIO
[ TriggData ':= ' ] < variable (VAR) of triggdata> ', '
[ Distance' := ' ] < expression (IN) of num>
[ '\ ' Start ] | [ '\ ' Next ] | [ '\ ' 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> ] ';'

```

### Información relacionada

| Para obtener más información sobre                              | Consulte  |
|---|---|
| Utilización de disparadores                                     | <a href="#">TriggL - Movimiento lineal del robot con eventos en la página 940</a><br><a href="#">TriggC - Movimiento circular del robot con eventos en la página 895</a><br><a href="#">TriggJ - Movimientos de ejes del robot a partir de eventos en la página 932</a> |
| Definición de eventos de E/S basados en la posición y el tiempo | <a href="#">TriggIO - Define un evento de E/S de posición o tiempo fijos cerca de un punto de paro en la página 926</a><br><a href="#">TriggEquip - Define un evento de E/S basado en la posición y el tiempo en la trayectoria en la página 914</a>                    |
| Definición de interrupciones basadas en la posición             | <a href="#">TriggInt - Define una interrupción dependiente de una posición en la página 921</a>   |
| Almacenamiento de datos de disparo                              | <a href="#">triggdata - Eventos de posicionamiento, trigg en la página 1805</a>   |
| Definición de diferentes modos de acción de disparo             | <a href="#">triggmode - Disparar modo de acción en la página 1811</a>   |
| Definición de operadores de comparación                         | <a href="#">opnum - Operador de comparación en la página 1715</a>   |

# 1 Instrucciones

---

## 1.303 TriggDataCopy - Copiar el contenido de una variable de tipo triggdata

RobotWare - OS

## 1.303 TriggDataCopy - Copiar el contenido de una variable de tipo triggdata

---

### Utilización

TriggDataCopy se utiliza para copiar el contenido en una variable triggdata.

---

### Ejemplos básicos

El ejemplo siguiente ilustra la instrucción TriggDataCopy.

#### Ejemplo 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;
  ...
```

El procedimiento MyTriggProcL anterior utiliza la instrucción TriggDataCopy para copiar los argumentos opcionales triggdata al lugar correcto de la matriz triggdata que se utiliza en la instrucción TriggL.

---

### Argumentos

TriggDataCopy Source Destination

Source

Tipo de dato: triggdata

La variable triggdata desde la que se copia.

Destination

Tipo de dato: triggdata

*Continúa en la página siguiente*

---

1.303 TriggDataCopy - Copiar el contenido de una variable de tipo triggdata  
 RobotWare - OS  
 Continuación

La variable `triggdata` a la que se copia.

**Ejecución de programas**

La instrucción `TriggDataCopy` se utiliza para copiar datos de una variable `triggdata` a otra variable `triggdata`. Esta instrucción puede resultar útil al trabajar con variables de matriz `triggdata`.

**Limitaciones**

`TriggDataCopy` no puede ejecutarse en un gestor UNDO o en rutinas de RAPID que estén conectadas a los siguientes eventos especiales del sistema: PowerOn, Stop, QStop, Restart, Reset o Step.

**Sintaxis**

```
TriggDataCopy
  [Source ':=' ] < variable (VAR) of triggdata > ','
  [Destination ':=' ] < variable (VAR) of triggdata > ';'

```

**Información relacionada**

| Para obtener más información sobre   | Consulte   |
|--------------------------------------|--|
| Movimiento lineal con disparadores   | <a href="#">TriggL - Movimiento lineal del robot con eventos en la página 940</a>  |
| Movimiento de ejes con disparadores  | <a href="#">TriggJ - Movimientos de ejes del robot a partir de eventos en la página 932</a>  |
| Movimiento circular con disparadores | <a href="#">TriggC - Movimiento circular del robot con eventos en la página 895</a>  |
| Definición de disparadores           | <a href="#">TriggIO - Define un evento de E/S de posición o tiempo fijos cerca de un punto de paro en la página 926</a><br><a href="#">TriggEquip - Define un evento de E/S basado en la posición y el tiempo en la trayectoria en la página 914</a><br><a href="#">TriggInt - Define una interrupción dependiente de una posición en la página 921</a><br><a href="#">TriggCheckIO - Define una comprobación de E/S en una posición fija en la página 904</a><br><a href="#">TriggRampAO - Define un evento AO de rampa de posición fija en la trayectoria en la página 965</a><br><a href="#">TriggSpeed - Define la velocidad del TCP en proporción a una salida analógica con un evento de escala fija de posición-tiempo en la página 973</a> |
| Manejo de <code>triggdata</code>     | <a href="#">triggdata - Eventos de posicionamiento, trigg en la página 1805</a><br><a href="#">TriggDataReset - Restablecer el contenido en una variable de tipo triggdata en la página 912</a><br><a href="#">TriggDataValid - Comprobar si el contenido de una variable de tipo triggdata es válido en la página 1537</a>  |

## 1 Instrucciones

---

### 1.304 TriggDataReset - Restablecer el contenido en una variable de tipo triggdata

RobotWare - OS

### 1.304 TriggDataReset - Restablecer el contenido en una variable de tipo triggdata

---

#### Utilización

TriggDataReset se utiliza para restablecer el contenido de una variable triggdata.

---

#### Ejemplos básicos

El ejemplo siguiente ilustra la instrucción TriggDataReset.

#### Ejemplo 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;
  ...
```

El procedimiento MyTriggProcL anterior utiliza la instrucción TriggDataReset para restablecer la matriz triggdata antes de su uso.

---

#### Argumentos

TriggDataReset TriggData

TriggData

Tipo de dato: triggdata

La variable triggdata que se desea restablecer.

*Continúa en la página siguiente*

## 1.304 TriggDataReset - Restablecer el contenido en una variable de tipo triggdata RobotWare - OS Continuación

### Ejecución de programas

La instrucción `TriggDataReset` se utiliza para eliminar cualquier condición de disparo utilizada anteriormente en una variable `triggdata`. Esta instrucción puede resultar útil al trabajar con variables de matriz `triggdata`.

### Limitaciones

`TriggDataReset` no puede ejecutarse en un gestor UNDO o en rutinas de RAPID que estén conectadas a los siguientes eventos especiales del sistema: PowerOn, Stop, QStop, Restart, Reset o Step.

### Sintaxis

```
TriggDataReset
  [TriggData ':='] < variable (VAR) of triggdata > ';'

```

### Información relacionada

| Para obtener más información sobre   | Consulte   |
|--------------------------------------|--|
| Movimiento lineal con disparadores   | <a href="#">TriggL - Movimiento lineal del robot con eventos en la página 940</a>  |
| Movimiento de ejes con disparadores  | <a href="#">TriggJ - Movimientos de ejes del robot a partir de eventos en la página 932</a>  |
| Movimiento circular con disparadores | <a href="#">TriggC - Movimiento circular del robot con eventos en la página 895</a>  |
| Definición de disparadores           | <a href="#">TriggIO - Define un evento de E/S de posición o tiempo fijos cerca de un punto de paro en la página 926</a><br><a href="#">TriggEquip - Define un evento de E/S basado en la posición y el tiempo en la trayectoria en la página 914</a><br><a href="#">TriggInt - Define una interrupción dependiente de una posición en la página 921</a><br><a href="#">TriggCheckIO - Define una comprobación de E/S en una posición fija en la página 904</a><br><a href="#">TriggRampAO - Define un evento AO de rampa de posición fija en la trayectoria en la página 965</a><br><a href="#">TriggSpeed - Define la velocidad del TCP en proporción a una salida analógica con un evento de escala fija de posición-tiempo en la página 973</a> |
| Manejo de <code>triggdata</code>     | <a href="#">triggdata - Eventos de posicionamiento, trigg en la página 1805</a><br><a href="#">TriggDataCopy - Copiar el contenido de una variable de tipo triggdata en la página 910</a><br><a href="#">TriggDataValid - Comprobar si el contenido de una variable de tipo triggdata es válido en la página 1537</a>  |

# 1 Instrucciones

1.305 TriggEquip - Define un evento de E/S basado en la posición y el tiempo en la trayectoria  
RobotWare - OS

## 1.305 TriggEquip - Define un evento de E/S basado en la posición y el tiempo en la trayectoria

### Utilización

TriggEquip (*Trigg Equipment*) se utiliza para definir condiciones y acciones para la activación de una señal digital, un grupo de señales digitales o una señal analógica de salida en una posición fija a lo largo de la trayectoria de movimiento del robot, con la posibilidad de aplicar una compensación de tiempo para el retardo del equipo externo.

Siempre debe usarse TriggIO (no TriggEquip) si se necesita una buena exactitud de los ajustes de E/S cerca de un punto de paro.

Los datos definidos se utilizan para la implementación o instrucciones TriggL, TriggC o TriggJ posteriores.

Esta instrucción sólo puede usarse en la tarea principal T\_ROB1 o, si se cuenta con un sistema *MultiMove*, en las tareas de movimiento.

### Ejemplos básicos

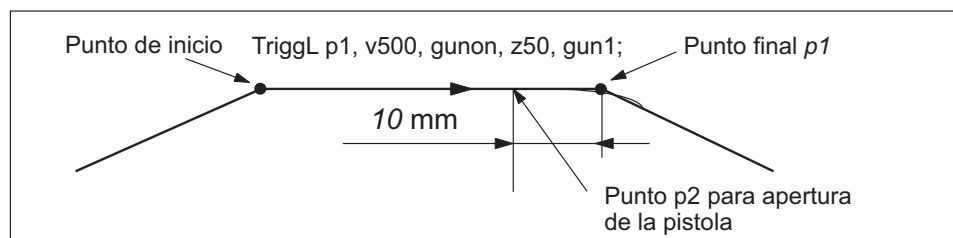
El ejemplo que aparece a continuación ilustra la instrucción TriggEquip:  
Consulte también [Más ejemplos en la página 918](#).

#### Ejemplo 1

```
VAR triggdata gunon;  
...  
TriggEquip gunon, 10, 0.1 \DOP:=gun, 1;  
TriggL p1, v500, gunon, z50, gun1;
```

La herramienta *gun1* empieza a abrirse cuando su TCP se encuentra 0,1 s antes del punto ficticio *p2* (10 mm antes del punto *p1*). La pistola está totalmente abierta cuando el TCP alcanza el punto *p2*.

En la figura se muestra un ejemplo de evento de E/S basado en una posición y un tiempo fijos.



xx0500002260

### Argumentos

```
TriggEquip TriggData Distance [\Start] | [\Next] EquipLag [\DOP]  
| [\GOp] | [\AOp] | [\ProcID] SetValue | SetDvalue [\Inhib]  
[\InhibSetValue] [\Mode]
```

TriggData

Tipo de dato: triggdata

*Continúa en la página siguiente*

## 1.305 TriggEquip - Define un evento de E/S basado en la posición y el tiempo en la trayectoria RobotWare - OS Continuación

Una variable para almacenar el `triggdata` devuelto por la instrucción. Estos datos `triggdata` se utilizan a continuación en las instrucciones `TriggL`, `TriggC` o `TriggJ` posteriores.

`Distance`

Tipo de dato: `num`

Define la posición de la trayectoria en la que debe producirse el evento de equipo de E/S.

Se especifica como la distancia en mm (valor positivo) desde el punto final de la trayectoria de movimiento hacia el punto de inicio (aplicable si no se utiliza el argumento `\Start` y `\Next`).

Consulte [Ejecución de programas en la página 917](#) para obtener más detalles.

[ `\Start` ]

Tipo de dato: `switch`

Se utiliza cuando la distancia del argumento `Distance` comienza en el punto de inicio del movimiento en lugar del punto final.

[ `\Next` ]

Tipo de dato: `switch`

Se utiliza cuando la distancia para el argumento `Distance` avanza hacia el siguiente punto programado. Si la `Distance` es mayor que la distancia al siguiente punto fino, el evento se ejecutará en el punto fino.

`EquipLag`

*Equipment Lag*

Tipo de dato: `num`

Especifica el retardo del equipo externo, en segundos.

Para la compensación del retardo de los equipos externos, utilice un valor de argumento positivo. Un valor positivo en el argumento significa que la señal de E/S es activada por el sistema de robot en el momento especificado, antes de que el TCP alcance físicamente la distancia especificada respecto del punto de inicio o final del movimiento.

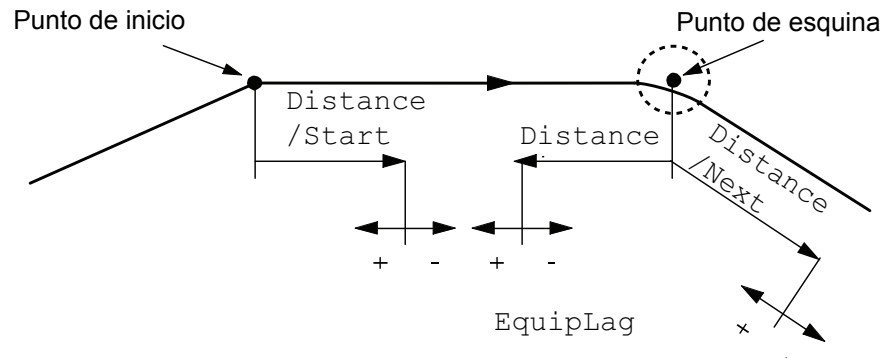
Un valor de argumento negativo significa que la señal de E/S es activada por el sistema de robot en el momento especificado, después de que el TCP físico haya sobrepasado la distancia especificada respecto del punto de inicio o final del movimiento.

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.305 TriggEquip - Define un evento de E/S basado en la posición y el tiempo en la trayectoria RobotWare - OS Continuación

En la figura se muestra el uso del argumento EquipLag.



xx0500002262

[ \DOP ]

### Digital Output

Tipo de dato: signaldo

El nombre de la señal cuando es necesario cambiar una señal digital de salida.

[ \GOP ]

### Group Output

Tipo de dato: signalgo

El nombre de la señal, cuando es necesario cambiar un grupo de señales digitales de salida.

[ \AOP ]

### Analog Output

Tipo de dato: signalao

El nombre de la señal cuando es necesario cambiar una señal analógica de salida.

[ \ProcID ]

### Process Identity

Tipo de dato: num

No implementado para uso del cliente.

(La identidad del proceso IPM que debe recibir el evento. El selector se especifica en el argumento SetValue.)

SetValue

Tipo de dato: num

El valor deseado para la señal (dentro del rango permitido para la señal actual). Si la señal es una señal digital, debe ser un valor entero. Si la señal es una señal digital de grupo, el valor permitido depende del número de señales del grupo. El valor máximo que puede usarse en el argumento SetValue es de 8388608, que es el valor que una señal digital de grupo de 23 bits puede tener como valor máximo (consulte los rangos válidos para num).

SetDvalue

Tipo de dato: dnum

Continúa en la página siguiente



1.305 TriggEquip - Define un evento de E/S basado en la posición y el tiempo en la trayectoria  
*RobotWare - OS*  
*Continuación*

El valor deseado para la señal (dentro del rango permitido para la señal actual). Si la señal es una señal digital, debe ser un valor entero. Si la señal es una señal digital de grupo, el valor permitido depende del número de señales del grupo. El valor máximo de los bits de señales que puede tener una señal digital de grupo es de 32. Con una variable `dnum` es posible cubrir los valores del 0 al 4294967295, que constituyen el rango que puede tener una señal digital de 32 bits.

[ \Inhib ]

***Inhibit***

Tipo de dato: `bool`

El nombre de un indicador de variable persistente para la inhibición del valor de la señal en tiempo de ejecución.

Si se utiliza este argumento opcional y el valor actual del indicador especificado es TRUE en la posición y el momento del establecimiento de la señal analógica, la señal especificada (`DOP`, `GOP` o `AOP`) se cambia a 0 en lugar del valor especificado.

[ \InhibitSetValue ]

***InhibitSetValue***

Tipo de dato: `bool`, `num` or `dnum`

El nombre de una variable persistente de tipo de dato `bool`, `num` o `dnum` o cualquier alias de estos tres tipos de datos.

Este argumento opcional solo puede utilizarse junto con el argumento opcional `Inhib`.

Si se utiliza este argumento opcional y el valor del indicador de variable persistente utilizado en el argumento opcional `Inhib` es TRUE en la posición y tiempo para configurar la señal, se lee el valor de la variable persistente utilizado en el argumento opcional `InhibitSetValue` y el valor se utiliza para configurar la señal `DOP`, `GOP` o `AOP`.

Si se utiliza una variable persistente booleana, el valor TRUE se traduce al valor 1, y FALSE se traduce al valor 0.

[ \Mode ]

Tipo de dato: `triggmode`

Se utiliza para especificar diferentes modos de acción al definir disparadores.

**Ejecución de programas**

Cuando se ejecuta la instrucción `TriggEquip`, la condición de disparo se almacena en una variable especificada para el argumento `TriggData`.

A continuación, cuando se ejecuta una de las instrucciones `TriggL`, `TriggC` o `TriggJ`, se aplica lo siguiente en cuanto a las definiciones de `TriggEquip`:

En la tabla se describe la distancia especificada en el argumento `Distance`:

|                     |                                  |
|---------------------|----------------------------------|
| Movimiento lineal   | La distancia en línea recta      |
| Movimiento circular | La longitud del arco del círculo |

*Continúa en la página siguiente*

# 1 Instrucciones

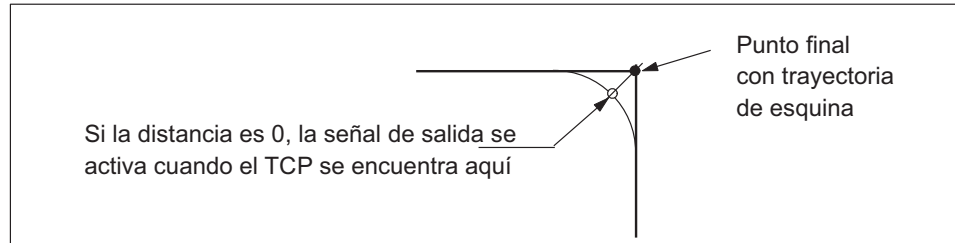
## 1.305 TriggEquip - Define un evento de E/S basado en la posición y el tiempo en la trayectoria

RobotWare - OS

Continuación

|                      |  |
|----------------------|--|
| Movimiento no lineal | La longitud de arco aproximada a lo largo de la trayectoria (para conseguir la exactitud adecuada, la distancia no debe superar la mitad de la longitud del arco). |
|----------------------|--|

La figura muestra la E/S en una posición y un tiempo fijos con una trayectoria de esquina.



xx0500002263

El evento dependiente de la posición y del tiempo se genera cuando se atraviesa el punto de inicio (punto final) si la distancia especificada respecto del punto final (punto de inicio) no se encuentra dentro de la longitud de movimiento de la instrucción actual (`TriggL...`). Cuando se usa el argumento `EquipLag` con un tiempo (retardo) negativo, la señal de E/S puede activarse después del punto final.

### Más ejemplos

A continuación aparecen más ejemplos de cómo usar la instrucción `TriggEquip`.

#### Ejemplo 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;
```

La señal analógica de salida `glue` cambia al valor 5.3 cuando el TCP atraviesa un punto situado a 1 mm después del punto de inicio `p1`, con una compensación de retardo del equipo de 0.05 s.

#### Ejemplo 2

```
...
TriggL p3, v500, glueflow, z50, tool1;
```

La señal analógica de salida `glue` cambia de nuevo al valor 5.3 cuando el TCP atraviesa un punto situado a 1 mm después del punto de inicio `p2`.

### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                  | Causa del error   |
|-------------------------|---|
| <code>ERR_AO_LIM</code> | El argumento programado <code>SetValue</code> para la señal analógica de salida especificada <code>AOp</code> está fuera de límite. |

Continúa en la página siguiente

## 1.305 TriggEquip - Define un evento de E/S basado en la posición y el tiempo en la trayectoria

RobotWare - OS

Continuación

| Nombre             | Causa del error   |
|--------------------|---|
| ERR_GO_LIM         | El argumento programado <code>SetValue</code> o <code>SetDvalue</code> para la señal digital de salida de grupo especificada <code>GOp</code> está fuera de límite.       |
| ERR_NO_ALIASIO_DEF | La variable de señal es una variable declarada en RAPID. No se ha conectado a una señal E/S definida en la configuración de E/S con la instrucción <code>AliasIO</code> . |

### Limitaciones

Los eventos de E/S basados en una distancia se han diseñado para los puntos de paso (trayectorias de esquina). El uso de puntos de paro daría como resultado una exactitud menor que la especificada a continuación.

En cuanto a la exactitud de los eventos de E/S con distancia y con puntos de paso, se aplica lo siguiente a la hora de activar una salida digital a una distancia específica del punto de inicio o del punto final con las instrucciones `TriggL` o `TriggC`:

- La exactitud especificada abajo es válida al utilizar un `EquipLag` positivo es menor que 40 ms, que equivale al retardo del servo del robot sin cambiar el parámetro *Event Preset Time* del sistema. El retardo puede variar de un tipo de robot a otro.
- La precisión especificada abajo es válida cuando se utiliza un `EquipLag` positivo menor que el *Event Preset Time* configurado en los parámetros del sistema.
- La exactitud especificada a continuación no es válida cuando se utiliza un `EquipLag` positivo mayor que el *Event Preset Time* configurado en los parámetros del sistema. En este caso, se utiliza un método aproximado que no tiene en cuenta las limitaciones dinámicas del robot. Entonces, debe usarse `SingArea \Wrist` para conseguir una precisión aceptable.
- La exactitud especificada a continuación es válida cuando se utiliza un `EquipLag` negativo.

Los valores absolutos típicos en cuanto a la exactitud para activar salidas digitales son de:  $\pm 5$  ms.

Los valores repetidos típicos en cuanto a la exactitud para activar salidas digitales son de:  $\pm 2$  ms.

`TriggEquip` no puede ejecutarse en un gestor UNDO o en rutinas de RAPID que estén conectadas a los siguientes eventos especiales del sistema: `PowerOn`, `Stop`, `QStop`, `Restart`, `Reset` o `Step`.

### Sintaxis

```
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> ] ','
```

Continúa en la página siguiente

# 1 Instrucciones

## 1.305 TriggEquip - Define un evento de E/S basado en la posición y el tiempo en la trayectoria

RobotWare - OS

Continuación

```
[ SetValue' :=' ] < expression (IN) of num>  
| [ SetDvalue' :=' ] < expression (IN) of dnum> ','  
[ '\ Inhib' :=' < persistent (PERS) of bool> ]  
[ '\ InhibSetValue' :=' < persistent (PERS) of anytype> ]  
[ '\ Mode' :=' < expression (IN) of trigemode> ] ';' 
```

### Información relacionada

| Para obtener más información sobre                         | Consulte  |
|--|---|
| Utilización de disparadores                                | <a href="#">TriggL - Movimiento lineal del robot con eventos en la página 940</a><br><a href="#">TriggC - Movimiento circular del robot con eventos en la página 895</a><br><a href="#">TriggJ - Movimientos de ejes del robot a partir de eventos en la página 932</a>                     |
| Definición de otros disparos                               | <a href="#">TriggIO - Define un evento de E/S de posición o tiempo fijos cerca de un punto de paro en la página 926</a><br><a href="#">TriggInt - Define una interrupción dependiente de una posición en la página 921</a>  |
| Definición de una comprobación de E/S en una posición fija | <a href="#">TriggCheckIO - Define una comprobación de E/S en una posición fija en la página 904</a>   |
| Almacenamiento de datos de disparo                         | <a href="#">triggdata - Eventos de posicionamiento, trigg en la página 1805</a>   |
| Definición de diferentes modos de acción de disparo        | <a href="#">triggmode - Disparar modo de acción en la página 1811</a>   |
| Establecimiento de E/S                                     | <a href="#">SetDO - Cambia el valor de una señal digital de salida en la página 704</a><br><a href="#">SetGO - Cambia el valor de un grupo de señales digitales de salida en la página 707</a><br><a href="#">SetAO - Cambia el valor de una señal analógica de salida en la página 694</a> |
| Configuración de Event preset time                         | <a href="#">Manual de referencia técnica - Parámetros del sistema</a>   |

## 1.306 TriggInt - Define una interrupción dependiente de una posición

### Utilización

`TriggInt` se utiliza para definir las condiciones y acciones de ejecución de una rutina de interrupción en una posición específica que se encuentra en la trayectoria de movimiento del robot.

Los datos definidos se utilizan para la implementación o instrucciones `TriggL`, `TriggC` o `TriggJ` posteriores.

Esta instrucción sólo puede usarse en la tarea principal `T_ROB1` o, si se cuenta con un sistema `MultiMove`, en las tareas de movimiento.

### Ejemplos básicos

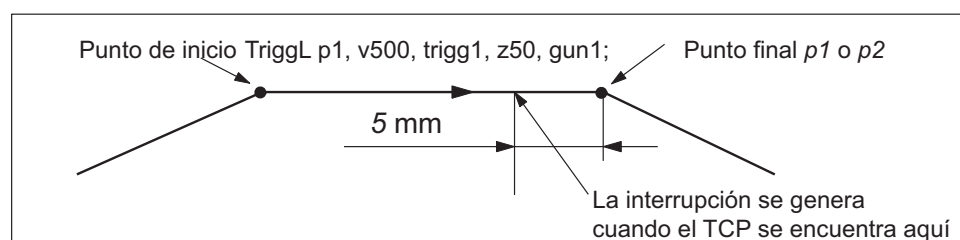
El ejemplo que aparece a continuación ilustra la instrucción `TriggInt`:

#### Ejemplo 1

```
VAR intnum intnol;
VAR triggdata trigg1;
...
PROC main()
  CONNECT intnol WITH trap1;
  TriggInt trigg1, 5, intnol;
  ...
  TriggL p1, v500, trigg1, z50, gun1;
  TriggL p2, v500, trigg1, z50, gun1;
  ...
  IDelete intnol;
```

Se ejecuta la rutina de interrupción `trap1` cuando el TCP se encuentra en una posición a 5 mm antes del punto `p1` o `p2` respectivamente.

En la figura se muestra un ejemplo de interrupción relacionada con una posición.



xx0500002251

### Argumentos

```
TriggInt TriggData Distance [\Start] | [\Next] | [\Time] Interrupt
[\Inhib] [\Mode]
```

#### TriggData

**Tipo de dato:** `triggdata`

Una variable para almacenar el `triggdata` devuelto por la instrucción. Estos datos `triggdata` se utilizan a continuación en las instrucciones `TriggL`, `TriggC` o `TriggJ` posteriores.

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.306 TriggInt - Define una interrupción dependiente de una posición

RobotWare - OS

Continuación

Distance

Tipo de dato: num

Define la posición en la que debe generarse la interrupción dentro de la trayectoria. Se especifica como la distancia en mm (valor positivo) desde el punto final de la trayectoria de movimiento (aplicable si no se utiliza el argumento `\Start` o `\Time`). Consulte [Ejecución de programas en la página 922](#) para obtener más detalles.

[ `\Start` ]

Tipo de dato: switch

Se utiliza cuando la distancia del argumento `Distance` comienza en el punto de inicio del movimiento en lugar del punto final.

[ `\Next` ]

Tipo de dato: switch

Se utiliza cuando la distancia para el argumento `Distance` avanza hacia el siguiente punto programado. Si la `Distance` es mayor que la distancia al siguiente punto fino, el evento se ejecutará en el punto fino.

[ `\Time` ]

Tipo de dato: switch

Se utiliza cuando el valor especificado para el argumento `Distance` es en realidad un tiempo en segundos (valor positivo) en lugar de una distancia. Las interrupciones relacionadas con una posición y basadas en tiempo sólo puede usarse con tiempos breves (< 0,5 s) antes de que el robot alcance el punto final de la instrucción. Consulte la sección *Limitaciones* para obtener más detalles.

Interrupt

Tipo de dato: intnum

La variable utilizada para identificar una interrupción.

[ `\Inhib` ]

*Inhibit*

Tipo de dato: bool

El nombre de un indicador de variable persistente para la inhibición de la ejecución de la rutina de interrupción.

Si se utiliza este argumento opcional y el valor real del indicador especificado es TRUE en la posición y el tiempo de la ejecución de la interrupción, la interrupción no se ejecuta.

[ `\Mode` ]

Tipo de dato: triggmode

Se utiliza para especificar diferentes modos de acción al definir disparadores.

---

### Ejecución de programas

Cuando se ejecuta la instrucción `TriggInt`, la información se almacena en una variable especificada para el argumento `TriggData` y se activa la interrupción especificada en la variable del argumento `Interrupt`.

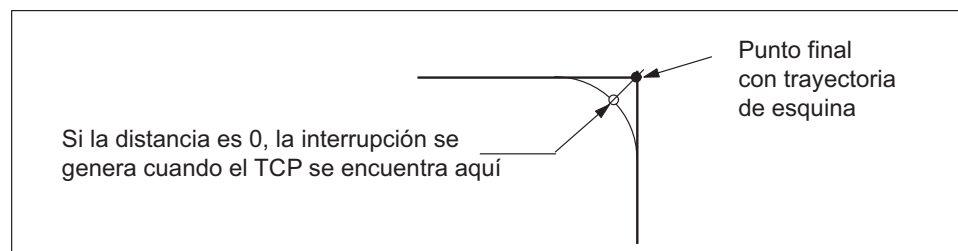
*Continúa en la página siguiente*

A continuación, cuando se ejecuta una de las instrucciones `TriggL`, `TriggC` o `TriggJ`, se aplica lo siguiente en cuanto a las definiciones de `TriggInt`:

En la tabla se describe la distancia especificada en el argumento `Distance`:

|                      |  |
|----------------------|--|
| Movimiento lineal    | La distancia en línea recta  |
| Movimiento circular  | La longitud del arco del círculo   |
| Movimiento no lineal | La longitud de arco aproximada a lo largo de la trayectoria (para conseguir la exactitud adecuada, la distancia no debe superar la mitad de la longitud del arco). |

La figura muestra una interrupción relacionada con una posición de una trayectoria de esquina.



xx0500002253

La interrupción relacionada con la posición se genera cuando se atraviesa el punto de inicio (punto final) si la distancia especificada respecto del punto final (punto de inicio) no se encuentra dentro de la longitud de movimiento de la instrucción actual (`TriggL...`).

Se considera que la interrupción es una interrupción segura. Una interrupción segura no puede ponerse en reposo con la instrucción `ISleep`. El evento de interrupción segura se coloca en la cola en caso de paro del programa y de ejecución paso a paso. La interrupción se ejecuta al iniciar nuevamente el modo continuo. El único momento en el que una interrupción segura se desecha es cuando la cola de interrupciones está llena. En este caso se genera un error. La interrupción no sobrevive al restablecimiento del programa, por ejemplo, `PP` a `main`.

### Más ejemplos

A continuación aparecen más ejemplos de cómo usar la instrucción `TriggInt`.

#### Ejemplo 1

En este ejemplo se describe la programación de las instrucciones que interactúan para generar interrupciones dependientes de una posición:

```
VAR intnum intno2;
```

```
VAR triggdata trigg2;
```

- Declaración de las variables `intno2` y `trigg2` ( no se inicia).

```
CONNECT intno2 WITH trap2;
```

- Asignación de números de interrupción almacenados en la variable `intno2`.
- El número de interrupción está asociado a la rutina de interrupción `trap2`.

```
TriggInt trigg2, 0, intno2;
```

Continúa en la página siguiente

# 1 Instrucciones

---

## 1.306 TriggInt - Define una interrupción dependiente de una posición

RobotWare - OS

Continuación

- El número de interrupción de la variable `intno2` se marca como utilizado.
- Se activa la interrupción.
- Las condiciones de disparo y el número de interrupción definidos se almacenan en la variable `trigg2`

```
TriggL p1, v500, trigg2, z50, gun1;
```

- El robot se mueve hacia el punto `p1`.
- Cuando el TCP alcanza el punto `p1`, se genera una interrupción y se ejecuta la rutina de interrupción `trap2`.

```
TriggL p2, v500, trigg2, z50, gun1;
```

- El robot se mueve hacia el punto `p2`.
- Cuando el TCP alcanza el punto `p2`, se genera una interrupción y se ejecuta la rutina de interrupción `trap2` de nuevo.

```
IDelete intno2;
```

- Se anula la asignación del número de interrupción de la variable `intno2`.

---

## Limitaciones

Los eventos de interrupción basados en una distancia (sin el argumento `\Time`) se han diseñado para los puntos de paso (trayectorias de esquina). Los eventos de interrupción basados en distancia, con puntos de paro, presentan una exactitud menor que la especificada a continuación.

Los eventos de interrupción basados en tiempo (con el argumento `\Time`) se han diseñado para los puntos de paro. Los eventos de interrupción basados en tiempo, con puntos de paso, presentan una exactitud menor que la especificada a continuación. Los eventos de E/S basados en tiempo sólo pueden especificarse a partir del punto final del movimiento. Este tiempo no puede superar el tiempo de frenado actual del robot, que tiene un máximo aproximado de 0,5 s (valores típicos a una velocidad de 500 mm/s: para IRB 2400 es 150 ms, y para IRB 6400 es 250 ms). Si el tiempo especificado es mayor que el tiempo de frenado actual, el evento se genera en cualquier caso, pero no hasta que se inicia el frenado (más tarde de lo especificado). La totalidad del tiempo de movimiento para el movimiento actual puede utilizarse durante los movimientos pequeños y rápidos.

Los valores absolutos típicos en cuanto a la exactitud de la generación de interrupciones son de +/- 5 ms. Los valores típicos en cuanto a la repetición la generación de interrupciones son de +/- 2 ms. Normalmente, existe un retardo de 2 a 30 ms entre la generación de interrupciones y la respuesta, en función del tipo de movimiento que se realiza en el momento de la interrupción. Consulte *Manual de referencia técnica - RAPID Overview*.

Para obtener la mayor exactitud al activar una salida en una posición fija a lo largo de la trayectoria del robot, utilice preferentemente las instrucciones `TriggIO` o `TriggEquip` en lugar de las instrucciones `TriggInt` con `SetDO/SetGO/SetAO` en rutinas de interrupción.

`TriggInt` no puede ejecutarse en un gestor UNDO o en rutinas de RAPID que estén conectadas a los siguientes eventos especiales del sistema: `PowerOn`, `Stop`, `QStop`, `Restart`, `Reset` o `Step`.

Continúa en la página siguiente



**Sintaxis**

```

TriggInt
  [TriggData '[:=' ] <variable (VAR) of triggdata>', '
  [Distance' '[:=' ] <expression (IN) of num>
  ['\ ' Start] | ['\ ' Next ] | ['\ ' Time]','
  [Interrupt '[:=' ] <variable (VAR) of intnum>
  ['\ ' Inhib '[:=' ] <persistent (PERS) of bool>]
  ['\ ' Mode '[:=' ] <expression (IN) of triggmode>'];'

```

**Información relacionada**

| Para obtener más información sobre                         | Consulte  |
|--|---|
| Utilización de disparadores                                | <a href="#">TriggL - Movimiento lineal del robot con eventos en la página 940</a><br><a href="#">TriggC - Movimiento circular del robot con eventos en la página 895</a><br><a href="#">TriggJ - Movimientos de ejes del robot a partir de eventos en la página 932</a> |
| Definición de E/S de posición fija                         | <a href="#">TriggIO - Define un evento de E/S de posición o tiempo fijos cerca de un punto de paro en la página 926</a><br><a href="#">TriggEquip - Define un evento de E/S basado en la posición y el tiempo en la trayectoria en la página 914</a>                    |
| Definición de una comprobación de E/S en una posición fija | <a href="#">TriggCheckIO - Define una comprobación de E/S en una posición fija en la página 904</a>   |
| Almacenamiento de datos de disparo                         | <a href="#">triggdata - Eventos de posicionamiento, trigg en la página 1805</a>   |
| Definición de diferentes modos de acción de disparo        | <a href="#">triggmode - Disparar modo de acción en la página 1811</a>   |
| Interrupciones   | <a href="#">Manual de referencia técnica - RAPID Overview</a>   |

## 1 Instrucciones

1.307 TriggIO - Define un evento de E/S de posición o tiempo fijos cerca de un punto de paro  
*RobotWare - OS*

### 1.307 TriggIO - Define un evento de E/S de posición o tiempo fijos cerca de un punto de paro

#### Utilización

TriggIO se utiliza para definir condiciones y acciones para el establecimiento de una señal digital, un grupo de señales digitales o una señal analógica de salida en una posición fija a lo largo de la trayectoria de movimiento del robot.

Siempre debe usarse TriggIO (no TriggEquip) si se necesita una buena exactitud de los ajustes de E/S cerca de un punto de paro.

Para obtener un evento de E/S en una posición determinada, TriggIO compensa el retardo del sistema de control (el retardo entre el robot y el servo), pero no los retardos de los equipos externos. Para la compensación de los dos retardos, utilice TriggEquip.

Los datos definidos se utilizan para la implementación o instrucciones TriggL, TriggC o TriggJ posteriores.

Esta instrucción sólo puede usarse en la tarea principal T\_ROB1 o, si se cuenta con un sistema *MultiMove*, en las tareas de movimiento.

#### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción TriggIO:

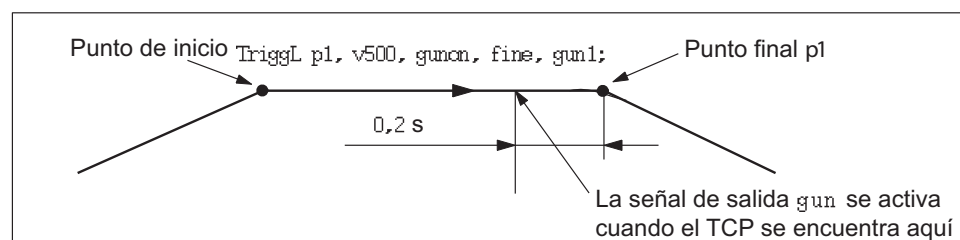
Consulte también [Más ejemplos en la página 929](#).

#### Ejemplo 1

```
VAR triggdata gunon;  
...  
TriggIO gunon, 0.2\Time\DOp:=gun, 1;  
TriggL p1, v500, gunon, fine, gun1;
```

La señal digital de salida `gun` cambia al valor 1 cuando el TCP se encuentra a 0,2 s antes del punto `p1`.

En la figura se muestra un ejemplo de evento de E/S basado en una posición fija.



xx0500002247

#### Argumentos

```
TriggIO TriggData Distance [\Start] | [\Time] [\DOp] | [\GOp] |  
[\AOp] | [\ProcID] SetValue | SetDvalue [\DODelay] [\Inhib]  
[\InhibSetValue] [\Mode]
```

TriggData

Tipo de dato: triggdata

*Continúa en la página siguiente*

## 1.307 TriggIO - Define un evento de E/S de posición o tiempo fijos cerca de un punto de paro RobotWare - OS Continuación

Una variable para almacenar el `triggdata` devuelto por la instrucción. Estos datos `triggdata` se utilizan a continuación en las instrucciones `TriggL`, `TriggC` o `TriggJ` posteriores.

`Distance`

Tipo de dato: `num`

Define la posición de la trayectoria en la que debe producirse el evento de E/S. Se especifica como la distancia en mm (valor positivo) desde el punto final de la trayectoria de movimiento (aplicable si no se utiliza el argumento `\Start` o `\Time`).

Consulte las secciones tituladas [Ejecución de programas en la página 929](#) y [Limitaciones en la página 930](#) para obtener más detalles.

[ `\Start` ]

Tipo de dato: `switch`

Se utiliza cuando la distancia del argumento `Distance` comienza en el punto de inicio del movimiento en lugar del punto final.

[ `\Time` ]

Tipo de dato: `switch`

Se utiliza cuando el valor especificado para el argumento `Distance` es en realidad un tiempo en segundos (valor positivo) en lugar de una distancia.

La posición fija de E/S basada en tiempo sólo puede usarse con tiempos breves (< 0,5 s) antes de que el robot alcance el punto final de la instrucción. Consulte la sección Limitaciones para obtener más detalles.

[ `\DOp` ]

**Digital Output**

Tipo de dato: `signaldo`

El nombre de la señal cuando es necesario cambiar una señal digital de salida.

[ `\GOP` ]

**Group Output**

Tipo de dato: `signalgo`

El nombre de la señal, cuando es necesario cambiar un grupo de señales digitales de salida.

[ `\AOp` ]

**Analog Output**

Tipo de dato: `signalao`

El nombre de la señal cuando es necesario cambiar una señal analógica de salida.

[ `\ProcID` ]

**Process Identity**

Tipo de dato: `num`

No implementado para uso del cliente.

(La identidad del proceso IPM que debe recibir el evento. El selector se especifica en el argumento `SetValue`.)

*Continúa en la página siguiente*

## 1 Instrucciones

---

1.307 TriggIO - Define un evento de E/S de posición o tiempo fijos cerca de un punto de paro

RobotWare - OS

Continuación

SetValue

Tipo de dato: num

El valor deseado para la señal (dentro del rango permitido para la señal actual). Si la señal es una señal digital, debe ser un valor entero. Si la señal es una señal digital de grupo, el valor permitido depende del número de señales del grupo. El valor máximo que puede usarse en el argumento SetValue es de 8388608, que es el valor que una señal digital de grupo de 23 bits puede tener como valor máximo (consulte los rangos válidos para num).

SetDvalue

Tipo de dato: dnum

El valor deseado para la señal (dentro del rango permitido para la señal actual). Si la señal es una señal digital, debe ser un valor entero. Si la señal es una señal digital de grupo, el valor permitido depende del número de señales del grupo. El valor máximo de los bits de señales que puede tener una señal digital de grupo es de 32. Con una variable dnum es posible cubrir los valores del 0 al 4294967295, que constituyen el rango que puede tener una señal digital de 32 bits.

[ \DODelay ]

*Digital Output Delay*

Tipo de dato: num

El retardo de tiempo en segundos (valor positivo) de una salida digital, un grupo de señales digitales o una salida analógica.

Sólo se usa para la definición de retardos en señales digitales una vez que el robot ha alcanzado la posición especificada. Si se omite este argumento, no se utilizará ningún retardo.

El retardo no está sincronizado con el movimiento.

[ \Inhib ]

*Inhibit*

Tipo de dato: bool

El nombre de un indicador de variable persistente para la inhibición del valor de la señal en tiempo de ejecución.

Si se utiliza este argumento opcional y el valor actual del indicador especificado es TRUE en la posición y el momento del establecimiento de la señal analógica, la señal especificada (DOP, GOP o AOP) se cambia a 0 en lugar del valor especificado.

[ \InhibSetValue ]

*InhibitSetValue*

Tipo de dato: bool, num or dnum

El nombre de una variable persistente de tipo de dato bool, num o dnum o cualquier alias de estos tres tipos de datos.

Este argumento opcional solo puede utilizarse junto con el argumento opcional Inhib.

Si se utiliza este argumento opcional y el valor del indicador de variable persistente utilizado en el argumento opcional Inhib es TRUE en la posición y tiempo para

Continúa en la página siguiente

## 1.307 TriggIO - Define un evento de E/S de posición o tiempo fijos cerca de un punto de paro

*RobotWare - OS*

*Continuación*

configurar la señal, se lee el valor de la variable persistente utilizado en el argumento opcional `InhibSetValue` y el valor se utiliza para configurar la señal `DOp`, `GOp` o `AOp`.

Si se utiliza una variable persistente booleana, el valor `TRUE` se traduce al valor 1, y `FALSE` se traduce al valor 0.

[ \Mode ]

Tipo de dato: `triggmode`

Se utiliza para especificar diferentes modos de acción al definir disparadores.

### Ejecución de programas

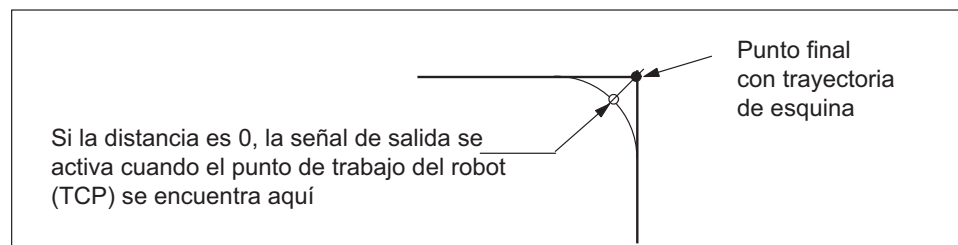
Cuando se ejecuta la instrucción `TriggIO`, la condición de disparo se almacena en una variable especificada en el argumento `TriggData`.

A continuación, cuando se ejecuta una de las instrucciones `TriggL`, `TriggC` o `TriggJ`, se aplica lo siguiente en cuanto a las definiciones de `TriggIO`:

En la tabla siguiente se describe la distancia especificada en el argumento `Distance`:

|                      |  |
|----------------------|--|
| Movimiento lineal    | La distancia en línea recta  |
| Movimiento circular  | La longitud del arco del círculo   |
| Movimiento no lineal | La longitud de arco aproximada a lo largo de la trayectoria (para conseguir la exactitud adecuada, la distancia no debe superar la mitad de la longitud del arco). |

La figura muestra la E/S en una posición fija con una trayectoria de esquina.



xx0500002248

La E/S con una posición fija se genera cuando se atraviesa el punto de inicio (punto final) si la distancia especificada respecto del punto final (punto de inicio) no se encuentra dentro de la longitud de movimiento de la instrucción actual (`Trigg...`).

### Más ejemplos

A continuación aparecen más ejemplos de cómo usar la instrucción `TriggIO`.

#### Ejemplo 1

```
VAR triggdata glueflow;

TriggIO glueflow, 1 \Start \AOp:=glue, 5.3;

MoveJ p1, v1000, z50, tool1;
TriggL p2, v500, glueflow, z50, tool1;
```

*Continúa en la página siguiente*

## 1 Instrucciones

### 1.307 TriggIO - Define un evento de E/S de posición o tiempo fijos cerca de un punto de paro

RobotWare - OS

Continuación

La señal analógica de salida `glue` cambia al valor `5.3` cuando el punto de trabajo (TCP) atraviesa un punto situado `1 mm` después del punto de inicio `p1`.

#### Ejemplo 2

```
...  
TriggL p3, v500, glueflow, z50, tool1;
```

La señal analógica de salida `glue` cambia de nuevo al valor `5.3` cuando el punto de (TCP) trabajo atraviesa un punto situado a `1 mm` después del punto de inicio `p2`.

#### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                          | Causa del error   |
|---------------------------------|---|
| <code>ERR_AO_LIM</code>         | El argumento programado <code>SetValue</code> para la señal analógica de salida especificada <code>AOp</code> está fuera de límite.   |
| <code>ERR_GO_LIM</code>         | El argumento programado <code>SetValue</code> o <code>SetDvalue</code> para la señal digital de salida de grupo especificada <code>GOp</code> está fuera de límite.                     |
| <code>ERR_NO_ALIASIO_DEF</code> | La variable de señal es una variable declarada en <code>RAPID</code> . No se ha conectado a una señal E/S definida en la configuración de E/S con la instrucción <code>AliasIO</code> . |

#### Limitaciones

Los eventos de E/S basados en una distancia (sin el argumento `\Time`) se han diseñado para los puntos de paso (trayectorias de esquina). Los eventos de E/S con el parámetro `distance=0` y que utilizan puntos de paro retardarán el disparo hasta que el robot haya alcanzado el punto con una exactitud de  $\pm 24$  ms.

Los eventos de E/S basados en tiempo (con el argumento `\Time`) se han diseñado para los puntos de paro. Los eventos de E/S basados en tiempo, con puntos de paso, presentan una exactitud menor que la especificada a continuación. Los eventos de E/S basados en tiempo sólo pueden especificarse a partir del punto final del movimiento. Este tiempo no puede superar el tiempo de frenado actual del robot, que tiene un máximo aproximado de `0,5 s` (valores típicos a una velocidad de `500 mm/s`: para `IRB 2400` es `150 ms`, y para `IRB 6400` es `250 ms`). Si el tiempo especificado es mayor que el tiempo de frenado actual, el evento se genera en cualquier caso, pero no hasta que se inicia el frenado (más tarde de lo especificado). La totalidad del tiempo de movimiento para el movimiento actual puede utilizarse durante los movimientos pequeños y rápidos.

Los valores absolutos típicos en cuanto a la exactitud al activar salidas digitales es de  $\pm 5$  ms. Los valores absolutos típicos en cuanto a la repetición al activar salidas digitales es de  $\pm 2$  ms.

`TriggIO` no puede ejecutarse en un gestor `UNDO` o en rutinas de `RAPID` que estén conectadas a los siguientes eventos especiales del sistema: `PowerOn`, `Stop`, `QStop`, `Restart`, `Reset` o `Step`.

Continúa en la página siguiente

## 1.307 TriggIO - Define un evento de E/S de posición o tiempo fijos cerca de un punto de paro

RobotWare - OS  
Continuación

### Sintaxis

```
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>'];'
```

### Información relacionada

| Para obtener más información sobre                              | Consulte  |
|---|---|
| Utilización de disparadores                                     | <a href="#">TriggL - Movimiento lineal del robot con eventos en la página 940</a><br><a href="#">TriggC - Movimiento circular del robot con eventos en la página 895</a><br><a href="#">TriggJ - Movimientos de ejes del robot a partir de eventos en la página 932</a>                     |
| Definición de eventos de E/S basados en la posición y el tiempo | <a href="#">TriggEquip - Define un evento de E/S basado en la posición y el tiempo en la trayectoria en la página 914</a>   |
| Definición de interrupciones basadas en la posición             | <a href="#">TriggInt - Define una interrupción dependiente de una posición en la página 921</a>   |
| Almacenamiento de datos de disparo                              | <a href="#">triggdata - Eventos de posicionamiento, trigg en la página 1805</a>   |
| Definición de diferentes modos de acción de disparo             | <a href="#">triggmode - Disparar modo de acción en la página 1811</a>   |
| Definición de una comprobación de E/S en una posición fija      | <a href="#">TriggCheckIO - Define una comprobación de E/S en una posición fija en la página 904</a>   |
| Establecimiento de E/S  | <a href="#">SetDO - Cambia el valor de una señal digital de salida en la página 704</a><br><a href="#">SetGO - Cambia el valor de un grupo de señales digitales de salida en la página 707</a><br><a href="#">SetAO - Cambia el valor de una señal analógica de salida en la página 694</a> |

# 1 Instrucciones

## 1.308 TriggJ - Movimientos de ejes del robot a partir de eventos RobotWare - OS

### 1.308 TriggJ - Movimientos de ejes del robot a partir de eventos

#### Utilización

TriggJ (*TriggJoint*) se utiliza para establecer señales de salida y/o ejecutar rutinas de interrupción en posiciones fijas aproximadas, al mismo tiempo que el robot se mueve rápidamente de un punto a otro cuando no es imprescindible que ese movimiento siga una línea recta.

Es posible definir uno o varios eventos (hasta un máximo de 25) mediante las instrucciones TriggIO, TriggEquip, TriggInt, TriggCheckIO, TriggSpeed, o TriggRampAO y hacer referencia posteriormente a estas definiciones en la instrucción TriggJ.

Esta instrucción sólo puede usarse en la tarea principal T\_ROB1 o, si se cuenta con un sistema *MultiMove*, en las tareas de movimiento.

#### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción TriggJ:

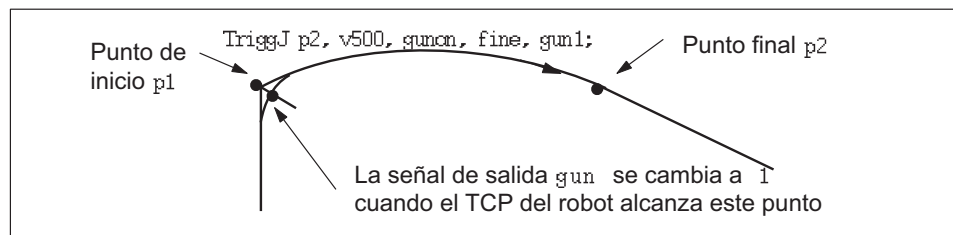
Consulte también [Más ejemplos en la página 937](#).

#### Ejemplo 1

```
VAR triggdata gunon;  
...  
TriggIO gunon, 0 \Start \DOp:=gun, 1;  
MoveL p1, v500, z50, gun1;  
TriggJ p2, v500, gunon, fine, gun1;
```

La señal digital de salida `gun` se activa cuando el TCP del robot atraviesa el punto central de la trayectoria de esquina del punto `p1`.

En la figura se muestra un ejemplo de evento de E/S basado en una posición fija.



xx0500002272

#### Argumentos

```
TriggJ [\Conc] ToPoint [\ID] Speed [\T] Trigg_1 | TriggArray [ \T2  
] [ \T3 ] [\T4] [\T5] [\T6] [\T7] [\T8] Zone [\Inpos] Tool  
[\WObj] [\TLoad]
```

[ \Conc ]

**Concurrent**

Tipo de dato:switch

Distintas instrucciones consecutivas se ejecutan mientras el robot está en movimiento. Este argumento puede usarse para evitar los paros no deseados,

*Continúa en la página siguiente*



## 1.308 TriggJ - Movimientos de ejes del robot a partir de eventos RobotWare - OS Continuación

causados por la sobrecarga de la CPU, cuando se utilizan puntos de paso. Esto resulta útil cuando los puntos programados están muy cercanos entre sí y se trabaja a velocidades elevadas.

Este argumento también resulta útil si, por ejemplo, no se requiere la comunicación con equipos externos ni la sincronización entre los equipos externos y los movimientos del robot. También puede usarse para ajustar la ejecución de la trayectoria del robot, para evitar el aviso 50024 Fallo en punto de paso o el error 50082 Limite de deceleración.

Cuando se utiliza el argumento `\Conc`, el número de instrucciones de movimiento seguidas está limitado a 5. En secciones de programa que incluyen `StorePath-RestoPath`, no se permite el uso de instrucciones con el argumento `\Conc`.

Si se omite este argumento, la instrucción posterior se ejecuta una vez que el robot ha alcanzado el punto de paro especificado, o 100 ms antes de la zona especificada.

Este argumento no puede usarse en los movimientos sincronizados coordinados en un sistema `MultiMove`.

ToPoint

Tipo de dato: `robtarget`

El punto de destino de los ejes del robot y de los ejes externos. Se define como una posición con nombre o se almacena directamente en la instrucción (marcada con un asterisco \* en la instrucción).

[ \ID ]

*Synchronization id*

Tipo de dato: `identno`

El argumento [ \ID ] es obligatorio en los sistemas `MultiMove`, si el movimiento es sincronizado o sincronizado coordinado. Este argumento no está permitido en ningún otro caso. El número de ID especificado debe ser el mismo en todas las tareas de programa que cooperan entre sí. Al usar el número de ID los movimientos no se mezclan en tiempo de ejecución.

Speed

Tipo de dato: `speeddata`

Los datos de velocidad que se aplican a los movimientos. Los datos de velocidad definen la velocidad del punto central de la herramienta, la reorientación de la misma y los ejes externos.

[ \T ]

*Time*

Tipo de dato: `num`

Este argumento se utiliza para especificar el tiempo total en segundos que dura el movimiento del robot. A continuación, se sustituye por los datos de velocidad correspondientes.

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.308 TriggJ - Movimientos de ejes del robot a partir de eventos

RobotWare - OS

Continuación

Trigg\_1

**Tipo de dato:** triggdata

La variable que hace referencia a las condiciones de disparo y la actividad de disparo, definida anteriormente en el programa usando las instrucciones TriggIO, TriggEquip, TriggInt, TriggCheckIO, TriggSpeed o TriggRampAO.

TriggArray

**Trigg Data Array Parameter**

**Tipo de dato:** triggdata

La variable de matriz que hace referencia a las condiciones de disparo y la actividad de disparo, definidas anteriormente en el programa usando las instrucciones TriggIO, TriggEquip, TriggInt, TriggSpeed, TriggCheckIO o TriggRampAO.

La matriz presenta una limitación de 25 elementos y es necesario definir de 1 a 25 condiciones de disparo.

No es posible utilizar los argumentos opcionales T2, T3, T4, T5, T6, T7 o T8 al mismo tiempo que se utiliza el argumento TriggArray.

[ \T2 ]

**Trigg 2**

**Tipo de dato:** triggdata

La variable que hace referencia a las condiciones de disparo y la actividad de disparo, definida anteriormente en el programa usando las instrucciones TriggIO, TriggEquip, TriggInt, TriggCheckIO, TriggSpeed o TriggRampAO.

[ \T3 ]

**Trigg 3**

**Tipo de dato:** triggdata

La variable que hace referencia a las condiciones de disparo y la actividad de disparo, definida anteriormente en el programa usando las instrucciones TriggIO, TriggEquip, TriggInt, TriggCheckIO, TriggSpeed o TriggRampAO.

[ \T4 ]

**Trigg 4**

**Tipo de dato:** triggdata

La variable que hace referencia a las condiciones de disparo y la actividad de disparo, definida anteriormente en el programa usando las instrucciones TriggIO, TriggEquip, TriggInt, TriggCheckIO, TriggSpeed o TriggRampAO.

[ \T5 ]

**Trigg 5**

**Tipo de dato:** triggdata

La variable que hace referencia a las condiciones de disparo y la actividad de disparo, definida anteriormente en el programa usando las instrucciones TriggIO, TriggEquip, TriggInt, TriggCheckIO, TriggSpeed o TriggRampAO.

*Continúa en la página siguiente*

[ \T6 ]

### *Trigg 6*

Tipo de dato: `triggdata`

La variable que hace referencia a las condiciones de disparo y la actividad de disparo, definida anteriormente en el programa usando las instrucciones `TriggIO`, `TriggEquip`, `TriggInt`, `TriggCheckIO`, `TriggSpeed` o `TriggRampAO`.

[ \T7 ]

### *Trigg 7*

Tipo de dato: `triggdata`

La variable que hace referencia a las condiciones de disparo y la actividad de disparo, definida anteriormente en el programa usando las instrucciones `TriggIO`, `TriggEquip`, `TriggInt`, `TriggCheckIO`, `TriggSpeed` o `TriggRampAO`.

[ \T8 ]

### *Trigg 8*

Tipo de dato: `triggdata`

La variable que hace referencia a las condiciones de disparo y la actividad de disparo, definida anteriormente en el programa usando las instrucciones `TriggIO`, `TriggEquip`, `TriggInt`, `TriggCheckIO`, `TriggSpeed` o `TriggRampAO`.

Zone

Tipo de dato: `zonedata`

Los datos de zona del movimiento. Los datos de zona describen el tamaño de la trayectoria de esquina generada.

[ \Inpos ]

### *In position*

Tipo de dato: `stoppointdata`

Este argumento se utiliza para especificar los criterios de convergencia para la posición del TCP del robot en el punto de paro. Los datos de puntos de paro sustituyen a la zona especificada en el parámetro `Zone`.

Tool

Tipo de dato: `tooldata`

La herramienta en uso durante el movimiento del robot. El punto central de la herramienta es el punto que se mueve hacia la posición de destino especificada.

[ \WObj ]

### *Work Object*

Tipo de dato: `wobjdata`

El objeto de trabajo (sistema de coordenadas) con el que está relacionada la posición de robot indicada en la instrucción.

Es posible omitir este argumento. Si se omite, la posición depende del sistema de coordenadas mundo. Si por otro lado se usa un TCP estacionario o ejes externos

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.308 TriggJ - Movimientos de ejes del robot a partir de eventos

RobotWare - OS

Continuación

coordinados, es necesario especificar este argumento para ejecutar un movimiento de ejes respecto del objeto de trabajo.

[ \TLoad ]

### *Total load*

Tipo de dato: loaddata

El argumento \TLoad describe la carga total usada durante el movimiento. La carga total es la carga de la herramienta más la carga útil transportada por la herramienta. Si se utiliza el argumento \TLoad, no se tiene en cuenta el valor de loaddata en los tooldata actuales.

Si el argumento \TLoad tiene el valor load0, el argumento \TLoad no se tiene en cuenta y se utilizan en su lugar los loaddata de los tooldata.

Para poder utilizar el argumento \TLoad, es necesario cambiar el valor del parámetro de sistema ModalPayloadMode a 0. Si ModalPayloadMode tiene el valor 0, ya no es posible utilizar la instrucción GripLoad.

La carga total puede identificarse con la rutina de servicio LoadIdentify. Si el parámetro de sistema ModalPayloadMode tiene el valor 0, el operador tiene la posibilidad de copiar los loaddata de la herramienta a una variable persistente loaddata existente o nueva al ejecutar la rutina de servicio.

Es posible realizar una ejecución de prueba del programa sin ninguna carga útil utilizando una señal de entrada digital conectada a la entrada de sistema SimMode (modo simulado). Si la señal de entrada digital tiene el valor 1, los loaddata del argumento opcional \TLoad no se tienen en cuenta y se utilizan en su lugar los loaddata de los tooldata actuales.



### **Nota**

La funcionalidad predeterminada de manejo de la carga útil es utilizar la instrucción GripLoad. Por tanto, el valor predeterminado del parámetro de sistema ModalPayloadMode es 1.

---

## Ejecución de programas

Consulte la instrucción MoveJ para obtener más información acerca del movimiento de ejes.

A medida que se cumplen las condiciones de disparo cuando el robot se sitúa más y más cerca del punto final, se realizan las actividades de disparo definidas. Las condiciones de disparo se cumplen a una distancia determinada del punto final de la instrucción o a una distancia determinada del punto de inicio de la instrucción, o bien en un momento determinado (limitado a un tiempo breve) antes del punto final de la instrucción.

Durante la ejecución paso a paso hacia delante, las actividades de E/S se realizan pero las rutinas de interrupción no se ejecutan. Durante la ejecución paso a paso hacia atrás, no se realiza ninguna actividad de disparo.

*Continúa en la página siguiente*

**Más ejemplos**

A continuación aparecen más ejemplos de cómo usar la instrucción TriggJ.

**Ejemplo 1**

```
VAR intnum intnol;
VAR triggdata trigg1;
...
PROC main()
  CONNECT intnol WITH trap1;
  TriggInt trigg1, 0.1 \Time, intnol;
  ...
  TriggJ p1, v500, trigg1, fine, gun1;
  TriggJ p2, v500, trigg1, fine, gun1;
  ...
  IDelete intnol;
```

Se ejecuta la rutina de interrupción trap1 cuando el punto de trabajo se encuentra en una posición 0,1 s antes del punto de paro p1 o p2, según corresponda.

**Ejemplo 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;
  ...
```

Las señales de salida digitales de la do1 a la do25 se activan durante el movimiento a p1. La distancia entre los valores de las señales es de 10 mm.

**Gestión de errores**

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema ERRNO cambiará a:

| Nombre     | Causa del error   |
|------------|---|
| ERR_AO_LIM | El argumento ScaleValue programado para la señal analógica de salida especificada AOp en algunas de las instrucciones TriggSpeed conectadas da lugar a la salida de límite de la señal analógica junto con la Speed programada en esta instrucción. |

Continúa en la página siguiente

# 1 Instrucciones

## 1.308 TriggJ - Movimientos de ejes del robot a partir de eventos

RobotWare - OS

Continuación

| Nombre         | Causa del error   |
|----------------|---|
| ERR_DIPLAG_LIM | El argumento <code>DipLag</code> programado en algunas de las instrucciones <code>TriggSpeed</code> conectadas es demasiado grande con respecto al tiempo preestablecido para los eventos en parámetros del sistema.                    |
| ERR_NORUNUNIT  | No hay ningún contacto con el dispositivo de E/S al introducir la instrucción, y el <code>triggdata</code> utilizado depende de un dispositivo de E/S en funcionamiento, es decir, que se utiliza una señal en <code>triggdata</code> . |
| ERR_CONC_MAX   | Se ha superado el número de instrucciones de movimiento seguidas con el argumento <code>\Conc</code> .  |

### Limitaciones

Si el punto de inicio actual se desvía del habitual, de forma que la longitud de posicionamiento total de la instrucción `TriggJ` es más corta de lo habitual (es decir, al principio de `TriggJ` con la posición del robot en el punto final), puede ocurrir que se cumplan varias de las condiciones de disparo, o incluso todas ellas, inmediatamente y en una misma posición. En estos casos, la secuencia en la que se realizan las actividades de disparo no estará definida. La lógica de programa del programa del usuario no puede basarse en una secuencia normal de actividades de disparo para un movimiento incompleto.

`TriggJ` no puede ejecutarse en un gestor UNDO o en rutinas de RAPID que estén conectadas a los siguientes eventos especiales del sistema: `PowerOn`, `Stop`, `QStop`, `Restart`, `Reset` o `Step`.

### Sintaxis

```
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 > ] ; '
```

Continúa en la página siguiente

### Información relacionada

| Para obtener más información sobre  | Consulte   |
|---|--|
| Movimiento lineal con disparadores  | <a href="#">TriggL - Movimiento lineal del robot con eventos en la página 940</a>  |
| Movimiento circular con disparadores  | <a href="#">TriggC - Movimiento circular del robot con eventos en la página 895</a>  |
| Definición de disparadores  | <a href="#">TriggIO - Define un evento de E/S de posición o tiempo fijos cerca de un punto de paro en la página 926</a><br><a href="#">TriggEquip - Define un evento de E/S basado en la posición y el tiempo en la trayectoria en la página 914</a><br><a href="#">TriggRampAO - Define un evento AO de rampa de posición fija en la trayectoria en la página 965</a><br><a href="#">TriggInt - Define una interrupción dependiente de una posición en la página 921</a><br><a href="#">TriggCheckIO - Define una comprobación de E/S en una posición fija en la página 904</a> |
| Manejo de <code>triggdata</code>  | <a href="#">triggdata - Eventos de posicionamiento, trigg en la página 1805</a><br><a href="#">TriggDataReset - Restablecer el contenido en una variable de tipo <code>triggdata</code> en la página 912</a><br><a href="#">TriggDataCopy - Copiar el contenido de una variable de tipo <code>triggdata</code> en la página 910</a><br><a href="#">TriggDataValid - Comprobar si el contenido de una variable de tipo <code>triggdata</code> es válido en la página 1537</a>   |
| Movimiento del robot mediante un movimiento de ejes   | <a href="#">MoveJ - Mueve el robot mediante un movimiento de ejes en la página 421</a>   |
| Movimiento de ejes  | <i>Manual de referencia técnica - RAPID Overview</i>   |
| Definición de carga   | <a href="#">loaddata - Datos de carga en la página 1694</a>  |
| Definición de velocidad   | <a href="#">speeddata - Datos de velocidad en la página 1767</a>   |
| Definición de datos de punto de paro  | <a href="#">stoppointdata - Datos de punto de paro en la página 1771</a>   |
| Definición de herramientas  | <a href="#">tooldata - Datos de herramienta en la página 1795</a>  |
| Definición de objetos de trabajo  | <a href="#">wobjdata - Datos del objeto de trabajo en la página 1823</a>   |
| Definición de datos de zona   | <a href="#">zonedata - Datos de zonas en la página 1831</a>  |
| Movimiento en general   | <i>Manual de referencia técnica - RAPID Overview</i>   |
| Ejemplo de cómo usar <code>TLoad</code> , carga total.  | <a href="#">MoveL - Mueve el robot siguiendo una trayectoria lineal en la página 447</a>   |
| Definición de la carga útil de un robot   | <a href="#">GripLoad - Define la carga útil de un robot en la página 244</a>   |
| <code>LoadIdentify</code> , rutina de servicio de identificación de carga                                 | <i>Manual del operador - IRC5 con FlexPendant</i>  |
| Señal de entrada de sistema <code>SimMode</code> para mover el robot en el modo simulado sin carga útil   | <i>Manual de referencia técnica - Parámetros del sistema</i>   |
| Parámetro de sistema <code>ModalPayloadMode</code> para la activación y la desactivación de la carga útil | <i>Manual de referencia técnica - Parámetros del sistema</i>   |

# 1 Instrucciones

## 1.309 TriggL - Movimiento lineal del robot con eventos

RobotWare - OS

### 1.309 TriggL - Movimiento lineal del robot con eventos

#### Utilización

TriggL (*Trigg Linear*) se utiliza para establecer señales de salida y/o ejecutar rutinas de interrupción en posiciones fijas al tiempo que se mueve el robot siguiendo un movimiento lineal.

Es posible definir uno o varios eventos (hasta un máximo de 25) mediante las instrucciones TriggIO, TriggEquip, TriggInt, TriggSpeed, TriggCheckIO, o TriggRampAO. Posteriormente, se hace referencia a estas definiciones en la instrucción TriggL.

Esta instrucción sólo puede usarse en la tarea principal T\_ROB1 o, si se cuenta con un sistema *MultiMove*, en las tareas de movimiento.

#### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción TriggL:

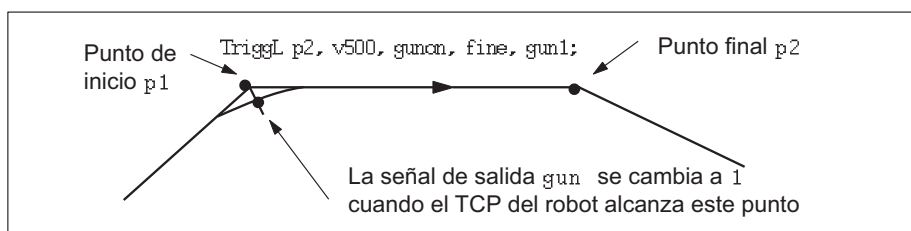
Consulte también [Más ejemplos en la página 945](#).

#### Ejemplo 1

```
VAR triggdata gunon;  
  
TriggIO gunon, 0 \Start \DOp:=gun, 1;  
MoveJ p1, v500, z50, gun1;  
TriggL p2, v500, gunon, fine, gun1;
```

La señal digital de salida `gun` se activa cuando el TCP del robot atraviesa el punto central de la trayectoria de esquina del punto `p1`.

En la figura se muestra un ejemplo de evento de E/S basado en una posición fija.



xx0500002291

#### Argumentos

```
TriggL [\Conc] ToPoint [\ID] Speed [\T] Trigg_1 | TriggArray [\T2]  
[\T3] [\T4] [\T5] [\T6] [\T7] [\T8] Zone [\Inpos] Tool [\WObj]  
[\Corr] [\TLoad]
```

[ \Conc ]

#### *Concurrent*

Tipo de dato:switch

Distintas instrucciones consecutivas se ejecutan mientras el robot está en movimiento. Este argumento puede usarse para evitar los paros no deseados, causados por la sobrecarga de la CPU, cuando se utilizan puntos de paso. Esto

*Continúa en la página siguiente*



resulta útil cuando los puntos programados están muy cercanos entre sí y se trabaja a velocidades elevadas.

Este argumento también resulta útil si, por ejemplo, no se requiere la comunicación con equipos externos ni la sincronización entre los equipos externos y los movimientos del robot. También puede usarse para ajustar la ejecución de la trayectoria del robot, para evitar el aviso 50024 Fallo en punto de paso o el error 50082 Limite de deceleración.

Cuando se utiliza el argumento `\Conc`, el número de instrucciones de movimiento seguidas está limitado a 5. En secciones de programa que incluyen `StorePath-RestoPath`, no se permite el uso de instrucciones con el argumento `\Conc`.

Si se omite este argumento y `ToPoint` no es un punto de paro, la instrucción siguiente se ejecuta algún tiempo antes de que el robot alcance la zona programada.

Este argumento no puede usarse en los movimientos sincronizados coordinados en un sistema `MultiMove`.

`ToPoint`

Tipo de dato: `robtarget`

El punto de destino de los ejes del robot y de los ejes externos. Se define como una posición con nombre o se almacena directamente en la instrucción (marcada con un asterisco \* en la instrucción).

[ `\ID` ]

*Synchronization id*

Tipo de dato: `identno`

El argumento [ `\ID` ] es obligatorio en los sistemas `MultiMove`, si el movimiento es sincronizado o sincronizado coordinado. Este argumento no está permitido en ningún otro caso. El número de ID especificado debe ser el mismo en todas las tareas de programa que cooperan entre sí. Al usar el número de ID los movimientos no se mezclan en tiempo de ejecución.

`Speed`

Tipo de dato: `speeddata`

Los datos de velocidad que se aplican a los movimientos. Los datos de velocidad definen la velocidad del punto central de la herramienta, los ejes externos y la reorientación de la herramienta.

[ `\T` ]

*Time*

Tipo de dato: `num`

Este argumento se utiliza para especificar el tiempo total en segundos que dura el movimiento del robot. A continuación, se sustituye por los datos de velocidad correspondientes.

`Trigg_1`

Tipo de dato: `triggdata`

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.309 TriggL - Movimiento lineal del robot con eventos

RobotWare - OS

Continuación

La variable que hace referencia a las condiciones de disparo y la actividad de disparo, definida anteriormente en el programa usando las instrucciones TriggIO, TriggEquip, TriggInt, TriggSpeed, TriggCheckIO o TriggRampAO.

TriggArray

### *Trigg Data Array Parameter*

Tipo de dato: triggdata

La variable de matriz que hace referencia a las condiciones de disparo y la actividad de disparo, definidas anteriormente en el programa usando las instrucciones TriggIO, TriggEquip, TriggInt, TriggSpeed, TriggCheckIO o TriggRampAO.

La matriz presenta una limitación de 25 elementos y es necesario definir de 1 a 25 condiciones de disparo.

No es posible utilizar los argumentos opcionales T2, T3, T4, T5, T6, T7 o T8 al mismo tiempo que se utiliza el argumento TriggArray.

[ \T2 ]

### *Trigg 2*

Tipo de dato: triggdata

La variable que hace referencia a las condiciones de disparo y la actividad de disparo, definida anteriormente en el programa usando las instrucciones TriggIO, TriggEquip, TriggInt, TriggSpeed, TriggCheckIO o TriggRampAO.

[ \T3 ]

### *Trigg 3*

Tipo de dato: triggdata

La variable que hace referencia a las condiciones de disparo y la actividad de disparo, definida anteriormente en el programa usando las instrucciones TriggIO, TriggEquip, TriggInt, TriggSpeed, TriggCheckIO o TriggRampAO.

[ \T4 ]

### *Trigg 4*

Tipo de dato: triggdata

La variable que hace referencia a las condiciones de disparo y la actividad de disparo, definida anteriormente en el programa usando las instrucciones TriggIO, TriggEquip, TriggInt, TriggSpeed, TriggCheckIO o TriggRampAO.

[ \T5 ]

### *Trigg 5*

Tipo de dato: triggdata

La variable que hace referencia a las condiciones de disparo y la actividad de disparo, definida anteriormente en el programa usando las instrucciones TriggIO, TriggEquip, TriggInt, TriggSpeed, TriggCheckIO o TriggRampAO.

[ \T6 ]

### *Trigg 6*

Tipo de dato: triggdata

Continúa en la página siguiente

La variable que hace referencia a las condiciones de disparo y la actividad de disparo, definida anteriormente en el programa usando las instrucciones `TriggIO`, `TriggEquip`, `TriggInt`, `TriggSpeed`, `TriggCheckIO` o `TriggRampAO`.

[ \T7 ]

### *Trigg 7*

Tipo de dato: `triggdata`

La variable que hace referencia a las condiciones de disparo y la actividad de disparo, definida anteriormente en el programa usando las instrucciones `TriggIO`, `TriggEquip`, `TriggInt`, `TriggSpeed`, `TriggCheckIO` o `TriggRampAO`.

[ \T8 ]

### *Trigg 8*

Tipo de dato: `triggdata`

La variable que hace referencia a las condiciones de disparo y la actividad de disparo, definida anteriormente en el programa usando las instrucciones `TriggIO`, `TriggEquip`, `TriggInt`, `TriggSpeed`, `TriggCheckIO` o `TriggRampAO`.

Zone

Tipo de dato: `zonedata`

Los datos de zona del movimiento. Los datos de zona describen el tamaño de la trayectoria de esquina generada.

[ \Inpos ]

### *In position*

Tipo de dato: `stoppointdata`

Este argumento se utiliza para especificar los criterios de convergencia para la posición del TCP del robot en el punto de paro. Los datos de puntos de paro sustituyen a la zona especificada en el parámetro `Zone`.

Tool

Tipo de dato: `tooldata`

La herramienta en uso durante el movimiento del robot. El punto central de la herramienta es el punto que se mueve hacia la posición de destino especificada.

[ \WObj ]

### *Work Object*

Tipo de dato: `wobjdata`

El objeto de trabajo (sistema de coordenadas) con el que está relacionada la posición de robot indicada en la instrucción.

Es posible omitir este argumento. Si se omite, la posición depende del sistema de coordenadas mundo. Si por otro lado se usa un TCP estacionario o ejes externos coordinados, es necesario especificar este argumento para ejecutar un movimiento lineal respecto del objeto de trabajo.

[ \Corr ]

### *Correction*

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.309 TriggL - Movimiento lineal del robot con eventos

RobotWare - OS

Continuación

Tipo de dato: `switch`

Los datos de corrección escritos en una entrada de corrección mediante una instrucción `CorrWrite` se añaden a la trayectoria y a la posición de destino si se utiliza este argumento.

Se requiere RobotWare, opción *Path Offset*, cuando se utiliza este argumento.

[ `\TLoad` ]

### Total load

Tipo de dato: `loaddata`

El argumento `\TLoad` describe la carga total usada durante el movimiento. La carga total es la carga de la herramienta más la carga útil transportada por la herramienta. Si se utiliza el argumento `\TLoad`, no se tiene en cuenta el valor de `loaddata` en los `tooldata` actuales.

Si el argumento `\TLoad` tiene el valor `load0`, el argumento `\TLoad` no se tiene en cuenta y se utilizan en su lugar los `loaddata` de los `tooldata`.

Para poder utilizar el argumento `\TLoad`, es necesario cambiar el valor del parámetro de sistema `ModalPayloadMode` a 0. Si `ModalPayloadMode` tiene el valor 0, ya no es posible utilizar la instrucción `GripLoad`.

La carga total puede identificarse con la rutina de servicio `LoadIdentify`. Si el parámetro de sistema `ModalPayloadMode` tiene el valor 0, el operador tiene la posibilidad de copiar los `loaddata` de la herramienta a una variable persistente `loaddata` existente o nueva al ejecutar la rutina de servicio.

Es posible realizar una ejecución de prueba del programa sin ninguna carga útil utilizando una señal de entrada digital conectada a la entrada de sistema `SimMode` (modo simulado). Si la señal de entrada digital tiene el valor 1, los `loaddata` del argumento opcional `\TLoad` no se tienen en cuenta y se utilizan en su lugar los `loaddata` de los `tooldata` actuales.



### Nota

La funcionalidad predeterminada de manejo de la carga útil es utilizar la instrucción `GripLoad`. Por tanto, el valor predeterminado del parámetro de sistema `ModalPayloadMode` es 1.

---

## Ejecución de programas

Consulte la instrucción `MoveL` para obtener más información acerca del movimiento lineal.

A medida que se cumplen las condiciones de disparo cuando el robot se sitúa más y más cerca del punto final, se realizan las actividades de disparo definidas. Las condiciones de disparo se cumplen a una distancia determinada del punto final de la instrucción o a una distancia determinada del punto de inicio de la instrucción, o bien en un momento determinado (limitado a un tiempo breve) antes del punto final de la instrucción.

Continúa en la página siguiente

Durante la ejecución paso a paso hacia delante, las actividades de E/S se realizan pero las rutinas de interrupción no se ejecutan. Durante la ejecución paso a paso hacia atrás, no se realiza ninguna actividad de disparo.

### Más ejemplos

A continuación aparecen más ejemplos de cómo usar la instrucción TriggL.

#### Ejemplo 1

```
VAR intnum intnol;
VAR triggdata trigg1;
...
PROC main()
  CONNECT intnol WITH trap1;
  TriggInt trigg1, 0.1 \Time, intnol;
  ...
  TriggL p1, v500, trigg1, fine, gun1;
  TriggL p2, v500, trigg1, fine, gun1;
  ...
  IDelete intnol;
```

La rutina de interrupción `trap1` se ejecuta si el punto de trabajo se encuentra en la posición situada 0.1 segundos antes del punto `p1` o `p2` respectivamente.

#### Ejemplo 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;
  MoveL p2, v500, z30, tool2;
  ...
```

Las señales de salida digitales de la `do1` a la `do25` se activan durante el movimiento a `p1`. La distancia entre los valores de las señales es de 10 mm.

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.309 TriggL - Movimiento lineal del robot con eventos

RobotWare - OS

Continuación

### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                      | Causa del error   |
|-----------------------------|---|
| <code>ERR_AO_LIM</code>     | El argumento <code>ScaleValue</code> programado para la señal analógica de salida especificada <code>AOp</code> en algunas de las instrucciones <code>TriggSpeed</code> conectadas da lugar a la salida de límite de la señal analógica junto con la <code>Speed</code> programada en esta instrucción. |
| <code>ERR_DIPLAG_LIM</code> | El argumento <code>DipLag</code> programado en algunas de las instrucciones <code>TriggSpeed</code> conectadas es demasiado grande con respecto al tiempo preestablecido para los eventos en parámetros del sistema.  |
| <code>ERR_NORUNUNIT</code>  | No hay ningún contacto con el dispositivo de E/S al introducir la instrucción, y el <code>triggdata</code> utilizado depende de un dispositivo de E/S en funcionamiento, es decir, que se utiliza una señal en <code>triggdata</code> .   |
| <code>ERR_CONC_MAX</code>   | Se ha superado el número de instrucciones de movimiento seguidas con el argumento <code>\Conc</code> .  |

### Limitaciones

Si el punto de inicio actual se desvía del habitual, de forma que la longitud de posicionamiento total de la instrucción `TriggL` es más corta de lo habitual (es decir, al principio de `TriggL` con la posición del robot en el punto final), puede ocurrir que se cumplan varias de las condiciones de disparo, o incluso todas ellas, inmediatamente y en una misma posición. En estos casos, la secuencia en la que se realizan las actividades de disparo no estará definida. La lógica de programa del programa del usuario no puede basarse en una secuencia normal de actividades de disparo para un movimiento incompleto.

`TriggL` no puede ejecutarse en un gestor UNDO o en rutinas de RAPID que estén conectadas a los siguientes eventos especiales del sistema: `PowerOn`, `Stop`, `QStop`, `Restart`, `Reset` o `Step`.

### Sintaxis

```
TriggL
  ['\ Conc ',']
  [ToPoint' :='] <expression (IN) of robtarg>
  ['\ 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>
```

Continúa en la página siguiente

```
[ '\ ' Inpos ' := ' <expression (IN) of stoppointdata> ] ', '
[ Tool ' := ' ] <persistent (PERS) of tooldata>
[ '\ ' WObj ' := ' <persistent (PERS) of wobjdata> ]
[ '\ ' Corr ]
[ '\ ' TLoad ' := ' <persistent (PERS) of loaddata> ] ; ;'
```

### Información relacionada

| Para obtener más información sobre       | Consulte   |
|--|--|
| Movimiento circular con disparadores     | <a href="#">TriggC - Movimiento circular del robot con eventos en la página 895</a>  |
| Movimiento de ejes con disparadores      | <a href="#">TriggJ - Movimientos de ejes del robot a partir de eventos en la página 932</a>  |
| Definición de disparadores               | <a href="#">TriggIO - Define un evento de E/S de posición o tiempo fijos cerca de un punto de paro en la página 926</a><br><a href="#">TriggEquip - Define un evento de E/S basado en la posición y el tiempo en la trayectoria en la página 914</a><br><a href="#">TriggInt - Define una interrupción dependiente de una posición en la página 921</a><br><a href="#">TriggCheckIO - Define una comprobación de E/S en una posición fija en la página 904</a><br><a href="#">TriggRampAO - Define un evento AO de rampa de posición fija en la trayectoria en la página 965</a><br><a href="#">TriggSpeed - Define la velocidad del TCP en proporción a una salida analógica con un evento de escala fija de posición-tiempo en la página 973</a> |
| Manejo de triggdata                      | <a href="#">triggdata - Eventos de posicionamiento, trigg en la página 1805</a><br><a href="#">TriggDataReset - Restablecer el contenido en una variable de tipo triggdata en la página 912</a><br><a href="#">TriggDataCopy - Copiar el contenido de una variable de tipo triggdata en la página 910</a><br><a href="#">TriggDataValid - Comprobar si el contenido de una variable de tipo triggdata es válido en la página 1537</a>  |
| Escritura en una entrada de corrección   | <a href="#">CorrWrite - Escribe en un generador de correcciones en la página 183</a>   |
| Movimiento lineal                        | <a href="#">Manual de referencia técnica - RAPID Overview</a>  |
| Definición de carga                      | <a href="#">loaddata - Datos de carga en la página 1694</a>  |
| Definición de velocidad                  | <a href="#">speeddata - Datos de velocidad en la página 1767</a>   |
| Definición de datos de punto de paro     | <a href="#">stoppointdata - Datos de punto de paro en la página 1771</a>   |
| Definición de herramientas               | <a href="#">tooldata - Datos de herramienta en la página 1795</a>  |
| Definición de objetos de trabajo         | <a href="#">wobjdata - Datos del objeto de trabajo en la página 1823</a>   |
| Definición de datos de zona              | <a href="#">zonedata - Datos de zonas en la página 1831</a>  |
| Movimiento en general                    | <a href="#">Manual de referencia técnica - RAPID Overview</a>  |
| Ejemplo de cómo usar TLoad, carga total. | <a href="#">MoveL - Mueve el robot siguiendo una trayectoria lineal en la página 447</a>   |
| Definición de la carga útil de un robot  | <a href="#">GripLoad - Define la carga útil de un robot en la página 244</a>   |

Continúa en la página siguiente

# 1 Instrucciones

---

## 1.309 TriggL - Movimiento lineal del robot con eventos

RobotWare - OS

Continuación

| Para obtener más información sobre   | Consulte   |
|--|--|
| LoadIdentify, rutina de servicio de identificación de carga  | <i>Manual del operador - IRC5 con FlexPendant</i>            |
| Señal de entrada de sistema <i>Sim-Mode</i> para mover el robot en el modo simulado sin carga útil   | <i>Manual de referencia técnica - Parámetros del sistema</i> |
| Parámetro de sistema <i>ModalPay-LoadMode</i> para la activación y la desactivación de la carga útil | <i>Manual de referencia técnica - Parámetros del sistema</i> |
| <i>Path Offset</i>   | <i>Application manual - Controller software IRC5</i>         |



## 1.310 TriggJIOs - Movimientos de ejes del robot con eventos de E/S

### Utilización

TriggJIOs (*Trigg Joint I/O*) se utiliza para establecer señales de salida al tiempo que se mueve el robot siguiendo un movimiento de ejes.

La instrucción TriggJIOs está optimizada para proporcionar una buena exactitud cuando se utilizan movimientos con zonas (compárese con TriggEquip/TriggL).

Esta instrucción sólo puede usarse en la tarea principal T\_ROB1 o, si se cuenta con un sistema *MultiMove*, en las tareas de movimiento.

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción TriggJIOs:

Consulte también [Más ejemplos en la página 961](#).

#### Ejemplo 1

```
VAR triggios gunon{1};

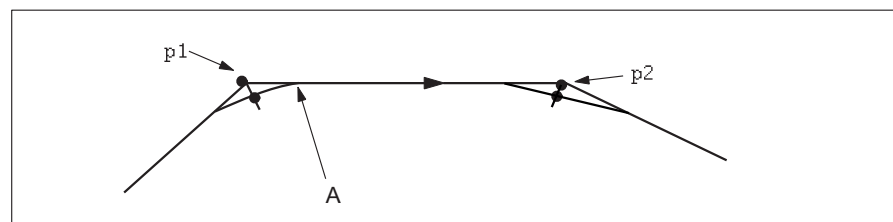
gunon{1}.used:=TRUE;
gunon{1}.distance:=3;
gunon{1}.start:=TRUE;
gunon{1}.signalname="gun";
gunon{1}.equiplag:=0;
gunon{1}.setvalue:=1;

MoveJ p1, v500, z50, gun1;
TriggJIOs p2, v500, \TriggData1:=gunon, z50, gun1;
MoveL p3, v500, z50, gun1;
```

La señal `gun` se activa cuando el TCP está 3 mm después del punto `p1`.

El código de RAPID y la figura muestran un ejemplo de evento de E/S basado en una posición fija.

```
TriggJIOs p2, v500, \TriggData1:=gunon, z50, gun1;
```



xx1500000304

**A** La señal de salida `gun` se cambia a 1 cuando el TCP del robot alcanza este punto.

### Argumentos

```
TriggJIOs [\Conc] ToPoint [\ID] Speed [\T] [\TriggData1]
[\TriggData2] [\TriggData3] Zone [\Inpos] Tool [\WObj]
[\TLoad]
```

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.310 TriggJIOs - Movimientos de ejes del robot con eventos de E/S

RobotWare - OS

Continuación

[ \Conc ]

### *Concurrent*

Tipo de dato: switch

Distintas instrucciones consecutivas se ejecutan mientras el robot está en movimiento. Este argumento puede usarse para evitar los paros no deseados, causados por la sobrecarga de la CPU, cuando se utilizan puntos de paso. Esto resulta útil cuando los puntos programados están muy cercanos entre sí y se trabaja a velocidades elevadas.

Este argumento también resulta útil si, por ejemplo, no se requiere la comunicación con equipos externos ni la sincronización entre los equipos externos y los movimientos del robot. También puede usarse para ajustar la ejecución de la trayectoria del robot, para evitar el aviso 50024 Fallo en punto de paso o el error 50082 Limite de deceleración.

Cuando se utiliza el argumento \Conc, el número de instrucciones de movimiento seguidas está limitado a 5. En secciones de programa que incluyen StorePath-RestoPath, no se permite el uso de instrucciones con el argumento \Conc.

Si se omite este argumento y ToPoint no es un punto de paro, la instrucción siguiente se ejecuta algún tiempo antes de que el robot alcance la zona programada.

Este argumento no puede usarse en los movimientos sincronizados coordinados en un sistema MultiMove.

ToPoint

Tipo de dato: robtarget

El punto de destino de los ejes del robot y de los ejes externos. Se define como una posición con nombre o se almacena directamente en la instrucción (marcada con un asterisco \* en la instrucción).

[ \ID ]

### *Synchronization id*

Tipo de dato: identno

El argumento [ \ID ] es obligatorio en los sistemas MultiMove, si el movimiento es sincronizado o sincronizado coordinado. Este argumento no está permitido en ningún otro caso. El número de ID especificado debe ser el mismo en todas las tareas de programa que cooperan entre sí. Al usar el número de ID los movimientos no se mezclan en tiempo de ejecución.

Speed

Tipo de dato: speeddata

Los datos de velocidad que se aplican a los movimientos. Los datos de velocidad definen la velocidad del punto central de la herramienta, los ejes externos y la reorientación de la herramienta.

[ \T ]

### *Time*

Continúa en la página siguiente

**Tipo de dato:** num

Este argumento se utiliza para especificar el tiempo total en segundos que dura el movimiento del robot. A continuación, se sustituye por los datos de velocidad correspondientes.

[ \TriggData1 ]

**Tipo de dato:** array of triggios

Una variable (matriz) que hace referencia a condiciones de disparo y actividad de disparo. Al utilizar este argumento, es posible establecer señales analógicas de salida, señales digitales de salida y señales digitales de salida de grupo. Si utiliza una señal digital de salida de grupo, existe una limitación de 23 señales por grupo.

[ \TriggData2 ]

**Tipo de dato:** array of triggstrgo

Una variable (matriz) que hace referencia a condiciones de disparo y actividad de disparo. Al utilizar este argumento, es posible establecer señales digitales de salida de grupo compuestas por 32 señales en el grupo y con un valor establecido máximo de 4.294.967.295. Sólo es posible utilizar señales digitales de salida de grupo.

[ \TriggData3 ]

**Tipo de dato:** array of triggiosdnum

Una variable (matriz) que hace referencia a condiciones de disparo y actividad de disparo. Al utilizar este argumento, es posible establecer señales analógicas de salida, señales digitales de salida y señales digitales de salida de grupo compuestas por 32 señales en el grupo y con un valor establecido máximo de 4294967295.

Zone

**Tipo de dato:** zonedata

Los datos de zona del movimiento. Los datos de zona describen el tamaño de la trayectoria de esquina generada.

[ \Inpos ]

*In position*

**Tipo de dato:** stoppointdata

Este argumento se utiliza para especificar los criterios de convergencia para la posición del TCP del robot en el punto de paro. Los datos de puntos de paro sustituyen a la zona especificada en el parámetro Zone.

Tool

**Tipo de dato:** tooldata

La herramienta en uso durante el movimiento del robot. El punto central de la herramienta es el punto que se mueve hacia la posición de destino especificada.

[ \WObj ]

*Work Object*

**Tipo de dato:** wobjdata

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.310 TriggJIos - Movimientos de ejes del robot con eventos de E/S

RobotWare - OS

Continuación

El objeto de trabajo (sistema de coordenadas) con el que está relacionada la posición de robot indicada en la instrucción.

Es posible omitir este argumento. Si se omite, la posición depende del sistema de coordenadas mundo. Si por otro lado se usa un TCP estacionario o ejes externos coordinados, es necesario especificar este argumento para ejecutar un movimiento lineal respecto del objeto de trabajo.

[ \TLoad ]

### *Total load*

Tipo de dato: loaddata

El argumento \TLoad describe la carga total usada durante el movimiento. La carga total es la carga de la herramienta más la carga útil transportada por la herramienta. Si se utiliza el argumento \TLoad, no se tiene en cuenta el valor de loaddata en los tooldata actuales.

Si el argumento \TLoad tiene el valor load0, el argumento \TLoad no se tiene en cuenta y se utilizan en su lugar los loaddata de los tooldata.

Para poder utilizar el argumento \TLoad, es necesario cambiar el valor del parámetro de sistema ModalPayloadMode a 0. Si ModalPayloadMode tiene el valor 0, ya no es posible utilizar la instrucción GripLoad.

La carga total puede identificarse con la rutina de servicio LoadIdentify. Si el parámetro de sistema ModalPayloadMode tiene el valor 0, el operador tiene la posibilidad de copiar los loaddata de la herramienta a una variable persistente loaddata existente o nueva al ejecutar la rutina de servicio.

Es posible realizar una ejecución de prueba del programa sin ninguna carga útil utilizando una señal de entrada digital conectada a la entrada de sistema SimMode (modo simulado). Si la señal de entrada digital tiene el valor 1, los loaddata del argumento opcional \TLoad no se tienen en cuenta y se utilizan en su lugar los loaddata de los tooldata actuales.



### **Nota**

La funcionalidad predeterminada de manejo de la carga útil es utilizar la instrucción GripLoad. Por tanto, el valor predeterminado del parámetro de sistema ModalPayloadMode es 1.

---

## Ejecución de programas

Consulte la instrucción MoveJ para obtener más información acerca del movimiento de ejes, [MoveJ - Mueve el robot mediante un movimiento de ejes en la página 421](#).

La instrucción TriggJIos permite configurar de 1 a 50 actividades de disparo diferentes a partir de señales de E/S a lo largo de una trayectoria de A a B. Las señales que pueden usarse son señales digitales de salida, señales digitales de salida de grupo y señales analógicas de salida. Las condiciones de disparo se cumplen a cierta distancia antes del punto final de la instrucción o una determinada distancia tras el punto de inicio de la instrucción.

Continúa en la página siguiente

La instrucción requiere el uso del argumento `TriggData1`, `TriggData2` o `TriggData3` o bien todos ellos. Sin embargo, el uso de cualquiera de los disparos. Para inhibir el uso de un disparo, el componente `used` puede cambiarse a `FALSE` en el elemento de matriz de los tipos de datos `triggios/triggstrgo/triggiosdnum`. Si no se utiliza ningún elemento de matriz, la instrucción `TriggJIos` se comporta como una instrucción `MoveJ` y no se realiza ninguna actividad de E/S.

Si se ejecuta el programa paso a paso hacia delante, las actividades de E/S se ejecutan. Durante la ejecución paso a paso hacia atrás, no se realiza ninguna actividad de E/S.

Si se define el componente `EquipLag` en el argumento `TriggData1`, `TriggData2` o `TriggData3` con un tiempo (retardo) negativo, la señal de E/S puede establecerse a continuación del punto de destino (`ToPoint`).

Si utiliza el argumento `TriggData2` o `TriggData3`, es posible utilizar valores hasta el 4294967295, que es el valor máximo que puede tener un grupo de señales digitales (el sistema admite como máximo 32 señales en una señal de grupo).

### Más ejemplos

A continuación aparecen más ejemplos de cómo usar la instrucción `TriggJIos`.

#### Ejemplo 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;
```

La señal digital de salida de grupo `go1` recibe el valor 55 a una distancia de 3 mm de `p1`. La señal analógica recibe el valor 10 a una distancia de 15 mm de `p1`. La señal digital de salida `do1` recibirá el valor 3 mm a partir de `ToPoint p2`.

#### Ejemplo 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;
```

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.310 TriggJIos - Movimientos de ejes del robot con eventos de E/S

RobotWare - OS

Continuación

La señal digital de salida de grupo `go1` recibe el valor 55 a una distancia de 3 mm de `p1`. La señal analógica de salida `ao1` recibe el valor 10 a una distancia de 15 mm de `p1`. La señal digital de salida `do1` recibirá el valor 3 mm a partir de `ToPoint p2`. Estos eventos de posición son definidos por la variable `mytriggios`. La variable `mytriggstrgo` establece que se produzcan eventos de posición a una distancia de entre 3 y 15 mm de `p1`. La señal `go2` recibe primero el valor 1 y luego el valor 800000. La señal recibe el valor 4294967295 a una distancia de 4 mm de `ToPoint p2`. Se trata del valor máximo para una señal digital de salida de 32 bits. La variable `mytriggiosdnum` establece que se produzcan tres eventos de posición a una distancia de 10 mm de `p1`. En primer lugar se cambia la señal `go3` a 4294967295, a continuación se cambia `ao2` a 5 y en último lugar se cambia `do2` a 1.

### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                     | Causa del error  |
|----------------------------|--|
| <code>ERR_NORUNUNIT</code> | Se ha perdido el contacto con el dispositivo de E/S.   |
| <code>ERR_GO_LIM</code>    | El argumento programado <code>setvalue</code> para la señal digital de salida de grupo especificada <code>signalname</code> está fuera de límites. (Se declara en <code>TriggData1</code> , <code>TriggData2</code> o <code>TriggData3</code> ). |
| <code>ERR_AO_LIM</code>    | El argumento programado <code>setvalue</code> para la señal analógica de salida especificada <code>signalname</code> está fuera de límites. (Se declara en <code>TriggData1</code> o <code>TriggData3</code> ).                                  |

### Limitaciones

Si el punto de inicio actual se desvía del habitual, de forma que la longitud de posicionamiento total de la instrucción `TriggJIos` es más corta de lo habitual (es decir, al principio de `TriggJIos` con la posición del robot en el punto final), puede ocurrir que se cumplan varias de las condiciones de disparo, o incluso todas ellas, inmediatamente y en una misma posición. En estos casos, la secuencia en la que se realizan las actividades de disparo no estará definida. La lógica de programa del programa del usuario no puede basarse en una secuencia normal de actividades de disparo para un movimiento incompleto.

La limitación en el número de disparos de la instrucción `TriggJIos` es de 50 para cada instrucción programada. Si los disparos se producen a una distancia más corta, es posible que el sistema no los gestione. Esto depende de cómo se realice el movimiento, la velocidad de TCP utilizada y la cercanía programada entre los disparos. Estas limitaciones existen pero resulta difícil predecir cuándo se producirán estos problemas.

Continúa en la página siguiente

TriggJIos no puede ejecutarse en un gestor UNDO o en rutinas de RAPID que estén conectadas a los siguientes eventos especiales del sistema: PowerOn, Stop, QStop, Restart, Reset o Step.

#### Limitaciones respecto a la exactitud

Los eventos de E/S basados en una distancia se han diseñado para los puntos de paso (trayectorias de esquina). El uso de puntos de paro daría como resultado una exactitud menor que la especificada a continuación.

En cuanto a la exactitud de los eventos de E/S con distancia y con puntos de paso, se aplica lo siguiente a la hora de activar una salida digital a una distancia específica del punto de inicio o del punto final con las instrucciones TriggLIos o TriggCIos:

- La exactitud especificada a continuación es válida cuando se utiliza un `equiplag` positivo<sup>1</sup> que es menor que 40 ms, que equivale al retardo del servo del robot sin cambiar el parámetro *Event Preset Time* del sistema. El retardo puede variar de un tipo de robot a otro.
- La precisión especificada abajo es válida cuando se utiliza un `equiplag` positivo menor que el *Event Preset Time* configurado en los parámetros del sistema.
- La exactitud especificada a continuación no es válida cuando se utiliza un `equiplag` positivo mayor que el *Event Preset Time* configurado en los parámetros del sistema. En este caso, se utiliza un método aproximado que no tiene en cuenta las limitaciones dinámicas del robot. Entonces, debe usarse `SingArea \Wrist` para conseguir una precisión aceptable.
- La exactitud especificada a continuación es válida cuando se utiliza un `equiplag` negativo.

<sup>1</sup> `equiplag` es un `dataobject` de tipo de datos `triggios`

Los valores absolutos típicos en cuanto a la exactitud para activar salidas digitales son de:  $\pm 5$  ms.

Los valores repetidos típicos en cuanto a la exactitud para activar salidas digitales son de:  $\pm 2$  ms.

#### Sintaxis

```
TriggJIos
['\' Conc ',']
[ToPoint ':=' ] <expression (IN) of robtarg>
['\' 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>]
['\' TLoad ':=' <persistent (PERS) of loaddata>'];'
```

Continúa en la página siguiente

# 1 Instrucciones

## 1.310 TriggJIOs - Movimientos de ejes del robot con eventos de E/S

RobotWare - OS

Continuación

### Información relacionada

| Para obtener más información sobre  | Consulte   |
|---|--|
| Movimientos lineales del robot con eventos de E/S   | <a href="#">TriggLIOs - Movimientos lineales del robot con eventos de E/S en la página 957</a> |
| Almacenamiento de condiciones de disparo y actividad de disparo   | <a href="#">triggios - Eventos de posicionamiento, trigg en la página 1806</a>                 |
| Almacenamiento de condiciones de disparo y actividad de disparos de una señal digital de grupo compuesta por 32 señales | <a href="#">triggstrgo - Positioning events, trigg en la página 1814</a>                       |
| Almacenamiento de condiciones de disparo y actividad de disparo   | <a href="#">triggiosdnum - Eventos de posicionamiento, trigg en la página 1809</a>             |
| Asignación de objetos de evento   | Manual de referencia técnica - Parámetros del sistema  |
| Movimiento lineal   | Manual de referencia técnica - RAPID Overview  |
| Movimiento en general   | Manual de referencia técnica - RAPID Overview  |
| Definición de carga   | <a href="#">loaddata - Datos de carga en la página 1694</a>                                    |
| Ejemplo de cómo usar TLoad, carga total.  | <a href="#">MoveL - Mueve el robot siguiendo una trayectoria lineal en la página 447</a>       |
| Definición de la carga útil de un robot   | <a href="#">GripLoad - Define la carga útil de un robot en la página 244</a>                   |
| LoadIdentify, rutina de servicio de identificación de carga   | Manual del operador - IRC5 con FlexPendant   |
| Señal de entrada de sistema SimMode para mover el robot en el modo simulado sin carga útil                              | Manual de referencia técnica - Parámetros del sistema  |
| Parámetro de sistema ModalPayloadMode para la activación y la desactivación de la carga útil                            | Manual de referencia técnica - Parámetros del sistema  |
| Path Offset   | Application manual - Controller software IRC5  |



## 1.311 TriggLIOS - Movimientos lineales del robot con eventos de E/S

### Utilización

TriggLIOS (disparar E/S lineal) se utiliza para establecer señales de salida al tiempo que se mueve el robot siguiendo un movimiento lineal.

La instrucción TriggLIOS está optimizada para proporcionar una buena exactitud cuando se utilizan movimientos con zonas (compárese con TriggEquip/TriggL).

Esta instrucción sólo puede usarse en la tarea principal T\_ROB1 o, si se cuenta con un sistema *MultiMove*, en las tareas de movimiento.

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción TriggLIOS:

Consulte también [Más ejemplos en la página 961](#).

#### Ejemplo 1

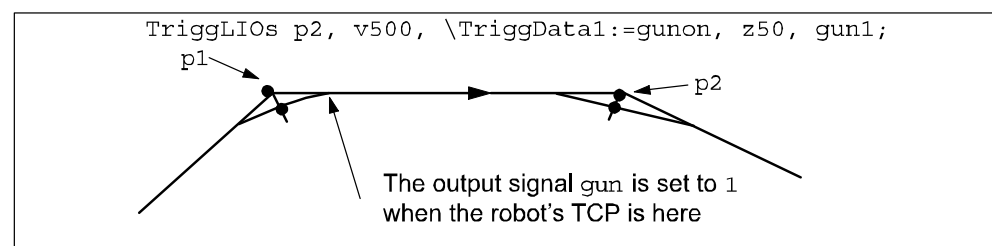
```
VAR triggios gunon{1};

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;
```

La señal `gun` se activa cuando el TCP está 3 mm después del punto `p1`.

En la figura se muestra un ejemplo de evento de E/S basado en una posición fija.



en0800000157

### Argumentos

```
TriggLIOS [\Conc] ToPoint [\ID] Speed [\T] [\TriggData1]
[\TriggData2] [\TriggData3] Zone [\Inpos] Tool [\WObj] [\Corr]
[\TLoad]
```

[ \Conc ]

**Concurrent**

Tipo de dato:switch

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.311 TriggLIOs - Movimientos lineales del robot con eventos de E/S

RobotWare - OS

Continuación

Distintas instrucciones consecutivas se ejecutan mientras el robot está en movimiento. Este argumento puede usarse para evitar los paros no deseados, causados por la sobrecarga de la CPU, cuando se utilizan puntos de paso. Esto resulta útil cuando los puntos programados están muy cercanos entre sí y se trabaja a velocidades elevadas.

Este argumento también resulta útil si, por ejemplo, no se requiere la comunicación con equipos externos ni la sincronización entre los equipos externos y los movimientos del robot. También puede usarse para ajustar la ejecución de la trayectoria del robot, para evitar el aviso 50024 Fallo en punto de paso o el error 50082 Limite de deceleración.

Cuando se utiliza el argumento `\Conc`, el número de instrucciones de movimiento seguidas está limitado a 5. En secciones de programa que incluyen `StorePath-RestoPath`, no se permite el uso de instrucciones con el argumento `\Conc`.

Si se omite este argumento y `ToPoint` no es un punto de paro, la instrucción siguiente se ejecuta algún tiempo antes de que el robot alcance la zona programada.

Este argumento no puede usarse en los movimientos sincronizados coordinados en un sistema `MultiMove`.

`ToPoint`

Tipo de dato: `robtarget`

El punto de destino de los ejes del robot y de los ejes externos. Se define como una posición con nombre o se almacena directamente en la instrucción (marcada con un asterisco \* en la instrucción).

[ `\ID` ]

*Synchronization id*

Tipo de dato: `identno`

El argumento [ `\ID` ] es obligatorio en los sistemas `MultiMove`, si el movimiento es sincronizado o sincronizado coordinado. Este argumento no está permitido en ningún otro caso. El número de ID especificado debe ser el mismo en todas las tareas de programa que cooperan entre sí. Al usar el número de ID los movimientos no se mezclan en tiempo de ejecución.

`Speed`

Tipo de dato: `speeddata`

Los datos de velocidad que se aplican a los movimientos. Los datos de velocidad definen la velocidad del punto central de la herramienta, los ejes externos y la reorientación de la herramienta.

[ `\T` ]

*Time*

Tipo de dato: `num`

Este argumento se utiliza para especificar el tiempo total en segundos que dura el movimiento del robot. A continuación, se sustituye por los datos de velocidad correspondientes.

*Continúa en la página siguiente*

[ \TriggData1 ]

Tipo de dato: array of triggios

Una variable (matriz) que hace referencia a condiciones de disparo y actividad de disparo. Al utilizar este argumento, es posible establecer señales analógicas de salida, señales digitales de salida y señales digitales de salida de grupo. Si utiliza una señal digital de salida de grupo, existe una limitación de 23 señales por grupo.

[ \TriggData2 ]

Tipo de dato: array of triggstrgo

Una variable (matriz) que hace referencia a condiciones de disparo y actividad de disparo. Al utilizar este argumento, es posible establecer señales digitales de salida de grupo compuestas por 32 señales en el grupo y con un valor establecido máximo de 4.294.967.295. Sólo es posible utilizar señales digitales de salida de grupo.

[ \TriggData3 ]

Tipo de dato: array of triggiosdnum

Una variable (matriz) que hace referencia a condiciones de disparo y actividad de disparo. Al utilizar este argumento, es posible establecer señales analógicas de salida, señales digitales de salida y señales digitales de salida de grupo compuestas por 32 señales en el grupo y con un valor establecido máximo de 4294967295.

Zone

Tipo de dato: zonedata

Los datos de zona del movimiento. Los datos de zona describen el tamaño de la trayectoria de esquina generada.

[ \Inpos ]

*In position*

Tipo de dato: stoppointdata

Este argumento se utiliza para especificar los criterios de convergencia para la posición del TCP del robot en el punto de paro. Los datos de puntos de paro sustituyen a la zona especificada en el parámetro Zone.

Tool

Tipo de dato: tooldata

La herramienta en uso durante el movimiento del robot. El punto central de la herramienta es el punto que se mueve hacia la posición de destino especificada.

[ \WObj ]

*Work Object*

Tipo de dato: wobjdata

El objeto de trabajo (sistema de coordenadas) con el que está relacionada la posición de robot indicada en la instrucción.

Es posible omitir este argumento. Si se omite, la posición depende del sistema de coordenadas mundo. Si por otro lado se usa un TCP estacionario o ejes externos

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.311 TriggLIos - Movimientos lineales del robot con eventos de E/S

RobotWare - OS

Continuación

coordinados, es necesario especificar este argumento para ejecutar un movimiento lineal respecto del objeto de trabajo.

[ \Corr ]

### *Correction*

Tipo de dato: switch

Los datos de corrección escritos en una entrada de corrección mediante una instrucción `CorrWrite` se añaden a la trayectoria y a la posición de destino si se utiliza este argumento.

Se requiere RobotWare, opción *Path Offset*, cuando se utiliza este argumento.

[ \TLoad ]

### *Total load*

Tipo de dato: loaddata

El argumento `\TLoad` describe la carga total usada durante el movimiento. La carga total es la carga de la herramienta más la carga útil transportada por la herramienta. Si se utiliza el argumento `\TLoad`, no se tiene en cuenta el valor de `loaddata` en los `tooldata` actuales.

Si el argumento `\TLoad` tiene el valor `load0`, el argumento `\TLoad` no se tiene en cuenta y se utilizan en su lugar los `loaddata` de los `tooldata`.

Para poder utilizar el argumento `\TLoad`, es necesario cambiar el valor del parámetro de sistema `ModalPayLoadMode` a 0. Si `ModalPayLoadMode` tiene el valor 0, ya no es posible utilizar la instrucción `GripLoad`.

La carga total puede identificarse con la rutina de servicio `LoadIdentify`. Si el parámetro de sistema `ModalPayLoadMode` tiene el valor 0, el operador tiene la posibilidad de copiar los `loaddata` de la herramienta a una variable persistente `loaddata` existente o nueva al ejecutar la rutina de servicio.

Es posible realizar una ejecución de prueba del programa sin ninguna carga útil utilizando una señal de entrada digital conectada a la entrada de sistema `SimMode` (modo simulado). Si la señal de entrada digital tiene el valor 1, los `loaddata` del argumento opcional `\TLoad` no se tienen en cuenta y se utilizan en su lugar los `loaddata` de los `tooldata` actuales.



### Nota

La funcionalidad predeterminada de manejo de la carga útil es utilizar la instrucción `GripLoad`. Por tanto, el valor predeterminado del parámetro de sistema `ModalPayLoadMode` es 1.

---

## Ejecución de programas

Consulte la instrucción `MoveL` para obtener más información acerca del movimiento lineal.

La instrucción `TriggLIos` permite configurar de 1 a 50 actividades de disparo diferentes a partir de señales de E/S a lo largo de una trayectoria de A a B. Las señales que pueden usarse son señales digitales de salida, señales digitales de salida de grupo y señales analógicas de salida. Las condiciones de disparo se

Continúa en la página siguiente

cumplen a cierta distancia antes del punto final de la instrucción o una determinada distancia tras el punto de inicio de la instrucción.

La instrucción requiere el uso del argumento `TriggData1`, `TriggData2` o `TriggData3`, o bien todos ellos. Sin embargo, el uso de cualquiera de los disparos. Para inhibir el uso de un disparo, el componente `used` puede cambiarse a `FALSE` en el elemento de matriz de los tipos de datos `triggios/triggstrgo/triggiosdnum`. Si no se utiliza ningún elemento de matriz, la instrucción `TriggLIOs` se comporta como una instrucción `MoveL` y no se realiza ninguna actividad de E/S.

Si se ejecuta el programa paso a paso hacia delante, las actividades de E/S se ejecutan. Durante la ejecución paso a paso hacia atrás, no se realiza ninguna actividad de E/S.

Si se define el componente `EquipLag` en el argumento `TriggData1`, `TriggData2` o `TriggData3` con un tiempo (retardo) negativo, la señal de E/S puede establecerse a continuación del punto de destino (`ToPoint`).

Si utiliza el argumento `TriggData2` o `TriggData3`, es posible utilizar valores hasta el 4294967295, que es el valor máximo que puede tener un grupo de señales digitales (el sistema admite como máximo 32 señales en una señal de grupo).

## Más ejemplos

A continuación aparecen más ejemplos de cómo usar la instrucción `TriggLIOs`.

### Ejemplo 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;
```

La señal digital de salida de grupo `go1` recibe el valor 55 a una distancia de 3 mm de `p1`. La señal analógica recibe el valor 10 a una distancia de 15 mm de `p1`. La señal digital de salida `do1` recibirá el valor 3 mm a partir de `ToPoint p2`.

### Ejemplo 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;
TriggLIOs p2, v500, \TriggData1:=mytriggios \TriggData2:=
    mytriggstrgo \TriggData3:=mytriggiosdnum, z50, gun1;
MoveL p3, v500, z50, gun1;
```

*Continúa en la página siguiente*

## 1 Instrucciones

### 1.311 TriggLIos - Movimientos lineales del robot con eventos de E/S

RobotWare - OS

Continuación

La señal digital de salida de grupo `go1` recibe el valor 55 a una distancia de 3 mm de `p1`. La señal analógica de salida `ao1` recibe el valor 10 a una distancia de 15 mm de `p1`. La señal digital de salida `do1` recibirá el valor 3 mm a partir de `ToPoint p2`. Estos eventos de posición son definidos por la variable `mytriggios`. La variable `mytriggstrgo` establece que se produzcan eventos de posición a una distancia de entre 3 y 15 mm de `p1`. La señal `go2` recibe primero el valor 1 y luego el valor 800000. La señal recibe el valor 4294967295 a una distancia de 4 mm de `ToPoint p2`. Se trata del valor máximo para una señal digital de salida de 32 bits. La variable `mytriggiosdnum` establece que se produzcan tres eventos de posición a una distancia de 10 mm de `p1`. En primer lugar se cambia la señal `go3` a 4294967295, a continuación se cambia `ao2` a 5 y en último lugar se cambia `do2` a 1.

#### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                     | Causa del error  |
|----------------------------|--|
| <code>ERR_NORUNUNIT</code> | Se ha perdido el contacto con el dispositivo de E/S.   |
| <code>ERR_GO_LIM</code>    | El argumento programado <code>setvalue</code> para la señal digital de salida de grupo especificada <code>signalname</code> está fuera de límites. (Se declara en <code>TriggData1</code> , <code>TriggData2</code> o <code>TriggData3</code> ). |
| <code>ERR_AO_LIM</code>    | El argumento programado <code>setvalue</code> para la señal analógica de salida especificada <code>signalname</code> está fuera de límites. (Se declara en <code>TriggData1</code> o <code>TriggData3</code> ).                                  |
| <code>ERR_CONC_MAX</code>  | Se ha superado el número de instrucciones de movimiento seguidas con el argumento <code>\Conc</code> .   |

#### Limitaciones

Si el punto de inicio actual se desvía del habitual, de forma que la longitud de posicionamiento total de la instrucción `TriggLIos` es más corta de lo habitual (es decir, al principio de `TriggLIos` con la posición del robot en el punto final), puede ocurrir que se cumplan varias de las condiciones de disparo, o incluso todas ellas, inmediatamente y en una misma posición. En estos casos, la secuencia en la que se realizan las actividades de disparo no estará definida. La lógica de programa del programa del usuario no puede basarse en una secuencia normal de actividades de disparo para un movimiento incompleto.

La limitación en el número de disparos de la instrucción `TriggLIos` es de 50 para cada instrucción programada. Si los disparos se producen a una distancia más corta, es posible que el sistema no los gestione. Esto depende de cómo se realice el movimiento, la velocidad de TCP utilizada y la cercanía programada entre los disparos. Estas limitaciones existen pero resulta difícil predecir cuándo se producirán estos problemas.

Continúa en la página siguiente

TriggLIOS no puede ejecutarse en un gestor UNDO o en rutinas de RAPID que estén conectadas a los siguientes eventos especiales del sistema: PowerOn, Stop, QStop, Restart, Reset o Step.

#### Limitaciones respecto a la exactitud

Los eventos de E/S basados en una distancia se han diseñado para los puntos de paso (trayectorias de esquina). El uso de puntos de paro daría como resultado una exactitud menor que la especificada a continuación.

En cuanto a la exactitud de los eventos de E/S con distancia y con puntos de paso, se aplica lo siguiente a la hora de activar una salida digital a una distancia específica del punto de inicio o del punto final con las instrucciones TriggLIOS o TriggCIOs:

- La exactitud especificada a continuación es válida cuando se utiliza un `equiplag` positivo<sup>1</sup> que es menor que 40 ms, que equivale al retardo del servo del robot sin cambiar el parámetro *Event Preset Time* del sistema. El retardo puede variar de un tipo de robot a otro.
- La precisión especificada abajo es válida cuando se utiliza un `equiplag` positivo menor que el *Event Preset Time* configurado en los parámetros del sistema.
- La exactitud especificada a continuación no es válida cuando se utiliza un `equiplag` positivo mayor que el *Event Preset Time* configurado en los parámetros del sistema. En este caso, se utiliza un método aproximado que no tiene en cuenta las limitaciones dinámicas del robot. Entonces, debe usarse `SingArea \Wrist` para conseguir una precisión aceptable.
- La exactitud especificada a continuación es válida cuando se utiliza un `equiplag` negativo.

<sup>1</sup> `equiplag` es un `dataobject` de tipo de datos `triggios`

Los valores absolutos típicos en cuanto a la exactitud para activar salidas digitales son de:  $\pm 5$  ms.

Los valores repetidos típicos en cuanto a la exactitud para activar salidas digitales son de:  $\pm 2$  ms.

#### Sintaxis

```
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>','
[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>'];'
```

Continúa en la página siguiente

# 1 Instrucciones

## 1.311 TriggLIOs - Movimientos lineales del robot con eventos de E/S

RobotWare - OS

Continuación

### Información relacionada

| Para obtener más información sobre  | Consulte  |
|---|---|
| Movimientos de ejes del robot con eventos de E/S  | <a href="#">TriggLIOs - Movimientos de ejes del robot con eventos de E/S en la página 949</a> |
| Almacenamiento de condiciones de disparo y actividad de disparo   | <a href="#">triggios - Eventos de posicionamiento, trigg en la página 1806</a>                |
| Almacenamiento de condiciones de disparo y actividad de disparos de una señal digital de grupo compuesta por 32 señales | <a href="#">triggstrgo - Positioning events, trigg en la página 1814</a>                      |
| Almacenamiento de condiciones de disparo y actividad de disparo   | <a href="#">triggiosdnum - Eventos de posicionamiento, trigg en la página 1809</a>            |
| Asignación de objetos de evento   | Manual de referencia técnica - Parámetros del sistema   |
| Movimiento lineal   | Manual de referencia técnica - RAPID Overview   |
| Movimiento en general   | Manual de referencia técnica - RAPID Overview   |
| Definición de carga   | <a href="#">loaddata - Datos de carga en la página 1694</a>                                   |
| Ejemplo de cómo usar TLoad, carga total.  | <a href="#">MoveL - Mueve el robot siguiendo una trayectoria lineal en la página 447</a>      |
| Definición de la carga útil de un robot   | <a href="#">GripLoad - Define la carga útil de un robot en la página 244</a>                  |
| LoadIdentify, rutina de servicio de identificación de carga   | Manual del operador - IRC5 con FlexPendant  |
| Señal de entrada de sistema SimMode para mover el robot en el modo simulado sin carga útil                              | Manual de referencia técnica - Parámetros del sistema   |
| Parámetro de sistema ModalPayloadMode para la activación y la desactivación de la carga útil                            | Manual de referencia técnica - Parámetros del sistema   |
| Path Offset   | Application manual - Controller software IRC5   |



## 1.312 TriggRampAO - Define un evento AO de rampa de posición fija en la trayectoria

### Utilización

TriggRampAO(*Trigg Ramp Analog Output*) se utiliza para definir condiciones y acciones para la aplicación de una rampa ascendente o descendente a un valor de señal analógica de salida en una posición fija a lo largo de la trayectoria de movimiento del robot, con la posibilidad de aplicar una compensación de tiempo para el retardo del equipo externo.

Los datos definidos se utilizan para la implementación o instrucciones TriggL, TriggC o TriggJ posteriores. Aparte de estas instrucciones, también es posible usar TriggRampAO en las instrucciones CapL o CapC.

El tipo de acciones disparo conectado a la misma instrucción TriggL/C/J puede ser TriggRampAO o cualquiera de las instrucciones TriggIO, TriggEquip, TriggSpeed, TriggInt o TriggCheckIO. Se permite cualquier tipo de combinación, si bien sólo se permite una única acción TriggSpeed con la misma señal dentro de la misma instrucción TriggL/C/J.

Esta instrucción sólo puede usarse en la tarea principal T\_ROB1 o, si se cuenta con un sistema *MultiMove*, en las tareas de movimiento.

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción TriggRampAO:

Consulte también [Más ejemplos en la página 970](#).

#### Ejemplo 1

```
VAR trigdata ramp_up;
...
TriggRampAO ramp_up, 0 \Start, 0.1, aolaser1, 8, 15;
MoveL p1, v200, z10, gun1;
TriggL p2, v200, ramp_up, z10, gun1;
```

La señal analógica aolaser1 inicia una rampa ascendente de su valor lógico desde el valor actual hasta el nuevo valor 8, cuando el gun1 de la herramienta se encuentra 0,1 s antes del centro de la trayectoria de esquina en p1. La totalidad de la rampa ascendente se completa mientras el robot se mueve 15 mm.

#### Ejemplo 2

```
VAR trigdata ramp_down;
...
TriggRampAO ramp_down, 15, 0.1, aolaser1, 2, 10;
MoveL p3, v200, z10, gun1;
TriggL p4, v200, ramp_down, z10, gun1;
```

La señal analógica aolaser1 inicia una rampa descendente de su valor lógico desde el valor actual hasta el nuevo valor 2, cuando el TCP de la herramienta gun1 se encuentra 15 mm y 0,1 s antes del centro de la trayectoria de esquina en p4. La totalidad de la rampa ascendente se completa mientras el robot se mueve 10 mm.

*Continúa en la página siguiente*

# 1 Instrucciones

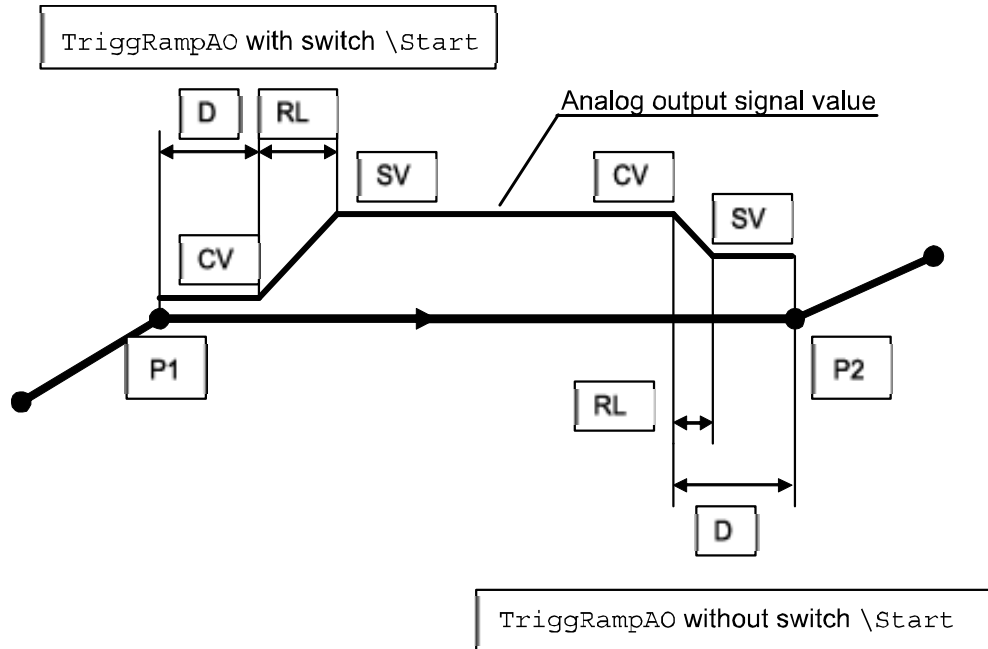
## 1.312 TriggRampAO - Define un evento AO de rampa de posición fija en la trayectoria

RobotWare - OS

Continuación

### Argumentos

```
TriggRampAO TriggData Distance [\Start] | [\Next] EquipLag AOutput
SetValve RampLength [\Time] [\Inhib] [\InhibSetValve] [\Mode]
```



xx0600003433

|    |  |
|----|--|
| D  | Parámetro Distance                                     |
| RL | Parámetro RampLength                                   |
| CV | Valor actual de la señal analógica                     |
| SV | Parámetro SetValve para el valor de la señal analógica |
| P1 | ToPoint para la instrucción de movimiento precedente   |
| P2 | ToPoint para la instrucción TriggL/C/J actual          |

TriggData

Tipo de dato: triggdata

Una variable para almacenar el triggdata devuelto por la instrucción. Estos datos triggdata pueden usarse en las instrucciones TriggL, TriggC, TriggJ, CapL o CapC posteriores.

Distance

Tipo de dato: num

Define la distancia desde el centro de la trayectoria de esquina en la que debe iniciarse la rampa de la salida analógica.

Se especifica como la distancia en mm (valor positivo) desde el punto final (ToPoint) de la trayectoria de movimiento (aplicable si no se utiliza el argumento \Start).

Consulte [Ejecución de programas en la página 969](#) para obtener más detalles.

Continúa en la página siguiente

## 1.312 TriggRampAO - Define un evento AO de rampa de posición fija en la trayectoria

RobotWare - OS

Continuación

[ \Start ]

Tipo de dato: switch

Se utiliza cuando la distancia del argumento `Distance` está relacionada con el punto de inicio del movimiento (`ToPoint` precedente) en lugar del punto final.

[ \Next ]

Tipo de dato: switch

Se utiliza cuando la distancia para el argumento `Distance` avanza hacia el siguiente punto programado. Si la `Distance` es mayor que la distancia al siguiente punto fino, el evento se ejecutará en el punto fino.

EquipLag

### Equipment Lag

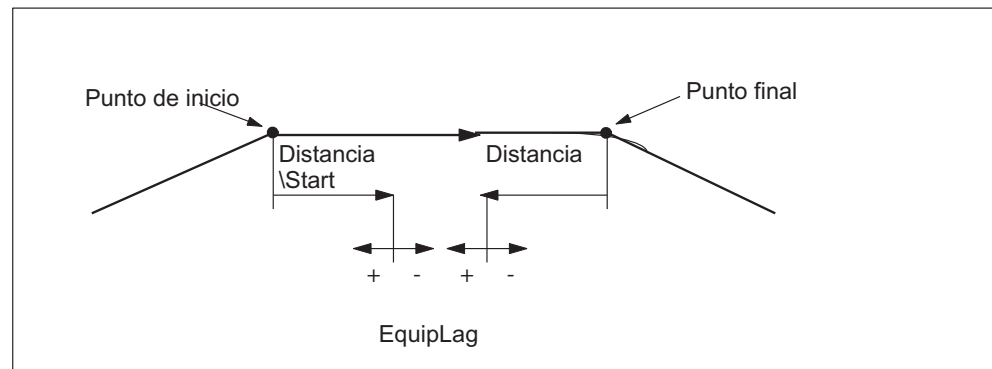
Tipo de dato: num

Especifica el retardo del equipo externo, en segundos.

Para la compensación del retardo de los equipos externos, utilice un valor de argumento positivo. Un valor de argumento positivo significa que el principio de la rampa de la señal AO es realizado por el sistema de robot en un momento especificado antes de que el TCP alcance físicamente el punto de la distancia especificada, respecto del punto inicial o final del movimiento.

Un valor de argumento negativo significa que el inicio de la rampa de velocidad de la señal AO lo realiza el sistema de robot en un momento determinado después de que el TCP haya sobrepasado físicamente el punto de distancia especificado en relación con el punto inicial o final del movimiento.

En la figura se muestra el uso del argumento `EquipLag`.



xx0500002262

AOutput

### Analog Output

Tipo de dato: signalao

El nombre de la señal de salida analógica.

SetValue

Tipo de dato: num

Continúa en la página siguiente

## 1 Instrucciones

---

### 1.312 TriggRampAO - Define un evento AO de rampa de posición fija en la trayectoria

RobotWare - OS

Continuación

El valor hasta el cual debe aumentar o descender en rampa la señal analógica de salida (debe estar dentro del valor de rango lógico permitido para la señal). La rampa comienza con el valor actual de la señal analógica de salida.

RampLength

Tipo de dato: num

La longitud de la rampa en mm a lo largo de la trayectoria de movimientos del TCP.

[ \Time ]

Tipo de dato: switch

Si se utiliza, RampLength especifica el tiempo de rampa en s en lugar de en longitud de la rampa.

Debe utilizarse si una instrucción TriggL, TriggC o TriggJ posterior especifica que el movimiento total debe realizarse por tiempo (argumento \T) en lugar de por velocidad.

[ \Inhib ]

*Inhibit*

Tipo de dato: bool

El nombre de un indicador de variable persistente para la inhibición del valor de la señal en tiempo de ejecución.

Si se utiliza este argumento opcional y el valor actual del indicador especificado es TRUE en la posición y el tiempo del inicio de la rampa de la señal de E/S, la señal especificada (AOutput) cambia a 0.

[ \InhibSetValue ]

*InhibitSetValue*

Tipo de dato: bool, num or dnum

El nombre de una variable persistente de tipo de dato bool, num o dnum o cualquier alias de estos tres tipos de datos.

Este argumento opcional solo puede utilizarse junto con el argumento opcional Inhib.

Si se utiliza este argumento opcional y el valor del indicador de variable persistente utilizado en el argumento opcional Inhib es TRUE en la posición y tiempo para configurar la señal, se lee el valor de la variable persistente utilizado en el argumento opcional InhibSetValue y el valor se utiliza para configurar la señal AOutput.

Si se utiliza una variable persistente booleana, el valor TRUE se traduce al valor 1, y FALSE se traduce al valor 0.

[ \Mode ]

Tipo de dato: triggmode

Se utiliza para especificar diferentes modos de acción al definir disparadores.

*Continúa en la página siguiente*

### Ejecución de programas

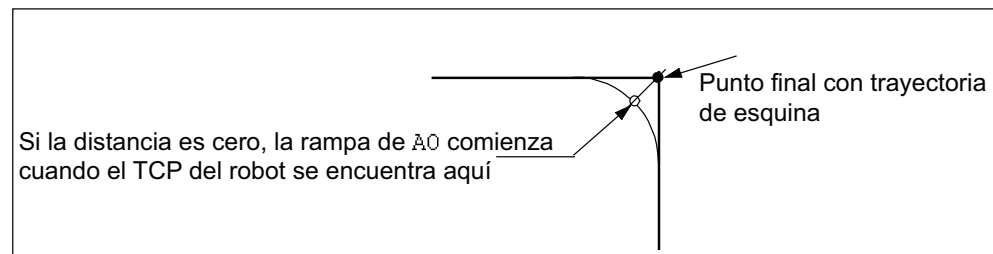
Cuando se ejecuta la instrucción `TriggRampAO`, la condición de disparo se almacena en una variable especificada para el argumento `TriggData`.

A continuación, cuando se ejecuta una de las instrucciones `TriggL`, `TriggC` o `TriggJ`, se aplica lo siguiente en cuanto a las definiciones de `TriggRampAO`:

En la tabla se describe la distancia especificada en el argumento `Distance`:

|                      |  |
|----------------------|--|
| Movimiento lineal    | La distancia en línea recta  |
| Movimiento circular  | La longitud del arco del círculo   |
| Movimiento no lineal | La longitud de arco aproximada a lo largo de la trayectoria (para conseguir la exactitud adecuada, la distancia no debe superar la mitad de la longitud del arco). |

La figura muestra la rampa de AO en una trayectoria de esquina.



xx0600003439

Características de ejecución de programas de `TriggRampAO` en conexión con cualquier `TriggL/C/J`:

- La rampa de la señal AO comienza cuando el robot alcanza el punto `Distance` especificado en la trayectoria del robot (con compensación del valor especificado de `EquipLag`)
- La función de rampa se realiza durante un periodo de tiempo calculado a partir del valor de `RampLength` especificado y la velocidad de TCP programada. El cálculo tiene en cuenta `VelSet`, la redefinición de velocidad manual y el máximo de 250 mm/s en el modo MAN, pero no tiene en cuenta ninguna otra limitación.
- La actualización del valor de señal AO desde el valor inicial (lectura actual) al `SetValue` especificado se realizará cada 10 ms, lo que produce una forma escalonada. Si el tiempo de rampa calculado o el tiempo de rampa especificado es de más de 0,5 s la frecuencia de rampa se ralentizará:
  - $\leq 0,5$  s produce como máximo 50 pasos de 10 ms cada uno
  - $\leq 1$  s produce como máximo 50 pasos de 20 ms cada uno
  - $\leq 1,5$  s produce como máximo 50 pasos de 30 ms cada uno, etcétera

La acción `TriggRampAO` también se realiza en el paso FWD, pero no en el modo de paso BWD.

Continúa en la página siguiente

## 1 Instrucciones

### 1.312 TriggRampAO - Define un evento AO de rampa de posición fija en la trayectoria

RobotWare - OS

Continuación

En cualquier tipo de paro (paro de programa, paro de emergencia), si la función de rampa está activa en esa ocasión:

- En caso de rampa ascendente, el valor de AO cambia momentáneamente a un valor anterior.
- En caso de rampa descendente, el valor de AO cambia momentáneamente al nuevo valor de SetValue.

#### Más ejemplos

A continuación aparecen más ejemplos de cómo usar la instrucción TriggRampAO.

#### Ejemplo 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;
```

En este ejemplo, se realiza tanto la rampa ascendente como la rampa descendente de AO en la misma instrucción TriggL y en la misma trayectoria de movimiento. Funciona sin ninguna interferencia de los valores de AO, siempre y cuando la trayectoria de movimiento sea lo suficientemente larga.

La señal analógica aolaser1 inicia una rampa ascendente de su valor lógico desde el valor actual hasta el nuevo valor 8, cuando el TCP de la herramienta gun1 se encuentra 0,1 s antes del centro de la trayectoria de esquina en p1. La totalidad de la rampa ascendente se completa mientras el robot se mueve 15 mm.

La señal analógica aolaser1 comenzará a reducir su valor lógico desde el valor 8 actual al nuevo valor 2 cuando el TCP de la herramienta gun1 es de 15 mm más 0,1 s antes del centro de la trayectoria en ángulo en p2. El descenso completo se realizará mientras el robot se desplaza 10 mm.

#### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema ERRNO cambiará a:

| Nombre             | Causa del error   |
|--------------------|---|
| ERR_AO_LIM         | El argumento programado SetValue para la señal analógica de salida especificada AOutput está fuera de límite.   |
| ERR_NO_ALIASIO_DEF | La variable de señal es una variable declarada en RAPID y no se ha conectado a una señal de E/S definida en la configuración de E/S con la instrucción AliasIO. |

#### Limitaciones

El valor de la señal analógica de salida no se compensa por la reducción de velocidad del TCP en la trayectoria de esquina ni durante otras fases de aceleración o deceleración (AO no es proporcional a la velocidad del TCP).

Continúa en la página siguiente

## 1.312 TriggRampAO - Define un evento AO de rampa de posición fija en la trayectoria *RobotWare - OS* *Continuación*

Sólo el punto de inicio de la rampa de AO se ejecutará en la posición especificada de la trayectoria. La rampa ascendente o descendente se realizará con un “cálculo muerto” y con alta exactitud:

- A una velocidad constante, la desviación del fin de la rampa de AO con respecto al especificado será reducida.
- Durante las fases de aceleración o deceleración, como por ejemplo cerca de puntos de paro, la desviación será mayor.
- Recomendación: Utilice trayectorias de esquina antes de las rampas ascendentes y a continuación de las rampas descendentes.

Si utiliza dos o más TriggRampAO en la misma señal analógica de salida y en conexión con la misma instrucción TriggL/C/J y las dos RampLength o varias de ellas están situadas en la misma parte de la trayectoria del robot, los valores de AO interactuarán entre sí.

El evento AO de rampa relacionada con la posición (+/- tiempo) comienza cuando se pase más allá del punto ToPoint anterior, si el valor especificado de Distance a partir del punto ToPoint no está dentro de la longitud del movimiento de la instrucción TriggL/C/J actual. El evento AO de rampa relacionada con la posición (+/- tiempo) comienza cuando se pase más allá del punto ToPoint anterior, si el valor especificado de Distance desde el punto ToPoint anterior no está dentro de la longitud del movimiento de la instrucción TriggL/C/J actual (con el argumento \Start).

No se admite el reinicio de la función AO de rampa a continuación de ningún tipo de paro (paro de programa, paro de emergencia, etc.).

En caso de reinicio tras una caída de alimentación, la instrucción TriggL/C/J comienza por el principio desde la posición de caída de alimentación actual.

TriggRampAO no puede ejecutarse en un gestor UNDO o en rutinas de RAPID que estén conectadas a los siguientes eventos especiales del sistema: PowerOn, Stop, QStop, Restart, Reset o Step.

### Sintaxis

```
TriggRampAO
[ TriggData ':= ' ] < variable (VAR) of triggdata > ', '
[ Distance ':= ' ] < expression (IN) of num >
[ '\ ' Start ] | [ '\ ' Next ] ', '
[ 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 triggmodes > ] ';'

```

*Continúa en la página siguiente*

# 1 Instrucciones

1.312 TriggRampAO - Define un evento AO de rampa de posición fija en la trayectoria

RobotWare - OS

Continuación

## Información relacionada

| Para obtener más información sobre                   | Consulte  |
|--|---|
| Utilización de disparadores                          | <a href="#">TriggL - Movimiento lineal del robot con eventos en la página 940</a><br><a href="#">TriggC - Movimiento circular del robot con eventos en la página 895</a><br><a href="#">TriggJ - Movimientos de ejes del robot a partir de eventos en la página 932</a> |
| Definición de otros disparos                         | <a href="#">TriggEquip - Define un evento de E/S basado en la posición y el tiempo en la trayectoria en la página 914</a>   |
| Almacenamiento de datos de disparo                   | <a href="#">triggdata - Eventos de posicionamiento, trigg en la página 1805</a><br><a href="#">triggmode - Disparar modo de acción en la página 1811</a>  |
| Establecimiento de una señal analógica de salida     | <a href="#">SetAO - Cambia el valor de una señal analógica de salida en la página 694</a><br><a href="#">signalxx - Señales digitales y analógicas en la página 1763</a>  |
| Configuración del tiempo preestablecido para eventos | <a href="#">Manual de referencia técnica - Parámetros del sistema</a>   |



1.313 TriggSpeed - Define la velocidad del TCP en proporción a una salida analógica con un evento de escala fija de posición-tiempo  
RobotWare - OS

### 1.313 TriggSpeed - Define la velocidad del TCP en proporción a una salida analógica con un evento de escala fija de posición-tiempo

#### Utilización

TriggSpeed se utiliza para definir condiciones y acciones para el control de una señal analógica de salida cuyo valor de salida es proporcional a la velocidad real del TCP. El inicio, el escalado y la finalización de la salida analógica pueden especificarse en una posición-tiempo fija a lo largo de la trayectoria de movimientos del robot. Es posible utilizar una compensación de tiempo para el retardo del equipo externo en el inicio, el escalado y la finalización de la salida analógica y también para los cambios de velocidad del robot.

Los datos definidos se utilizan en una o varias instrucciones TriggL, TriggC o TriggJ posteriores.

Esta instrucción sólo puede usarse en la tarea principal T\_ROB1 o, si se cuenta con un sistema MultiMove, en las tareas de movimiento.

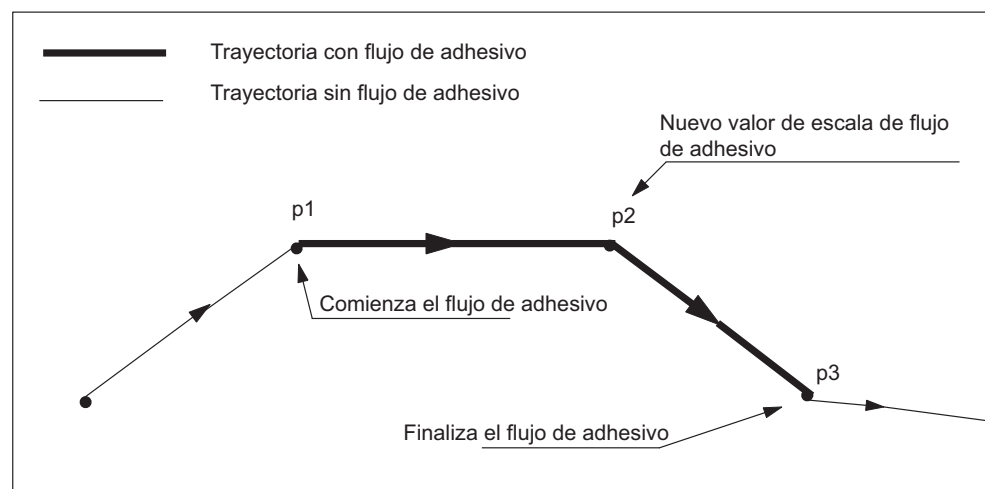
#### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción TriggSpeed:  
Consulte también [Más ejemplos en la página 979](#).

#### Ejemplo 1

```
VAR trigdata 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;
```

La figura siguiente muestra un ejemplo de secuencia con TriggSpeed.



xx0500002329

El flujo de adhesivo (salida analógica glue\_ao) con valor de escala de 0.8 comienza cuando el TCP está 0.05 s antes del punto p1, el nuevo valor de escala

*Continúa en la página siguiente*

# 1 Instrucciones

---

1.313 TriggSpeed - Define la velocidad del TCP en proporción a una salida analógica con un evento de escala fija de posición-tiempo

RobotWare - OS

Continuación

de flujo de adhesivo 1 cuando el TCP está 10 mm y 0.05 s antes del punto p2, y el flujo de adhesivo termina (valor de escala 9) cuando el TCP está 0.05 s antes del punto p3.

Cualquier reducción de velocidad del robot recibe una compensación de tiempo de forma que la señal de salida analógica glue\_ao se vea afectada 0.04 s antes de que se produzca la reducción de velocidad del TCP.

En caso de rebasamiento del valor lógico calculado de la salida analógica glue\_ao, se activa la señal de salida digital glue\_err. Si no hay más desbordamiento, glue\_err se pone a cero.

---

## Argumentos

```
TriggSpeed TriggData Distance [\Start] | [\Next] ScaleLag AOp
ScaleValue [\DipLag] [\ErrDO] [\Inhib] [\InhibSetValue]
[\Mode]
```

TriggData

Tipo de dato: triggdata

Una variable para almacenar el triggdata devuelto por la instrucción. Estos datos triggdata se utilizan a continuación en las instrucciones TriggL, TriggC o TriggJ posteriores.

Distance

Tipo de dato: num

Define la posición en la trayectoria para el cambio del valor de salida analógico. Se especifica como la distancia en mm (valor positivo) desde el punto final de la trayectoria de movimiento (aplicable si no se utiliza el argumento \Start).

Consulte [Ejecución de programas en la página 978](#) para obtener más detalles.

[ \Start ]

Tipo de dato: switch

Se utiliza cuando la distancia del argumento Distance comienza en el punto de inicio del movimiento en lugar del punto final.

[ \Next ]

Tipo de dato: switch

Se usa cuando la distancia para el argumento Distance se envía al siguiente punto programado.

Este argumento puede utilizarse para mover el cambio del valor de la salida analógica después del punto final de la trayectoria de movimiento. Solo debe utilizarse si se desea ajustar el cambio de la señal para que esté cerca después

Continúa en la página siguiente

## 1.313 TriggSpeed - Define la velocidad del TCP en proporción a una salida analógica con un evento de escala fija de posición-tiempo

RobotWare - OS

Continuación

del ToPoint de la instrucción de movimiento usada (instrucciones TriggL, TriggC o TriggJ).



### Nota

El programador debe tener cuidado al utilizar el argumento \Next. En función de la distancia y argumentos usados en la siguiente condición de disparo, puede que haya un solape en las acciones, y puede que el resultado no sea el esperado.

ScaleLag

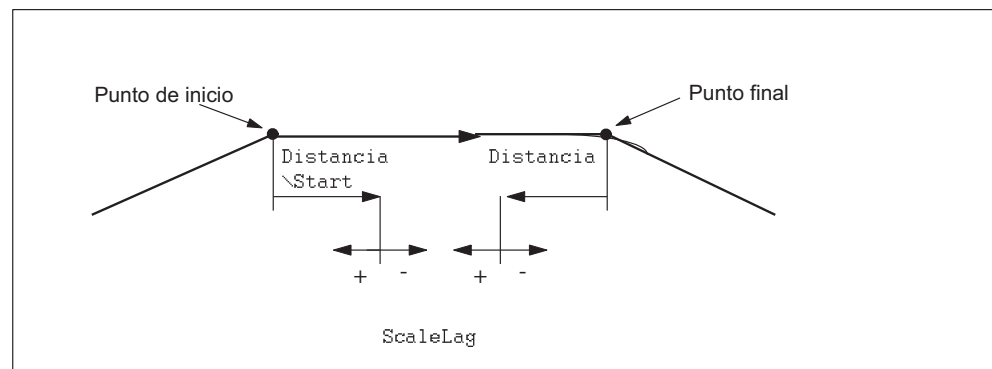
Tipo de dato: num

Especifique el retardo de tiempo en s (valor positivo) del equipo externo antes del cambio del valor de la salida analógica (inicio, escalado y finalización).

Para la compensación del retardo del equipo externo, el valor de este argumento significa que la señal de salida es activada por el robot en el momento especificado, antes de que el TCP físico alcance la distancia especificada respecto del punto de inicio o final del movimiento.

Este argumento también puede usarse para extender la salida analógica más allá del punto final. Establezca el tiempo en segundos durante el cual el robot debe mantener la señal analógica. Establezca el tiempo con un signo negativo. El límite es -0,10 segundos.

En la figura siguiente se ilustra el uso del argumento ScaleLag



xx0500002330

AOp

### Analog Output

Tipo de dato: signalao

El nombre de la señal de salida analógica.

ScaleValue

Tipo de dato: num

El valor de escala de la señal de salida analógica.

El valor físico de salida de la señal analógica es calculado por el robot:

- Valor lógico de salida = Valor de escala \* velocidad actual del TCP en mm/s.

Continúa en la página siguiente

# 1 Instrucciones

1.313 TriggSpeed - Define la velocidad del TCP en proporción a una salida analógica con un evento de escala fija de posición-tiempo

RobotWare - OS

Continuación

- Valor físico de salida = De acuerdo con la definición de la configuración de la señal analógica de salida actual, tomando como entrada el valor lógico de salida indicado más arriba.

[ \DipLag ]

Tipo de dato: num

Especifique el retardo de tiempo en s (valor positivo) del equipo externo antes del cambio del valor de la salida analógica debido a los cambios de velocidad del robot.

Para la compensación del retardo del equipo externo, el valor de este argumento significa que la señal de salida analógica es activada por el robot en el momento especificado, antes de que se produzca la reducción de velocidad del TCP.



## Nota

Este argumento sólo puede ser utilizado por el robot en la primera instrucción TriggSpeed (en combinación con una de las instrucciones TriggL, TriggC o TriggJ) en una secuencia de varias instrucciones TriggSpeed. El primer valor de argumento especificado es válido para todas las demás instrucciones TriggSpeed de la secuencia.

[ \ErrDO ]

## Error Digital Output

Tipo de dato: signaldo

El nombre de la señal de salida digital para informar del desbordamiento del valor analógico.

Si, durante el movimiento, el cálculo del valor lógico de salida analógica de la señal indicada en el argumento AOp da lugar a un desbordamiento debido a un exceso de velocidad, esta señal se activa y el valor físico de la salida analógica se reduce al valor máximo. Si no hay más desbordamiento, la señal se pone a cero.



## Nota

Este argumento sólo puede ser utilizado por el robot en la primera instrucción TriggSpeed (en combinación con una de las instrucciones TriggL, TriggC o TriggJ) en una secuencia de varias instrucciones TriggSpeed. El primer valor de argumento especificado es válido para todas las demás instrucciones TriggSpeed de la secuencia.

[ \Inhib ]

## Inhibit

Tipo de dato: bool

El nombre de un indicador de variable persistente para la inhibición del valor de la señal en tiempo de ejecución.

Continúa en la página siguiente

1.313 TriggSpeed - Define la velocidad del TCP en proporción a una salida analógica con un evento de escala fija de posición-tiempo

RobotWare - OS

Continuación

Si se utiliza este argumento opcional y el valor actual del indicador especificado es TRUE en el momento del establecimiento de la señal analógica, la señal especificada AOp se cambia a 0 en lugar del valor especificado.



### Nota

Este argumento sólo puede ser utilizado por el robot en la primera instrucción TriggSpeed (en combinación con una de las instrucciones TriggL, TriggC o TriggJ) en una secuencia de varias instrucciones TriggSpeed. El primer valor de argumento especificado es válido para todas las demás instrucciones TriggSpeed de la secuencia.

[ \InhibSetValue ]

### InhibitSetValue

Tipo de dato: bool, num or dnum

El nombre de una variable persistente de tipo de dato bool, num o dnum o cualquier alias de estos tres tipos de datos.

Este argumento opcional solo puede utilizarse junto con el argumento opcional Inhib.

Si se utiliza este argumento opcional y el valor del indicador de variable persistente utilizado en el argumento opcional Inhib es TRUE en la posición y tiempo para configurar la señal, se lee el valor de la variable persistente utilizado en el argumento opcional InhibitSetValue y el valor se utiliza para configurar la señal AOp.

Si se utiliza una variable persistente booleana, el valor TRUE se traduce al valor 1, y FALSE se traduce al valor 0.



### Nota

Este argumento sólo puede ser utilizado por el robot en la primera instrucción TriggSpeed (en combinación con una de las instrucciones TriggL, TriggC o TriggJ) en una secuencia de varias instrucciones TriggSpeed. El primer valor de argumento especificado es válido para todas las demás instrucciones TriggSpeed de la secuencia.

[ \Mode ]

Tipo de dato: trigemode

Se utiliza para especificar diferentes modos de acción al definir disparadores.



### Nota

Este argumento sólo puede ser utilizado por el robot en la primera instrucción TriggSpeed (en combinación con una de las instrucciones TriggL, TriggC o TriggJ) en una secuencia de varias instrucciones TriggSpeed. El primer valor de argumento especificado es válido para todas las demás instrucciones TriggSpeed de la secuencia.

Continúa en la página siguiente

# 1 Instrucciones

1.313 TriggSpeed - Define la velocidad del TCP en proporción a una salida analógica con un evento de escala fija de posición-tiempo

RobotWare - OS

Continuación

## Ejecución de programas

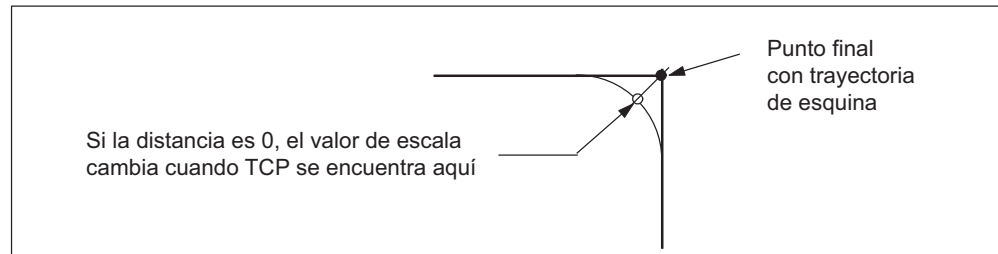
Cuando se ejecuta la instrucción `TriggSpeed`, la condición de disparo se almacena en una variable especificada para el argumento `TriggData`.

A continuación, cuando se ejecuta una de las instrucciones `TriggL`, `TriggC` o `TriggJ`, se aplica lo siguiente en cuanto a las definiciones de `TriggSpeed`:

En cuanto a la distancia especificada en el argumento `Distance`, consulte la tabla siguiente:

|                      |  |
|----------------------|--|
| Movimiento lineal    | La distancia en línea recta  |
| Movimiento circular  | La longitud del arco del círculo   |
| Movimiento no lineal | La longitud de arco aproximada a lo largo de la trayectoria (para conseguir la exactitud adecuada, la distancia no debe superar la mitad de la longitud del arco). |

La figura siguiente muestra el evento de valor de escala fija de posición-tiempo en una trayectoria de esquina.



xx0500002331

El evento de valor de escala dependiente de posición-tiempo se genera cuando se atraviesa el punto de inicio (punto final) si la distancia especificada respecto del punto final (punto de inicio) no se encuentra dentro de la longitud de movimiento de la instrucción actual (`TriggL`, `TriggC` o `TriggJ`).

La primera instrucción `TriggSpeed` utilizada por una de las instrucciones `TriggL`, `TriggC` o `TriggJ` creará internamente en el sistema un proceso con el mismo nombre que la señal de salida analógica. El mismo proceso será utilizado por todas las instrucciones `TriggL`, `TriggC` o `TriggJ` posteriores que hagan referencia al mismo nombre de señal, configurado por una instrucción `TriggSpeed`.

El proceso cambia inmediatamente la salida analógica a 0 en el caso de un paro de emergencia de programa. En el caso de un paro de programa, la señal de salida analógica permanece proporcional a la velocidad del TCP hasta que el robot se detiene. El proceso se mantiene "vivo" y listo para un reinicio. Cuando el robot

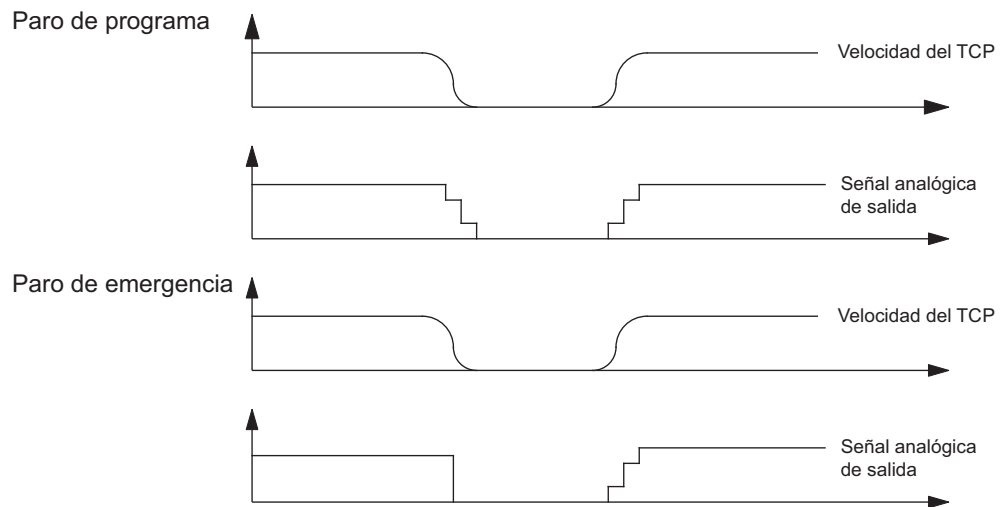
Continúa en la página siguiente

## 1.313 TriggSpeed - Define la velocidad del TCP en proporción a una salida analógica con un evento de escala fija de posición-tiempo

RobotWare - OS

Continuación

reanuda su funcionamiento, la señal es proporcional a la velocidad del TCP desde el mismo momento del inicio.



xx0500002332

El proceso “muere” tras gestionar un evento de escala con valor 0, siempre y cuando no haya ninguna instrucción TriggL, TriggC o TriggJ en la cola en ese momento.

### Más ejemplos

A continuación aparecen más ejemplos de la instrucción TriggSpeed.

#### Ejemplo 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;
```

La señal analógica de salida `flowsignal` se establece en un valor lógico =  $(0.5 * \text{velocidad real del TCP en mm/s}) / 0.05$  s antes de que el TCP atravesase un punto situado 10 mm después del punto de inicio `p`. El valor de salida se ajusta de forma proporcional a la velocidad real del TCP durante el movimiento hacia `p2`.

...

```
TriggL p3, v500, flow, z10, tool1;
```

El robot se mueve de `p2` a `p3`, manteniendo el valor de la salida analógica de forma proporcional a la velocidad actual del TCP. El valor de la salida analógica se reduce 0.03 s antes de que el robot reduzca la velocidad del TCP mientras pasa por la trayectoria de esquina `z10`.

#### Ejemplo 2

```
VAR triggdata glueflow;
VAR triggdata glueflowend;
TriggSpeed glueflow, 0, 0.05, glue_ao, 1;
TriggSpeed glueflowend, 25 \Next, 0, glue_ao, 0;
TriggL p1, v500, glueflow, z50, gun1;
TriggL p2, v500, glueflow, z50, gun1;
TriggL p3, v500, glueflowend, z50, gun1;
```

Continúa en la página siguiente

# 1 Instrucciones

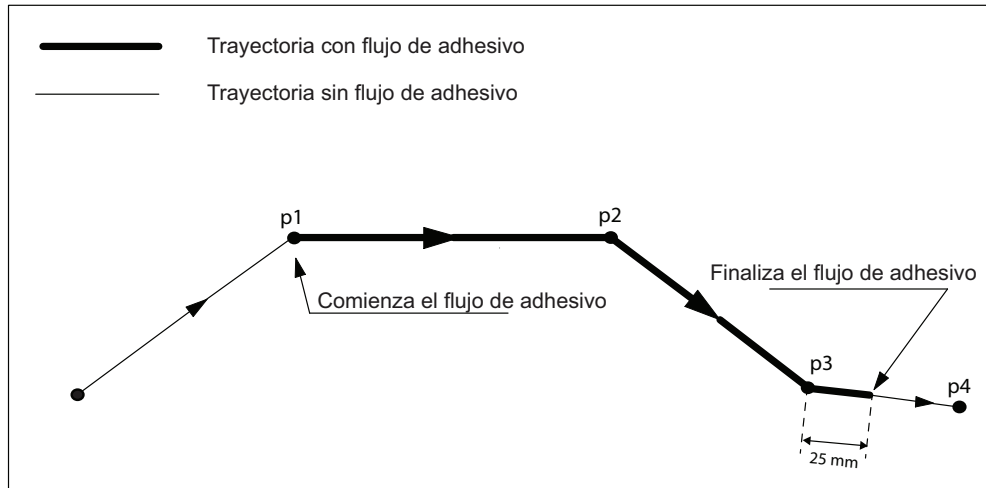
1.313 TriggSpeed - Define la velocidad del TCP en proporción a una salida analógica con un evento de escala fija de posición-tiempo

RobotWare - OS

Continuación

```
MoveL p4, v500, z50, gun1;
```

La figura siguiente muestra un ejemplo de secuencia con TriggSpeed y uso del argumento \Next



xx1800000010

El flujo de adhesivo (salida analógica glue\_ao) con valor de escala 0.8 comienza cuando TCP está 0.05 s antes del punto p1. El flujo de adhesivo termina (valor de escala 0) cuando TCP está 25 mm después del punto p3.

## Limitaciones

A continuación se muestran las limitaciones de la instrucción TriggSpeed.

### Exactitud de un evento de valor de escala dependiente de posición-tiempo

Los valores absolutos típicos en cuanto a la exactitud de los eventos de valor de escala son de  $\pm 5$  ms.

Los valores de repetición típicos en cuanto a la exactitud de los eventos de valor de escala son de  $\pm 2$  ms.

### Exactitud de la adaptación a los cambios de velocidad del TCP (fases de deceleración y aceleración)

Los valores absolutos típicos en cuanto a la exactitud de la adaptación a los cambios de velocidad del TCP son de  $\pm 5$  ms.

Los valores de repetición típicos en cuanto a la exactitud de la adaptación a los cambios de velocidad del TCP son de  $\pm 2$  ms (el valor depende del valor configurado en *Path resolution*).

Continúa en la página siguiente



1.313 TriggSpeed - Define la velocidad del TCP en proporción a una salida analógica con un evento de escala fija de posición-tiempo

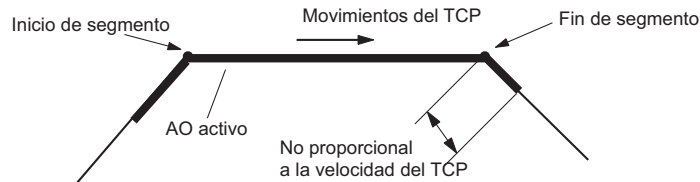
RobotWare - OS

Continuación

Negativo ScaleLag

Si se usa un valor negativo en el parámetro `ScaleLag` para trasladar la escala cero a la siguiente orden de movimiento, la señal de salida digital no se pone a cero si se produce un paro de programa. Los paros de emergencia siempre ponen a cero la señal analógica.

La señal analógica deja de ser proporcional a la velocidad del TCP en el punto final de la orden de movimiento.



xx0500002333

TriggSpeed no puede ejecutarse en un gestor UNDO o en rutinas de RAPID que estén conectadas a los siguientes eventos especiales del sistema: PowerOn, Stop, QStop, Restart, Reset o Step.

Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre             | Causa del error   |
|--------------------|---|
| ERR_NO_ALIASIO_DEF | La variable de señal es una variable declarada en RAPID. No se ha conectado a una señal E/S definida en la configuración de E/S con la instrucción <code>AliasIO</code> . |

Dos órdenes de movimiento consecutivas dadas con instrucciones `TriggL/TriggSpeed`. Un valor negativo en el parámetro `ScaleLag` hace posible trasladar el evento de escala de la primera orden de movimiento al principio de la

Continúa en la página siguiente

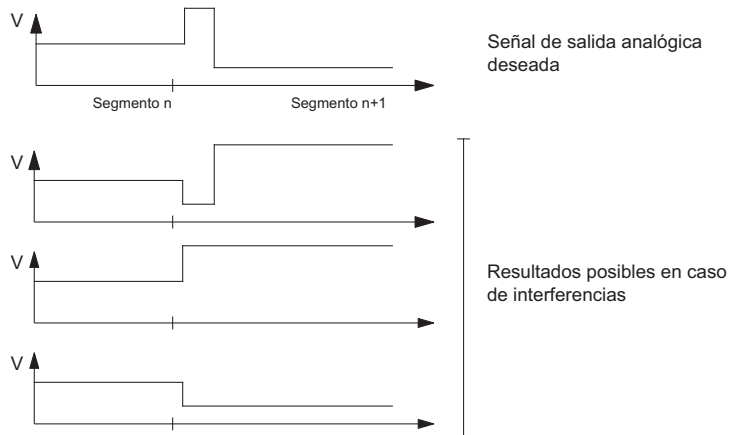
# 1 Instrucciones

1.313 TriggSpeed - Define la velocidad del TCP en proporción a una salida analógica con un evento de escala fija de posición-tiempo

RobotWare - OS

Continuación

segunda orden de movimiento. Si la segunda orden de movimiento se escala al principio, no hay ningún control si las dos escalas interfieren.



xx0500002334

## Parámetros del sistema relacionados

El parámetro del sistema *Event Preset Time* se utiliza para retrasar el robot para permitir la activación y el control de equipos externos antes de que el robot atraviese la posición.

En la tabla siguiente se muestra la recomendación para la configuración del parámetro del sistema *Event Preset Time*, con un valor de retardo de servo típico de 0,040 s.

| ScaleLag          | DipLag             | Valor de <i>Event Preset Time</i> necesario para evitar los errores de tiempo de ejecución | Valor de <i>Event Preset Time</i> recomendado para obtener la máxima exactitud |
|-------------------|--------------------|--|--|
| ScaleLag > DipLag | Siempre            | DipLag, si DipLag > ServoLag   | ScaleLag en s más 0,090 s  |
| ScaleLag < DipLag | DipLag < Servo Lag | - " -  | 0.090 s  |
| - " -             | DipLag > Servo Lag | - " -  | DipLag en s más 0.030 s  |

## Sintaxis

```
TriggSpeed
[ TriggData ':' ] < variable (VAR) of triggdata> ','
[ Distance' ':' ] < expression (IN) of num>
[ '\ Start ] | [ '\ Next ] ','
[ 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> ]
```

Continúa en la página siguiente

## 1.313 TriggSpeed - Define la velocidad del TCP en proporción a una salida analógica con un evento de escala fija de posición-tiempo

RobotWare - OS

Continuación

```
[ '\ ' Inhib' :=' < persistent ( PERS ) of bool > ]
[ '\ ' InhibSetValue' :=' < persistent ( PERS ) of anytype> ]
[ '\ ' Mode' :=' < expression ( IN ) of triggmode> ] ';' ]
```

### Información relacionada

| Para obtener más información sobre | Consulte  |
|------------------------------------|---|
| Utilización de disparadores        | <a href="#">TriggL - Movimiento lineal del robot con eventos en la página 940</a><br><a href="#">TriggC - Movimiento circular del robot con eventos en la página 895</a><br><a href="#">TriggJ - Movimientos de ejes del robot a partir de eventos en la página 932</a>   |
| Definición de otros disparos       | <a href="#">TriggIO - Define un evento de E/S de posición o tiempo fijos cerca de un punto de paro en la página 926</a><br><a href="#">TriggInt - Define una interrupción dependiente de una posición en la página 921</a><br><a href="#">TriggEquip - Define un evento de E/S basado en la posición y el tiempo en la trayectoria en la página 914</a> |
| Almacenamiento de datos de disparo | <a href="#">triggdata - Eventos de posicionamiento, trigg en la página 1805</a><br><a href="#">triggmode - Disparar modo de acción en la página 1811</a>  |
| Configuración de Event preset time | Manual de referencia técnica - Parámetros del sistema   |
| Advanced RAPID                     | Application manual - Controller software IRC5   |

# 1 Instrucciones

---

1.314 TriggStopProc - Genera datos de reinicio para las señales de disparo ante paros  
RobotWare - OS

## 1.314 TriggStopProc - Genera datos de reinicio para las señales de disparo ante paros

---

### Utilización

La instrucción `TriggStopProc` crea un proceso interno de supervisión en el sistema para la puesta a cero de las señales de proceso especificadas y la generación de datos de reinicio en una variable persistente especificada, cada vez que se detiene el programa (`STOP`) o se produce un paro de emergencia (`QSTOP`) en el sistema.

`TriggStopProc` y el tipo de dato `restartdata` han sido creados para su uso en el reinicio tras un paro de programa (`STOP`) o un paro de emergencia (`QSTOP`) de las instrucciones del propio proceso definidas en `RAPID` (rutinas `NOSTEPIN`).

En una rutina de evento `RESTART` definida por el usuario, es posible analizar los datos de reinicio actuales, retroceder por la trayectoria con la instrucción `StepBwdPath` y activar señales de proceso adecuadas antes del reinicio del movimiento.

Esta instrucción sólo puede usarse en la tarea principal `T_ROB1` o, si se cuenta con un sistema *MultiMove*, en las tareas de movimiento.

Recuerde que en el caso de los sistemas *MultiMove*, sólo un proceso de soporte `TriggStopProc` con el nombre de señal "shadow" (argumento `ShadowDO`) puede estar activa en el sistema cada vez. Esto significa que `TriggStopProc` supervisa el paro de programa o el paro de emergencia en la tarea de programa en la que fue ejecutado por última vez.

---

### Argumentos

`TriggStopProc RestartRef [\DO] [\G01] [\G02] [\G03] [\G04] ShadowDO`

`RestartRef`

#### *Restart Reference*

Tipo de dato: `restartdata`

La variable persistente en la que estarán disponibles los datos de reinicio después de cada detención de la ejecución del programa.

`[\DO1]`

#### *Digital Output 1*

Tipo de dato: `signaldo`

La variable de señal de una señal digital de proceso que debe ponerse a cero y supervisarse en los datos de reinicio cuando se detiene la ejecución del programa.

`[\G01]`

#### *Group Output 1*

Tipo de dato: `signalgo`

La variable de señal de una señal digital de grupo de proceso que debe ponerse a cero y supervisarse en los datos de reinicio cuando se detiene la ejecución del programa.

*Continúa en la página siguiente*

## 1.314 TriggStopProc - Genera datos de reinicio para las señales de disparo ante paros RobotWare - OS Continuación

[\GO2]

### *Group Output 2*

Tipo de dato: `signalgo`

La variable de señal de una señal digital de grupo de proceso que debe ponerse a cero y supervisarse en los datos de reinicio cuando se detiene la ejecución del programa.

[\GO3]

### *Group Output 3*

Tipo de dato: `signalgo`

La variable de señal de una señal digital de grupo de proceso que debe ponerse a cero y supervisarse en los datos de reinicio cuando se detiene la ejecución del programa.

[\GO4]

### *Group Output 4*

Tipo de dato: `signalgo`

La variable de señal de una señal digital de grupo de proceso que debe ponerse a cero y supervisarse en los datos de reinicio cuando se detiene la ejecución del programa.

Uno de los parámetros opcionales D01, GO1 . . . GO4 debe ser utilizado.

ShadowDO

### *Shadow Digital Output*

Tipo de dato: `signaldo`

La variable de señal de la señal digital que debe reflejar si el proceso está activo a lo largo de la trayectoria del robot.

Esta señal no será puesta a cero por el proceso `TriggStopProc` ante un `STOP` o `QSTOP`, pero sus valores se reflejarán en `restartdata`.

---

## Ejecución de programas

### Configuración y ejecución de `TriggStopProc`

La llamada a `TriggStopProc` debe realizarse desde los dos lugares siguientes:

- La rutina de evento `START` o la parte inicial del programa (el cambio del `PP` a `main` elimina el proceso interno para `TriggStopProc`)
- La rutina de evento `POWERON` (el apagado elimina el proceso interno para `TriggStopProc`)

El nombre interno del proceso para `TriggStopProc` es el mismo que el nombre de señal del argumento `ShadowDO`. Si `TriggStopProc`, con el mismo nombre de señal en el argumento `ShadowDO`, se ejecuta dos veces desde la misma tarea de programa u otra, sólo estará activo el último `TriggStopProc` ejecutado.

La ejecución de `TriggStopProc` sólo inicia la supervisión de las señales de E/S con los paros `STOP` y `QSTOP`.

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.314 TriggStopProc - Genera datos de reinicio para las señales de disparo ante paros

RobotWare - OS

Continuación

### Paro de programa STOP

El proceso `TriggStopProc` comprende los pasos siguientes:

- 1 Se espera hasta que el robot se haya detenido a lo largo de la trayectoria.
- 2 Se almacena el valor actual (valor previo de acuerdo con `restartdata`) de todas las señales de proceso utilizadas. Se ponen a cero todas las señales de proceso utilizadas, excepto `ShadowDO`.
- 3 Haga lo siguiente durante el próximo intervalo de tiempo, aproximadamente 500 ms, si algunas señales de proceso cambian su valor durante este periodo:
  - Guarde de nuevo el valor actual (valor posterior acorde con `restatdata`)
  - Poner la señal a cero excepto `ShadowDO`
  - Contar el número de transicciones de valor (flancos) de la señal `ShadowDO`
- 4 Se actualiza la variable persistente especificada con el dato de reinicio.

### Paro de emergencia (QSTOP)

El proceso `TriggStopProc` comprende los pasos siguientes:

- 1 Se hace el paso siguiente lo antes posible.
- 2 Se almacena el valor actual (valor previo de acuerdo con `restartdata`) de todas las señales de proceso utilizadas. Se ponen a cero todas las señales de proceso utilizadas, excepto `ShadowDO`.
- 3 Haga lo siguiente durante el próximo intervalo de tiempo, aproximadamente 500 ms, si algunas señales de proceso cambian su valor durante este periodo:
  - Guarde de nuevo su valor actual (valor posterior acorde con `restatdata`)
  - Poner la señal a cero excepto `ShadowDO`
  - Contar el número de transicciones de valor (flancos) de la señal `ShadowDO`
- 4 Se actualiza la variable persistente especificada con el dato de reinicio.

### Área crítica para el reinicio del proceso

Tanto el servo del robot como el equipo externo presentan ciertos retardos. Todas las instrucciones de la familia `Trigg` han sido diseñadas de forma que todas las señales se establezcan en lugares adecuados a lo largo de la trayectoria del robot, independientemente de los distintos retardos en el equipo externo, con el fin de obtener los mejores resultados de proceso posibles. Por ello, el establecimiento de las señales de E/S pueden retrasarse internamente en el sistema de 0 a 80 ms, una vez que el robot se detiene con un paro de programa (STOP) o tras el registro de un paro de emergencia (QSTOP). Debido a esta desventaja de las funciones de reinicio, tanto el valor previo como el valor posterior, así como los flancos de la señal correspondiente, se introducen en los datos de reinicio.

Si este intervalo crítico de 0 a 80 ms coincide con los casos de proceso de aplicación siguientes, resulta difícil realizar un buen reinicio de proceso:

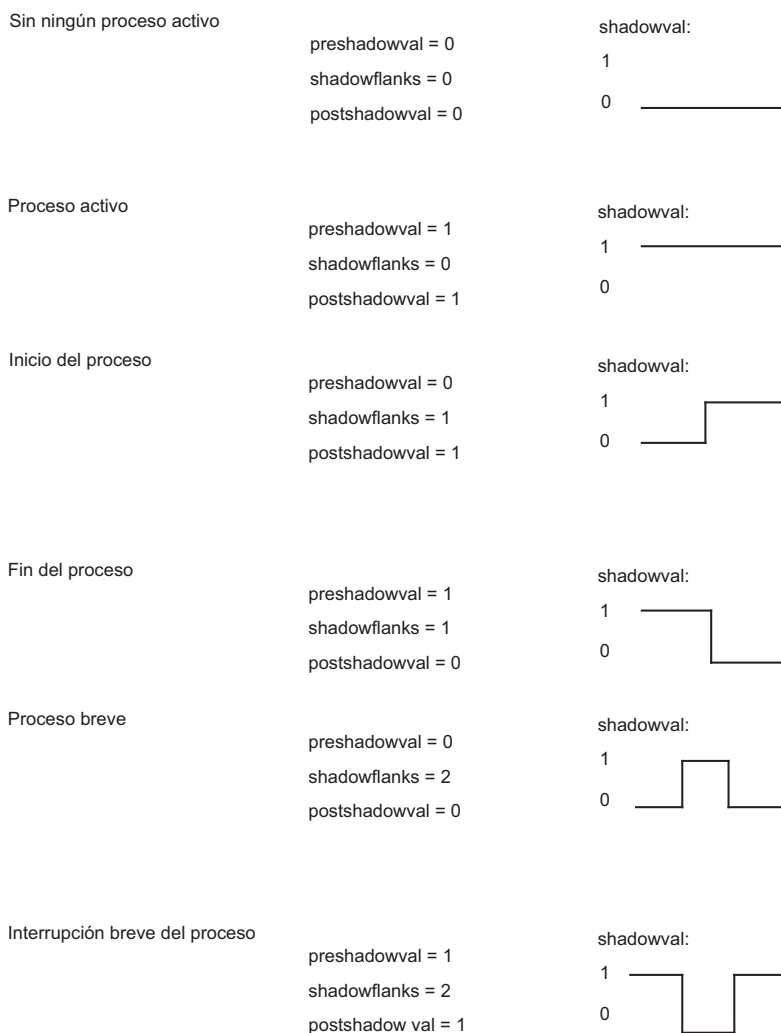
- Al principio del proceso de aplicación
- Al final del proceso de aplicación

Continúa en la página siguiente

## 1.314 TriggStopProc - Genera datos de reinicio para las señales de disparo ante paros RobotWare - OS Continuación

- Durante un proceso de aplicación breve
- Durante una breve interrupción en el proceso de aplicación

En la figura siguiente se muestran las fases de proceso de STOP o QSTOP dentro del intervalo crítico de 0-80 ms



xx0500002326

### Realización de un reinicio

Un reinicio de las instrucciones del proceso (rutinas NOSTEPIN) a lo largo de la trayectoria del robot debe realizarse en una rutina de evento RESTART.

La rutina de evento RESTART puede componerse de los pasos siguientes:

|    | Acción   |
|----|--|
| 1. | Tras un paro QSTOP, la recuperación de la trayectoria se realiza al iniciarse el programa. |

Continúa en la página siguiente

# 1 Instrucciones

## 1.314 TriggStopProc - Genera datos de reinicio para las señales de disparo ante paros

RobotWare - OS

Continuación

|    | Acción   |
|----|--|
| 2. | Se analizan los datos de reinicio desde el último paro STOP o QSTOP.   |
| 3. | Se determina la estrategia de reinicio del proceso partiendo del resultado del análisis, por ejemplo: <ul style="list-style-type: none"><li>• Proceso activo, se procesa el reinicio.</li><li>• Proceso inactivo, no se procesa el reinicio.</li><li>• Se realizan las acciones adecuadas en función del tipo de aplicación de proceso:<ul style="list-style-type: none"><li>- Inicio del proceso</li><li>- Fin del proceso</li><li>- Proceso breve</li><li>- Interrupción breve del proceso</li></ul></li></ul> |
| 4. | Se retrocede sobre la trayectoria.   |
| 5. | La reanudación del programa da lugar al reinicio del movimiento.   |

Si se está en espera en cualquier rutina de evento STOP o QSTOP hasta que el proceso TriggStopProc haya concluido, por ejemplo con WaitUntil (myproc.restartstop=TRUE), \MaxTime:=2; el usuario debe siempre poner a cero el indicador de la rutina de evento RESTART, por ejemplo con myproc.restartstop:=FALSE. A continuación, el reinicio queda completado.

### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema ERRNO cambiará a:

| Nombre             | Causa del error   |
|--------------------|---|
| ERR_NO_ALIASIO_DEF | La variable de señal es una variable declarada en RAPID. No se ha conectado a una señal E/S definida en la configuración de E/S con la instrucción AliasIO. |
| ERR_NORUNUNIT      | Se ha perdido el contacto con el dispositivo de E/S.  |

### Limitaciones

No se admite el reinicio de las instrucciones del proceso tras una caída de alimentación.

TriggStopProc no puede ejecutarse en un gestor UNDO o en rutinas de RAPID que estén conectadas a los siguientes eventos especiales del sistema: PowerOn, Stop, QStop, Restart, Reset o Step.

### Sintaxis

```
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> ';
```

Continúa en la página siguiente



## 1.314 TriggStopProc - Genera datos de reinicio para las señales de disparo ante paros RobotWare - OS Continuación

### Información relacionada

| Para obtener más información sobre | Consulte   |
|------------------------------------|--|
| Instrucciones de proceso           | <a href="#">TriggL - Movimiento lineal del robot con eventos en la página 940</a><br><a href="#">TriggC - Movimiento circular del robot con eventos en la página 895</a> |
| Datos de reinicio                  | <a href="#">restartdata - Datos de reinicio de señales de disparo en la página 1740</a>  |
| Retroceso por la trayectoria       | <a href="#">StepBwdPath - Retrocede un paso a lo largo de la trayectoria en la página 811</a>  |
| Advanced RAPID                     | <a href="#">Application manual - Controller software IRC5</a>  |

# 1 Instrucciones

---

1.315 TryInt - Comprobar si un objeto de dato es un entero válido  
RobotWare - OS

## 1.315 TryInt - Comprobar si un objeto de dato es un entero válido

---

### Utilización

`TryInt` se utiliza para comprobar si un objeto de dato determinado es un entero válido.

---

### Ejemplos básicos

Los ejemplos siguientes ilustran la instrucción `TryInt`.

#### Ejemplo 1

```
VAR num myint := 4;
...
TryInt myint;
```

Se evalúa el valor de `myint` y, dado que 4 es un entero válido, la ejecución del programa continúa.

#### Ejemplo 2

```
VAR dnum mydnum := 20000000;
...
TryInt mydnum;
```

Se evalúa el valor de `mydnum` y, dado que 20000000 es un entero `dnum` válido, la ejecución del programa continúa.

#### Ejemplo 3

```
VAR num myint := 5.2;
...
TryInt myint;
...
ERROR
  IF ERRNO = ERR_INT_NOTVAL THEN
    myint := Round(myint);
    RETRY;
  ENDIF
```

Se evalúa el valor de `myint` y, dado que 5.2 no es un entero válido, se generará un error. En el gestor de errores, `myint` se redondea a 5 y la instrucción `TryInt` se ejecuta una vez más.

---

### Argumentos

`TryInt DataObj | DataObj2`

`DataObj`

**Data Object**

Tipo de dato: `num`

El objeto de datos en el que se desea comprobar si el valor es un entero válido.

`DataObj2`

**Data Object 2**

Tipo de dato: `dnum`

El objeto de datos en el que se desea comprobar si el valor es un entero válido.

*Continúa en la página siguiente*

### Ejecución de programas

Se comprueba el objeto de dato indicado:

- Si es un entero válido, la ejecución continúa con la instrucción siguiente.
- Si no es un entero válido, la ejecución continúa en el gestor de errores del procedimiento actual.

### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                      | Causa del error  |
|-----------------------------|--|
| <code>ERR_INT_NOTVAL</code> | DataObj contiene un valor decimal.   |
| <code>ERR_INT_MAXVAL</code> | <ul style="list-style-type: none"> <li>• El valor de DataObj es mayor o menor que el rango de valor entero del tipo de dato <code>num</code>.</li> <li>• El valor de DataObj2 es mayor o menor que el rango de valor entero del tipo de dato <code>dnum</code>.</li> </ul> |

Recuerde que un valor de `3.0` se evalúa como un entero, dado que el valor `.0` puede omitirse.

### Sintaxis

```
TryInt
  [ DataObj '[:=' ] < expression (IN) of num>
  | [ DataObj2 '[:=' ] < expression (IN) of dnum>' ;'
```

### Información relacionada

| Para obtener más información sobre | Consulte  |
|------------------------------------|---|
| Tipo de dato <code>num</code>      | <a href="#">num - Valores numéricos en la página 1712</a> |

# 1 Instrucciones

---

1.316 TRYNEXT - Salta sobre una instrucción que ha provocado un error  
*RobotWare-OS*

## 1.316 TRYNEXT - Salta sobre una instrucción que ha provocado un error

---

### Utilización

La instrucción TRYNEXT se utiliza para reanudar la ejecución después de un error, empezando por la instrucción que sigue a la instrucción que provocó el error.

---

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción TryNext:

#### Ejemplo 1

```
reg2 := reg3/reg4;  
...  
ERROR  
  IF ERRNO = ERR_DIVZERO THEN  
    reg2:=0;  
    TRYNEXT;  
  ENDIF
```

Se intenta dividir `reg3` por `reg4`. Si `reg4` es igual a 0 (lo cual da lugar a una división por cero), se realiza un salto al gestor de errores, donde se asigna a `reg2` el valor 0. A continuación, la instrucción TRYNEXT se utiliza para continuar con la siguiente instrucción.

---

### Ejecución de programas

La ejecución del programa continúa en la instrucción siguiente a la instrucción que provocó el error.

---

### Limitaciones

La instrucción sólo puede existir en el gestor de errores de la rutina.

---

### Sintaxis

```
TRYNEXT ; '
```

---

### Información relacionada

| Para obtener más información sobre | Consulte   |
|------------------------------------|--|
| Gestores de errores                | <i>Manual de referencia técnica - RAPID Overview</i> |

### 1.317 TuneReset - Restablecimiento del ajuste del servo

#### Utilización

`TuneReset` se utiliza para devolver el comportamiento dinámico de todos los ejes del robot y de las unidades mecánicas a sus valores normales.

Esta instrucción sólo puede usarse en la tarea principal `T_ROB1` o, si se cuenta con un sistema *MultiMove*, en las tareas de movimiento.

#### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción `TuneReset`:

##### Ejemplo 1

```
TuneReset ;
```

Se restablecen los valores de ajuste de todos los ejes a 100%.

#### Ejecución de programas

Se restablecen a 100% los valores de ajuste de todos los ejes.

Los valores de ajuste predeterminados del servo de todos los ejes se ajustan automáticamente mediante la ejecución de la instrucción `TuneReset` en los casos siguientes:

- en un **Reinicio**.
- Cuando se carga un nuevo programa.
- Cuando se inicia la ejecución del programa desde el principio

#### Sintaxis

```
TuneReset ' ; '
```

#### Información relacionada

| Para obtener más información sobre | Consulte  |
|------------------------------------|---|
| Ajuste de servos                   | <a href="#">TuneServo - Ajuste de servos en la página 994</a> |

# 1 Instrucciones

---

1.318 TuneServo - Ajuste de servos  
RobotWare - OS

## 1.318 TuneServo - Ajuste de servos

---

### Utilización

TuneServo se utiliza para ajustar el comportamiento dinámico de los distintos ejes del robot.

Para la mayoría de las aplicaciones no es necesario usar TuneServo, pero para algunas otras, TuneServo se debe obtener la precisión deseada. El uso de TuneServo se puede sustituir en muchos casos seleccionando un *Motion Process Mode* predefinido, consulte *Manual de referencia técnica - Parámetros del sistema*, o modificando un *Motion Process Mode* predefinido.

En el caso de los ejes externos, puede usarse TuneServo para la adaptación de la carga.

Evite ejecutar instrucciones TuneServo mientras el robot está en movimiento. Puede dar lugar a un par de torsión momentáneamente alto, lo que puede causar indicaciones de error y paros.

Esta instrucción sólo puede usarse en la tarea principal T\_ROB1 o, si se cuenta con un sistema *MultiMove*, en las tareas de movimiento.



#### Nota

Para obtener el ajuste óptimo, resulta esencial que se utilicen datos de carga correctos. Compruébelo antes de usar TuneServo.



#### ¡AVISO!

Un uso incorrecto de TuneServo puede dar lugar a movimientos de oscilación o pares que pueden causar daños en el robot. Debe tenerlo en cuenta y tener cuidado al usar TuneServo.

---

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción TuneServo:

#### Ejemplo 1

```
TuneServo MHA160R1, 1, 110 \Type:= TUNE_KP;
```

Activación del tipo de ajuste TUNE\_KP con un valor de ajuste del 110% en el eje 1 de la unidad mecánica MHA160R1.

---

### Descripción

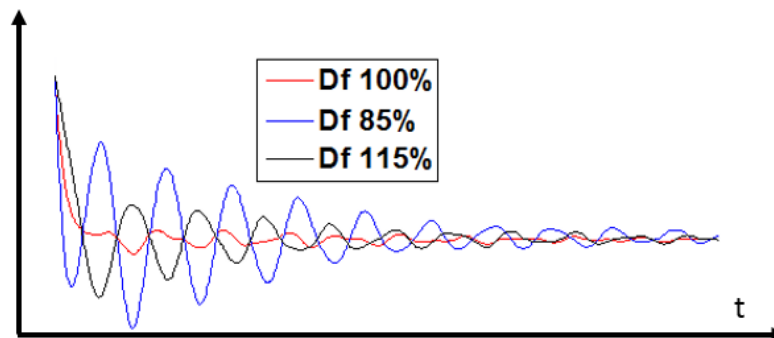
Reducir las vibraciones y el riesgo de sobrepasar posiciones - TUNE\_DF

TUNE\_DF se puede usar para ajustar la frecuencia de resonancia magnética predicha de un eje concreto. Un valor de ajuste del 95% reduce la frecuencia de resonancia en un 5%. El uso más común de TUNE\_DF es para compensar una base de rigidez inadecuada, es decir, una base flexible. En este caso, el valor de ajuste para los ejes 1 y 2 se reduce habitualmente a un valor de entre el 80% y el 99%.

*Continúa en la página siguiente*

El uso de TUNE\_DF para los ejes 3-6 es poco común y normalmente no se recomienda. Una excepción es el ajuste de los ejes 4-6 para compensar la frecuencia de resonancia de una carga útil flexible y extendida.

Correctamente ajustado, ni demasiado alto ni demasiado bajo, TUNE\_DF reduce las vibraciones y el riesgo de sobrepasar posiciones. Tenga cuidado al ajustar TUNE\_DF, dado que tanto un valor de ajuste demasiado alto como uno demasiado bajo puede afectar considerablemente al movimiento. Un ejemplo de ello se muestra en la figura siguiente. En este caso, un valor de ajuste del 100% proporciona el mejor resultado.



xx1400001280

El valor de ajuste puede optimizarse automáticamente utilizando TuneMaster, que es lo recomendado.

En el caso del ajuste manual, un ejemplo de fragmento de código RAPID para ajustar el eje 1 es el siguiente:

```
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;
```

Aquí, el valor de ajuste se cambia en pasos del 5%; también podrían usarse pasos del 2%. Tenga en cuenta que el movimiento debe ser corto; 2 grados es un valor típico. El robot se debe situar en una posición de área de trabajo normal. Se debe elegir mediante inspección visual el valor de ajuste que permita minimizar las vibraciones y el riesgo de sobrepasar posiciones.

También se pueden reducir las vibraciones y el riesgo de sobrepasar posiciones reduciendo el valor de ajuste para TUNE\_DH o reduciendo la aceleración mediante AccSet. En muchos casos, esta es la mejor solución. Sin embargo, si un problema se puede resolver mediante TUNE\_DF, el tiempo de ciclo no se ve afectado y el uso de TUNE\_DF es por tanto la mejor solución.

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.318 TuneServo - Ajuste de servos

RobotWare - OS

Continuación

En el caso de los robots que disponen de *Mounting Stiffness Factor*, consulte *Motion Process Mode* en *Manual de referencia técnica - Parámetros del sistema*; el uso de *Mounting Stiffness Factor* para compensar una base flexible, reemplaza al uso de TUNE\_DF.

### Reducir las vibraciones y el riesgo de sobrepasar posiciones - TUNE\_DH

TUNE\_DH puede usarse para aumentar la suavidad de la trayectoria del robot ajustando el ancho de banda efectivo del sistema. El valor de ajuste solo se puede reducir y los valores superiores al 100% no afectarán a los movimientos. Un valor de ajuste inferior al 100% reduce el ancho de banda y aumenta la suavidad, por lo que reduce vibraciones y el riesgo de sobrepasar posiciones.

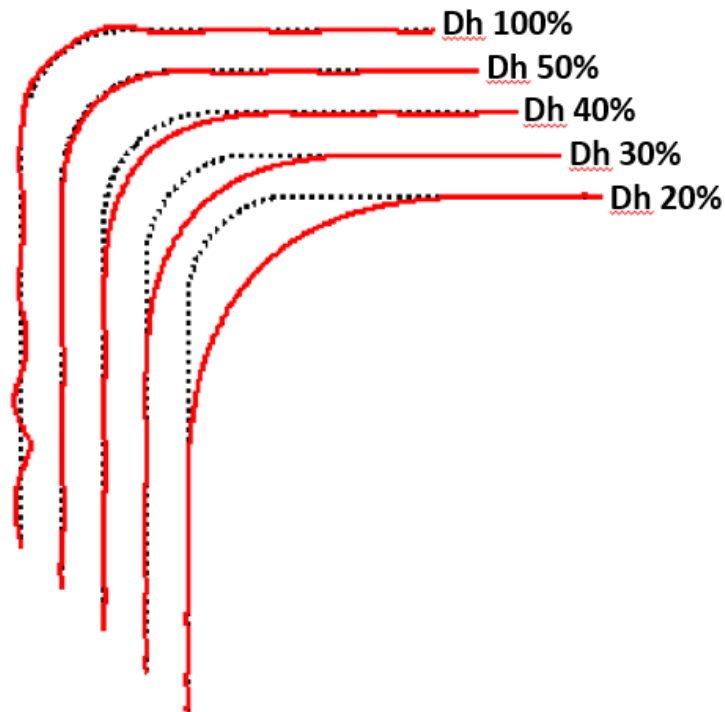
TUNE\_DH solo aumenta el tiempo de ciclo en los puntos finos, mientras que las reducciones de aceleración aumentan el tiempo de ciclo a lo largo de toda la trayectoria del robot. Por lo tanto, el uso de TUNE\_DH puede ser una forma muy eficiente de reducir vibraciones y el riesgo de sobrepasar posiciones en el tiempo de ciclo en comparación con la reducción de mediante la instrucción AccSet. A alta velocidad, se observarán zonas de esquina mayores que las programadas cuando se utiliza TUNE\_DH. Por lo tanto, la utilización de TUNE\_DH reduce los errores de trayectoria causados por vibraciones, pero aumenta los errores de trayectoria a alta velocidad al tomar atajos en las zonas de esquina. Los atajos aumentan al reducir el valor de ajuste y aumentar la velocidad. Si estos atajos no son aceptables, se recomienda AccSet en vez de TUNE\_DH.

La figura que aparece a continuación muestra el efecto de la reducción del valor de ajuste y que una vibración indeseada se puede eliminar con un valor de ajuste

*Continúa en la página siguiente*



adecuado. Con los valores de ajuste más pequeños, el atajo en la zona de esquina se vuelve detectable.



xx1400001281

Basta con ejecutar la instrucción `TuneServo` con el argumento `\Type:=TUNE_DH` para un eje. Todos los ejes de la misma unidad mecánica reciben automáticamente el mismo valor.

Ejemplos:

- Corte con una velocidad de hasta 300 mm/s. Un valor de ajuste del 50% reduce las vibraciones indeseadas.  
Esto se combina a veces con `AccSet`, por ejemplo: `AccSet 50,100;`
- Manejo de materiales a alta velocidad. Un valor de ajuste del 15% reduce las vibraciones indeseadas.



### ¡CUIDADO!

Nunca cambie el valor de ajuste cuando el robot esté en movimiento y tenga cuidado cuando use valores de ajuste pequeños (por debajo del 30%) ya que el robot tomará atajos en las zonas de esquina.

Continúa en la página siguiente

# 1 Instrucciones

---

## 1.318 TuneServo - Ajuste de servos

RobotWare - OS

Continuación

Sólo para uso interno de ABB - TUNE\_DK, TUNE\_DL, TUNE\_DG, TUNE\_DI



**¡AVISO!**

Sólo para uso interno de ABB. No utilice estos tipos de ajuste. Un uso incorrecto puede dar lugar a movimientos de oscilación o pares que pueden causar daños en el robot.

Ajuste de ejes externos - TUNE\_KP, TUNE\_KV, TUNE\_TI

Estos tipos de ajustes afectan a la ganancia de control de posición (kp), la ganancia de control de velocidad (kv) y el tiempo de integración de control de velocidad (ti) de los ejes externos. Estos parámetros se usan para adaptar los ejes externos a distintas inercias de carga. El ajuste básico de los ejes externos también puede simplificarse mediante estos tipos de ajuste.

Ajuste de ejes de robot - TUNE\_KP, TUNE\_KV, TUNE\_TI

Estos parámetros pueden usarse para cambiar el comportamiento del controlador de servo. TUNE\_KP afecta a la ganancia equivalente del controlador de posición, TUNE\_KV afecta a la ganancia equivalente del controlador de velocidad y TUNE\_TI afecta a la acción integral del controlador.

Aumentar el valor de ajuste para TUNE\_KV aumenta la rigidez de servo del robot y puede ser útil en aplicaciones de contacto dado que la rigidez total del sistema de robot depende tanto de la rigidez de servo como de la rigidez mecánica. Un valor de ajuste aumentado para TUNE\_KV también reduce los errores de trayectoria a baja velocidad y puede ser útil en aplicaciones de corte y soldadura cuya velocidad sea menor de 100 mm/s. Los valores de ajuste típicos son 150% - 200%. Un valor de ajuste demasiado alto provoca vibraciones en los motores y debe evitarse. Tenga siempre cuidado y vigile si aumenta el nivel de ruido del motor al ajustar TUNE\_KV y no use valores de ajuste más altos de lo que se necesitan para satisfacer los requisitos de la aplicación. Un valor de ajuste demasiado alto también puede aumentar las vibraciones debidas a las resonancias mecánicas.

Un valor de ajuste aumentado para TUNE\_KP y un valor de ajuste reducido para TUNE\_TI aumentan la rigidez de servo y reducen los errores de trayectoria de baja velocidad en la región de baja frecuencia. Los valores de ajuste normales para TUNE\_KP son 150% - 300%, y para TUNE\_TI 20% - 50%. En la mayoría de los casos, TUNE\_KV es el parámetro más importante y ni TUNE\_KP ni TUNE\_TI necesitan ajuste. Un valor de ajuste demasiado alto para TUNE\_KP o uno demasiado bajo para TUNE\_TI también pueden aumentar las vibraciones debidas a las resonancias mecánicas.

Ejemplo:

- Los robots de las aplicaciones de eliminación de rebabas necesitan una rigidez de servo mayor para reducir los errores de trayectoria. TUNE\_KV 175%, TUNE\_KP 250%, y TUNE\_TI 30%.

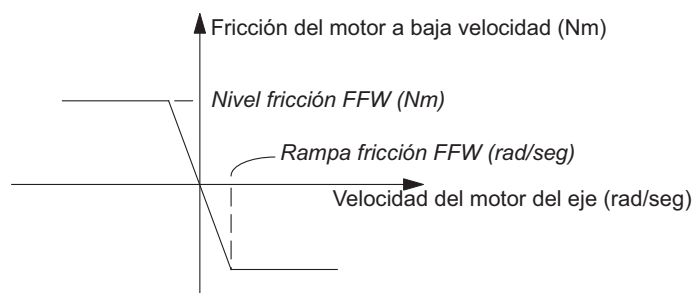
Esto se combina frecuentemente con AccSet, por ejemplo: AccSet  
30,100;.

Continúa en la página siguiente

## Compensación de fricción - TUNE\_FRIC\_LEV, TUNE\_FRIC\_RAMP

Estos tipos de ajuste pueden usarse para reducir los errores de trayectoria del robot que se deban a la fricción y a los movimientos no deseados que se producen a baja velocidad (de 10 a 200 mm/s). Estos errores de trayectoria aparecen cuando un eje del robot cambia de sentido de movimiento. Para activar la compensación de fricción para un eje, cambie a Yes el parámetro de sistema Motion/Control Parameters/Friction FFW On.

El modelo de fricción es un nivel constante con el signo opuesto al sentido de avance del eje. *Friction FFW Level (Nm)* es el nivel absoluto de fricción a (baja) velocidad y es mayor que *Friction FFW Ramp (rad/s)*. Consulte la figura siguiente, que muestra un modelo de fricción.



xx0500002188

TUNE\_FRIC\_LEV redefine el valor del parámetro de sistema *Friction FFW Level*.

El ajuste de *Friction FFW Level* (con TUNE\_FRIC\_LEV) en cada eje del robot puede aumentar considerablemente la exactitud de la trayectoria del robot en el rango de velocidad de 20-100 mm/s. En robots grandes (especialmente en el caso de la familia IRB 6400) el efecto será mínimo dado que en estos robots dominan otros métodos de seguimiento de errores.

TUNE\_FRIC\_RAMP redefine el valor del parámetro de sistema *Friction FFW Ramp*. En la mayoría de los casos, no es necesario ajustar el valor de *Friction FFW Ramp*. El valor predeterminado es adecuado.

**Ajuste un eje cada vez.** Cambie el valor de ajuste en pequeños incrementos y determine el nivel que permita reducir al mínimo el error de trayectoria del robot en la trayectoria en la que este eje concreto cambie de sentido de movimiento. Repita el mismo procedimiento con el siguiente eje, etc.

Los valores finales de ajuste pueden transferirse a los parámetros del sistema.

Ejemplo:

- Friction FFW Level = 1. Valor de ajuste final (TUNE\_FRIC\_LEV) = 150%.
- Cambie el valor de Friction FFW Level a 1,5 y el valor de ajuste al 100% (el valor predeterminado), para conseguir el valor equivalente.

## Argumentos

TuneServo MecUnit Axis TuneValue [\Type]

MecUnit

*Mechanical Unit*

Tipo de dato: mecunit

Continúa en la página siguiente

# 1 Instrucciones

---

## 1.318 TuneServo - Ajuste de servos

RobotWare - OS

Continuación

El nombre de la unidad mecánica.

Axis

Tipo de dato: num

El número del eje actual de la unidad mecánica (del 1 al 6).

TuneValue

Tipo de dato: num

El valor de ajuste, en porcentaje (de 1 a 500). El 100% es el valor normal.

[ \Type ]

Tipo de dato: tunetype

El tipo de ajuste del servo. Los tipos disponibles son TUNE\_DF, TUNE\_KP, TUNE\_KV, TUNE\_TI, TUNE\_FRIC\_LEV, TUNE\_FRIC\_RAMP, TUNE\_DG, TUNE\_DH, TUNE\_DI. Los tipos TUNE\_DK y TUNE\_DL son sólo para uso interno de ABB.

Puede omitir este argumento si utiliza el tipo de ajuste TUNE\_DF.

---

### Ejecución de programas

El tipo de ajuste especificado y el valor de ajuste se activan para el eje especificado. Este valor se aplica a todos los movimientos hasta que se programa un nuevo valor para el eje actual o hasta que se restablecen los tipos y valores de ajuste de todos los ejes mediante la instrucción TuneReset.

Los valores de ajuste predeterminados del servo de todos los ejes se ajustan automáticamente mediante la ejecución de la instrucción TuneReset en los casos siguientes:

- en un Reinicio.
- Cuando se carga un nuevo programa.
- Cuando se inicia la ejecución del programa desde el principio

---

### Limitaciones

Cualquier ajuste de servo activo se devuelve siempre a los valores predeterminados en caso de una caída de alimentación.

Esta limitación puede gestionarse en el programa del usuario al reanudar la ejecución después de una caída de alimentación.

---

### Sintaxis

```
TuneServo
  [MecUnit ':=' ] < variable (VAR) of mecunit> ', '
  [Axis ':=' ] < expression (IN) of num> ', '
  [TuneValue ':=' ] < expression (IN) of num>
  ['\ ' Type ':=' <expression (IN) of tunetype>] ';'
```

---

### Información relacionada

|                                    |   |
|------------------------------------|---|
| Para obtener más información sobre | Consulte                                      |
| Otros parámetros de movimiento     | Manual de referencia técnica - RAPID Overview |

Continúa en la página siguiente

| Para obtener más información sobre                                       | Consulte   |
|--|--|
| Tipos de ajuste de servo   | <a href="#">tunetype - Tipo de ajuste de servo en la página 1819</a>                                     |
| Restablecimiento de todos los ajustes de servo                           | <a href="#">TuneReset - Restablecimiento del ajuste del servo en la página 993</a>                       |
| MotionProcessModeSet - Configuración del modo de proceso de movimientos. | <a href="#">MotionProcessModeSet - Configuración del modo de proceso de movimientos en la página 377</a> |
| Ajuste de ejes externos  | <a href="#">Application manual - Additional axes and stand alone controller</a>                          |
| Compensación de fricción   | <a href="#">Manual de referencia técnica - Parámetros del sistema</a>                                    |

# 1 Instrucciones

---

1.319 UIMsgBox - Cuadro de mensaje de usuario de tipo básico  
RobotWare - OS

## 1.319 UIMsgBox - Cuadro de mensaje de usuario de tipo básico

---

### Utilización

UIMsgBox (*User Interaction Message Box*) se usa para comunicarse con el usuario del sistema de robot a través de un dispositivo de usuario disponible, como el FlexPendant. Se escribe un mensaje para el operador, que a su vez responde con la selección de un botón. A continuación, la selección de usuario se transfiere al programa.

---

### Ejemplos básicos

Los ejemplos siguientes ilustran la instrucción UIMsgBox.

Consulte también [Más ejemplos en la página 1007](#).

#### Ejemplo 1

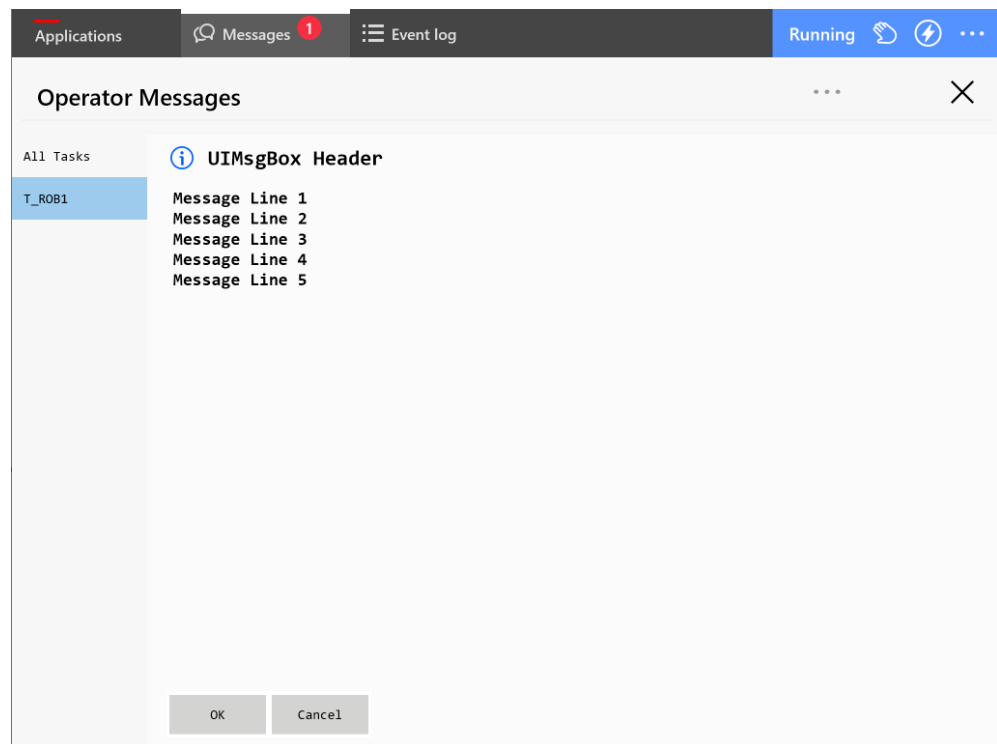
```
UIMsgBox "Continue the program ?";
```

Se muestra el mensaje "Continue the program ?". El programa continúa tan pronto como el usuario presiona el botón predeterminado OK.

#### Ejemplo 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;
```

*Continúa en la página siguiente*



xx2000001833

Se muestra en la pantalla del FlexPendant el cuadro de mensaje anterior, con icono, título, líneas de mensaje de la 1 a la 5 y pulsadores definidos por el usuario. La ejecución espera hasta que se presiona OK o Cancelar. En otras palabras, se asigna a `answer` el valor 1 (OK) ó 5 (Cancelar) en función de cuál de los botones se presione. Si la respuesta es OK, se llama a `my_proc`.

Recuerde que las líneas de mensaje de la 1 a la 5 se muestran en las líneas de la 1 a la 5 separadas (no se usa el modificador `\Wrap`).

### Argumentos

UIMsgBox [`\Header`] `MsgLine1` [`MsgLine2`] [`MsgLine3`] [`MsgLine4`] [`MsgLine5`] [`Wrap`] [`Buttons`] [`Icon`] [`Image`] [`Result`] [`MaxTime`] [`DIBreak`] [`DIPassive`] [`DOBreak`] [`DOPassive`] [`PersBoolBreak`] [`PersBoolPassive`] [`BreakFlag`] [`UIActiveSignal`]

[`\Header`]

Tipo de dato: `string`

El texto de título que debe escribirse en la parte superior del cuadro de mensaje. Máximo 40 caracteres.

`MsgLine1`

*Message Line 1*

Tipo de dato: `string`

La línea 1 del texto a escribir en la pantalla. Máximo 55 caracteres.

[`\MsgLine2`]

*Message Line 2*

Continúa en la página siguiente

# 1 Instrucciones

---

## 1.319 UIMsgBox - Cuadro de mensaje de usuario de tipo básico

RobotWare - OS

Continuación

Tipo de dato: `string`

Se escribe la línea 2 de texto adicional en la pantalla. Máximo 55 caracteres.

[`\MsgLine3`]

*Message Line 3*

Tipo de dato: `string`

Se escribe la línea 3 de texto adicional en la pantalla. Máximo 55 caracteres.

[`\MsgLine4`]

*Message Line 4*

Tipo de dato: `string`

Se escribe la línea 4 de texto adicional en la pantalla. Máximo 55 caracteres.

[`\MsgLine5`]

*Message Line 5*

Tipo de dato: `string`

Se escribe la línea 5 de texto adicional en la pantalla. Máximo 55 caracteres.

[`\Wrap`]

Tipo de dato: `switch`

Si se selecciona, todas las cadenas de `MsgLine1 ... MsgLine5` se concatenan para formar una cadena con un solo espacio entre las distintas cadenas individuales y distribuida en el número mínimo posible de líneas.

De forma predeterminada, cada cadena de mensaje de *MsgLine1 ... MsgLine5* aparece en una línea separada en la pantalla.

[`\Buttons`]

Tipo de dato: `buttondata`

Define los pulsadores que se desea mostrar. Sólo puede mostrarse una de las combinaciones de botones predefinidas del tipo `buttondata`. Consulte [Datos predefinidos en la página 1007](#).

De forma predeterminada, el sistema muestra el botón OK. (`\Buttons:=btn OK`).

[`\Icon`]

Tipo de dato: `icondata`

Define el icono a mostrar. Sólo puede usarse uno de los iconos predefinidos de tipo `icondata`. Consulte [Datos predefinidos en la página 1007](#).

De forma predeterminada, no se usa ningún icono.

[`\Image`]

Tipo de dato: `string`

El nombre de la imagen que debe utilizarse. Para iniciar sus propias imágenes, las imágenes deben estar situadas en el directorio `HOME :` del sistema activo o directamente en el sistema activo.

La recomendación es situar los archivos en el directorio `HOME :` de forma que se incluyan en las operaciones de copia de seguridad y restauración.

*Continúa en la página siguiente*



Se requiere un **Reinicio**, tras lo cual el FlexPendant cargará las imágenes.

Una exigencia del sistema es que se use la opción de RobotWare *FlexPendant Interface*.

La imagen a mostrar puede tener 185 píxeles de anchura y 300 píxeles de altura. Si la imagen tiene un tamaño mayor, sólo se muestran 185 \* 300 píxeles de la imagen a partir de su parte superior izquierda.

No es posible especificar ningún valor exacto en cuanto al tamaño que una imagen puede tener o el número de imágenes que es posible cargar en el FlexPendant. Depende del tamaño de los demás archivos cargados en el FlexPendant. La ejecución de los programas continúa simplemente si se usa una imagen que no está cargada en el FlexPendant.

[\Result]

Tipo de dato: `btnres`

La variable cuyo valor se devuelve (de 0 a 7) en función de qué botón se presione. Sólo es posible usar una de las constantes predefinidas del tipo `btnres` para evaluar la selección del usuario. Consulte [Datos predefinidos en la página 1007](#).

En caso de cualquier tipo de interrupción del sistema, como `\MaxTime`, `\DIBreak` o `\DOBreak` o en caso de que `\Buttons:=btnNone, resUnkwn` se devuelve igual a 0.

[\MaxTime]

Tipo de dato: `num`

El periodo máximo, en segundos, que debe esperar el programa para continuar con la ejecución. Si no se selecciona ningún botón en ese periodo, el programa sigue ejecutándose en el gestor de errores, a no ser que se utilice el indicador `BreakFlag` (que se documenta a continuación). La constante `ERR_TP_MAXTIME` puede usarse para comprobar si ha transcurrido ya el tiempo máximo establecido.

[\DIBreak]

**Digital Input Break**

Tipo de dato: `signal`

La señal digital de entrada que puede interrumpir el diálogo con el operador. Si no se selecciona ningún botón cuando la señal cambia a 1 (o si ya tiene el valor 1), el programa sigue ejecutándose en el gestor de errores, a no ser que se utilice el indicador `BreakFlag` (que se documenta a continuación). La constante `ERR_TP_DIBREAK` puede usarse para comprobar si esto ha ocurrido.

[\DIPassive]

**Digital Input Passive**

Tipo de dato: `switch`

Este modificador redefine el comportamiento predeterminado con el argumento opcional `DIBreak`. En lugar de reaccionar cuando la señal cambia a 1 (o ya tiene el valor 1), la instrucción debe continuar en el gestor de errores (si no se utiliza `BreakFlag`) cuando la señal `DIBreak` cambia a 0 (o ya tiene el valor 0). La constante `ERR_TP_DIBREAK` puede usarse para comprobar si esto ha ocurrido.

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.319 UIMsgBox - Cuadro de mensaje de usuario de tipo básico

RobotWare - OS

Continuación

[ \DOBreak ]

### *Digital Output Break*

Tipo de dato: `signaldo`

La señal digital de salida que puede interrumpir el diálogo con el operador. Si no se selecciona ningún botón cuando la señal cambia a 1 (o si ya tiene el valor 1), el programa sigue ejecutándose en el gestor de errores, a no ser que se utilice el indicador `BreakFlag` (que se documenta a continuación). La constante `ERR_TP_DOBREAK` puede usarse para comprobar si esto ha ocurrido.

[ \DOPassive ]

### *Digital Output Passive*

Tipo de dato: `switch`

Este modificador redefine el comportamiento predeterminado con el argumento opcional `DOBreak`. En lugar de reaccionar cuando la señal cambia a 1 (o ya tiene el valor 1), la instrucción debe continuar en el gestor de errores (si no se utiliza `BreakFlag`) cuando la señal `DOBreak` cambia a 0 (o ya tiene el valor 0). La constante `ERR_TP_DOBREAK` puede usarse para comprobar si esto ha ocurrido.

[ \PersBoolBreak ]

### *Persistent Boolean Break*

Tipo de dato: `bool`

El booleano persistente que puede interrumpir el diálogo con el operador. Si no se selecciona ningún botón cuando el booleano persistente cambia a `TRUE` (o ya tiene el valor `TRUE`) entonces el programa continuará ejecutándose en el gestor de errores, a no ser que se utilice `BreakFlag` (que se documenta a continuación). La constante `ERR_TP_PERSBOOLBREAK` puede usarse para comprobar si esto ha ocurrido.

[ \PersBoolPassive ]

### *Persistent Boolean Passive*

Tipo de dato: `switch`

Este interruptor redefine el comportamiento con el argumento opcional `PersBoolBreak`. En lugar de reaccionar cuando el booleano persistente cambia a `TRUE` (o ya tiene el valor `TRUE`), la instrucción debe continuar en el gestor de errores (si no se utiliza `BreakFlag`) cuando el booleano persistente `PersBoolBreak` cambia a `FALSE` (o ya tiene el valor `FALSE`). La constante `ERR_TP_PERSBOOLBREAK` puede usarse para comprobar si esto ha ocurrido.

[ \BreakFlag ]

Tipo de dato: `errnum`

Una variable que contiene el código de error si se utiliza `MaxTime`, `DIBreak`, `DOBreak`, o `PersBoolBreak`. Si se omite esta variable opcional, se ejecuta el gestor de errores. Las constantes `ERR_TP_MAXTIME`, `ERR_TP_DIBREAK`, `ERR_TP_DOBREAK`, y `ERR_TP_PERSBOOLBREAK` pueden usarse para seleccionar el motivo.

*Continúa en la página siguiente*

[\UIActiveSignal]

**Tipo de dato:** signaldo

La señal digital de salida utilizada en el argumento opcional `UIActiveSignal` se establece en 1 cuando se activa el cuadro de mensaje en `FlexPendant`. Cuando se ha realizado la selección de usuario y la ejecución continúa, la señal se vuelve a establecer en 0.

No existe ninguna supervisión de parada o reinicio. La señal se establece en 0 cuando la instrucción está preparada o cuando se mueve el PP.

### Ejecución de programas

Se muestra un cuadro de mensaje con icono, título, líneas de mensaje, imágenes y botones, de acuerdo con los argumentos del programa. La ejecución del programa espera hasta que el usuario seleccione un botón o que el cuadro de mensaje sea interrumpido por un tiempo límite o una acción de señal. La opción seleccionada por el usuario y el motivo de la interrupción se devuelven al programa.

El nuevo cuadro de mensaje del nivel de rutina TRAP toma el foco del cuadro de mensaje del nivel básico.

### Datos predefinidos

```
!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 btnAbtrRtryIgn := 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;
```

### Más ejemplos

A continuación aparecen más ejemplos de cómo usar la instrucción `UIMsgBox`.

#### Ejemplo 1

```
VAR errnum err_var;
```

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.319 UIMsgBox - Cuadro de mensaje de usuario de tipo básico

RobotWare - OS

Continuación

```
...
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
CASE ERR_TP_DIBREAK:
    ! Input signal break, signal di5 has been set to 1
DEFAULT:
    ! Not such case defined
ENDTEST
```

El cuadro de mensaje se muestra hasta que una condición de interrupción sea verdadera. El operador no puede responder ni eliminar el cuadro de mensaje porque se ha configurado `btnNone` para el argumento `\Buttons`. El cuadro de mensaje desaparece cuando `di5` ha cambiado a 1 o en el tiempo límite (tras 60 segundos).

### Ejemplo 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
```

El cuadro de mensaje se muestra hasta que una condición de interrupción sea verdadera. El operador no puede responder ni eliminar el cuadro de mensaje porque se ha configurado `btnNone` para el argumento `\Buttons`. El cuadro de mensaje desaparece cuando `di5` ha cambiado a 0 o en el tiempo límite (tras 60 segundos).

### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                          | Causa del error   |
|---------------------------------|---|
| <code>ERR_NO_ALIASIO_DEF</code> | La variable de señal es una variable declarada en <code>RAPID</code> . No se ha conectado a una señal E/S definida en la configuración de E/S con la instrucción <code>AliasIO</code> . |
| <code>ERR_TP_NO_CLIENT</code>   | No hay ningún cliente, por ejemplo un <code>FlexPendant</code> , a cargo de la instrucción.   |

Continúa en la página siguiente

Si no se usa el parámetro `\BreakFlag`, estas situaciones pueden ser gestionadas en el gestor de errores:

- Si se alcanza el tiempo límite (parámetro `\MaxTime`) antes de que responda el operador, la variable de sistema `ERRNO` cambia a `ERR_TP_MAXTIME` y la ejecución continúa en el gestor de errores.
- Si se activa la entrada digital (parámetro `\DIBreak`) antes de que responda el operador, la variable de sistema `ERRNO` cambia a `ERR_TP_DIBREAK` y la ejecución continúa en el gestor de errores.
- Si se activa la salida digital (parámetro `\DOBreak`) antes de la acción del operador, la variable de sistema `ERRNO` cambia a `ERR_TP_DOBREAK` y la ejecución continúa en el gestor de errores.
- Si se configura un booleano persistente (parámetro `\PersBoolBreak`) antes de la acción del operador, la variable de sistema `ERRNO` cambia a `ERR_TP_PERSBOOLBREAK` y la ejecución continúa en el gestor de errores.

### Limitaciones

Evite usar un valor demasiado pequeño para el parámetro de tiempo límite `\MaxTime` si `UIMsgBox` se ejecuta frecuentemente, por ejemplo, en un bucle. Si lo hace, puede dar lugar a un comportamiento impredecible del rendimiento del sistema, por ejemplo, la ralentización de la respuesta del FlexPendant.

### Sintaxis

```
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]
['\PersBoolBreak ':=' <persistent (PERS) of bool>]
['\PersBoolPassive]
['\BreakFlag':=' <var or pers (INOUT) of errnum>]
['\UIActiveSignal ':=' <variable (VAR) of signaldo>] '';
```

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.319 UIMsgBox - Cuadro de mensaje de usuario de tipo básico

RobotWare - OS

Continuación

### Información relacionada

| Para obtener más información sobre                               | Consulte   |
|--|--|
| Datos de visualización de iconos                                 | <a href="#">icondata - Datos de visualización de iconos en la página 1683</a>                  |
| Datos de pulsador  | <a href="#">buttondata - Datos de botón en la página 1614</a>                                  |
| Datos de resultado de pulsador                                   | <a href="#">btnres - Datos de resultado de pulsador en la página 1611</a>                      |
| Cuadro de mensaje de interacción con el usuario de tipo avanzado | <a href="#">UIMessageBox - Cuadro de mensaje de usuario de tipo avanzado en la página 1577</a> |
| Introducción de número de interacción con el usuario             | <a href="#">UINumEntry - Introducción de número de usuario en la página 1585</a>               |
| Ajuste de número de interacción con el usuario                   | <a href="#">UINumTune - Ajuste de número de usuario en la página 1592</a>                      |
| Introducción alfanumérica de interacción con el usuario          | <a href="#">UIAlphaEntry - Introducción alfanumérica del usuario en la página 1545</a>         |
| Vista de lista de interacción con el usuario                     | <a href="#">UIListView - Vista de lista de usuario en la página 1568</a>                       |
| Sistema conectado al FlexPendant, etc.                           | <a href="#">UIClientExist - Existe cliente de usuario en la página 1552</a>                    |
| FlexPendant Interface  | <a href="#">Especificaciones del producto - Controlador IRC5</a>                               |
| Borrado de la ventana de operador                                | <a href="#">TPErase - Borra el texto mostrado en el Flex-Pendant en la página 877</a>          |

## 1.320 UIMsgWrite - Cuadro de diálogo de mensaje de usuario tipo sin espera RobotWare - OS

### 1.320 UIMsgWrite - Cuadro de diálogo de mensaje de usuario tipo sin espera

#### Utilización

`UIMsgWrite` (*User Interaction Message Write*) se utiliza para comunicarse con el usuario del sistema del robot en un dispositivo de usuario disponible, como por ejemplo `FlexPendant`. Se escribe un mensaje para el operador.

#### Ejemplos básicos

Los siguientes ejemplos ilustran la instrucción `UIMsgWrite`.

##### Ejemplo 1

```
VAR string myHeader := "Signal error!";

UIMsgWrite myHeader, "Set signal dil high please!" \Icon:=iconInfo;
WaitDI dil, 1;
UIMsgWriteAbort;
```

Se muestra el mensaje "*Set signal dil high please!*". El programa continúa y el mensaje se elimina cuando se activa la señal `dil`.

##### Ejemplo 2

```
VAR string myHeader := "Signal Error!";
VAR string myMsgArray{5}:= ["Set", "signal", "dil", "high",
"please!"];
UIMsgWrite myHeader, myMsgArray, \Icon:=iconInfo
\Image:="MyImage.jpg";
WaitDI dil, 1;
UIMsgWriteAbort;
```

Se muestra el mensaje incluyendo la cabecera, cinco líneas de mensaje, icono e imagen. El programa continúa y el mensaje se elimina cuando se activa la señal `dil`.

Recuerde que las líneas de mensaje de la 1 a la 5 se muestran en las líneas de la 1 a la 5 separadas (no se usa el modificador `\Wrap`).

#### Argumentos

```
UIMsgWrite Header Message | MsgArray [\Wrap] [\Icon] [\Image]
[\PersBool] | [\PersBoolName] [\AbortValue] [\UIActiveSignal]
```

##### Header

Tipo de dato: `string`

El texto de cabecera que debe escribirse en la parte superior del cuadro de mensaje. Máximo 40 caracteres.

##### Message

Tipo de dato: `string`

Una línea de texto que se escribirá en la pantalla. Máximo 50 caracteres.

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.320 UIMsgWrite - Cuadro de diálogo de mensaje de usuario tipo sin espera

RobotWare - OS

Continuación

MsgArray

(*Message Array*)

Tipo de dato: `string`

Varias líneas de texto de una matriz para escribir en la pantalla. Solo es posible usar uno de los parámetros `Message` o `MsgArray` al mismo tiempo.

El máximo espacio es de 5 líneas con 50 caracteres cada una.

[`\Wrap`]

Tipo de dato: `switch`

Si se selecciona, todas las líneas de `MsgArray` se concatenarán para formar una cadena con un solo espacio entre las distintas cadenas individuales y distribuida en el número mínimo posible de líneas.

De forma predeterminada, cada línea de `MsgArray` aparece en una línea separada en la pantalla.

[`\Icon`]

Tipo de dato: `icondata`

Define el icono que se va a mostrar. Solo puede utilizarse uno de los iconos predefinidos del tipo `icondata`; consulte [Datos predefinidos en la página 1014](#).

De forma predeterminada, no se usa ningún icono.

[`\Image`]

Tipo de dato: `string`

El nombre de la imagen que debe utilizarse. Para iniciar sus propias imágenes, las imágenes deben estar situadas en el directorio *HOME*: del sistema activo o directamente en el sistema activo.

La recomendación es situar los archivos en el directorio *HOME*: de forma que se incluyan en las operaciones de copia de seguridad y restauración.

Se requiere un reinicio y, a continuación, el FlexPendant carga las imágenes.

Una exigencia del sistema es que se use la opción de RobotWare *FlexPendant Interface*.

La imagen a mostrar puede tener 185 píxeles de anchura y 300 píxeles de altura. Si la imagen tiene un tamaño mayor, solo se muestran 185 \* 300 píxeles de la imagen a partir de su parte superior izquierda.

No es posible especificar ningún valor exacto en cuanto al tamaño que una imagen puede tener o el número de imágenes que es posible cargar en el FlexPendant. Depende del tamaño de los demás archivos cargados en el FlexPendant. La ejecución de los programas continúa simplemente si se usa una imagen que no está cargada en el FlexPendant.

[`\PersBool`]

(*Persistent Bool*)

Tipo de dato: `bool`

Continúa en la página siguiente



## 1.320 UIMsgWrite - Cuadro de diálogo de mensaje de usuario tipo sin espera RobotWare - OS Continuación

El mensaje se mostrará siempre que el `bool` especificado sea `FALSE`. Si se utilizara el parámetro `AbortValue`, se mostrará el mensaje siempre que el `bool` especificado sea diferente en valor de `AbortValue`.

Solo puede utilizarse una variable booleana completa `PERS` o `TASK PERS`.

`[\PersBoolName]`

*(Persistent Bool Name)*

Tipo de dato: `string`

El mensaje se mostrará siempre que el `bool` especificado sea `FALSE`. Si se utilizara el parámetro `AbortValue`, se mostrará el mensaje siempre que el `bool` especificado sea diferente en valor de `AbortValue`.

Solo puede utilizarse un nombre de variable booleana completa `PERS` o `TASK PERS`.

Si se utiliza el argumento `\PersBoolName`, es posible utilizar una variable booleana persistente declarada en otra tarea en la instrucción `UIMsgWrite`.

`[\AbortValue]`

Tipo de dato: `bool`

Solo válido si está presente `PersBool`. El valor esperado para `PersBool`. El mensaje se mostrará hasta que `PersBool` sea igual a `AbortValue`.

`[\UIActiveSignal]`

Tipo de dato: `signaldo`

La señal digital de salida utilizada en un argumento opcional `UIActiveSignal` se establece en 1 cuando el cuadro de mensaje se activa en `FlexPendant`. La señal se sitúa en 0 cuando el cuadro de mensaje se retira con la instrucción `UIMsgWriteAbort` o cuando se cumple la expresión `PersBool`.

No existe ninguna supervisión de parada o reinicio. La señal se establece en 0 cuando se mueve el PP.

---

### Ejecución de programas

El mensaje con icono, cabecera, líneas de mensaje, imagen y expresiones no cumplidas se muestran de acuerdo con los argumentos programados. El mensaje se muestra hasta que es cancelado por `UIMsgWriteAbort` (o cuando se cumple la expresión `PersBool`, si se utiliza el argumento `PersBool`).

- Un nuevo mensaje en el nivel básico sustituirá un mensaje anterior.
- Un nuevo mensaje en el nivel de `TRAP` sustituirá un mensaje anterior en el nivel básico, y permanecerá activo al volver al nivel básico.
- Un nuevo mensaje en una rutina de servicio siempre se cancelará al volver al nivel básico. Por lo tanto, cualquier mensaje activo en el nivel básico se reactivará.
- Un nuevo mensaje de usuario generado por las siguientes instrucciones y funciones sustituirá un mensaje generado por `UIMsgWrite`:

`TPReadFK`, `TPReadDnum`, `TPReadNum`, `UIMsgBox`, `UIMessageBox`,  
`UIDnumEntry`, `UIDnumTune`, `UINumEntry`, `UINumTune`,

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.320 UIMsgWrite - Cuadro de diálogo de mensaje de usuario tipo sin espera

RobotWare - OS

Continuación

UIAlphaEntry, UICollection. Un cuadro de mensaje de una de las instrucciones de espera WaitAI, WaitAO, WaitGI, WaitGO, WaitDI, WaitDO, WaitUntil también sustituirá un mensaje generado por UIMsgWrite.

### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                          | Causa del error   |
|---------------------------------|---|
| <code>ERR_NO_ALIASIO_DEF</code> | La variable de señal es una variable declarada en <code>RAPID</code> . No se ha conectado a una señal E/S definida en la configuración de E/S con la instrucción <code>AliasIO</code> .               |
| <code>ERR_NORUNUNIT</code>      | Se ha perdido el contacto con el dispositivo de E/S.  |
| <code>ERR_SIG_NOT_VALID</code>  | La señal de E/S no está disponible. Los motivos pueden ser que el dispositivo de E/S no está en funcionamiento o que exista un error en la configuración (válido solamente para el bus de campo ICI). |
| <code>ERR_SYM_ACCESS</code>     | El objeto de datos utilizado en <code>[\PersBoolName]</code> no existe.   |

### Datos predefinidos

Se han predefinido en el sistema las constantes siguientes del tipo de dato `icondata`.

| Valor | Constante                | Icono                |
|-------|--------------------------|----------------------|
| 0     | <code>iconNone</code>    | Ningún icono         |
| 1     | <code>iconInfo</code>    | Icono de información |
| 2     | <code>iconWarning</code> | Icono de aviso       |
| 3     | <code>iconError</code>   | Icono de error       |

### Sintaxis

```
UIMsgWrite
[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>]
['\ PersBool ::= <pers (IN) of bool>]
| ['\ PersBoolName ::= <pers (IN) of string>]
['\ AbortValue ::= <var or pers (IN) of bool>]
['\ UIActiveSignal ::= <variable (VAR) of signaldo>] ';'
```

### Información relacionada

| Para obtener más información sobre | Consulte  |
|------------------------------------|---|
| Datos de visualización de iconos   | <a href="#">icondata - Datos de visualización de iconos en la página 1683</a> |

Continúa en la página siguiente

## 1.320 UIMsgWrite - Cuadro de diálogo de mensaje de usuario tipo sin espera

RobotWare - OS

Continuación

| Para obtener más información sobre               | Consulte   |
|--|--|
| Cancelar cuadro de diálogo de mensaje de usuario | <a href="#">UIMsgWriteAbort - Cancelar cuadro de diálogo de mensaje de usuario tipo sin espera en la página 1016</a> |
| Sistema conectado al FlexPendant, etc.           | <a href="#">UIClientExist - Existe cliente de usuario en la página 1552</a>  |
| FlexPendant Interface                            | <a href="#">Especificaciones del producto - Controlador IRC5</a>   |
| Cyclic bool                                      | <a href="#">Application manual - Controller software IRC5</a>  |

# 1 Instrucciones

---

1.321 UIMsgWriteAbort - Cancelar cuadro de diálogo de mensaje de usuario tipo sin espera  
RobotWare - OS

## 1.321 UIMsgWriteAbort - Cancelar cuadro de diálogo de mensaje de usuario tipo sin espera

---

### Utilización

UIMsgWriteAbort se utiliza para cancelar un mensaje activo que ha sido iniciado previamente por una instrucción UIMsgWrite.

---

### Ejemplos básicos

El siguiente ejemplo ilustra la instrucción UIMsgWriteAbort.

#### Ejemplo 1

```
VAR string myHeader := "Signal error!";

UIMsgWrite myHeader, "Set signal di1 high please!", iconInfo;
WaitDI di1, 1;
UIMsgWriteAbort;
```

Se muestra el mensaje "Set signal di1 high please!". El programa continúa y el mensaje se elimina cuando se activa la señal di1.

---

### Sintaxis

```
UIMsgWriteAbort ';' ;'
```

---

### Información relacionada

| Para obtener más información sobre               | Consulte   |
|--|--|
| Datos de visualización de iconos                 | <a href="#">icondata - Datos de visualización de iconos en la página 1683</a>                          |
| Escribir cuadro de diálogo de mensaje de usuario | <a href="#">UIMsgWrite - Cuadro de diálogo de mensaje de usuario tipo sin espera en la página 1011</a> |
| Sistema conectado al FlexPendant, etc.           | <a href="#">UIClientExist - Existe cliente de usuario en la página 1552</a>                            |
| FlexPendant Interface                            | <a href="#">Especificaciones del producto - Controlador IRC5</a>                                       |
| Cyclic bool                                      | <a href="#">Application manual - Controller software IRC5</a>  |

## 1.322 UIShow - Visualización de interfaz de usuario

### Utilización

UIShow (*User Interface Show*) se usa para comunicarse con el usuario del sistema de robot a través de un dispositivo de usuario disponible, como el FlexPendant. Con UIShow tanto las aplicaciones individuales como las aplicaciones estándar pueden ser iniciadas desde un programa de RAPID.

### Ejemplos básicos

Los ejemplos siguientes ilustran la instrucción UIShow.

Los ejemplos 1 y 2 sólo funcionan si los archivos TpsViewMyAppl.dll y TpsViewMyAppl.gtpu.dll están presentes en el directorio HOME: y se ha realizado un Reinicio.

#### Ejemplo 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;
```

El código anterior inicia la vista TpsViewMyAppl con el comando de inicialización Cmd1 y a continuación actualiza la vista con Cmd2.

#### Ejemplo 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;
```

El código anterior inicia la vista TpsViewMyAppl con el comando de inicialización Cmd1. A continuación, inicia otra vista con el comando de inicialización Cmd2.

#### Ejemplo 3

```
CONST string Name:="tpsviewbackupandrestore.dll";
CONST string Type:="ABB.Robotics.Tps.Views.TpsViewBackupAndRestore";
```

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.322 UIShow - Visualización de interfaz de usuario

RobotWare - OS

Continuación

```
VAR num mystatus:=0;  
...  
UIShow Name, Type \Status:=mystatus;
```

Inicio de las aplicaciones estándar de copia de seguridad y restauración.

### Ejemplo 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;
```

Inicie una aplicación creada con ScreenMaker.

---

### Argumentos

```
UIShow AssemblyName TypeName [\InitCmd] [\InstanceId] [\Status]  
[\NoCloseBtn]
```

AssemblyName

Tipo de dato: string

El nombre del conjunto que contiene la vista.

TypeName

Tipo de dato: string

Éste es el nombre de la vista (del tipo a crear). Éste es el nombre completo del nombre del tipo, es decir, con su espacio de nombres incluido.

[\InitCmd]

*Init Command*

Tipo de dato: string

Una cadena de datos de inicialización entregada a la vista.

[\InstanceId]

Tipo de dato: uishownum

Un parámetro que representa un token utilizado para identificar una vista. Si se muestra una vista a continuación de la llamada a `UIShow`, se devuelve un valor que identifica la vista. A continuación, este token puede usarse en otras llamadas a `UIShow` para activar una vista que ya está en funcionamiento. Si el valor identifica una vista existente (en ejecución), se activa la vista. Si no existe, se crea una nueva instancia. Esto significa que este parámetro puede usarse para determinar si se iniciará una nueva instancia o no. Si su valor identifica a una vista ya iniciada, esta vista se activará, independientemente de los valores de los demás parámetros. Una recomendación es usar una variable `InstanceId` exclusiva para cada nueva aplicación que se prevea iniciar con la instrucción `UIShow`.

El parámetro debe ser una variable persistente y el motivo para ello es que esta variable debe conservar su valor, incluso si el puntero de programa se mueve a `Main`. Si se ejecuta la misma instrucción `UIShow` que antes y se usa la misma variable, se activa la misma siempre y cuando siga abierta. Si la vista ha sido cerrada, se inicia una nueva vista.

Continúa en la página siguiente

[\Status]

Tipo de dato: num

Status indica si la operación tuvo éxito o no. Recuerde que si se usa esta opción, la ejecución de RAPID permanecerá en espera hasta que se completa la instrucción, es decir, la vista se inicia.

Este parámetro opcional se usa principalmente para fines de depuración. (Consulte *Gestión de errores*)

| Status | Descripción   |
|--------|---|
| 0      | Correcto  |
| -1     | No queda espacio en el FlexPendant para la nueva vista. Es posible tener abiertas un máximo de 6 vistas a la vez en el FlexPendant. |
| -2     | No se encuentra el conjunto. No existe.   |
| -3     | El archivo se ha encontrado, pero no es posible cargarlo.   |
| -4     | El conjunto existe, pero no puede crearse ninguna instancia nueva.  |
| -5     | El valor de typename no es válido para este conjunto.   |
| -6     | InstanceID no se corresponde con el conjunto a cargar.  |

[\NoCloseBtn]

*No Close Button*

Tipo de dato: switch

NoCloseBtn desactiva el botón Cerrar de la vista.

## Ejecución de programas

La instrucción UIShow se utiliza para iniciar aplicaciones individuales en el FlexPendant. Para iniciar aplicaciones individuales, los conjuntos deben estar situados en el directorio HOME: del sistema activo, ya sea directamente en el sistema activo o en una opción adicional. La recomendación es situar los archivos en el directorio HOME:, de forma que se incluyan en las operaciones de copia de seguridad y restauración. Se requiere un Reinicio, tras lo cual el FlexPendant carga los nuevos conjuntos.

Una exigencia del sistema es que se use la opción de RobotWare *FlexPendant Interface*.

También es posible iniciar aplicaciones estándar, como la copia de seguridad y la restauración. En este caso, no existe la exigencia de tener instalada la opción de RobotWare *FlexPendant Interface*.

Si se usa el argumento \Status, la ejecución del programa espera hasta el inicio de la aplicación. Si los errores de la aplicación no son gestionados, lo único que se supervisa es el resultado del inicio. Sin el parámetro \Status, se ordena al FlexPendant que inicie la aplicación, pero no existe ninguna comprobación para determinar si el inicio es posible o no.

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.322 UIShow - Visualización de interfaz de usuario

RobotWare - OS

Continuación

### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                        | Causa del error   |
|-------------------------------|---|
| <code>ERR_TP_NO_CLIENT</code> | No hay ningún cliente, por ejemplo un FlexPendant, a cargo de la instrucción. |

Si se usa el parámetro `\Status`, estas situaciones pueden ser gestionadas en el gestor de errores:

- Si no queda espacio libre en el FlexPendant para el conjunto, la variable de sistema `ERRNO` cambia a `ERR_UISHOW_FULL` y la ejecución continúa en el gestor de errores. El FlexPendant puede tener abiertas 6 vistas a la vez.
- Si algo sale mal al intentar iniciar una vista, la variable de sistema `ERRNO` cambia a `ERR_UISHOW_FATAL` y la ejecución continúa en el gestor de errores.

### Limitaciones

Al utilizar la instrucción `UIShow` para iniciar aplicaciones individuales, existe la exigencia de que el sistema esté dotado de la opción *FlexPendant Interface*.

Las aplicaciones iniciadas con la instrucción `UIShow` no sobreviven a las situaciones de caída de alimentación. Es posible usar la rutina de evento `POWER ON` para volver a configurar la aplicación.

### Sintaxis

```
UIShow
  [AssemblyName ':=' ] < expression (IN) of string > ','
  [TypeName ':=' ] < expression (IN) of string > ','
  [ '\InitCmd' :=' < expression (IN) of string > ]
  [ '\InstanceId ':=' < persistent (PERS) of uishownum > ]
  [ '\Status ':=' < variable (VAR) of num > ]
  [ '\NoCloseBtn ]';'
```

### Información relacionada

| Para obtener más información sobre                        | Consulte  |
|---|---|
| FlexPendant interface                                     | <i>Especificaciones del producto - Controlador IRC5</i>                                       |
| Creación de aplicaciones individuales para el FlexPendant | <a href="http://developercenter.robotstudio.com/">http://developercenter.robotstudio.com/</a> |
| uishownum   | <i>uishownum - ID de instancia para UIShow en la página 1820</i>                              |
| Borrado de la ventana de operador                         | <i>TPERase - Borra el texto mostrado en el FlexPendant en la página 877</i>                   |



## 1.323 UnLoad - Descargar un módulo de programa durante la ejecución

### Utilización

UnLoad se utiliza para descargar un módulo de programa de la memoria durante la ejecución.

El módulo de programa debe haberse cargado previamente en la memoria de programa mediante las instrucciones Load o StartLoad - WaitLoad.

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción UnLoad:

Consulte también [Más ejemplos en la página 1022](#), a continuación.

#### Ejemplo 1

```
UnLoad diskhome \File:="PART_A.MOD";
```

Realizar una operación UnLoad para descargar de la memoria de programas el módulo de programa PART\_A.MOD, que fue cargado anteriormente en la memoria Load. (Consulte la instrucción Load). diskhome es una constante de cadena predefinida "HOME:".

### Argumentos

```
UnLoad [\ErrIfChanged] | [\Save] FilePath [\File]
```

[\ErrIfChanged]

Tipo de dato: switch

Si se usa este argumento y el módulo ha cambiado desde su carga en el sistema, la instrucción generará el código de recuperación de errores ERR\_NOTSAVED.

[\Save]

Tipo de dato: switch

Si se utiliza este argumento, el módulo de programa se guarda antes de que comience la descarga. El módulo de programa se guarda en la posición original especificada en la instrucción Load o StartLoad.

FilePath

Tipo de dato: string

La ruta y el nombre del archivo que se descargará de la memoria de programa. La ruta y el nombre del archivo deben ser los mismos que en la instrucción Load o StartLoad ejecutadas anteriormente. El nombre de archivo se excluye cuando se utiliza el argumento \File .

[\File]

Tipo de dato: string

Cuando se excluye el nombre del archivo en el argumento FilePath, es necesario definirlo con este argumento. El nombre del archivo debe ser el mismo que en la instrucción Load o StartLoad ejecutadas anteriormente.

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.323 UnLoad - Descargar un módulo de programa durante la ejecución

RobotWare - OS

Continuación

---

### Ejecución de programas

Para poder ejecutar una instrucción `UnLoad` en el programa, es necesario haber ejecutado anteriormente en el programa una instrucción `Load` o `StartLoad` - `WaitLoad`.

La ejecución del programa espera a que el módulo de programa termine de cargarse antes de que la ejecución continúe con la instrucción siguiente.

A continuación, se descarga al módulo de programa y se vinculan los demás módulos de programa.

Para obtener más información, consulte las instrucciones `Load` o `StartLoad-Waitload`.

---

### Más ejemplos

A continuación aparecen más ejemplos de cómo usar la instrucción `UnLoad`.

#### Ejemplo 1

```
UnLoad "HOME:/DOORDIR/DOOR1.MOD" ;
```

Realizar una operación `UnLoad` para descargar de la memoria de programas el módulo de programa `DOOR1.MOD`, que fue cargado anteriormente en la memoria.

#### Ejemplo 2

```
UnLoad "HOME:" \File:="DOORDIR/DOOR1.MOD" ;
```

Lo mismo que en el ejemplo 1 anterior pero con otra sintaxis.

#### Ejemplo 3

```
Unload \Save, "HOME:" \File:="DOORDIR/DOOR1.MOD" ;
```

Lo mismo que en los ejemplos 1 y 2 anteriores, pero guarda el módulo de programa antes de la descarga.

---

### Limitaciones

No se permite descargar módulos de programa que se están ejecutando (puntero de programa en el módulo).

Las rutinas TRAP, los eventos de E/S del sistema y otras tareas de programa no pueden ejecutarse durante la descarga.

Evite tener movimientos en curso durante la descarga.

La detención del programa durante la ejecución de la instrucción `UnLoad` puede dar lugar a un paro protegido con los motores apagados y un mensaje de error "20025 Tiempo excesivo Orden paro" en el FlexPendant.

---

### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                    | Causa del error  |
|---------------------------|--|
| <code>ERR_NOTSAVED</code> | Se utiliza el argumento <code>\ErrIfChanged</code> y el módulo se ha cambiado. |

Continúa en la página siguiente

---

## 1.323 UnLoad - Descargar un módulo de programa durante la ejecución

RobotWare - OS

Continuación

| Nombre     | Causa del error   |
|------------|---|
| ERR_UNLOAD | No es posible descargar el archivo de la instrucción UnLoad debido a que se está ejecutando el módulo o a que se indica una ruta incorrecta (cuando el módulo no se ha cargado con Load o StartLoad). |

### Sintaxis

```
UnLoad
['\ErrIfChanged ',''] | ['\Save ','']
[FilePath':=']<expression (IN) of string>
['\File':=' <expression (IN) of string>'];'
```

### Información relacionada

| Para obtener más información sobre | Consulte   |
|------------------------------------|--|
| Comprobar referencias de programa  | <a href="#">CheckProgRef - Comprobar referencias de programa en la página 131</a>  |
| Carga de un módulo de programa     | <a href="#">Load - Carga un módulo de programa durante la ejecución en la página 347</a><br><a href="#">StartLoad - Carga de programa durante la ejecución en la página 792</a><br><a href="#">WaitLoad - Conectar un módulo cargado a una tarea en la página 1064</a> |

# 1 Instrucciones

## 1.324 UnpackRawBytes - Desempaquetar datos de un dato de tipo rawbytes

RobotWare - OS

## 1.324 UnpackRawBytes - Desempaquetar datos de un dato de tipo rawbytes

### Utilización

UnpackRawBytes se utiliza para desempaquetar el contenido de un contenedor de tipo rawbytes en variables de tipo byte, num, dnum o string.

### Ejemplos básicos

El ejemplo siguiente ilustra la instrucción UnpackRawBytes.

#### Ejemplo 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;
```

De acuerdo con el protocolo conocido por el programador, el mensaje se envía al dispositivo "chan1:". A continuación, la respuesta es leída desde el dispositivo.

Por ejemplo, la respuesta contiene lo siguiente:

| Número de byte: | Contenido:   |
|-----------------|--|
| 1-4             | entero 5   |
| 5-8             | flotante 234.6   |
| 9-25            | cadena "This is real fun!". Se trata de una cadena ISO 8859-1 (Latin-1), con caracteres de un solo byte. |
| 26              | valor hexadecimal '4D'   |
| 27              | Código ASCII 122, es decir, z  |
| 28-36           | entero 4294967295  |
| 37-40           | entero 4294967295  |

```
UnpackRawBytes raw_data_in, 1, integer \IntX := DINT;
```

El contenido de integer será el número entero 5.

```
UnpackRawBytes raw_data_in, 5, float \Float4;
```

El contenido de float será el número con decimales 234,6.

```
UnpackRawBytes raw_data_in, 9, string1 \ASCII:=17;
```

El contenido de string1 será el número entero "This is real fun!".

```
UnpackRawBytes raw_data_in, 26, byte1 \Hex1;
```

Continúa en la página siguiente

## 1.324 UnpackRawBytes - Desempaqueta datos de un dato de tipo rawbytes

RobotWare - OS

Continuación

El contenido de `byte1` será el valor hexadecimal '4D'.

```
UnpackRawBytes raw_data_in, 27, data1 \ASCII:=1;
```

El contenido de `data1` será 122, el código ASCII de "z".

```
UnpackRawBytes raw_data_in, 28, bigInt \IntX := LINT;
```

El contenido de `bigInt` será el número entero 4294967295.

```
UnpackRawBytes raw_data_in, 37, bigInt \IntX := UDINT;
```

El contenido de `bigInt` será el número entero 4294967295.

## Argumentos

```
UnpackRawBytes RawData [ \Network ] StartIndex Value [\Hex1 ] | [
  \IntX ] | [ \Float4 ] | [ \ASCII ]
```

Consulte [Combinación de los argumentos en la página 1026](#).

RawData

Tipo de dato: rawbytes

El contenedor de variable del cual se desempaquetarán los datos.

[ \Network ]

Tipo de dato: switch

Indica que los valores enteros y flotantes deben desempaquetarse con la representación big-endian (orden de red) de `RawData`. Tanto ProfiBus como InterBus utilizan big-endian.

Sin este modificador, los valores enteros y flotantes se desempaquetan con la representación little-endian (sin el orden de la red) de `RawData`. DeviceNet utiliza little-endian.

Sólo relevante junto con el parámetro opcional `\IntX - UINT, UDINT, , ULINT, INT, DINT, LINT y \Float4`.

StartIndex

Tipo de dato: num

`StartIndex`, entre 1 y 1.024, indica a partir de dónde debe empezar el desempaqueado de `RawData`.

Value

Tipo de dato: anytype

La variable que contiene los datos desempaqueados de `RawData`.

Los tipos de datos permitidos son: `byte`, `num`, `dnum` o `string`. No se permite el uso de matrices.

[ \Hex1 ]

Tipo de dato: switch

Los datos a desempaquear y situar en `Value` tienen el formato hexadecimal en 1 `byte` y se convierten al formato decimal en una variable de tipo `byte`.

[ \IntX ]

Tipo de dato: inttypes

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.324 UnpackRawBytes - Desempaqueta datos de un dato de tipo rawbytes

RobotWare - OS

Continuación

Los datos a desempaquetar tienen el formato que corresponde a la constante especificada del tipo `inttypes`. Los datos se convertirán a una variable de tipo `num` o `dnum` que contiene un entero y se almacenan en `Value`.

Consulte [Datos predefinidos en la página 1026](#).

[ `\Float4` ]

Tipo de dato: `switch`

Los datos a desempaquetar y situar en `Value` tienen el formato flotante de 4 bytes y se convertirán en una variable de tipo `num` que contiene un valor de coma flotante.

[ `\ASCII` ]

Tipo de dato: `num`

Los datos a desempaquetar y situar en `Value` tienen el formato `byte` o `string`.

Si `Value` es de tipo `byte`, los datos se interpretarán como código ASCII y se convertirán al formato `byte` (1 carácter).

Si `Value` es del tipo `string`, los datos se almacenarán como `string` (de 1 a 80 caracteres). Los datos de cadena no son terminados con NULL en el caso de los datos de tipo `rawbytes`.

### Combinación de los argumentos

Se debe utilizar uno de los argumentos `\Hex1`, `\IntX`, `\Float4`, o `\ASCII`.

Se permiten las combinaciones siguientes:

| Tipo de dato de <code>Value</code> : | Parámetros opcionales permitidos:          |
|--------------------------------------|--|
| <code>num<sup>i</sup></code>         | <code>\IntX</code>                         |
| <code>dnum<sup>ii</sup></code>       | <code>\IntX</code>                         |
| <code>num</code>                     | <code>\Float4</code>                       |
| <code>string</code>                  | <code>\ASCII</code> (de 1 a 80 caracteres) |
| <code>byte</code>                    | <code>\Hex1</code> <code>\ASCII</code>     |

<sup>i</sup> Debe ser un entero dentro del rango de valor de la constante simbólica seleccionada, `USINT`, `UINT`, `UDINT`, `SINT`, `INT` o `DINT`.

<sup>ii</sup> Debe ser un entero dentro del rango de valor de la constante simbólica seleccionada, `USINT`, `UINT`, `UDINT`, `ULINT`, `SINT`, `INT`, `DINT` o `LINT`.

### Ejecución de programas

Durante la ejecución del programa, se desempaquetan los datos del contenedor del tipo `rawbytes` en una variable de tipo `anytype`.

En caso de un reinicio tras una caída de alimentación, todos los archivos o dispositivos de E/S abiertos del sistema se cierran y el descriptor de E/S de la variable del tipo `iodev` se restablece.

### Datos predefinidos

Se han definido las constantes simbólicas siguientes para el tipo de dato `inttypes`. Puede usarlas para especificar el entero en el parámetro `\IntX`.

| Constante simbólica | Valor constante | Formato de entero          | Rango de valores enteros |
|---------------------|-----------------|----------------------------|--------------------------|
| <code>USINT</code>  | 1               | Entero de 1 byte sin signo | 0 ... 255                |

Continúa en la página siguiente

## 1.324 UnpackRawBytes - Desempaqueta datos de un dato de tipo rawbytes

RobotWare - OS

Continuación

| Constante simbólica | Valor constante | Formato de entero           | Rango de valores enteros   |
|---------------------|-----------------|-----------------------------|--|
| UINT                | 2               | Entero de 2 byte sin signo  | 0 ... 65 535   |
| UDINT               | 4               | Entero de 4 byte sin signo  | 0 ... 8 388 608 <sup>i</sup><br>0 ... 4 294 967 295 <sup>ii</sup>                        |
| ULINT               | 8               | Entero de 8 byte sin signo  | 0 ... 4 503 599 627 370 496 <sup>iii</sup>   |
| SINT                | - 1             | Entero de 1 bytes con signo | - 128... 127   |
| INT                 | - 2             | Entero de 2 bytes con signo | - 32 768 ... 32 767  |
| DINT                | - 4             | Entero de 4 bytes con signo | - 8 388 607 ... 8 388 608 <sup>i</sup><br>-2 147 483 648 ... 2 147 483 647 <sup>iv</sup> |
| LINT                | - 8             | Entero de 8 bytes con signo | - 4 503 599 627 370 496... 4 503 599 627 370 496 <sup>iii</sup>                          |

- <sup>i</sup> Limitación de RAPID para el almacenamiento de enteros en el tipo de dato num.  
<sup>ii</sup> Rango al utilizar una variable dnum e inttype UDINT.  
<sup>iii</sup> Limitación de RAPID para el almacenamiento de enteros en el tipo de dato dnum.  
<sup>iv</sup> Rango al utilizar una variable dnum e inttype DINT.

## Sintaxis

```
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> ';'
```

## Información relacionada

| Para obtener más información sobre                              | Consulte   |
|---|--|
| rawbytes datos  | <a href="#">rawbytes - Datos sin formato en la página 1736</a>   |
| Obtención de la longitud de un dato rawbytes                    | <a href="#">RawBytesLen - Obtiene la longitud de un dato de tipo rawbytes en la página 1432</a>        |
| Borrado del contenido de un dato de tipo rawbytes               | <a href="#">ClearRawBytes - Borra el contenido de un dato de tipo rawbytes en la página 146</a>        |
| Copiado del contenido de un dato de tipo rawbytes               | <a href="#">CopyRawBytes - Copia el contenido de un dato de tipo rawbytes en la página 171</a>         |
| Empaquetamiento de un encabezado de DeviceNet en datos rawbytes | <a href="#">PackDNHeader - Empaqueta un encabezado de DeviceNet en datos rawbytes en la página 496</a> |
| Empaquetamiento de datos en datos rawbytes                      | <a href="#">PackRawBytes - Empaqueta datos en un dato de tipo rawbytes en la página 499</a>            |
| Escritura de un dato rawbytes                                   | <a href="#">WriteRawBytes - Escribe un dato de tipo rawbytes en la página 1111</a>                     |
| Lectura de un dato rawbytes                                     | <a href="#">ReadRawBytes - Lee datos de tipo rawbytes en la página 589</a>                             |

Continúa en la página siguiente

# 1 Instrucciones

---

## 1.324 UnpackRawBytes - Desempaqueta datos de un dato de tipo rawbytes

RobotWare - OS

Continuación

| Para obtener más información sobre        | Consulte   |
|---|--|
| Funciones para bits/bytes                 | <i>Manual de referencia técnica - RAPID Overview</i> |
| Funciones para cadenas de caracteres      | <i>Manual de referencia técnica - RAPID Overview</i> |
| Gestión de archivos y dispositivos de E/S | <i>Application manual - Controller software IRC5</i> |



### 1.325 VelSet - Cambia la velocidad programada

#### Utilización

`VelSet` se utiliza para aumentar o reducir la velocidad programada de todas las instrucciones de movimiento posteriores. Esta instrucción también se utiliza para limitar la velocidad máxima de TCP.

Esta instrucción sólo puede usarse en la tarea principal `T_ROB1` o, si se cuenta con un sistema *MultiMove*, en las tareas de movimiento.

#### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción `VelSet`:

Consulte también [Más ejemplos en la página 1030](#).

#### Ejemplo 1

```
VelSet 50, 800;
```

Todas las velocidades programadas se reducen hasta el 50% del valor indicado en la instrucción. No se permite que TCP tenga una velocidad superior a los 800 mm/s.

#### Argumentos

```
VelSet Override Max
```

##### Override

Tipo de dato: num

La velocidad deseada, como un porcentaje de la velocidad programada. El 100% corresponde a la velocidad programada.

##### Max

Tipo de dato: num

La velocidad máxima del TCP en mm/s.

#### Ejecución de programas

La velocidad programada se aplica a la siguiente instrucción de movimiento ejecutada y es válida hasta que se ejecute una nueva instrucción `VelSet`.

El argumento `Override` afecta a lo siguiente:

- Todos los componentes de velocidad (TCP, orientación, ejes externos de rotación y ejes externos lineales) de `speeddata`.
- La redefinición de velocidad programada en la instrucción de posicionamiento (el argumento `\V`).
- Movimientos temporizados.

El argumento `Override` no afecta a lo siguiente:

- La velocidad de soldado de `welddata`.
- El calentamiento y la velocidad de llenado de `seamdata`.

El argumento `Max` solo limita la velocidad de la TCP si está es más lenta que la velocidad programada.

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.325 VelSet - Cambia la velocidad programada

RobotWare - OS

Continuación

Los valores predeterminados de `Override` y `Max` son 100 % y `vmax.v_tcp` mm/s respectivamente. Estos valores se establecen automáticamente

- cuando se utiliza el modo de reinicio **Restablecer RAPID**
- al cargar un nuevo programa o un nuevo módulo
- al iniciar la ejecución del programa desde el principio
- al mover el puntero del programa a `main`
- al mover el puntero del programa a una rutina
- al mover el puntero de programa de una forma que se pierde el orden de la ejecución.



### Nota

La velocidad máxima de TCP para el tipo de robot utilizado puede cambiarse en los parámetros *Motion* de configuración del sistema, escriba *Motion Planner* y el atributo *Linear Max Speed*. La función de RAPID `MaxRobSpeed` devuelve el mismo valor.

## Más ejemplos

A continuación aparecen más ejemplos de cómo usar la instrucción `VelSet`.

### Ejemplo 1

```
VelSet 50, 800;  
MoveL p1, v1000, z10, tool1;  
MoveL p2, v2000, z10, tool1;  
MoveL p3, v1000\T:=5, z10, tool1;
```

La velocidad es 500 mm/s hasta el punto `p1` y 800 mm/s hasta el punto `p2`. Tarda 10 segundos en desplazarse de `p2` hasta `p3`.

## Limitaciones

La velocidad máxima no se tiene en cuenta cuando se especifica un tiempo en la instrucción de movimiento.

## Sintaxis

```
VelSet  
[ Override ':=' ] < expression (IN) of num > ','  
[ Max ':=' ] < expression (IN) of num > ';' 
```

## Información relacionada

| Para obtener más información sobre            | Consulte   |
|---|--|
| Reducción de la aceleración                   | <a href="#">AccSet - Reduce la aceleración en la página 25</a>                   |
| Velocidad máxima del TCP para el robot actual | <a href="#">MaxRobSpeed - Velocidad máxima del robot en la página 1373</a>       |
| Datos de parámetros de movimiento             | <a href="#">motsetdata - Datos de parámetros de movimiento en la página 1705</a> |
| Definición de velocidad                       | <a href="#">speeddata - Datos de velocidad en la página 1767</a>                 |

Continúa en la página siguiente

|   |  |
|---|--|
| <b>Para obtener más información sobre</b> | <b>Consulte</b>                                      |
| Instrucciones de posicionamiento          | <i>Manual de referencia técnica - RAPID Overview</i> |

# 1 Instrucciones

---

1.326 WaitAI - Espera hasta que se establece un valor de señal analógica de entrada  
RobotWare - OS

## 1.326 WaitAI - Espera hasta que se establece un valor de señal analógica de entrada

---

### Utilización

WaitAI (*Wait Analog Input*) se usa para esperar hasta que se establece un valor de señal analógica de entrada.

---

### Ejemplos básicos

Los ejemplos siguientes ilustran la instrucción WaitAI.

#### Ejemplo 1

```
WaitAI ail, \GT, 5;
```

La ejecución del programa sólo continúa una vez que la entrada analógica ail tiene un valor mayor que 5.

#### Ejemplo 2

```
WaitAI ail, \LT, 5;
```

La ejecución del programa sólo continúa una vez que la entrada analógica ail tiene un valor inferior a 5.

---

### Argumentos

```
WaitAI Signal [\LT] | [\GT] Value [\MaxTime] [\ValueAtTimeout]  
[\Visualize] [\Header] [\Message] | [\MsgArray] [\Wrap]  
[\Icon] [\Image] [\VisualizeTime] [\UIActiveSignal]
```

Signal

Tipo de dato: signalai

El nombre de la señal analógica de entrada.

[\LT]

*Less Than*

Tipo de dato: switch

Si se usa este parámetro, la instrucción WaitAI espera hasta que el valor de la señal analógica es inferior que el valor de Value.

[\GT]

*Greater Than*

Tipo de dato: switch

Si se usa este parámetro, la instrucción WaitAI espera hasta que el valor de la señal analógica es mayor que el valor de Value.

Value

Tipo de dato: num

El valor deseado para la señal.

[\MaxTime]

*Maximum Time*

Tipo de dato: num

*Continúa en la página siguiente*

## 1.326 WaitAI - Espera hasta que se establece un valor de señal analógica de entrada RobotWare - OS Continuación

El periodo máximo permitido para el tiempo de espera, expresado en segundos. Si el tiempo se agota antes de que se cumpla la condición, se llama al gestor de errores si lo hay, con el código de error `ERR_WAIT_MAXTIME`. Si no hay ningún gestor de errores, se detiene la ejecución.

`[\ValueAtTimeout]`

Tipo de dato: `num`

Si la instrucción alcanza el tiempo límite, el valor actual de la señal se almacena en esta variable. La variable sólo se define si la variable de sistema `ERRNO` recibe el valor `ERR_WAIT_MAXTIME`.

`[\Visualize]`

Tipo de dato: `switch`

Si está seleccionado, se activa la visualización. La visualización consta de un cuadro de mensaje con la condición que no se cumple, icono, título, líneas de mensaje e imagen, de acuerdo con los argumentos del programa.

`[\Header]`

Tipo de dato: `string`

El texto de cabecera se escribirá en la parte superior del cuadro de mensaje. 40 caracteres como máximo. Si no se utilizara ningún argumento `\Header`, se mostrará un mensaje predeterminado.

`[\Message]`

Tipo de dato: `string`

Se escribirá una línea de texto en la pantalla. Máximo 50 caracteres.

`[\MsgArray]`

*(Message Array)*

Tipo de dato: `string`

Varias líneas de texto de una matriz para escribir en la pantalla. Solo es posible usar uno de los parámetros `\Message` o `\MsgArray` al mismo tiempo.

El máximo espacio es de 5 líneas con 50 caracteres cada una.

`[\Wrap]`

Tipo de dato: `switch`

Si se selecciona, todas las cadenas especificadas en el argumento `\MsgArray` se concatenan para formar una cadena con un solo espacio entre las distintas cadenas individuales y distribuida en el número mínimo posible de líneas.

De forma predeterminada, cada una de las cadenas del argumento `\MsgArray` aparece en una línea separada de la pantalla.

`[\Icon]`

Tipo de dato: `icondata`

Define el icono a mostrar. Sólo puede usarse uno de los iconos predefinidos de tipo `icondata`. Consulte [Datos predefinidos en la página 1683](#).

De forma predeterminada, no se usa ningún icono.

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.326 WaitAI - Espera hasta que se establece un valor de señal analógica de entrada

RobotWare - OS

Continuación

[\Image]

Tipo de dato: `string`

El nombre de la imagen que debe utilizarse. Para iniciar sus propias imágenes, las imágenes deben estar situadas en el directorio *HOME*: del sistema activo o directamente en el sistema activo.

La recomendación es situar los archivos en el directorio *HOME*: de forma que se incluyan en las operaciones de copia de seguridad y restauración.

Se requiere un reinicio y, a continuación, el FlexPendant carga las imágenes.

Una exigencia del sistema es que se use la opción de RobotWare *FlexPendant Interface*.

La imagen a mostrar puede tener 185 píxeles de anchura y 300 píxeles de altura. Si la imagen tiene un tamaño mayor, solo se muestran 185 \* 300 píxeles de la imagen a partir de su parte superior izquierda.

No es posible especificar ningún valor exacto en cuanto al tamaño que una imagen puede tener o el número de imágenes que es posible cargar en el FlexPendant. Depende del tamaño de los demás archivos cargados en el FlexPendant. La ejecución de los programas continúa simplemente si se usa una imagen que no está cargada en el FlexPendant.

[\VisualizeTime]

Tipo de dato: `num`

El tiempo de espera antes del cuadro de mensaje debe aparecer en FlexPendant. Si se utilizan los argumentos `\VisualizeTime` y `\MaxTime`, el tiempo utilizado en el argumento `\MaxTime` tiene que ser mayor que el tiempo utilizado en el argumento `\VisualizeTime`.

El tiempo predeterminado para la visualización si no se utiliza el argumento `\VisualizeTime` es 5 s. Valor mínimo 1 s. No hay límite de valor máximo. Resolución 0.001 s.

[\UIActiveSignal]

Tipo de dato: `signaldo`

La señal digital de salida utilizada en el argumento opcional `UIActiveSignal` se establece en 1 cuando se activa el cuadro de mensaje de visualización en FlexPendant. Cuando se ha retirado el cuadro de mensaje (cuando se cumple la condición), la señal se vuelve a establecer en 0.

No existe ninguna supervisión de parada o reinicio. La señal se establece en 0 cuando la instrucción está preparada o cuando se mueve el PP.

---

### Ejecución de programas

Si el valor de la señal es correcto cuando se ejecuta la instrucción, el programa sencillamente continúa con la instrucción siguiente.

Si el valor de la señal no es correcto, el robot entra en un estado de espera y el programa continúa tan pronto como la señal cambie al valor correcto. El cambio se detecta mediante una interrupción, lo cual genera una respuesta rápida (no sondeada).

*Continúa en la página siguiente*

## 1.326 WaitAI - Espera hasta que se establece un valor de señal analógica de entrada *RobotWare - OS* *Continuación*

Cuando el robot está en espera, el tiempo se supervisa. De forma predeterminada, el robot puede esperar para siempre, pero el tiempo de espera máximo puede especificarse con el argumento opcional `\MaxTime`. Si se sobrepasa este tiempo máximo, se genera un error.

Si la ejecución del programa se detiene y se reanuda a continuación, la instrucción evalúa el valor actual de la señal. Se rechaza cualquier cambio durante el paro del programa.

En el modo manual, después de esperar más de 3 s, aparecerá una ventana de alerta que pregunta si se debe simular la instrucción. Puede configurarse que la alerta no aparezca, estableciendo el parámetro del sistema *SimulateMenu* en NO, consulte *Manual de referencia técnica - Parámetros del sistema*, tema *Controller*, tipo *General RAPID*.

Si se utiliza el modificador `\Visualize`, aparece un cuadro de mensaje en FlexPendant acorde con los argumentos programados. Si no se utiliza el argumento `\Header`, aparece un texto de cabecera predeterminado. Cuando la ejecución de la instrucción `WaitAI` está preparada, el cuadro de mensaje se elimina de FlexPendant.

El nuevo cuadro de mensaje del nivel de rutina TRAP toma el foco del cuadro de mensaje del nivel básico.

---

### Más ejemplos

A continuación aparecen más ejemplos de la instrucción `WaitAI`.

#### Ejemplo 1

```
VAR num myvalattimeout:=0;
WaitAI ail, \LT, 5 \MaxTime:=4 \ValueAtTimeout:=myvalattimeout;
ERROR
  IF ERRNO=ERR_WAIT_MAXTIME THEN
    TPWrite "Value of ail at timeout:" + ValToStr(myvalattimeout);
    TRYNEXT;
  ELSE
    ! No error recovery handling
  ENDIF
```

La ejecución del programa sólo continúa si `ail` es menor que 5 o al alcanzar el tiempo límite. En caso de que se alcance el tiempo límite, el valor de la señal `ail` en el momento del tiempo límite puede registrarse sin otra lectura de la señal.

#### Ejemplo 2

```
WaitAI ail \GT, 5 \Visualize \Header:="Waiting for signal"
  \MsgArray:["Movement will not start until", "the condition
  below is TRUE"] \Icon:=iconError;
MoveL p40, v500, z20, L10tip;
..
```

Si no se cumple la condición, entonces la cabecera y el mensaje especificado en los argumentos opcionales `\Header` y `\MsgArray` se escribirán en la pantalla del FlexPendant junto con la condición que no se cumple.

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.326 WaitAI - Espera hasta que se establece un valor de señal analógica de entrada

RobotWare - OS

Continuación

### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                          | Causa del error   |
|---------------------------------|---|
| <code>ERR_AO_LIM</code>         | El argumento programado <code>Value</code> para la señal analógica de entrada especificada <code>Signal</code> está fuera de límites.   |
| <code>ERR_NO_ALIASIO_DEF</code> | La variable de señal es una variable declarada en <code>RAPID</code> . No se ha conectado a una señal E/S definida en la configuración de E/S con la instrucción <code>AliasIO</code> .               |
| <code>ERR_NORUNUNIT</code>      | Se ha perdido el contacto con el dispositivo de E/S.  |
| <code>ERR_SIG_NOT_VALID</code>  | La señal de E/S no está disponible. Los motivos pueden ser que el dispositivo de E/S no está en funcionamiento o que exista un error en la configuración (válido solamente para el bus de campo ICI). |
| <code>ERR_WAIT_MAXTIME</code>   | Existe un tiempo límite (parámetro <code>\MaxTime</code> ) antes de que la señal cambie al valor correcto.  |

### Sintaxis

```
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> ]
[ '\ VisualizeTime ':' <expression (IN) of num> ]
[ '\ UIActiveSignal ':' <variable (VAR) of signaldo> ] ';' ;'
```

### Información relacionada

| Para obtener más información sobre                          | Consulte   |
|---|--|
| Espera hasta que se cumple una condición                    | <a href="#">WaitUntil - Esperar hasta que se cumple una condición en la página 1084</a>                        |
| Espera durante un periodo determinado                       | <a href="#">WaitTime - Esperar una cantidad de tiempo determinada en la página 1082</a>                        |
| Espera hasta que se activa o desactiva una salida analógica | <a href="#">WaitAO - Espera hasta que se establece un valor de señal analógica de salida en la página 1037</a> |



---

## 1.327 WaitAO - Espera hasta que se establece un valor de señal analógica de salida

---

### Utilización

WaitAO (*Wait Analog Output*) se usa para esperar hasta que se establece un valor de señal analógica de salida.

---

### Ejemplos básicos

Los ejemplos siguientes ilustran la instrucción WaitAO.

#### Ejemplo 1

```
WaitAO a01, \GT, 5;
```

La ejecución del programa sólo continúa una vez que la salida analógica a01 tiene un valor mayor que 5.

#### Ejemplo 2

```
WaitAO a01, \LT, 5;
```

La ejecución del programa sólo continúa una vez que la salida analógica a01 tiene un valor menor que 5.

---

### Argumentos

```
WaitAO Signal [\LT] | [\GT] Value [\MaxTime] [\ValueAtTimeout]
[\Visualize] [\Header] [\Message] | [\MsgArray] [\Wrap]
[\Icon] [\Image] [\VisualizeTime] [\UIActiveSignal]
```

Signal

Tipo de dato: signalao

El nombre de la señal de salida analógica.

[\LT]

*Less Than*

Tipo de dato: switch

Si se usa este parámetro, la instrucción WaitAO espera hasta que el valor de la señal analógica es inferior que el valor de Value.

[\GT]

*Greater Than*

Tipo de dato: switch

Si se usa este parámetro, la instrucción WaitAO espera hasta que el valor de la señal analógica es mayor que el valor de Value.

Value

Tipo de dato: num

El valor deseado para la señal.

[\MaxTime]

*Maximum Time*

Tipo de dato: num

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.327 WaitAO - Espera hasta que se establece un valor de señal analógica de salida

RobotWare - OS

Continuación

El periodo máximo permitido para el tiempo de espera, expresado en segundos. Si el tiempo se agota antes de que se cumpla la condición, se llama al gestor de errores si lo hay, con el código de error `ERR_WAIT_MAXTIME`. Si no hay ningún gestor de errores, se detiene la ejecución.

`[\ValueAtTimeout]`

Tipo de dato: `num`

Si la instrucción alcanza el tiempo límite, el valor actual de la señal se almacena en esta variable. La variable sólo se define si la variable de sistema `ERRNO` recibe el valor `ERR_WAIT_MAXTIME`.

`[\Visualize]`

Tipo de dato: `switch`

Si está seleccionado, se activa la visualización. La visualización consta de un cuadro de mensaje con la condición que no se cumple, icono, título, líneas de mensaje e imagen, de acuerdo con los argumentos del programa.

`[\Header]`

Tipo de dato: `string`

El texto de cabecera se escribirá en la parte superior del cuadro de mensaje. 40 caracteres como máximo. Si no se utilizara ningún argumento `\Header`, se mostrará un mensaje predeterminado.

`[\Message]`

Tipo de dato: `string`

Se escribirá una línea de texto en la pantalla. Máximo 50 caracteres.

`[\MsgArray]`

*(Message Array)*

Tipo de dato: `string`

Varias líneas de texto de una matriz para escribir en la pantalla. Solo es posible usar uno de los parámetros `\Message` o `\MsgArray` al mismo tiempo.

El máximo espacio es de 5 líneas con 50 caracteres cada una.

`[\Wrap]`

Tipo de dato: `switch`

Si se selecciona, todas las cadenas especificadas en el argumento `\MsgArray` se concatenan para formar una cadena con un solo espacio entre las distintas cadenas individuales y distribuida en el número mínimo posible de líneas.

De forma predeterminada, cada una de las cadenas del argumento `\MsgArray` aparece en una línea separada de la pantalla.

`[\Icon]`

Tipo de dato: `icondata`

Define el icono a mostrar. Sólo puede usarse uno de los iconos predefinidos de tipo `icondata`. Consulte [Datos predefinidos en la página 1683](#).

De forma predeterminada, no se usa ningún icono.

*Continúa en la página siguiente*

## 1.327 WaitAO - Espera hasta que se establece un valor de señal analógica de salida *RobotWare - OS* *Continuación*

[`\Image`]

Tipo de dato: `string`

El nombre de la imagen que debe utilizarse. Para iniciar sus propias imágenes, las imágenes deben estar situadas en el directorio *HOME*: del sistema activo o directamente en el sistema activo.

La recomendación es situar los archivos en el directorio *HOME*: de forma que se incluyan en las operaciones de copia de seguridad y restauración.

Se requiere un reinicio y, a continuación, el FlexPendant carga las imágenes.

Una exigencia del sistema es que se use la opción de RobotWare *FlexPendant Interface*.

La imagen a mostrar puede tener 185 píxeles de anchura y 300 píxeles de altura. Si la imagen tiene un tamaño mayor, solo se muestran 185 \* 300 píxeles de la imagen a partir de su parte superior izquierda.

No es posible especificar ningún valor exacto en cuanto al tamaño que una imagen puede tener o el número de imágenes que es posible cargar en el FlexPendant. Depende del tamaño de los demás archivos cargados en el FlexPendant. La ejecución de los programas continúa simplemente si se usa una imagen que no está cargada en el FlexPendant.

[`\VisualizeTime`]

Tipo de dato: `num`

El tiempo de espera antes del cuadro de mensaje debe aparecer en FlexPendant. Si se utilizan los argumentos `\VisualizeTime` y `\MaxTime`, el tiempo utilizado en el argumento `\MaxTime` tiene que ser mayor que el tiempo utilizado en el argumento `\VisualizeTime`.

El tiempo predeterminado para la visualización si no se utiliza el argumento `\VisualizeTime` es 5 s. Valor mínimo 1 s. No hay límite de valor máximo. Resolución 0.001 s.

[`\UIActiveSignal`]

Tipo de dato: `signaldo`

La señal digital de salida utilizada en el argumento opcional `UIActiveSignal` se establece en 1 cuando se activa el cuadro de mensaje de visualización en FlexPendant. Cuando se ha retirado el cuadro de mensaje (cuando se cumple la condición), la señal se vuelve a establecer en 0.

No existe ninguna supervisión de parada o reinicio. La señal se establece en 0 cuando la instrucción está preparada o cuando se mueve el PP.

---

### Ejecución de programas

Si el valor de la señal es correcto cuando se ejecuta la instrucción, el programa sencillamente continúa con la instrucción siguiente.

Si el valor de la señal no es correcto, el robot entra en un estado de espera y el programa continúa tan pronto como la señal cambie al valor correcto. El cambio se detecta mediante una interrupción, lo cual genera una respuesta rápida (no sondeada).

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.327 WaitAO - Espera hasta que se establece un valor de señal analógica de salida

RobotWare - OS

Continuación

Cuando el robot está en espera, el tiempo se supervisa. De forma predeterminada, el robot puede esperar para siempre, pero el tiempo de espera máximo puede especificarse con el argumento opcional `\MaxTime`. Si se sobrepasa este tiempo máximo, se genera un error.

Si la ejecución del programa se detiene y se reanuda a continuación, la instrucción evalúa el valor actual de la señal. Se rechaza cualquier cambio durante el paro del programa.

En el modo manual, después de esperar más de 3 s, aparecerá una ventana de alerta que pregunta si se debe simular la instrucción. Puede configurarse que la alerta no aparezca, estableciendo el parámetro del sistema `SimulateMenu` en NO, consulte *Manual de referencia técnica - Parámetros del sistema*, tema *Controller*, tipo *General RAPID*.

Si se utiliza el modificador `\Visualize`, aparece un cuadro de mensaje en FlexPendant acorde con los argumentos programados. Si no se utiliza el argumento `\Header`, aparece un texto de cabecera predeterminado. Cuando la ejecución de la instrucción `WaitAO` está preparada, el cuadro de mensaje se elimina de FlexPendant.

El nuevo cuadro de mensaje del nivel de rutina TRAP toma el foco del cuadro de mensaje del nivel básico.

---

### Más ejemplos

A continuación aparecen más ejemplos de la instrucción `WaitAO`.

#### Ejemplo 1

```
VAR num myvalattimeout:=0;
WaitAO aol, \LT, 5 \MaxTime:=4 \ValueAtTimeout:=myvalattimeout;
ERROR
  IF ERRNO=ERR_WAIT_MAXTIME THEN
    TPWrite "Value of aol at timeout:" + ValToStr(myvalattimeout);
    TRYNEXT;
  ELSE
    ! No error recovery handling
  ENDIF
```

La ejecución del programa sólo continúa si `aol` es menor que 5 o al alcanzar el tiempo límite. En caso de que se alcance el tiempo límite, el valor de la señal `aol` en el momento del tiempo límite puede registrarse sin otra lectura de la señal.

#### Ejemplo 2

```
WaitAO aol \GT, 5 \Visualize \Header:="Waiting for signal"
  \MsgArray:["Movement will not start until", "the condition
  below is TRUE"] \Icon:=iconError;
MoveL p40, v500, z20, L10tip;
..
```

Si no se cumple la condición, entonces la cabecera y el mensaje especificado en los argumentos opcionales `\Header` y `\MsgArray` se escribirán en la pantalla del FlexPendant junto con la condición que no se cumple.

---

Continúa en la página siguiente

## Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                          | Causa del error   |
|---------------------------------|---|
| <code>ERR_AO_LIM</code>         | El argumento programado <code>Value</code> para la señal analógica de salida especificada <code>Signal</code> está fuera de límites.  |
| <code>ERR_NO_ALIASIO_DEF</code> | La variable de señal es una variable declarada en <code>RAPID</code> . No se ha conectado a una señal E/S definida en la configuración de E/S con la instrucción <code>AliasIO</code> .               |
| <code>ERR_NORUNUNIT</code>      | Se ha perdido el contacto con el dispositivo de E/S.  |
| <code>ERR_SIG_NOT_VALID</code>  | La señal de E/S no está disponible. Los motivos pueden ser que el dispositivo de E/S no está en funcionamiento o que exista un error en la configuración (válido solamente para el bus de campo ICI). |
| <code>ERR_WAIT_MAXTIME</code>   | Existe un tiempo límite (parámetro <code>\MaxTime</code> ) antes de que la señal cambie al valor correcto.  |

## Sintaxis

```

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> ]
  [ '\ ' VisualizeTime ':= ' <expression (IN) of num> ]
  [ '\ ' UIActiveSignal ':= ' <variable (VAR) of signaldo> ] ';'

```

## Información relacionada

| Para obtener más información sobre                           | Consulte  |
|--|---|
| Espera hasta que se cumple una condición                     | <a href="#">WaitUntil - Esperar hasta que se cumple una condición en la página 1084</a>                         |
| Espera durante un periodo determinado                        | <a href="#">WaitTime - Esperar una cantidad de tiempo determinada en la página 1082</a>                         |
| Espera hasta que se activa o desactiva una entrada analógica | <a href="#">WaitAI - Espera hasta que se establece un valor de señal analógica de entrada en la página 1032</a> |

# 1 Instrucciones

---

## 1.328 WaitDI - Espera hasta que se activa una señal digital de entrada

RobotWare - OS

## 1.328 WaitDI - Espera hasta que se activa una señal digital de entrada

---

### Utilización

WaitDI (*Wait Digital Input*) se usa para esperar hasta que se activa una entrada digital.

---

### Ejemplos básicos

Los ejemplos siguientes ilustran la instrucción WaitDI.

#### Ejemplo 1

```
WaitDI di4, 1;
```

La ejecución del programa sólo continúa después de que se ha activado la entrada di4.

#### Ejemplo 2

```
WaitDI grip_status, 0;
```

La ejecución del programa sólo continúa después de que se ha restablecido la entrada grip\_status.

#### Ejemplo 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;  
..
```

Si no se cumple la condición, entonces la cabecera y el mensaje especificado en los argumentos opcionales `\Header` y `\MsgArray` se escribirán en la pantalla del FlexPendant junto con la condición que no se cumple.

---

### Argumentos

```
WaitDI Signal Value [\MaxTime] [\TimeFlag] [\Visualize] [\Header]  
  [\Message] | [\MsgArray] [\Wrap] [\Icon] [\Image]  
  [\VisualizeTime] [\UIActiveSignal]
```

Signal

Tipo de dato: signaldi

El nombre de la señal.

Value

Tipo de dato: dionum

El valor deseado para la señal.

[\MaxTime]

**Maximum Time**

Tipo de dato: num

El periodo máximo permitido para el tiempo de espera, expresado en segundos. Si el tiempo se agota antes de que se cumpla la condición, se llama al gestor de errores si lo hay, con el código de error `ERR_WAIT_MAXTIME`. Si no hay ningún gestor de errores, se detiene la ejecución.

*Continúa en la página siguiente*

[\TimeFlag]

### *Timeout Flag*

Tipo de dato: `bool`

El parámetro de salida que contiene el valor TRUE si se agota el tiempo máximo de espera permitido antes de que se cumpla la condición. Si se incluye este parámetro en la instrucción, no se considera un error si llega a agotarse el tiempo límite. Este argumento no se tiene en cuenta si el argumento `MaxTime` no se incluye en la instrucción.

[\Visualize]

Tipo de dato: `switch`

Si está seleccionado, se activa la visualización. La visualización consta de un cuadro de mensaje con la condición que no se cumple, icono, título, líneas de mensaje e imagen, de acuerdo con los argumentos del programa.

[\Header]

Tipo de dato: `string`

El texto de cabecera se escribirá en la parte superior del cuadro de mensaje. 40 caracteres como máximo. Si no se utilizara ningún argumento `\Header`, se mostrará un mensaje predeterminado.

[\Message]

Tipo de dato: `string`

Se escribirá una línea de texto en la pantalla. Máximo 50 caracteres.

[\MsgArray]

### *(Message Array)*

Tipo de dato: `string`

Varias líneas de texto de una matriz para escribir en la pantalla. Solo es posible usar uno de los parámetros `\Message` o `\MsgArray` al mismo tiempo.

El máximo espacio es de 5 líneas con 50 caracteres cada una.

[\Wrap]

Tipo de dato: `switch`

Si se selecciona, todas las cadenas especificadas en el argumento `\MsgArray` se concatenan para formar una cadena con un solo espacio entre las distintas cadenas individuales y distribuida en el número mínimo posible de líneas.

De forma predeterminada, cada una de las cadenas del argumento `\MsgArray` aparece en una línea separada de la pantalla.

[\Icon]

Tipo de dato: `icondata`

Define el icono a mostrar. Sólo puede usarse uno de los iconos predefinidos de tipo `icondata`. Consulte [Datos predefinidos en la página 1683](#).

De forma predeterminada, no se usa ningún icono.

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.328 WaitDI - Espera hasta que se activa una señal digital de entrada

RobotWare - OS

Continuación

[\Image]

Tipo de dato: `string`

El nombre de la imagen que debe utilizarse. Para iniciar sus propias imágenes, las imágenes deben estar situadas en el directorio *HOME*: del sistema activo o directamente en el sistema activo.

La recomendación es situar los archivos en el directorio *HOME*: de forma que se incluyan en las operaciones de copia de seguridad y restauración.

Se requiere un reinicio y, a continuación, el FlexPendant carga las imágenes.

Una exigencia del sistema es que se use la opción de RobotWare *FlexPendant Interface*.

La imagen a mostrar puede tener 185 píxeles de anchura y 300 píxeles de altura. Si la imagen tiene un tamaño mayor, solo se muestran 185 \* 300 píxeles de la imagen a partir de su parte superior izquierda.

No es posible especificar ningún valor exacto en cuanto al tamaño que una imagen puede tener o el número de imágenes que es posible cargar en el FlexPendant. Depende del tamaño de los demás archivos cargados en el FlexPendant. La ejecución de los programas continúa simplemente si se usa una imagen que no está cargada en el FlexPendant.

[\VisualizeTime]

Tipo de dato: `num`

El tiempo de espera antes del cuadro de mensaje debe aparecer en FlexPendant. Si se utilizan los argumentos `\VisualizeTime` y `\MaxTime`, el tiempo utilizado en el argumento `\MaxTime` tiene que ser mayor que el tiempo utilizado en el argumento `\VisualizeTime`.

El tiempo predeterminado para la visualización si no se utiliza el argumento `\VisualizeTime` es 5 s. Valor mínimo 1 s. No hay límite de valor máximo. Resolución 0.001 s.

[\UIActiveSignal]

Tipo de dato: `signaldo`

La señal digital de salida utilizada en el argumento opcional `UIActiveSignal` se establece en 1 cuando se activa el cuadro de mensaje de visualización en FlexPendant. Cuando se ha retirado el cuadro de mensaje (cuando se cumple la condición), la señal se vuelve a establecer en 0.

No existe ninguna supervisión de parada o reinicio. La señal se establece en 0 cuando la instrucción está preparada o cuando se mueve el PP.

---

### Ejecución de programas

Si el valor de la señal es correcto cuando se ejecuta la instrucción, el programa sencillamente continúa con la instrucción siguiente.

Si el valor de la señal no es correcto, el robot entra en un estado de espera. Cuando la señal cambia al valor correcto, el programa continúa. El cambio se detecta mediante una interrupción, lo cual genera una respuesta rápida (no sondeada).

*Continúa en la página siguiente*



### 1.328 WaitDI - Espera hasta que se activa una señal digital de entrada RobotWare - OS Continuación

Cuando el robot está en espera, el tiempo se supervisa. De forma predeterminada, el robot puede esperar para siempre, pero el tiempo de espera máximo puede especificarse con el argumento opcional `\MaxTime`. Si se sobrepasa este tiempo máximo, se genera un error.

Si la ejecución del programa se detiene y se reanuda a continuación, la instrucción evalúa el valor actual de la señal. Se rechaza cualquier cambio durante el paro del programa.

En el modo manual, después de esperar más de 3 s, aparecerá una ventana de alerta que pregunta si se debe simular la instrucción. Puede configurarse que la alerta no aparezca, estableciendo el parámetro del sistema `SimulateMenu` en NO, consulte *Manual de referencia técnica - Parámetros del sistema*, tema *Controller*, tipo *General RAPID*.

Si se utiliza el modificador `\Visualize`, aparece un cuadro de mensaje en FlexPendant acorde con los argumentos programados. Si no se utiliza el argumento `\Header`, aparece un texto de cabecera predeterminado. Cuando la ejecución de la instrucción `WaitDI` está preparada, el cuadro de mensaje se elimina de FlexPendant.

El nuevo cuadro de mensaje del nivel de rutina TRAP toma el foco del cuadro de mensaje del nivel básico.

#### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                          | Causa del error   |
|---------------------------------|---|
| <code>ERR_NO_ALIASIO_DEF</code> | La variable de señal es una variable declarada en RAPID. No se ha conectado a una señal E/S definida en la configuración de E/S con la instrucción <code>AliasIO</code> .                             |
| <code>ERR_NORUNUNIT</code>      | Se ha perdido el contacto con el dispositivo de E/S.  |
| <code>ERR_SIG_NOT_VALID</code>  | La señal de E/S no está disponible. Los motivos pueden ser que el dispositivo de E/S no está en funcionamiento o que exista un error en la configuración (válido solamente para el bus de campo ICI). |
| <code>ERR_WAIT_MAXTIME</code>   | Existe un tiempo límite (parámetro <code>\MaxTime</code> ) antes de que la señal cambie al valor correcto.  |

#### Sintaxis

```
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> ]
```

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.328 WaitDI - Espera hasta que se activa una señal digital de entrada

RobotWare - OS

Continuación

```
['\' VisualizeTime ' :=' <expression (IN) of num>]  
['\' UIActiveSignal ' :=' <variable (VAR) of signaldo>] ';' 
```

---

### Información relacionada

| Para obtener más información sobre       | Consulte  |
|--|---|
| Espera hasta que se cumple una condición | <a href="#">WaitUntil - Esperar hasta que se cumple una condición en la página 1084</a> |
| Espera durante un periodo determinado    | <a href="#">WaitTime - Esperar una cantidad de tiempo determinada en la página 1082</a> |

---

## 1.329 WaitDO - Espera hasta que se activa una señal digital de salida

---

### Utilización

WaitDO (*Wait Digital Output*) se usa para esperar hasta que se activa una salida digital.

---

### Ejemplos básicos

Los ejemplos siguientes ilustran la instrucción WaitDO.

#### Ejemplo 1

```
WaitDO do4, 1;
```

La ejecución del programa sólo continúa después de que se ha activado la salida do4.

#### Ejemplo 2

```
WaitDO grip_status, 0;
```

La ejecución del programa sólo continúa después de que se ha desactivado la salida grip\_status.

#### Ejemplo 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;
..
```

Si no se cumple la condición, entonces la cabecera y el mensaje especificado en los argumentos opcionales \Header y \MsgArray se escribirán en la pantalla del FlexPendant junto con la condición que no se cumple.

---

### Argumentos

```
WaitDO Signal Value [\MaxTime] [\TimeFlag] [\Visualize] [\Header]
      [\Message] | [\MsgArray] [\Wrap] [\Icon] [\Image]
      [\VisualizeTime] [\UIActiveSignal]
```

Signal

**Tipo de dato:** signaldo

**El nombre de la señal.**

Value

**Tipo de dato:** dionum

**El valor deseado para la señal.**

[\MaxTime]

**Maximum Time**

**Tipo de dato:** num

El periodo máximo permitido para el tiempo de espera, expresado en segundos. Si el tiempo se agota antes de que se cumpla la condición y no se utiliza el argumento TimeFlag, es posible llamar al gestor de errores con el código de error ERR\_WAIT\_MAXTIME. Si no hay ningún gestor de errores, se detiene la ejecución.

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.329 WaitDO - Espera hasta que se activa una señal digital de salida

RobotWare - OS

Continuación

[\TimeFlag]

*Timeout Flag*

Tipo de dato: `bool`

El parámetro de salida que contiene el valor TRUE si se agota el tiempo máximo de espera permitido antes de que se cumpla la condición. Si se incluye este parámetro en la instrucción, no se considera un error si llega a agotarse el tiempo límite. Este argumento no se tiene en cuenta si el argumento `MaxTime` no se incluye en la instrucción.

[\Visualize]

Tipo de dato: `switch`

Si está seleccionado, se activa la visualización. La visualización consta de un cuadro de mensaje con la condición que no se cumple, icono, título, líneas de mensaje e imagen, de acuerdo con los argumentos del programa.

[\Header]

Tipo de dato: `string`

El texto de cabecera se escribirá en la parte superior del cuadro de mensaje. 40 caracteres como máximo. Si no se utilizara ningún argumento `\Header`, se mostrará un mensaje predeterminado.

[\Message]

Tipo de dato: `string`

Se escribirá una línea de texto en la pantalla. Máximo 50 caracteres.

[\MsgArray]

*(Message Array)*

Tipo de dato: `string`

Varias líneas de texto de una matriz para escribir en la pantalla. Solo es posible usar uno de los parámetros `\Message` o `\MsgArray` al mismo tiempo.

El máximo espacio es de 5 líneas con 50 caracteres cada una.

[\Wrap]

Tipo de dato: `switch`

Si se selecciona, todas las cadenas especificadas en el argumento `\MsgArray` se concatenan para formar una cadena con un solo espacio entre las distintas cadenas individuales y distribuida en el número mínimo posible de líneas.

De forma predeterminada, cada una de las cadenas del argumento `\MsgArray` aparece en una línea separada de la pantalla.

[\Icon]

Tipo de dato: `icondata`

Define el icono a mostrar. Sólo puede usarse uno de los iconos predefinidos de tipo `icondata`. Consulte [Datos predefinidos en la página 1683](#).

De forma predeterminada, no se usa ningún icono.

*Continúa en la página siguiente*

## 1.329 WaitDO - Espera hasta que se activa una señal digital de salida *RobotWare - OS* *Continuación*

[\Image]

Tipo de dato: `string`

El nombre de la imagen que debe utilizarse. Para iniciar sus propias imágenes, las imágenes deben estar situadas en el directorio *HOME*: del sistema activo o directamente en el sistema activo.

La recomendación es situar los archivos en el directorio *HOME*: de forma que se incluyan en las operaciones de copia de seguridad y restauración.

Se requiere un reinicio y, a continuación, el FlexPendant carga las imágenes.

Una exigencia del sistema es que se use la opción de RobotWare *FlexPendant Interface*.

La imagen a mostrar puede tener 185 píxeles de anchura y 300 píxeles de altura. Si la imagen tiene un tamaño mayor, solo se muestran 185 \* 300 píxeles de la imagen a partir de su parte superior izquierda.

No es posible especificar ningún valor exacto en cuanto al tamaño que una imagen puede tener o el número de imágenes que es posible cargar en el FlexPendant. Depende del tamaño de los demás archivos cargados en el FlexPendant. La ejecución de los programas continúa simplemente si se usa una imagen que no está cargada en el FlexPendant.

[\VisualizeTime]

Tipo de dato: `num`

El tiempo de espera antes del cuadro de mensaje debe aparecer en FlexPendant. Si se utilizan los argumentos `\VisualizeTime` y `\MaxTime`, el tiempo utilizado en el argumento `\MaxTime` tiene que ser mayor que el tiempo utilizado en el argumento `\VisualizeTime`.

El tiempo predeterminado para la visualización si no se utiliza el argumento `\VisualizeTime` es 5 s. Valor mínimo 1 s. No hay límite de valor máximo. Resolución 0.001 s.

[\UIActiveSignal]

Tipo de dato: `signaldo`

La señal digital de salida utilizada en el argumento opcional `UIActiveSignal` se establece en 1 cuando se activa el cuadro de mensaje de visualización en FlexPendant. Cuando se ha retirado el cuadro de mensaje (cuando se cumple la condición), la señal se vuelve a establecer en 0.

No existe ninguna supervisión de parada o reinicio. La señal se establece en 0 cuando la instrucción está preparada o cuando se mueve el PP.

---

### Ejecución de programas

Si el valor de la señal de salida es correcto cuando se ejecuta la instrucción, el programa sencillamente continúa con la instrucción siguiente.

Si el valor de la señal de salida no es correcto, el robot pasa a un estado de espera. Cuando la señal cambia al valor correcto, el programa continúa. El cambio se detecta mediante una interrupción, lo cual genera una respuesta rápida (no sondeada).

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.329 WaitDO - Espera hasta que se activa una señal digital de salida

RobotWare - OS

Continuación

Cuando el robot está en espera, el tiempo se supervisa. De forma predeterminada, el robot puede esperar para siempre, pero el tiempo de espera máximo puede especificarse con el argumento opcional `\MaxTime`. Si se sobrepasa este tiempo máximo, se genera un error.

Si la ejecución del programa se detiene y se reanuda a continuación, la instrucción evalúa el valor actual de la señal. Se rechaza cualquier cambio durante el paro del programa.

En el modo manual, después de esperar más de 3 s, aparecerá una ventana de alerta que pregunta si se debe simular la instrucción. Puede configurarse que la alerta no aparezca, estableciendo el parámetro del sistema `SimulateMenu` en NO, consulte *Manual de referencia técnica - Parámetros del sistema*, tema *Controller*, tipo *General RAPID*.

Si se utiliza el modificador `\Visualize`, aparece un cuadro de mensaje en FlexPendant acorde con los argumentos programados. Si no se utiliza el argumento `\Header`, aparece un texto de cabecera predeterminado. Cuando la ejecución de la instrucción `WaitDO` está preparada, el cuadro de mensaje se elimina de FlexPendant.

El nuevo cuadro de mensaje del nivel de rutina TRAP toma el foco del cuadro de mensaje del nivel básico.

### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                          | Causa del error   |
|---------------------------------|---|
| <code>ERR_NO_ALIASIO_DEF</code> | La variable de señal es una variable declarada en RAPID. No se ha conectado a una señal E/S definida en la configuración de E/S con la instrucción <code>AliasIO</code> .                             |
| <code>ERR_NORUNUNIT</code>      | Se ha perdido el contacto con el dispositivo de E/S.  |
| <code>ERR_SIG_NOT_VALID</code>  | La señal de E/S no está disponible. Los motivos pueden ser que el dispositivo de E/S no está en funcionamiento o que exista un error en la configuración (válido solamente para el bus de campo ICI). |
| <code>ERR_WAIT_MAXTIME</code>   | Existe un tiempo límite (parámetro <code>\MaxTime</code> ) antes de que la señal cambie al valor correcto.  |

### Sintaxis

```
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> ]
```

Continúa en la página siguiente

## 1.329 WaitDO - Espera hasta que se activa una señal digital de salida

RobotWare - OS

Continuación

```
['\ ' VisualizeTime ' := ' <expression ( IN ) of num > ]  
['\ ' UIActiveSignal ' := ' <variable ( VAR ) of signaldo > ] ';' 
```

### Información relacionada

| Para obtener más información sobre                 | Consulte   |
|--|--|
| Espera hasta que se cumple una condición           | <a href="#">WaitUntil - Esperar hasta que se cumple una condición en la página 1084</a>            |
| Espera durante un periodo determinado              | <a href="#">WaitTime - Esperar una cantidad de tiempo determinada en la página 1082</a>            |
| Espera hasta que se activa o desactiva una entrada | <a href="#">WaitDI - Espera hasta que se activa una señal digital de entrada en la página 1042</a> |

# 1 Instrucciones

---

1.330 WaitGI - Espera hasta que se activa un grupo de entradas digitales  
RobotWare - OS

## 1.330 WaitGI - Espera hasta que se activa un grupo de entradas digitales

---

### Utilización

WaitGI (*Wait Group digital Input*) se utiliza para esperar hasta que un grupo de señales digitales de entrada es cambiado a los valores especificados.

---

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción WaitGI:  
Consulte también [Más ejemplos en la página 1056](#).

#### Ejemplo 1

```
WaitGI gi4, 5;
```

La ejecución del programa sólo continúa después de que la entrada gi4 tiene el valor 5.

#### Ejemplo 2

```
WaitGI grip_status, 0;
```

La ejecución del programa sólo continúa después de que se ha restablecido la entrada grip\_status.

---

### Argumentos

```
WaitGI Signal [\NOTEQ] | [\LT] | [\GT] Value | Dvalue [\MaxTime]  
[\ValueAtTimeout] | [\DvalueAtTimeout] [\Visualize] [\Header]  
[\Message] | [\MsgArray] [\Wrap] [\Icon] [\Image]  
[\VisualizeTime] [\UIActiveSignal]
```

Signal

Tipo de dato: signalgi

El nombre de la señal de grupo de entradas digitales.

[\NOTEQ]

**NOT Equal**

Tipo de dato: switch

Si se usa este parámetro, la instrucción WaitGI espera hasta que el valor de la señal digital de grupo divide al valor de Value.

[\LT]

**Less Than**

Tipo de dato: switch

Si se usa este parámetro, la instrucción WaitGI espera hasta que el valor de la señal digital de grupo es inferior que el valor de Value.

[\GT]

**Greater Than**

Tipo de dato: switch

Si se usa este parámetro, la instrucción WaitGI espera hasta que el valor de la señal digital de grupo es mayor que el valor de Value.

---

*Continúa en la página siguiente*



## 1.330 WaitGI - Espera hasta que se activa un grupo de entradas digitales

RobotWare - OS

Continuación

Value

Tipo de dato: num

El valor deseado para la señal. Debe ser un valor entero dentro del rango de trabajo de la señal digital de grupo de entradas que se utiliza. El valor permitido depende del número de señales del grupo. El valor máximo que puede usarse en el argumento Value es de 8388608, que es el valor que una señal digital de grupo de 23 bits puede tener como valor máximo.

Dvalue

Tipo de dato: dnum

El valor deseado para la señal. Debe ser un valor entero dentro del rango de trabajo de la señal digital de grupo de entradas que se utiliza. El valor permitido depende del número de señales del grupo. El valor máximo de los bits de señales que puede tener una señal digital de grupo es de 32. Con una variable dnum es posible cubrir los valores del 0 al 4294967295, que constituyen el rango que puede tener una señal digital de 32 bits.

[ \MaxTime ]

*Maximum Time*

Tipo de dato: num

El periodo máximo permitido para el tiempo de espera, expresado en segundos. Si el tiempo se agota antes de que se cumpla la condición, se llama al gestor de errores (si lo hay) con el código de error ERR\_WAIT\_MAXTIME. Si no hay ningún gestor de errores, se detiene la ejecución.

[ \ValueAtTimeout ]

Tipo de dato: num

Si la instrucción alcanza el tiempo límite, el valor actual de la señal se almacena en esta variable. La variable sólo se define si la variable de sistema ERRNO recibe el valor ERR\_WAIT\_MAXTIME. Si se utiliza el argumento Dvalue, use el argumento DvalueAtTimeout para almacenar el valor actual en la señal (motivo: limitación del valor entero máximo para num).

Los valores de señal del 0 al 8.388.608 se almacenan siempre como un entero exacto.

[ \DvalueAtTimeout ]

Tipo de dato: dnum

Si la instrucción alcanza el tiempo límite, el valor actual de la señal se almacena en esta variable. La variable sólo se define si la variable de sistema ERRNO recibe el valor ERR\_WAIT\_MAXTIME.

Los valores de señal del 0 al 4.294.967.295 se almacenan siempre como un entero exacto.

[ \Visualize ]

Tipo de dato: switch

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.330 WaitGI - Espera hasta que se activa un grupo de entradas digitales

RobotWare - OS

Continuación

Si está seleccionado, se activa la visualización. La visualización consta de un cuadro de mensaje con la condición que no se cumple, icono, título, líneas de mensaje e imagen, de acuerdo con los argumentos del programa.

[ \Header ]

Tipo de dato: `string`

El texto de cabecera se escribirá en la parte superior del cuadro de mensaje. 40 caracteres como máximo. Si no se utilizara ningún argumento `\Header`, se mostrará un mensaje predeterminado.

[ \Message ]

Tipo de dato: `string`

Se escribirá una línea de texto en la pantalla. Máximo 50 caracteres.

[ \MsgArray ]

*(Message Array)*

Tipo de dato: `string`

Varias líneas de texto de una matriz para escribir en la pantalla. Solo es posible usar uno de los parámetros `\Message` o `\MsgArray` al mismo tiempo.

El máximo espacio es de 5 líneas con 50 caracteres cada una.

[ \Wrap ]

Tipo de dato: `switch`

Si se selecciona, todas las cadenas especificadas en el argumento `\MsgArray` se concatenan para formar una cadena con un solo espacio entre las distintas cadenas individuales y distribuida en el número mínimo posible de líneas.

De forma predeterminada, cada una de las cadenas del argumento `\MsgArray` aparece en una línea separada de la pantalla.

[ \Icon ]

Tipo de dato: `icondata`

Define el icono a mostrar. Sólo puede usarse uno de los iconos predefinidos de tipo `icondata`. Consulte [Datos predefinidos en la página 1683](#).

De forma predeterminada, no se usa ningún icono.

[ \Image ]

Tipo de dato: `string`

El nombre de la imagen que debe utilizarse. Para iniciar sus propias imágenes, las imágenes deben estar situadas en el directorio *HOME*: del sistema activo o directamente en el sistema activo.

La recomendación es situar los archivos en el directorio *HOME*: de forma que se incluyan en las operaciones de copia de seguridad y restauración.

Se requiere un reinicio y, a continuación, el FlexPendant carga las imágenes.

Una exigencia del sistema es que se use la opción de RobotWare *FlexPendant Interface*.

*Continúa en la página siguiente*

## 1.330 WaitGI - Espera hasta que se activa un grupo de entradas digitales *RobotWare - OS* *Continuación*

La imagen a mostrar puede tener 185 píxeles de anchura y 300 píxeles de altura. Si la imagen tiene un tamaño mayor, solo se muestran 185 \* 300 píxeles de la imagen a partir de su parte superior izquierda.

No es posible especificar ningún valor exacto en cuanto al tamaño que una imagen puede tener o el número de imágenes que es posible cargar en el FlexPendant. Depende del tamaño de los demás archivos cargados en el FlexPendant. La ejecución de los programas continúa simplemente si se usa una imagen que no está cargada en el FlexPendant.

[`\VisualizeTime`]

Tipo de dato: num

El tiempo de espera antes del cuadro de mensaje debe aparecer en FlexPendant. Si se utilizan los argumentos `\VisualizeTime` y `\MaxTime`, el tiempo utilizado en el argumento `\MaxTime` tiene que ser mayor que el tiempo utilizado en el argumento `\VisualizeTime`.

El tiempo predeterminado para la visualización si no se utiliza el argumento `\VisualizeTime` es 5 s. Valor mínimo 1 s. No hay límite de valor máximo. Resolución 0.001 s.

[`\UIActiveSignal`]

Tipo de dato: signaldo

La señal digital de salida utilizada en el argumento opcional `UIActiveSignal` se establece en 1 cuando se activa el cuadro de mensaje de visualización en FlexPendant. Cuando se ha retirado el cuadro de mensaje (cuando se cumple la condición), la señal se vuelve a establecer en 0.

No existe ninguna supervisión de parada o reinicio. La señal se establece en 0 cuando la instrucción está preparada o cuando se mueve el PP.

---

### Ejecución de programas

Si el valor de la señal es correcto cuando se ejecuta la instrucción, el programa sencillamente continúa con la instrucción siguiente.

Si el valor de la señal no es correcto, el robot entra en un estado de espera y el programa continúa tan pronto como la señal cambie al valor correcto. El cambio se detecta mediante una interrupción, lo cual genera una respuesta rápida (no sondeada).

Cuando el robot está en espera, el tiempo se supervisa. De forma predeterminada, el robot puede esperar para siempre, pero el tiempo de espera máximo puede especificarse con el argumento opcional `\MaxTime`. Si se sobrepasa este tiempo máximo, se genera un error.

Si la ejecución del programa se detiene y se reanuda a continuación, la instrucción evalúa el valor actual de la señal. Se rechaza cualquier cambio durante el paro del programa.

En el modo manual, después de esperar más de 3 s, aparecerá una ventana de alerta que pregunta si se debe simular la instrucción. Puede configurarse que la alerta no aparezca, estableciendo el parámetro del sistema `SimulateMenu` en NO,

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.330 WaitGI - Espera hasta que se activa un grupo de entradas digitales

RobotWare - OS

Continuación

consulte *Manual de referencia técnica - Parámetros del sistema*, tema *Controller*, tipo *General RAPID*.

Si se utiliza el modificador `\Visualize`, aparece un cuadro de mensaje en FlexPendant acorde con los argumentos programados. Si no se utiliza el argumento `\Header`, aparece un texto de cabecera predeterminado. Cuando la ejecución de la instrucción `WaitGI` está preparada, el cuadro de mensaje se elimina de FlexPendant.

El nuevo cuadro de mensaje del nivel de rutina TRAP toma el foco del cuadro de mensaje del nivel básico.

---

### Más ejemplos

A continuación aparecen más ejemplos de la instrucción `WaitGI`.

#### Ejemplo 1

```
WaitGI gil,\NOTEQ,0;
```

La ejecución del programa sólo continúa después de que la entrada `gil` es distinta del valor 0.

#### Ejemplo 2

```
WaitGI gil,\LT,1;
```

La ejecución del programa sólo continúa una vez que `gil` es menor que 1.

#### Ejemplo 3

```
WaitGI gil,\GT,0;
```

La ejecución del programa sólo continúa una vez que `gil` es mayor que 0.

#### Ejemplo 4

```
VAR num myvalattimeout:=0;
WaitGI gil, 5 \MaxTime:=4 \ValueAtTimeout:=myvalattimeout;
ERROR
  IF ERRNO=ERR_WAIT_MAXTIME THEN
    TPWrite "Value of gil at timeout:" + ValToStr(myvalattimeout);
    TRYNEXT;
  ELSE
    ! No error recovery handling
  ENDIF
```

La ejecución del programa sólo continúa si `gil` es igual a 5 o al alcanzar el tiempo límite. En caso de que se alcance el tiempo límite, el valor de la señal `gil` en el momento del tiempo límite puede registrarse sin otra lectura de la señal.

#### Ejemplo 5

```
WaitGI gil, 4, \Visualize \Header:="Waiting for signal"
  \MsgArray:["Movement will not start until", "the condition
  below is TRUE"] \Icon:=iconError;
MoveL p40, v500, z20, L10tip;
..
```

Si no se cumple la condición, entonces la cabecera y el mensaje especificado en los argumentos opcionales `\Header` y `\MsgArray` se escribirán en la pantalla del FlexPendant junto con la condición que no se cumple.

*Continúa en la página siguiente*

## Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                          | Causa del error   |
|---------------------------------|---|
| <code>ERR_GO_LIM</code>         | El argumento programado <code>Value</code> o <code>Dvalue</code> para la señal digital de entrada de grupo especificada <code>Signal</code> está fuera de límite.                                     |
| <code>ERR_NO_ALIASIO_DEF</code> | La variable de señal es una variable declarada en <code>RAPID</code> . No se ha conectado a una señal E/S definida en la configuración de E/S con la instrucción <code>AliasIO</code> .               |
| <code>ERR_NORUNUNIT</code>      | Se ha perdido el contacto con el dispositivo de E/S.  |
| <code>ERR_SIG_NOT_VALID</code>  | La señal de E/S no está disponible. Los motivos pueden ser que el dispositivo de E/S no está en funcionamiento o que exista un error en la configuración (válido solamente para el bus de campo ICI). |
| <code>ERR_WAIT_MAXTIME</code>   | Existe un tiempo límite (parámetro <code>\MaxTime</code> ) antes de que la señal cambie al valor correcto.  |

## Sintaxis

```

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> ]
[ '\ Image ':' <expression (IN) of string> ]
[ '\ VisualizeTime ':' <expression (IN) of num> ]
[ '\ UIActiveSignal ':' <variable (VAR) of signaldo> ] ';'

```

## Información relacionada

| Para obtener más información sobre                                 | Consulte  |
|--|---|
| Espera hasta que se cumple una condición                           | <a href="#">WaitUntil - Esperar hasta que se cumple una condición en la página 1084</a>             |
| Espera durante un periodo determinado                              | <a href="#">WaitTime - Esperar una cantidad de tiempo determinada en la página 1082</a>             |
| Espera hasta que un grupo de salidas digitales se activa/desactiva | <a href="#">WaitGO - Espera hasta que se activa un grupo de salidas digitales en la página 1058</a> |

## 1 Instrucciones

---

1.331 WaitGO - Espera hasta que se activa un grupo de salidas digitales  
RobotWare - OS

### 1.331 WaitGO - Espera hasta que se activa un grupo de salidas digitales

---

#### Utilización

WaitGO (*Wait Group digital Output*) se utiliza para esperar hasta que un grupo de señales digitales de salida cambia a un valor especificado.

---

#### Ejemplos básicos

Los ejemplos siguientes ilustran la instrucción WaitGO.

Consulte también [Más ejemplos en la página 1062](#).

#### Ejemplo 1

```
WaitGO go4, 5;
```

La ejecución del programa sólo continúa después de que la salida go4 tiene el valor 5.

#### Ejemplo 2

```
WaitGO grip_status, 0;
```

La ejecución del programa sólo continúa después de que la salida grip\_status se restablece.

---

#### Argumentos

```
WaitGO Signal [\NOTEQ] | [\LT] | [\GT] Value | Dvalue [\MaxTime]  
[\ValueAtTimeout] | [\DvalueAtTimeout] [\Visualize] [\Header]  
[\Message] | [\MsgArray] [\Wrap] [\Icon] [\Image]  
[\VisualizeTime] [\UIActiveSignal]
```

Signal

Tipo de dato: signalgo

El nombre de la señal de grupo de salidas digitales.

[\NOTEQ]

**NOT Equal**

Tipo de dato: switch

Si se usa este parámetro, la instrucción WaitGO espera hasta que el valor de la señal digital de grupo divide al valor de Value.

[\LT]

**Less Than**

Tipo de dato: switch

Si se usa este parámetro, la instrucción WaitGO espera hasta que el valor de la señal digital de grupo es inferior que el valor de Value.

[\GT]

**Greater Than**

Tipo de dato: switch

Si se usa este parámetro, la instrucción WaitGO espera hasta que el valor de la señal digital de grupo es mayor que el valor de Value.

*Continúa en la página siguiente*

Value

Tipo de dato: num

El valor deseado para la señal. Debe ser un valor entero dentro del rango de trabajo de la señal digital de grupo de salidas que se utiliza. El valor permitido depende del número de señales del grupo. El valor máximo que puede usarse en el argumento Value es de 8388608, que es el valor que una señal digital de grupo de 23 bits puede tener como valor máximo.

Dvalue

Tipo de dato: dnum

El valor deseado para la señal. Debe ser un valor entero dentro del rango de trabajo de la señal digital de grupo de salidas que se utiliza. El valor permitido depende del número de señales del grupo. El valor máximo de los bits de señales que puede tener una señal digital de grupo es de 32. Con una variable dnum es posible cubrir los valores del 0 al 4294967295, que constituyen el rango que puede tener una señal digital de 32 bits.

[ \MaxTime ]

*Maximum Time*

Tipo de dato: num

El periodo máximo permitido para el tiempo de espera, expresado en segundos. Si el tiempo se agota antes de que se cumpla la condición, se llama al gestor de errores si lo hay, con el código de error ERR\_WAIT\_MAXTIME. Si no hay ningún gestor de errores, se detiene la ejecución.

[ \ValueAtTimeout ]

Tipo de dato: num

Si la instrucción alcanza el tiempo límite, el valor actual de la señal se almacena en esta variable. La variable sólo se define si la variable de sistema ERRNO recibe el valor ERR\_WAIT\_MAXTIME. Si se utiliza el argumento Dvalue, use el argumento DvalueAtTimeout para almacenar el valor actual en la señal (motivo: limitación del valor entero máximo para num).

Los valores de señal del 0 al 8.388.608 se almacenan siempre como un entero exacto.

[ \DvalueAtTimeout ]

Tipo de dato: dnum

Si la instrucción alcanza el tiempo límite, el valor actual de la señal se almacena en esta variable. La variable sólo se define si la variable de sistema ERRNO recibe el valor ERR\_WAIT\_MAXTIME.

Los valores de señal del 0 al 4.294.967.295 se almacenan siempre como un entero exacto.

[ \Visualize ]

Tipo de dato: switch

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.331 WaitGO - Espera hasta que se activa un grupo de salidas digitales

RobotWare - OS

Continuación

Si está seleccionado, se activa la visualización. La visualización consta de un cuadro de mensaje con la condición que no se cumple, icono, título, líneas de mensaje e imagen, de acuerdo con los argumentos del programa.

[ \Header ]

Tipo de dato: `string`

El texto de cabecera se escribirá en la parte superior del cuadro de mensaje. 40 caracteres como máximo. Si no se utilizara ningún argumento `\Header`, se mostrará un mensaje predeterminado.

[ \Message ]

Tipo de dato: `string`

Se escribirá una línea de texto en la pantalla. Máximo 50 caracteres.

[ \MsgArray ]

*(Message Array)*

Tipo de dato: `string`

Varias líneas de texto de una matriz para escribir en la pantalla. Solo es posible usar uno de los parámetros `\Message` o `\MsgArray` al mismo tiempo.

El máximo espacio es de 5 líneas con 50 caracteres cada una.

[ \Wrap ]

Tipo de dato: `switch`

Si se selecciona, todas las cadenas especificadas en el argumento `\MsgArray` se concatenan para formar una cadena con un solo espacio entre las distintas cadenas individuales y distribuida en el número mínimo posible de líneas.

De forma predeterminada, cada una de las cadenas del argumento `\MsgArray` aparece en una línea separada de la pantalla.

[ \Icon ]

Tipo de dato: `icondata`

Define el icono a mostrar. Sólo puede usarse uno de los iconos predefinidos de tipo `icondata`. Consulte [Datos predefinidos en la página 1683](#).

De forma predeterminada, no se usa ningún icono.

[ \Image ]

Tipo de dato: `string`

El nombre de la imagen que debe utilizarse. Para iniciar sus propias imágenes, las imágenes deben estar situadas en el directorio `HOME:` del sistema activo o directamente en el sistema activo.

La recomendación es situar los archivos en el directorio `HOME:` de forma que se incluyan en las operaciones de copia de seguridad y restauración.

Se requiere un reinicio y, a continuación, el FlexPendant carga las imágenes.

Una exigencia del sistema es que se use la opción de RobotWare *FlexPendant Interface*.

*Continúa en la página siguiente*



## 1.331 WaitGO - Espera hasta que se activa un grupo de salidas digitales RobotWare - OS Continuación

La imagen a mostrar puede tener 185 píxeles de anchura y 300 píxeles de altura. Si la imagen tiene un tamaño mayor, solo se muestran 185 \* 300 píxeles de la imagen a partir de su parte superior izquierda.

No es posible especificar ningún valor exacto en cuanto al tamaño que una imagen puede tener o el número de imágenes que es posible cargar en el FlexPendant. Depende del tamaño de los demás archivos cargados en el FlexPendant. La ejecución de los programas continúa simplemente si se usa una imagen que no está cargada en el FlexPendant.

[`\VisualizeTime`]

Tipo de dato: num

El tiempo de espera antes del cuadro de mensaje debe aparecer en FlexPendant. Si se utilizan los argumentos `\VisualizeTime` y `\MaxTime`, el tiempo utilizado en el argumento `\MaxTime` tiene que ser mayor que el tiempo utilizado en el argumento `\VisualizeTime`.

El tiempo predeterminado para la visualización si no se utiliza el argumento `\VisualizeTime` es 5 s. Valor mínimo 1 s. No hay límite de valor máximo. Resolución 0.001 s.

[`\UIActiveSignal`]

Tipo de dato: signaldo

La señal digital de salida utilizada en el argumento opcional `UIActiveSignal` se establece en 1 cuando se activa el cuadro de mensaje de visualización en FlexPendant. Cuando se ha retirado el cuadro de mensaje (cuando se cumple la condición), la señal se vuelve a establecer en 0.

No existe ninguna supervisión de parada o reinicio. La señal se establece en 0 cuando la instrucción está preparada o cuando se mueve el PP.

---

### Ejecución de programas

Si el valor de la señal es correcto cuando se ejecuta la instrucción, el programa sencillamente continúa con la instrucción siguiente.

Si el valor de la señal no es correcto, el robot entra en un estado de espera y el programa continúa tan pronto como la señal cambie al valor correcto. El cambio se detecta mediante una interrupción, lo cual genera una respuesta rápida (no sondeada).

Cuando el robot está en espera, el tiempo se supervisa. De forma predeterminada, el robot puede esperar para siempre, pero el tiempo de espera máximo puede especificarse con el argumento opcional `\MaxTime`. Si se sobrepasa este tiempo máximo, se genera un error.

Si la ejecución del programa se detiene y se reanuda a continuación, la instrucción evalúa el valor actual de la señal. Se rechaza cualquier cambio durante el paro del programa.

En el modo manual, después de esperar más de 3 s, aparecerá una ventana de alerta que pregunta si se debe simular la instrucción. Puede configurarse que la alerta no aparezca, estableciendo el parámetro del sistema `SimulateMenu` en NO,

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.331 WaitGO - Espera hasta que se activa un grupo de salidas digitales

RobotWare - OS

Continuación

consulte *Manual de referencia técnica - Parámetros del sistema*, tema *Controller*, tipo *General RAPID*.

Si se utiliza el modificador `\Visualize`, aparece un cuadro de mensaje en FlexPendant acorde con los argumentos programados. Si no se utiliza el argumento `\Header`, aparece un texto de cabecera predeterminado. Cuando la ejecución de la instrucción `WaitGO` está preparada, el cuadro de mensaje se elimina de FlexPendant.

El nuevo cuadro de mensaje del nivel de rutina TRAP toma el foco del cuadro de mensaje del nivel básico.

---

### Más ejemplos

A continuación aparecen más ejemplos de la instrucción `WaitGO`.

#### Ejemplo 1

```
WaitGO go1,\NOTEQ,0;
```

La ejecución del programa sólo continúa después de que la entrada `go1` es distinta del valor 0.

#### Ejemplo 2

```
WaitGO go1,\LT,1;
```

La ejecución del programa sólo continúa una vez que `go1` es menor que 1.

#### Ejemplo 3

```
WaitGO go1,\GT,0;
```

La ejecución del programa sólo continúa una vez que `go1` es mayor que 0.

#### Ejemplo 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
```

La ejecución del programa sólo continúa si `go1` es igual a 5 o al alcanzar el tiempo límite. En caso de que se alcance el tiempo límite, el valor de la señal `go1` en el momento del tiempo límite puede registrarse sin otra lectura de la señal.

#### Ejemplo 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;
..
```

Si no se cumple la condición, entonces la cabecera y el mensaje especificado en los argumentos opcionales `\Header` y `\MsgArray` se escribirán en la pantalla del FlexPendant junto con la condición que no se cumple.

*Continúa en la página siguiente*

## Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                          | Causa del error   |
|---------------------------------|---|
| <code>ERR_GO_LIM</code>         | El argumento programado <code>Value</code> o <code>Dvalue</code> para la señal digital de salida de grupo especificada <code>Signal</code> está fuera de límite.                                      |
| <code>ERR_NO_ALIASIO_DEF</code> | La variable de señal es una variable declarada en <code>RAPID</code> . No se ha conectado a una señal E/S definida en la configuración de E/S con la instrucción <code>AliasIO</code> .               |
| <code>ERR_NORUNUNIT</code>      | Se ha perdido el contacto con el dispositivo de E/S.  |
| <code>ERR_SIG_NOT_VALID</code>  | La señal de E/S no está disponible. Los motivos pueden ser que el dispositivo de E/S no está en funcionamiento o que exista un error en la configuración (válido solamente para el bus de campo ICI). |
| <code>ERR_WAIT_MAXTIME</code>   | Existe un tiempo límite (parámetro <code>\MaxTime</code> ) antes de que la señal cambie al valor correcto.  |

## Sintaxis

```

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> ]
[ '\ Image ':' <expression (IN) of string> ]
[ '\ VisualizeTime ':' <expression (IN) of num> ]
[ '\ UIActiveSignal ':' <variable (VAR) of signaldo> ] ';'

```

## Información relacionada

| Para obtener más información sobre                                  | Consulte   |
|---|--|
| Espera hasta que se cumple una condición                            | <a href="#">WaitUntil - Esperar hasta que se cumple una condición en la página 1084</a>              |
| Espera durante un periodo determinado                               | <a href="#">WaitTime - Esperar una cantidad de tiempo determinada en la página 1082</a>              |
| Espera hasta que un grupo de entradas digitales se activa/desactiva | <a href="#">WaitGI - Espera hasta que se activa un grupo de entradas digitales en la página 1052</a> |

# 1 Instrucciones

---

## 1.332 WaitLoad - Conectar un módulo cargado a una tarea

RobotWare - OS

## 1.332 WaitLoad - Conectar un módulo cargado a una tarea

---

### Utilización

WaitLoad se utiliza para conectar con el módulo cargado con la instrucción StartLoad en la tarea de programa.

El módulo de programa cargado se añade a los módulos que ya existen en la memoria de programa.

Los módulos cargados con StartLoad deben ser conectados a la tarea de programa con la instrucción WaitLoad para poder utilizar cualquiera de sus símbolos o rutinas.

WaitLoad también puede descargar un módulo de programa si se utilizan los modificadores opcionales. De esta forma se reduce el número de vínculos (1 en lugar de 2).

WaitLoad también puede comprobar si hay referencias no resueltas si se utiliza el modificador opcional \CheckRef.

---

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción WaitLoad:

Consulte también [Más ejemplos en la página 1065](#).

#### Ejemplo 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";
```

Carga en la memoria de programas el módulo PART\_A.MOD existente en HOME:. En paralelo, mueve el robot. A continuación, conecta el nuevo módulo de programa a la tarea de programa y llama a la rutina routine\_x del módulo PART\_A.

---

### Argumentos

```
WaitLoad [\UnloadPath] [\UnloadFile] LoadNo [\CheckRef]
```

[\UnloadPath]

Tipo de dato: string

La ruta y el nombre del archivo que se descargará de la memoria de programa. El nombre de archivo se excluye cuando se utiliza el argumento \UnloadFile.

[\UnloadFile]

Tipo de dato: string

Cuando se excluye el nombre del archivo en el argumento \UnloadPath es necesario definirlo con este argumento.

*Continúa en la página siguiente*

LoadNo

Tipo de dato: loadsession

Ésta es una referencia a la sesión de carga, creada por la instrucción StartLoad que se necesita para conectar a la tarea de programa el módulo cargado.

[ \CheckRef ]

Tipo de dato: switch

Busca referencias no resueltas en la tarea de programa después de la carga del módulo. Si no se usa, no se realiza ninguna búsqueda de referencias no resueltas.

---

### Ejecución de programas

A continuación, la instrucción WaitLoad espera primero a que se complete la carga, si no se ha completado aún, y después el módulo será vinculado e inicializado. La inicialización del módulo cargado devuelve todas las variables del nivel de módulo a sus valores iniciales.

Las referencias no resueltas se aceptarán siempre para las operaciones de carga StartLoad - WaitLoad si no se utiliza el parámetro \CheckRef, pero será un error de tiempo de ejecución al ejecutar una referencia no resuelta.

El sistema se inicia con la operación de descarga, si se especifica. Si la descarga del módulo falla, no se carga ningún nuevo módulo.

Si se produce algún error en la operación de carga, incluidas las referencias no resueltas si se usa el modificador \CheckRef, el módulo cargado no estará ya disponible en la memoria de programas.

Para conseguir una estructura de programa idónea, fácil de comprender y mantener, todas las operaciones de carga y descarga de módulos de programa deben hacerse en el módulo principal ("main") que siempre está presente en la memoria de programa durante la ejecución.

Para cargar un programa que contiene un procedimiento principal desde un programa principal (que tiene su propio procedimiento main), consulte la instrucción Load.

---

### Más ejemplos

A continuación aparecen más ejemplos de la instrucción WaitLoad.

#### Ejemplo 1

```
StartLoad "HOME:/DOORDIR/DOOR2.MOD", load1;
...
WaitLoad \UnloadPath:="HOME:/DOORDIR/DOOR1.MOD", load1;
```

Carga en la memoria de programa el módulo de programa DOOR2.MOD desde el directorio DOORDIR de HOME: y conecta el nuevo módulo a la tarea. El módulo de programa DOOR1.MOD se descarga de la memoria de programa.

#### Ejemplo 2

```
StartLoad "HOME:" \File:="DOORDIR/DOOR2.MOD", load1;
! The robot can do some other work
WaitLoad \UnloadPath:="HOME:" \File:="DOORDIR/DOOR1.MOD", load1;
```

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.332 WaitLoad - Conectar un módulo cargado a una tarea

RobotWare - OS

Continuación

Equivale a las instrucciones siguientes, pero el robot puede hacer otras operaciones durante el tiempo de carga. Además, las hará más rápido (con un solo vínculo en lugar de los dos vínculos que aparecen a continuación).

```
Load "HOME:" \File:="DOORDIR/DOOR2.MOD";  
UnLoad "HOME:" \File:="DOORDIR/DOOR1.MOD";
```

### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre         | Causa del error   |
|----------------|---|
| ERR_FILNOTFND  | No se encuentra el archivo especificado en la instrucción <code>StartLoad</code> .  |
| ERR_IOERROR    | Algún tipo de problema para leer el archivo que se va a cargar.   |
| ERR_UNKPROC    | El argumento <code>LoadNo</code> hace referencia a una sesión de carga desconocida.   |
| ERR_PRGMEMFULL | Imposible cargar el módulo porque la memoria de programas está llena.   |
| ERR_LOADED     | El módulo de programa ya está cargado en la memoria de programas.   |
| ERR_SYNTAX     | El módulo cargado contiene errores de sintaxis.   |
| ERR_LINKREF    | El módulo cargado da lugar a errores de vínculo no recuperables. <ul style="list-style-type: none"><li>El módulo cargado da lugar a errores de vínculo no recuperables.</li><li><code>WaitLoad</code> se utiliza con el modificador <code>\CheckRef</code> para comprobar cualquier error de referencia y la memoria del programa contiene referencia sin resolver.</li></ul> |

Los errores siguientes sólo pueden producirse cuando el argumento `\UnloadPath` se usa en la instrucción `WaitLoad`:

| Nombre     | Causa del error  |
|------------|--|
| ERR_UNLOAD | <ul style="list-style-type: none"><li>No es posible descargar el módulo especificado en el argumento <code>\UnloadPath</code> porque se están ejecutando acciones dentro del módulo.</li><li>No es posible descargar el módulo especificado en el argumento <code>\UnloadPath</code> porque el módulo de programa no ha sido cargado con <code>Load</code> o <code>StartLoad-WaitLoad</code> desde el programa de RAPID.</li></ul> |

Si se produce alguno de estos errores, el módulo real será descargado y no estará disponible en el gestor `ERROR`.



#### Nota

**RETRY** no puede usarse para la recuperación de errores generados por `WaitLoad`.

Continúa en la página siguiente

**Limitaciones**

No es posible cambiar el valor actual de una variable PERS cargando el mismo módulo con un nuevo valor de inicialización para el valor real de la variable PERS.

Ejemplo:

- Se carga en el sistema el archivo `my_module.mod` con la declaración `PERS num my_pers:=1;`.
- El archivo `my_module.mod` se edita en el disco con el nuevo valor de variable persistente, por ejemplo `PERS num my_pers:=3;`
- Se ejecuta el código que aparece más abajo.
- Después de cargar de nuevo `my_module.mod`, el valor de `my_pers` sigue siendo 1 en lugar de 3.

```
StartLoad \Dynamic, "HOME:/my_module.mod", load1;
...
WaitLoad \UnloadPath:="HOME:/my_module.mod", load1;
```

Esta limitación es consecuencia de las características de las variables PERS. El valor actual de la variable PERS no es alterado por el valor de inicialización de la variable PERS cargada nuevamente si la variable PERS ya se está utilizando en el momento de la carga.

Los problemas descritos anteriormente no se producen si se ejecuta en su lugar el código siguiente:

```
Unload "HOME:/my_module.mod";
StartLoad \Dynamic, "HOME:/my_module.mod", load1;
...
WaitLoad load1;
```

Otra opción es utilizar una constante CONST para el valor de inicialización y hacer la asignación siguiente al comienzo de la ejecución del nuevo módulo: `my_pers := my_const;`

**Sintaxis****WaitLoad**

```
[ '\ ' UnloadPath ':=' <expression (IN) of string> ', ' ]
[ '\ ' UnloadFile ':=' <expression (IN) of string> ', ' ]
[ LoadNo ':=' ] <variable (VAR) of loadsession>
[ '\ ' CheckRef ] ';' ;'
```

**Información relacionada**

| Para obtener más información sobre                | Consulte  |
|---|---|
| Carga de módulos de programa durante la ejecución | <a href="#">StartLoad - Carga de programa durante la ejecución en la página 792</a>             |
| Sesión de carga                                   | <a href="#">loadsession - Sesión de carga de programa en la página 1702</a>                     |
| Carga de un módulo de programa                    | <a href="#">Load - Carga un módulo de programa durante la ejecución en la página 347</a>        |
| Descarga de un módulo de programa                 | <a href="#">Unload - Descargar un módulo de programa durante la ejecución en la página 1021</a> |

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.332 WaitLoad - Conectar un módulo cargado a una tarea

RobotWare - OS

Continuación

| Para obtener más información sobre                              | Consulte  |
|---|---|
| Cancelación de la carga de un módulo de programa                | <a href="#">CancelLoad - Cancela la carga de un módulo en la página 81</a>        |
| Comprobar referencias de programa                               | <a href="#">CheckProgRef - Comprobar referencias de programa en la página 131</a> |
| Llamada a procedimiento con enlazamiento en tiempo de ejecución | <i>Manual de referencia técnica - RAPID Overview</i>                              |



### 1.333 WaitRob - Esperar hasta un punto de paro o una velocidad cero

#### Utilización

WaitRob (*Wait Robot*) espera hasta que el robot y los ejes externos han llegado al punto de paro o tienen una velocidad cero.

#### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción WaitRob:  
Consulte también [Más ejemplos en la página 1069](#).

#### Ejemplo 1

```
WaitRob \InPos;
```

La ejecución del programa espera hasta que el robot y los ejes externos hayan llegado al punto de paro.

#### Argumentos

```
WaitRob [\InPos] | [\ZeroSpeed]
```

[\InPos]

##### *In Position*

Tipo de dato: switch

Si se utiliza este argumento, el robot y los ejes externos deben haber llegado al punto de paro (ToPoint de la instrucción de movimiento actual) para que pueda continuar la ejecución.

[\ZeroSpeed]

##### *Zero Speed*

Tipo de dato: switch

Si se utiliza este argumento, el robot y los ejes externos deben presentar una velocidad cero para que pueda continuar la ejecución.

Si no se introduce ninguno de los argumentos \InPos ni \ZeroSpeed, aparece un mensaje de error.

#### Más ejemplos

A continuación aparecen más ejemplos de cómo usar la instrucción WaitRob.

#### Ejemplo 1

```
PROC stop_event()  
  WaitRob \ZeroSpeed;  
  SetDO rob_moving, 0;  
ENDPROC
```

Este ejemplo muestra una rutina de evento que ejecuta un paro de programa. La señal digital de salida rob\_moving tiene el valor 1 mientras el robot se está moviendo y cambia a 0 cuando el robot y los ejes externos se han detenido tras un paro de programa.

*Continúa en la página siguiente*

# 1 Instrucciones

---

1.333 WaitRob - Esperar hasta un punto de paro o una velocidad cero

RobotWare - OS

Continuación

---

## Sintaxis

```
WaitRob  
[ '\ ' InPos ] | [ '\ ' ZeroSpeed ] ';' 
```

---

## Información relacionada

| Para obtener más información sobre     | Consulte   |
|--|--|
| Movimiento en general                  | <i>Manual de referencia técnica - RAPID Overview</i> , sección <i>Principios de movimiento y E/S</i>   |
| Otras instrucciones de posicionamiento | <i>Manual de referencia técnica - RAPID Overview</i> , sección <i>Resumen sobre RAPID - Movimiento</i> |
| Definición de datos de punto de paro   | <a href="#">stoppointdata - Datos de punto de paro en la página 1771</a>                               |

### 1.334 WaitSensor - Espera a la conexión de un sensor

#### Utilización

WaitSensor se conecta a un objeto de la ventana de inicio de una unidad mecánica con sensor.

#### Ejemplos básicos

A continuación aparecen algunos ejemplos básicos de la instrucción WaitSensor. Consulte también [Más ejemplos en la página 1072](#).

#### Ejemplo 1

```
WaitSensor Ssyncl;
```

El programa se conecta al primer objeto de la cola de objetos que se encuentre dentro de la ventana de inicio del sensor. Si no hay ningún objeto en la ventana de inicio, la ejecución se detiene y espera algún objeto.

#### Argumentos

```
WaitSensor MechUnit [\RelDist ][\PredTime][\MaxTime][\TimeFlag]
```

MechUnit

##### *Unidad mecánica*

Tipo de dato: mecunit

La unidad mecánica en movimiento con la que está relacionada la posición de robot de la instrucción.

[\RelDist]

##### *Distancia relativa*

Tipo de dato: num

Espera a que un objeto entre en la ventana de inicio y pase más allá de la distancia especificada por el argumento. Si el objeto ya está conectado, la ejecución se detiene hasta que el objeto pasa más allá de la distancia indicada. Si el objeto ya ha pasado más allá de la distancia relativa, la ejecución continúa.

[\PredTime]

##### *Tiempo de predicción*

Tipo de dato: num

Espera a que un objeto entre en la ventana de inicio y pase más allá de la distancia especificada por el argumento. Si el objeto ya está conectado, la ejecución se detiene hasta que el objeto pasa más allá de la distancia indicada. Si el objeto ya ha pasado más allá de la predicción de tiempo, la ejecución continúa.

[\MaxTime]

##### *Tiempo máximo*

Tipo de dato: num

El periodo máximo permitido para el tiempo de espera, expresado en segundos. Si este tiempo se agota antes de la conexión al sensor o que se alcance \RelDist,

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.334 WaitSensor - Espera a la conexión de un sensor

### Machine Synchronization

#### Continuación

se llama al gestor de errores (si lo hay) con el código de error `ERR_WAIT_MAXTIME`. Si no hay ningún gestor de errores, se detiene la ejecución.

`[\TimeFlag]`

*Indicador de tiempo límite*

Tipo de dato: `bool`

El parámetro de salida que contiene el valor `TRUE` si se agota el tiempo máximo de espera permitido antes de la conexión al sensor o que se alcance `\RelDist`. Si se incluye este parámetro en la instrucción, no se considera un error si llega a agotarse el tiempo límite.

Este argumento no se tiene en cuenta si el argumento `MaxTime` no se incluye en la instrucción.

---

### Ejecución de programas

Si no hay ningún objeto en la ventana de inicio, la ejecución del programa se detiene. Si hay un objeto de trabajo presente, éste se conecta al sensor y la ejecución prosigue.

Si se ejecuta una segunda instrucción `WaitSensor` mientras la conexión ya existe, se devuelve un error a no ser que se utilice el argumento opcional `\RelDist`.

---

### Más ejemplos

A continuación aparecen más ejemplos de la instrucción.

#### Ejemplo 1

```
WaitSensor Ssync1\RelDist:=500.0;
```

Si no hay conexión, se espera a que el objeto entre en la ventana de inicio y a continuación se espera a que el objeto sobrepase un punto situado a 500 mm en el sensor.

Si ya existe una conexión al objeto, se espera a que el objeto sobrepase los 500 mm.

#### Ejemplo 2

```
WaitSensor Ssync1\RelDist:=0.0;
```

Si no hay conexión, se espera la presencia de un objeto en la ventana de inicio.

Si ya existe una conexión, se continúa la ejecución dado que el objeto ya ha sobrepasado la posición de 0,0 mm.

#### Ejemplo 3

```
WaitSensor Ssync1;  
WaitSensor Ssync1\RelDist:=0.0;
```

La primera instrucción `WaitSensor` se conecta al objeto que se encuentra en la ventana de inicio. La segunda instrucción `WaitSensor` retorna inmediatamente si el objeto sigue conectado, pero esperará al siguiente objeto si el objeto anterior se ha movido más allá de la distancia máxima o ha sido soltado.

#### Ejemplo 4

```
WaitSensor Ssync1\RelDist:=0.5\PredTime:=0.1;
```

*Continúa en la página siguiente*

## 1.334 WaitSensor - Espera a la conexión de un sensor Machine Synchronization Continuación

La instrucción `WaitSensor` retorna inmediatamente si el objeto ha pasado más allá de los 0,5 metros pero, de lo contrario, espera a que un objeto alcance  $=RelDist - C1speed * PredTime$ . En este caso, el objetivo es adelantarse a los retardos antes de iniciar una nueva instrucción de movimiento.

### Ejemplo 5

```
WaitSensor Ssync1\RelDist:=0.5\MaxTime:=0.1\TimeFlag:=flag1;
```

La instrucción `WaitSensor` retorna inmediatamente si el objeto ha pasado más allá de los 0,5 metros pero, de lo contrario, espera a un objeto durante 0,1 s. Si ningún objeto sobrepasa la posición de 0,5 metros durante estos 0,1 s, la instrucción retornará con `flag1 = TRUE`.

### Limitaciones

Se requieren 50 ms para conectarse al primer objeto de la ventana de inicio. Después de la conexión, una segunda instrucción `WaitSensor` con el argumento opcional `\RelDist` sólo requiere el tiempo normal de ejecución de una instrucción de RAPID.

### Gestión de errores

Si se producen los errores siguientes durante la ejecución de la instrucción `WaitSensor`, la variable de sistema `ERRNO` cambia de valor. Estos errores pueden ser gestionados en el gestor de errores.

| Nombre                        | Causa del error   |
|-------------------------------|---|
| <code>ERR_CNV_NOT_ACT</code>  | El sensor no está activado.   |
| <code>ERR_CNV_CONNECT</code>  | La instrucción <code>WaitSensor</code> ya está conectada.   |
| <code>ERR_CNV_DROPPED</code>  | El objeto que estaba esperando la instrucción <code>WaitSensor</code> ha sido desechado por otra tarea. |
| <code>ERR_WAIT_MAXTIME</code> | El objeto no llegó a tiempo y no hay ningún <code>TimeFlag</code> .                                     |
| <code>ERR_CNV_OBJ_LOST</code> | El objeto ha pasado el <code>StartwindowWidth</code> sin conectarse.                                    |

### Sintaxis

```
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 > ] ;'
```

### Información relacionada

| Para obtener más información sobre   | Consulte   |
|--------------------------------------|--|
| Colocación de un objeto en el sensor | <a href="#">DropSensor - Colocación de un objeto en el sensor en la página 191</a> |
| Sincronización con un sensor         | <a href="#">SyncToSensor - Sincronización con un sensor en la página 866</a>       |
| Sincronización con un sensor         | <a href="#">SyncToSensor - Sincronización con un sensor en la página 866</a>       |

Continúa en la página siguiente

# 1 Instrucciones

---

1.334 WaitSensor - Espera a la conexión de un sensor

*Machine Synchronization*

*Continuación*

|   |  |
|---|--|
| <b>Para obtener más información sobre</b> | <b>Consulte</b>                                      |
| <i>Machine Synchronization</i>            | <i>Application manual - Controller software IRC5</i> |

## 1.335 WaitSyncTask - Esperar en un punto de sincronización con otras tareas de programa

### Utilización

WaitSyncTask se utiliza para sincronizar varias tareas de programa en un punto especial de cada uno de los programas. Cada tarea de programa espera hasta que todas las tareas de programa hayan alcanzado el punto de sincronización designado.



#### Nota

La instrucción WaitSyncTask sólo sincroniza la ejecución del programa. Para sincronizar tanto la ejecución del programa como los movimientos del robot, la instrucción de Move previa a WaitSyncTask debe ser un punto de paro en todas las tareas de programa implicadas. También es posible sincronizar tanto la ejecución del programa como los movimientos del robot mediante WaitSyncTask \Inpos ... en todas las tareas de programa implicadas.



#### ¡AVISO!

Para conseguir un funcionamiento seguro de la sincronización, el punto de reunión (parámetro SyncID) debe tener un nombre exclusivo en cada tarea de programa. Este nombre debe ser también el mismo en el caso de las tareas de programa que deben coincidir en el punto de reunión.

### Ejemplos básicos

Los ejemplos siguientes ilustran la instrucción WaitSyncTask.

Consulte también [Más ejemplos en la página 1077](#).

#### Ejemplo 1

##### Ejemplo de programa de la tarea T\_ROB1

```
PERS tasks task_list{2} := [ ["T_ROB1"], ["T_ROB2"] ];
VAR syncident sync1;

...
WaitSyncTask sync1, task_list;
...
```

#### Ejemplo 2

##### Ejemplo de programa de la tarea T\_ROB2

```
PERS tasks task_list{2} := [ ["T_ROB1"], ["T_ROB2"] ];
VAR syncident sync1;

...
WaitSyncTask sync1, task_list;
...
```

La tarea de programa que llegue en primer lugar a WaitSyncTask con la identidad sync1 espera hasta que la otra tarea de programa llegue a su WaitSyncTask con

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.335 WaitSyncTask - Esperar en un punto de sincronización con otras tareas de programa

### Multitasking

### Continuación

la misma identidad `sync1`. A continuación, las dos tareas de programa `T_ROB1` y `T_ROB2` prosiguen su ejecución.

---

### Argumentos

```
WaitSyncTask [\InPos] SyncID TaskList [\TimeOut]
```

`[\InPos]`

#### *In Position*

Tipo de dato: `switch`

Si se usa este argumento, los ejes del robot y los ejes externos deben estar completamente parados antes de que esta tarea de programa empiece a esperar a otras tareas de programa para que alcancen el punto de reunión especificado en la instrucción `WaitSyncTask`.

`SyncID`

#### *Synchronization identity*

Tipo de dato: `syncident`

Una variable que especifica el nombre del punto de sincronización (reunión). El tipo de dato `syncident` es de un tipo sin valor y sólo se utiliza como un identificador para asignar un nombre a la posición de sincronización.

La variable debe ser definida y tener un nombre igual en todas las tareas de programa cooperantes. Se recomienda definir siempre la variable global en cada tarea de programa (`VAR syncident ...`).

`TaskList`

Tipo de dato: `tasks`

Una variable persistente que especifica en una lista de tareas (matriz) el nombre (`string`) de las tareas de programa que deben coincidir en el punto de sincronización cuyo nombre se especifica en el argumento `SyncID`.

La variable persistente debe ser definida y tener un nombre igual y el mismo contenido en todas las tareas de programa cooperantes. Se recomienda definir siempre la variable global en el sistema (`PERS tasks ...`).

`[\TimeOut]`

Tipo de dato: `num`

El tiempo máximo que debe esperarse hasta que las demás tareas de programa lleguen hasta el punto de sincronización. Tiempo límite en segundos (resolución 0,001 s). Si este argumento no se especifica, la tarea de programa esperará de forma indefinida.

Si el tiempo se agota antes de que todas las tareas de programa alcancen el punto de sincronización, se llama al gestor de errores si lo hay, con el código de error `ERR_WAITSYNCTASK`. Si no hay ningún gestor de errores, se detiene la ejecución.

*Continúa en la página siguiente*



#### Ejecución de programas

La tarea de programa actual espera en `WaitSyncTask`, hasta que las otras tareas de programa indicadas en `TaskList` hayan alcanzado el mismo punto `SyncID`.

En ese momento, la tarea de programa correspondiente continuará ejecutando su siguiente instrucción.

Es posible programar `WaitSyncTask` entre instrucciones de movimiento con una zona de esquina intercalada. En función del equilibrio de temporización entre las tareas de programa en el momento de la ejecución, el sistema puede:

- Con la mejor temporización, mantener todas las zonas de esquina.
- Con la peor temporización, mantener sólo la zona de esquina de la tarea de programa que alcance la instrucción `WaitSyncTask` en último lugar. En el caso de las tareas de programa, esto da como resultado puntos de paro.

Es posible excluir las tareas de programa de las funciones de pruebas del panel de selección de tareas del `FlexPendant`.

Es posible utilizar los principios siguientes:

- Principio 1) Excluir permanentemente en el ciclo la tarea de programa del panel de selección de tareas antes del inicio desde `Main` (tras cambiar el `PP` a `Main`). Esta desconexión será válida durante todo el ciclo del programa.
- Principio 2) Excluir temporalmente la tarea de programa del panel de selección de tareas entre varias instrucciones `WaitSyncTask` del ciclo de programa. El sistema sólo ejecutará las otras tareas conectadas, pero forzará al usuario con un mensaje de error a conectar las tareas de programa excluidas antes de pasar por la instrucción `WaitSyncTask` cooperativa.
- Principio 3) Si se ejecuta de acuerdo con el principio 2, es posible excluir algún ciclo permanente de tarea de programa del panel de selección de tareas para su posterior ejecución según el principio 2, ejecutando la rutina de servicio `SkipTaskExec`.

Recuerde que el panel de selección de tareas está bloqueado mientras el sistema esté funcionando con movimiento sincronizado.

#### Más ejemplos

A continuación aparecen más ejemplos de la instrucción `WaitSyncTask`.

#### Ejemplo 1

##### Ejemplo de programa de la tarea `T_ROB1`

```
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
```

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.335 WaitSyncTask - Esperar en un punto de sincronización con otras tareas de programa

### Multitasking

### Continuación

La tarea de programa T\_ROB1 espera en la instrucción WaitSyncTask hasta que sus unidades mecánicas están en posición y, a continuación, espera a que la tarea de programa T\_ROB2 alcance el punto de sincronización con la misma identidad. Después de esperar durante 60 s, se llama al gestor de errores con ERRNO cambiado al valor ERR\_WAITSYNCTASK. A continuación, se llama de nuevo a la instrucción WaitSyncTask para una espera adicional de 60 s.

### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema ERRNO cambiará a:

| Nombre           | Causa del error   |
|------------------|---|
| ERR_WAITSYNCTASK | Se ha agotado un tiempo límite porque WaitSyncTask no está preparado. |

### Limitación

Si esta instrucción va precedida de una instrucción de movimiento, ésta última debe programarse con un punto de paro (zonedata fine), no un punto de paso. De lo contrario, no será posible reanudar la ejecución tras una caída de suministro eléctrico.

WaitSyncTask \InPos no puede ejecutarse en rutinas de RAPID que estén conectadas a los siguientes eventos especiales del sistema: PowerOn, Stop, QStop, Restart o Step.

### Sintaxis

```
WaitSyncTask
  [ '\ ' InPos ', ' ]
  [ SyncID ' := ' ] < variable (VAR) of syncident > ', '
  [ TaskList ' := ' ] < persistent array { * } (PERS) of tasks >
  [ '\ ' TimeOut ' := ' < expression (IN) of num > ] ' ; '
```

### Información relacionada

| Para obtener más información sobre                | Consulte   |
|---|--|
| Especificación de tareas de programa cooperativas | <a href="#">tasks - Tareas de programa RAPID en la página 1791</a>                 |
| Identidad para punto de sincronización            | <a href="#">syncident - Identidad de punto de sincronización en la página 1786</a> |

## 1.336 WaitTestAndSet - Esperar a que la variable cambie a FALSE y activarla a continuación

### Utilización

`WaitTestAndSet` espera que el valor de una variable persistente de tipo `bool` cambie a `FALSE`. Una vez que la variable cambia al valor `FALSE`, la instrucción cambia el valor a `TRUE` y permite proseguir con la ejecución. La variable persistente puede usarse como un semáforo binario para tareas de sincronización y exclusión mutua.

Esta instrucción tiene la misma funcionalidad subyacente que la función `TestAndSet`, pero `WaitTestAndSet` espera siempre y cuando el valor `bool` sea `FALSE`, mientras que la instrucción `TestAndSet` finaliza inmediatamente.

No se recomienda usar la instrucción `WaitTestAndSet` en rutinas TRAP, gestores de deshacer o rutinas de evento.

A continuación se enumeran algunos de los recursos que pueden necesitar protección de acceso al mismo tiempo:

- Uso de algunas rutinas de RAPID que presentan problemas de funcionamiento cuando se ejecutan en paralelo
- Uso del FlexPendant - Registro del operador

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción `WaitTestAndSet`:  
Consulte también [Más ejemplos en la página 1080](#).

#### Ejemplo 1

##### MAIN program task:

```
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 program task:

```
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;
```

Para evitar mezclar las líneas en el registro del operador, una de MAIN y una de BACK1, el uso de la función `WaitTestAndSet` garantiza que las tres líneas de cada tarea no se separen.

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.336 WaitTestAndSet - Esperar a que la variable cambie a FALSE y activarla a continuación

RobotWare - OS

Continuación

Si la tarea de programa MAIN activa el semáforo

WaitTestAndSet(tproutine\_inuse) en primer lugar, la tarea de programa BACK1 debe esperar hasta que la tarea de programa MAIN libere el semáforo.

---

### Argumentos

WaitTestAndSet Object

Object

Tipo de dato: bool

Objeto de datos definido por el usuario para usarlo como semáforo.- El objeto de datos debe ser una variable persistente PERS. Si se utilizan funciones

WaitTestAndSet entre tareas de programa diferentes, el objeto debe ser de tipo PERS global.

---

### Ejecución de programas

Esta instrucción comprueba y define, en un solo paso indivisible, la variable persistente definida por el usuario como en el ejemplo de código que aparece a continuación:

- Si tiene el valor FALSE, lo cambia a TRUE
- Si tiene el valor TRUE, , espera hasta que cambie a FALSE y lo cambia a TRUE

```
IF Object = FALSE THEN
  Object := TRUE;
ELSE
  ! Wait until it become FALSE
  WaitUntil Object = FALSE;
  Object := TRUE;
ENDIF
```

A continuación, la instrucción queda completada. Para evitar problemas, dado que las variables persistentes conservan su valor si el puntero de programa PP se mueve a Main, cambie siempre el objeto de semáforo a FALSE en el evento de rutina START.

---

### Más ejemplos

A continuación aparecen más ejemplos de la instrucción WaitTestAndSet.

#### Ejemplo 1

```
PERS bool semPers:= FALSE;
...
PROC doit(...)
  WaitTestAndSet semPers;
  ...
  semPers := FALSE;
ENDPROC
```

Continúa en la página siguiente

## 1.336 WaitTestAndSet - Esperar a que la variable cambie a FALSE y activarla a continuación

RobotWare - OS

Continuación



### Nota

Si la ejecución del programa se detiene en la rutina `doit` y el puntero de programa se traslada a `main`, la variable `semPers` no se restablecerá. Para evitarlo, devuelva la variable `semPers` a `FALSE` en la rutina de evento `START`.

### Sintaxis

```
WaitTestAndSet  
[ Object ':= ' ] < persistent (PERS) of bool> ';' 
```

### Información relacionada

| Para obtener más información sobre  | Consulte  |
|---|---|
| Comprobación de una variable y activarla si está desactivada (tipo sondeado con WaitTime) | <a href="#">TestAndSet - Comprueba una variable y la establece si no está establecida en la página 1523</a> |

# 1 Instrucciones

---

1.337 WaitTime - Esperar una cantidad de tiempo determinada  
RobotWare - OS

## 1.337 WaitTime - Esperar una cantidad de tiempo determinada

---

### Utilización

WaitTime se usa para esperar una cantidad de tiempo determinado. Esta instrucción también puede usarse para esperar hasta que los ejes del robot y los ejes externos se hayan parado.

---

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción WaitTime:  
Consulte también [Más ejemplos en la página 1082](#), a continuación.

#### Ejemplo 1

```
WaitTime 0.5;
```

La ejecución del programa espera 0,5 segundos.

---

### Argumentos

```
WaitTime [\InPos] Time
```

[\InPos]

#### *In Position*

Tipo de dato: switch

Si se usa este argumento, los ejes del robot y los ejes externos deben estar completamente parados antes de que empiece a contar el tiempo de espera. Este argumento sólo puede usarse si la tarea controla las unidades mecánicas.

Time

Tipo de dato: num

El tiempo, en segundos, que debe esperar la ejecución del programa. Valor mínimo 0 s. Valor máximo sin límites. Resolución 0,001 s.

---

### Ejecución de programas

La ejecución del programa se detiene temporalmente durante la cantidad de tiempo especificada. Sin embargo, la gestión de interrupciones y otras funciones parecidas permanecen activas.

En el modo manual, después de esperar más de 3 s, aparecerá una ventana de alerta que pregunta si se debe simular la instrucción. Puede configurarse que la alerta no aparezca, estableciendo el parámetro del sistema *SimulateMenu* en NO, consulte *Manual de referencia técnica - Parámetros del sistema*, tema *Controller*, tipo *General RAPID*.

---

### Más ejemplos

A continuación aparecen más ejemplos de cómo usar la instrucción WaitTime.

#### Ejemplo 1

```
WaitTime \InPos,0;
```

La ejecución del programa espera hasta que los ejes del robot y los ejes externos se hayan detenido completamente.

---

*Continúa en la página siguiente*

### Limitaciones

El argumento `\Inpos` no puede usarse junto con `SoftServo`.

Si esta instrucción va precedida de una instrucción `Move`, esta instrucción `Move` debe programarse con un punto de paro (zonedata fine), no un punto de paso. De lo contrario, no será posible reanudar la ejecución tras una caída de suministro eléctrico.

`WaitTime \Inpos` no puede ejecutarse en rutinas de RAPID que estén conectadas a los siguientes eventos especiales del sistema: `PowerOn`, `Stop`, `QStop`, `Restart` o `Step`.

### Sintaxis

```
WaitTime
  ['\ ' InPos ',']
  [ Time ' := ' ] <expression (IN) of num> ';'

```

### Información relacionada

| Para obtener más información sobre             | Consulte   |
|--|--|
| Espera hasta que se cumple una condición       | <a href="#">WaitUntil - Esperar hasta que se cumple una condición en la página 1084</a>            |
| Espera hasta que se activa o desactiva una E/S | <a href="#">WaitDI - Espera hasta que se activa una señal digital de entrada en la página 1042</a> |

# 1 Instrucciones

---

## 1.338 WaitUntil - Esperar hasta que se cumple una condición

RobotWare - OS

## 1.338 WaitUntil - Esperar hasta que se cumple una condición

---

### Utilización

WaitUntil se usa para esperar hasta que se cumpla una condición lógica. Por ejemplo, puede usarse para esperar hasta que se activan una o varias entradas.

---

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción WaitUntil:

Consulte también [Más ejemplos en la página 1087](#).

#### Ejemplo 1

```
WaitUntil di4 = 1;
```

La ejecución del programa sólo continúa después de que se ha activado la entrada di4.

---

### Argumentos

```
WaitUntil [\InPos] Cond [\MaxTime] [\TimeFlag] [\PollRate]
          [\Visualize] [\Header] [\Message] | [\MsgArray] [\Wrap]
          [\Icon] [\Image] [\VisualizeTime] [\UIActiveSignal]
```

[\InPos]

#### *In Position*

Tipo de dato: switch

Si se utiliza este argumento, el robot y los ejes externos deben haber llegado al punto de paro (ToPoint de la instrucción de movimiento actual) para que pueda continuar la ejecución. Este argumento sólo puede usarse si la tarea controla las unidades mecánicas.

Cond

Tipo de dato: bool

La expresión lógica que debe esperarse.

[\MaxTime]

Tipo de dato: num

El periodo máximo permitido para el tiempo de espera, expresado en segundos. Si el tiempo se agota antes de que se active la condición, se llama al gestor de errores si lo hay, con el código de error ERR\_WAIT\_MAXTIME. Si no hay ningún gestor de errores, se detiene la ejecución.

[\TimeFlag]

#### *Timeout Flag*

Tipo de dato: bool

El parámetro de salida que contiene el valor TRUE si se agota el tiempo máximo de espera permitido antes de que se cumpla la condición. Si se incluye este parámetro en la instrucción, no se considera un error si llega a agotarse el tiempo límite. Este argumento no se tiene en cuenta si el argumento MaxTime no se incluye en la instrucción.

*Continúa en la página siguiente*



[\PollRate]

### *Polling Rate*

Tipo de dato: `num`

La frecuencia de sondeo en segundos para comprobar si la condición del argumento `Cond` tiene el valor `TRUE`. Esto significa que `WaitUntil` comprueba primero la condición de inmediato y si no `TRUE`, tras el tiempo especificado hasta `TRUE`. El valor mínimo de la frecuencia de sondeo es 0,004 s. Si no se usa este argumento, se cambia la tasa de sondeo predeterminada a 0,1 s.

[\Visualize]

Tipo de dato: `switch`

Si está seleccionado, se activa la visualización. La visualización consta de un cuadro de mensaje con la condición lógica que no se cumple, icono, título, líneas de mensaje e imagen, de acuerdo con los argumentos del programa.

[\Header]

Tipo de dato: `string`

El texto de cabecera se escribirá en la parte superior del cuadro de mensaje. 40 caracteres como máximo. Si no se utilizara ningún argumento `\Header`, se mostrará un mensaje predeterminado.

[\Message]

Tipo de dato: `string`

Se escribirá una línea de texto en la pantalla. Máximo 50 caracteres.

[\MsgArray]

### *(Message Array)*

Tipo de dato: `string`

Varias líneas de texto de una matriz para escribir en la pantalla. Solo es posible usar uno de los parámetros `\Message` o `\MsgArray` al mismo tiempo.

El máximo espacio es de 5 líneas con 50 caracteres cada una.

[\Wrap]

Tipo de dato: `switch`

Si se selecciona, todas las cadenas especificadas en el argumento `\MsgArray` se concatenan para formar una cadena con un solo espacio entre las distintas cadenas individuales y distribuida en el número mínimo posible de líneas.

De forma predeterminada, cada una de las cadenas del argumento `\MsgArray` aparece en una línea separada de la pantalla.

[\Icon]

Tipo de dato: `icondata`

Define el icono a mostrar. Sólo puede usarse uno de los iconos predefinidos de tipo `icondata`. Consulte [Datos predefinidos en la página 1683](#).

De forma predeterminada, no se usa ningún icono.

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.338 WaitUntil - Esperar hasta que se cumple una condición

RobotWare - OS

Continuación

[\Image]

Tipo de dato: `string`

El nombre de la imagen que debe utilizarse. Para iniciar sus propias imágenes, las imágenes deben estar situadas en el directorio *HOME*: del sistema activo o directamente en el sistema activo.

La recomendación es situar los archivos en el directorio *HOME*: de forma que se incluyan en las operaciones de copia de seguridad y restauración.

Se requiere un reinicio y, a continuación, el FlexPendant carga las imágenes.

Una exigencia del sistema es que se use la opción de RobotWare *FlexPendant Interface*.

La imagen a mostrar puede tener 185 píxeles de anchura y 300 píxeles de altura. Si la imagen tiene un tamaño mayor, solo se muestran 185 \* 300 píxeles de la imagen a partir de su parte superior izquierda.

No es posible especificar ningún valor exacto en cuanto al tamaño que una imagen puede tener o el número de imágenes que es posible cargar en el FlexPendant. Depende del tamaño de los demás archivos cargados en el FlexPendant. La ejecución de los programas continúa simplemente si se usa una imagen que no está cargada en el FlexPendant.

[\VisualizeTime]

Tipo de dato: `num`

El tiempo de espera antes del cuadro de mensaje debe aparecer en FlexPendant. Si se utilizan los argumentos `\VisualizeTime` y `\MaxTime`, el tiempo utilizado en el argumento `\MaxTime` tiene que ser mayor que el tiempo utilizado en el argumento `\VisualizeTime`.

El tiempo predeterminado para la visualización si no se utiliza el argumento `\VisualizeTime` es 5 s. Valor mínimo 1 s. No hay límite de valor máximo. Resolución 0.001 s.

[\UIActiveSignal]

Tipo de dato: `signaldo`

La señal digital de salida utilizada en el argumento opcional `UIActiveSignal` se establece en 1 cuando se activa el cuadro de mensaje de visualización en FlexPendant. Cuando se ha retirado el cuadro de mensaje (cuando se cumple la condición), la señal se vuelve a establecer en 0.

No existe ninguna supervisión de parada o reinicio. La señal se establece en 0 cuando la instrucción está preparada o cuando se mueve el PP.

---

### Ejecución de programas

Si no se cumple la condición programada al ejecutar una instrucción `WaitUntil`, la condición se comprueba de nuevo cada 100 ms (o según el valor especificado en el argumento `Pollrate`).

Cuando el robot está en espera, el tiempo se supervisa. De forma predeterminada, el robot puede esperar para siempre, pero el tiempo de espera máximo puede especificarse con el argumento opcional `\MaxTime`. Si se sobrepasa este tiempo máximo, se genera un error.

Continúa en la página siguiente

En el modo manual, después de esperar más de 3 s, aparecerá una ventana de alerta que pregunta si se debe simular la instrucción. Puede configurarse que la alerta no aparezca, estableciendo el parámetro del sistema *SimulateMenu* en NO, consulte *Manual de referencia técnica - Parámetros del sistema*, tema *Controller*, tipo *General RAPID*.

Si se utiliza el modificador `\Visualize`, aparece un cuadro de mensaje en FlexPendant acorde con los argumentos programados. Si no se utiliza el argumento `\Header`, aparece un texto de cabecera predeterminado. Cuando la ejecución de la instrucción `WaitUntil` está preparada, el cuadro de mensaje se elimina de FlexPendant.

El nuevo cuadro de mensaje del nivel de rutina TRAP toma el foco del cuadro de mensaje del nivel básico.

### Más ejemplos

A continuación aparecen más ejemplos de cómo usar la instrucción `WaitUntil`.

#### Ejemplo 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
```

Si no se cumplen las dos condiciones de entrada en un plazo de 60 segundos, se escribe un mensaje de error en la pantalla del FlexPendant.

#### Ejemplo 2

```
WaitUntil \Inpos, di4 = 1;
```

La ejecución del programa espera hasta que el robot se ha detenido y hasta que se activa la entrada `di4`.

#### Ejemplo 3

```
WaitUntil di4 = 1 \MaxTime:=5.5;
..
ERROR
  ERROR
    IF ERRNO = ERR_WAIT_MAXTIME THEN
      RAISE;
    ELSE
      Stop;
    ENDIF
```

La ejecución del programa espera hasta que la entrada `di4` se active. Si el dispositivo de E/S ha sido desactivado o si se alcanza el tiempo límite de espera, la ejecución continúa en el gestor de errores.

#### Ejemplo 4

```
WaitUntil di1 = 1 AND di2 = 1 \MaxTime := 60 \Visualize;
```

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.338 WaitUntil - Esperar hasta que se cumple una condición

RobotWare - OS

Continuación

```
..
ERROR
  IF ERRNO = ERR_WAIT_MAXTIME THEN
    RAISE;
  ELSE
    Stop;
  ENDIF
```

Si no se cumplen las dos condiciones de entrada en un plazo de 5 segundos, se escribe un mensaje de error en la pantalla del FlexPendant. Si la condición no se cumple en los 60 siguientes segundos, la ejecución continúa en el gestor de errores.

### Ejemplo 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;
..
```

Si no se cumplen las dos condiciones de entrada, entonces la cabecera y el mensaje especificado en los argumentos opcionales `\Header` y `\MsgArray` se escribirán en la pantalla del FlexPendant junto con las condiciones que no se cumplen.

### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre             | Causa del error   |
|--------------------|---|
| ERR_NO_ALIASIO_DEF | La variable de señal es una variable declarada en RAPID. No se ha conectado a una señal E/S definida en la configuración de E/S con la instrucción <code>AliasIO</code> .                             |
| ERR_NORUNUNIT      | Existe una señal utilizada en la condición, y no existe ningún contacto con el dispositivo de E/S.  |
| ERR_SIG_NOT_VALID  | La señal de E/S no está disponible. Los motivos pueden ser que el dispositivo de E/S no está en funcionamiento o que exista un error en la configuración (válido solamente para el bus de campo ICI). |
| ERR_WAIT_MAXTIME   | Existe un tiempo límite (parámetro <code>\MaxTime</code> ) antes de que la condición haya cambiado al valor correcto.   |

### Limitación

El argumento `\Inpos` no puede usarse junto con `SoftServo`.

Si esta instrucción va precedida de una instrucción `Move`, esta instrucción `Move` debe programarse con un punto de paro (zonedata fine), no un punto de paso. De lo contrario, no será posible reanudar la ejecución tras una caída de suministro eléctrico.

`WaitUntil \Inpos` no puede ejecutarse en rutinas de RAPID que estén conectadas a los siguientes eventos especiales del sistema: `PowerOn`, `Stop`, `QStop`, `Restart` o `Step`.

Continúa en la página siguiente

WaitUntil \Inpos no puede utilizarse junto con StopMove para detectar si el movimiento se ha parado. La instrucción WaitUntil puede quedar en suspenso para siempre en ese caso. No detecta que el movimiento se ha detenido, pero sí detecta que los ejes del robot y los ejes externos han alcanzado el último ToPoint programado (MoveX, SearchX, TriggX).

### Sintaxis

```

WaitUntil
  ['\ ' InPos ',']
  [Cond ':=' ] <expression (IN) of bool>
  ['\ ' 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>]
  ['\ ' UIActiveSignal ':=' <variable (VAR) of signaldo>] ';'

```

### Información relacionada

| Para obtener más información sobre                 | Consulte   |
|--|--|
| Espera hasta que se activa o desactiva una entrada | <a href="#">WaitDI - Espera hasta que se activa una señal digital de entrada en la página 1042</a> |
| Espera durante un tiempo determinado               | <a href="#">WaitTime - Esperar una cantidad de tiempo determinada en la página 1082</a>            |
| Expresiones  | <a href="#">Manual de referencia técnica - RAPID Overview</a>                                      |

# 1 Instrucciones

---

## 1.339 WaitWObj - Esperar a un objeto de trabajo en un transportador *Conveyor Tracking*

### 1.339 WaitWObj - Esperar a un objeto de trabajo en un transportador

---

#### Utilización

WaitWObj (*Wait Work Object*) establece una conexión a un objeto de trabajo de la ventana de inicio de la unidad mecánica de transportador.

---

#### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción WaitWObj:  
Consulte también [Más ejemplos en la página 1091](#).

#### Ejemplo 1

```
WaitWObj wobj_on_cnv1;
```

El programa se conecta al primer objeto de la cola de objetos que se encuentre dentro de la ventana de inicio del controlador. Si no hay ningún objeto en la ventana de inicio, la ejecución espera algún objeto.

---

#### Argumentos

```
WaitWObj WObj [ \RelDist ][\MaxTime][\TimeFlag]
```

WObj

##### *Work Object*

Tipo de dato: wobjdata

El objeto de trabajo móvil (sistema de coordenadas) con el que está relacionada la posición de robot indicada en la instrucción. El transportador de la unidad mecánica debe especificarse con `ufmec` en el objeto de trabajo.

[ \RelDist ]

##### *Relative Distance*

Tipo de dato: num

Espera a que un objeto entre en la ventana de inicio y pase más allá de la distancia especificada por el argumento. Si el objeto ya está conectado, la ejecución espera hasta que el objeto pasa más allá de la distancia indicada. Si el objeto ya ha pasado más allá de la `\RelDist` la ejecución continúa.

[\MaxTime]

##### *Maximum Time*

Tipo de dato: num

El periodo máximo permitido para el tiempo de espera, expresado en segundos. Si el tiempo se agota antes de la conexión del objeto o antes de que se alcance la distancia `\RelDist`, se llama al gestor de errores si lo hay, con el código de error `ERR_WAIT_MAXTIME`. Si no hay ningún gestor de errores, se detiene la ejecución.

[\TimeFlag]

##### *Timeout Flag*

Tipo de dato: bool

*Continúa en la página siguiente*

## 1.339 WaitWObj - Esperar a un objeto de trabajo en un transportador *Conveyor Tracking* *Continuación*

El parámetro de salida que contiene el valor `TRUE` si se agota el tiempo máximo de espera permitido antes de que se conecte el objeto o se alcance la distancia `\RelDist`. Si se incluye este parámetro en la instrucción, no se considera un error si llega a agotarse el tiempo límite. Este argumento no se tiene en cuenta si el argumento `MaxTime` no se incluye en la instrucción.

### Ejecución de programas

Si no hay ningún objeto en la ventana de inicio, la ejecución del programa espera. Si hay un objeto presente, éste se conecta al transportador y la ejecución prosigue.

Si se ejecuta una segunda instrucción `WaitWObj` mientras la conexión ya existe, se devuelve un error a no ser que se utilice el argumento opcional `\RelDist`.

### Más ejemplos

A continuación aparecen más ejemplos de la instrucción `WaitWObj`.

#### Ejemplo 1

```
WaitWObj wobj_on_cnv1\RelDist:=500.0;
```

Si no hay conexión, se espera a que el objeto entre en la ventana de inicio y a continuación se espera a que el objeto sobrepase un punto situado a 500 mm en el transportador.

Si ya existe una conexión al objeto, se espera a que el objeto sobrepase los 500 mm.

Si no hay conexión, se espera la presencia de un objeto en la ventana de inicio.

#### Ejemplo 2

```
WaitWObj wobj_on_cnv1\RelDist:=0.0;
```

Si ya existe una conexión, se continúa la ejecución dado que el objeto ya ha sobrepasado la posición de 0,0 mm.

#### Ejemplo 3

```
WaitWObj wobj_on_cnv1;  
WaitWObj wobj_on_cnv1\RelDist:=0.0;
```

La primera instrucción `WaitWObj` se conecta al objeto que se encuentra en la ventana de inicio. La segunda instrucción `WaitWObj` retorna inmediatamente si el objeto sigue conectado. Sin embargo, esperará al siguiente objeto si el objeto anterior se ha movido más allá de la distancia máxima o ha sido soltado.

#### Ejemplo 4

```
WaitWObj wobj_on_cnv1\RelDist:=500.0\MaxTime:=0.1 \Timeflag:=flag1;
```

La instrucción `WaitWObj` retorna inmediatamente si el objeto ha pasado más allá de los 500 mm pero, de lo contrario, espera a un objeto durante 0,1 s. Si ningún objeto sobrepasa la posición de 500 mm durante estos 0,1 s, la instrucción retornará con `flag1 = TRUE`.

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.339 WaitWObj - Esperar a un objeto de trabajo en un transportador

### Conveyor Tracking

#### Continuación

#### Limitaciones

Se requieren 50 ms para conectarse al primer objeto de la ventana de inicio. Después de la conexión, una segunda instrucción `WaitWObj` con el argumento opcional `\RelDist` sólo requiere el tiempo normal de ejecución de una instrucción de `RAPID`.

#### Gestión de errores

Si se producen los errores siguientes durante la ejecución de la instrucción `WaitWObj`, la variable de sistema `ERRNO` cambia de valor. Estos errores pueden ser gestionados en el gestor de errores.

| Nombre                        | Causa del error   |
|-------------------------------|---|
| <code>ERR_CNV_NOT_ACT</code>  | El transportador no está activado.  |
| <code>ERR_CNV_CONNECT</code>  | La instrucción <code>WaitWObj</code> ya está conectada.   |
| <code>ERR_CNV_DROPPED</code>  | El objeto que estaba esperando la instrucción <code>WaitWObj</code> ha sido desechado por otra tarea. |
| <code>ERR_WAIT_MAXTIME</code> | El objeto no llegó a tiempo y no hay ningún <code>Timeflag</code> .                                   |
| <code>ERR_CNV_OBJ_LOST</code> | El objeto ha pasado el <code>StartwindowWidth</code> sin conectarse.                                  |

#### Sintaxis

```
WaitWObj
  [ WObj ':='] < persistent (PERS) of wobjdata> ';'
  [ '\ RelDist ':=> < expression (IN) of num > ]
  [ '\ MaxTime ':=> <expression (IN) of num>]
  [ '\ TimeFlag ':=> <variable (VAR) of bool>] ';
```

#### Información relacionada

| Para obtener más información sobre                     | Consulte   |
|--|--|
| Colocación de un objeto de trabajo en un transportador | <a href="#">DropWObj - Suelta un objeto de trabajo sobre un transportador en la página 193</a> |
| Seguimiento de transportadores                         | <i>Application manual - Conveyor tracking</i>  |



## 1.340 WarmStart - Reinicio del controlador

### Utilización

WarmStart se utiliza para reiniciar el controlador.

Los parámetros del sistema pueden cambiarse desde RAPID con la instrucción WriteCfgData. Debe reiniciar el controlador para que los cambios de algunos parámetros del sistema entren en vigor. El reinicio puede realizarse con la instrucción WarmStart.

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción WarmStart:

#### Ejemplo 1

```
WriteCfgData "/MOC/MOTOR_CALIB/rob1_1", "cal_offset", offset1;
WarmStart;
```

Escribe el valor de la variable de tipo num de tipo offset1 como offset de calibración del eje 1 de rob1 y genera un reinicio del controlador.

### Ejecución de programas

Warmstart se aplica inmediatamente y el puntero de programa se sitúa en la instrucción siguiente.

### Sintaxis

```
WarmStart ';' ;
```

### Información relacionada

| Para obtener más información sobre                   | Consulte   |
|--|--|
| Escritura de un atributo de un parámetro del sistema | <a href="#">WriteCfgData - Escribe un atributo de un parámetro del sistema en la página 1107</a> |
| Configuración  | <i>Manual de referencia técnica - Parámetros del sistema</i>                                     |
| <i>Advanced RAPID</i>                                | <i>Especificaciones del producto - Controller software IRC5</i>                                  |

# 1 Instrucciones

---

## 1.341 WHILE - Repetir siempre y cuando se cumpla una condición

RobotWare - OS

## 1.341 WHILE - Repetir siempre y cuando se cumpla una condición

---

### Utilización

WHILE se usa cuando es necesario repetir un conjunto de instrucciones siempre y cuando la evaluación de la expresión de la condición determinada dé como resultado el valor TRUE.



#### Recomendación

Si es posible determinar el número de repeticiones, puede usarse la instrucción FOR.

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción WHILE:

#### Ejemplo 1

```
WHILE reg1 < reg2 DO
  ...
  reg1 := reg1 + 1;
ENDWHILE
```

Repita las instrucciones que se encuentran dentro del bloque WHILE siempre y cuando `reg1 < reg2`.

### Argumentos

```
WHILE Condition DO ... ENDWHILE
```

Condition

Tipo de dato: bool

La condición cuya evaluación debe dar como resultado TRUE para que se ejecuten las instrucciones que se encuentran dentro del bloque WHILE.

### Ejecución de programas

- 1 Se evalúa la expresión de la condición. Si la evaluación de la expresión da como resultado TRUE, se ejecutan las instrucciones que se encuentran dentro del bloque WHILE.
- 2 A continuación, la expresión de la condición se evalúa de nuevo y, si el resultado de la evaluación es TRUE, se ejecutan de nuevo las instrucciones del bloque WHILE.
- 3 Este proceso continúa hasta que el resultado de la evaluación de la expresión dé como resultado FALSE.

En ese momento, la iteración se termina y la ejecución del programa continúa en la instrucción que sigue al bloque WHILE.

Si el resultado de la evaluación de la expresión da como resultado FALSE desde el principio, no se llegan a ejecutar las instrucciones del bloque WHILE y el control del programa se transfiere inmediatamente a la instrucción que sigue al bloque WHILE.

*Continúa en la página siguiente*

## 1.341 WHILE - Repetir siempre y cuando se cumpla una condición

RobotWare - OS

Continuación

### Sintaxis

```
WHILE <conditional expression> DO
  <statement list>
ENDWHILE
```

### Información relacionada

| Para obtener más información sobre        | Consulte  |
|---|---|
| Expresiones                               | <i>Manual de referencia técnica - RAPID Overview, sección Características básicas - Expresiones</i> |
| Repetición un número determinado de veces | <a href="#">FOR - Repite un número determinado de veces en la página 219</a>                        |

# 1 Instrucciones

---

1.342 WorldAccLim - Control de aceleración en el sistema de coordenadas mundo  
RobotWare - OS

## 1.342 WorldAccLim - Control de aceleración en el sistema de coordenadas mundo

---

### Utilización

WorldAccLim (*World Acceleration Limitation*) se usa para limitar la aceleración y deceleración de la herramienta (y de la carga útil) en el sistema de coordenadas mundo.

La limitación se consigue conjuntamente en el punto central de gravedad de la herramienta usada, la carga útil real (si la hay) y la brida de montaje del robot.

Esta instrucción sólo puede usarse en la tarea principal T\_ROB1 o, si se cuenta con un sistema *MultiMove*, en las tareas de movimiento.

### Ejemplos básicos

Los ejemplos siguientes ilustran la instrucción WorldAccLim.

#### Ejemplo 1

```
WorldAccLim \On := 3.5;
```

La aceleración y la deceleración están limitadas a 3,5 m/s<sup>2</sup>.

#### Ejemplo 2

```
WorldAccLim \Off;
```

La aceleración/deceleración se restablece a su valor máximo (predeterminado).

### Argumentos

```
WorldAccLim [\On] | [\Off]
```

[\On]

Tipo de dato: num

El valor absoluto de la limitación de aceleración/deceleración en m/s<sup>2</sup>.

[\Off]

Tipo de dato: switch

Sin límites. Aceleración máxima (predeterminada).

### Ejecución de programas

La limitación de la aceleración y la deceleración se aplica a la siguiente instrucción de movimiento del robot ejecutada y es válida hasta que se ejecute una nueva instrucción WorldAccLim.

La aceleración máxima (WorldAccLim \Off) se establece automáticamente en los casos siguientes:

- cuando se utiliza el modo de reinicio **Restablecer RAPID**
- al cargar un nuevo programa o un nuevo módulo
- al iniciar la ejecución del programa desde el principio
- al mover el puntero del programa a main
- al mover el puntero del programa a una rutina
- al mover el puntero de programa de una forma que se pierde el orden de la ejecución.

*Continúa en la página siguiente*

## 1.342 WorldAccLim - Control de aceleración en el sistema de coordenadas mundo

RobotWare - OS

Continuación

Se recomienda usar sólo un tipo de limitación de la aceleración. Si se usa una combinación de instrucciones `WorldAccLim`, `AccSet` y `PathAccLim`, el sistema reduce la aceleración y deceleración en el orden siguiente:

- Según `WorldAccLim`
- Según `AccSet`
- Según `PathAccLim`

### Limitaciones

La aceleración mínima permitida es de 0,1 m/s<sup>2</sup>.

Los siguientes modelos de robot no se admiten ni pueden utilizar la instrucción `WorldAccLim`:

- IRB 340, IRB 360, IRB 540, IRB 1400, IRB 1410

### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                       | Causa del error   |
|------------------------------|---|
| <code>ERR_ACC_TOO_LOW</code> | El argumento <code>On</code> está establecido en un valor demasiado bajo. |

### Sintaxis

```
WorldAccLim
[ '\ On ' := ' <expression (IN) of num> ] | [ '\ Off ] ';'

```

### Información relacionada

| Para obtener más información sobre                     | Consulte   |
|--|--|
| Instrucciones de posicionamiento                       | <a href="#">Manual de referencia técnica - RAPID Overview</a>  |
| Datos de parámetros de movimiento                      | <a href="#">motsetdata - Datos de parámetros de movimiento en la página 1705</a>                         |
| Reducción de la aceleración                            | <a href="#">AccSet - Reduce la aceleración en la página 25</a>   |
| Limitación de aceleración a lo largo de la trayectoria | <a href="#">PathAccLim - Reduce la aceleración del TCP a lo largo de la trayectoria en la página 504</a> |

# 1 Instrucciones

---

1.343 Write - Escribe en un archivo o dispositivo de E/S alfanumérico  
RobotWare - OS

## 1.343 Write - Escribe en un archivo o dispositivo de E/S alfanumérico

---

### Utilización

`Write` se usa para escribir en un archivo o dispositivo de E/S serie alfanumérico. Es posible escribir el valor de determinados datos, además de texto.

---

### Ejemplos básicos

Los siguientes ejemplos ilustran la función `Write`:

#### Ejemplo 1

```
Write logfile, "Execution started";
```

Se escribe el texto `Execution started` en el archivo al que se hace referencia en `logfile`.

#### Ejemplo 2

```
VAR num reg1:=5;  
...  
Write logfile, "No of produced parts="\Num:=reg1;
```

Se escribe el texto `No of produced parts=5` en el archivo al que se hace referencia en `logfile`.

---

### Argumentos

```
Write IODevice String [\Num] | [\Bool] | [\Pos] | [\Orient] |  
[\Dnum] [\NoNewLine]
```

IODevice

**Tipo de dato:** `iodev`

El nombre (referencia) del archivo o dispositivo de E/S actual.

String

**Tipo de dato:** `string`

El texto a escribir.

[\Num]

**Numeric**

**Tipo de dato:** `num`

El dato cuyo valor numérico se desea escribir a continuación de la cadena de texto.

[\Bool]

**Boolean**

**Tipo de dato:** `bool`

El dato cuyo valor lógico se desea escribir a continuación de la cadena de texto.

[\Pos]

**Position**

**Tipo de dato:** `pos`

El dato cuya posición se desea escribir a continuación de la cadena de texto.

---

*Continúa en la página siguiente*

1.343 Write - Escribe en un archivo o dispositivo de E/S alfanumérico  
*RobotWare - OS*  
*Continuación*

[ \Orient ]

*Orientation*

Tipo de dato: orient

El dato cuya orientación se desea escribir a continuación de la cadena de texto.

[ \Dnum ]

*Numeric*

Tipo de dato: dnum

El dato cuyo valor numérico se desea escribir a continuación de la cadena de texto.

[ \NoNewLine ]

Tipo de dato: switch

Omite el carácter de salto de línea que suele indicar el final del texto. Es decir, la siguiente instrucción `write` escribe a continuación en la misma línea.

**Ejecución de programas**

La cadena de texto se escribe en un archivo o un dispositivo de E/S especificado. También se escribe un carácter de salto de línea (LF), pero también puede omitirse si se usa el argumento `\NoNewLine`.

Si se usa uno de los argumentos `\Num`, `\Bool`, `\Pos` o `\Orient`, su valor se convierte en primer lugar en una cadena de texto, antes de añadirla a la primera cadena. La conversión del valor a una cadena de texto se realiza de la forma siguiente:

| Argumento            | Valor                  | Cadena de texto          |
|----------------------|------------------------|--------------------------|
| <code>\Num</code>    | 23                     | "23"                     |
| <code>\Num</code>    | 1.141367               | "1.14137"                |
| <code>\Bool</code>   | TRUE                   | "TRUE"                   |
| <code>\Pos</code>    | [1817.3,905.17,879.11] | "[1817.3,905.17,879.11]" |
| <code>\Orient</code> | [0.96593,0,0.25882,0]  | "[0.96593,0,0.25882,0]"  |
| <code>\Dnum</code>   | 4294967295             | "4294967295"             |

El valor se convierte en una cadena con un formato estándar de RAPID. Esto significa en principio 6 dígitos significativos. Si la parte decimal es menor que 0,000005 o mayor que 0,999995, el número se redondea a un entero.

En caso de un reinicio tras una caída de alimentación, todos los archivos o dispositivos de E/S abiertos del sistema se cierran y el descriptor de E/S de la variable del tipo `iodev` se restablece.

**Limitaciones**

Los argumentos `\Num`, `\Dnum`, `\Bool`, `\Pos` y `\Orient` son excluyentes entre sí y por tanto no pueden usarse simultáneamente en una misma instrucción.

Esta instrucción sólo puede usarse con dispositivos de E/S o archivos que hayan sido abiertos para escritura.

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.343 Write - Escribe en un archivo o dispositivo de E/S alfanumérico

RobotWare - OS

Continuación

### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                   | Causa del error                           |
|--------------------------|---|
| <code>ERR_FILEACC</code> | Se produjo un error durante la escritura. |

### Sintaxis

```
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 ] ';'
  [ 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 ]
  [ '\ ISOLatin1Encoding ] ';'

```

### Información relacionada

| Para obtener más información sobre        | Consulte   |
|---|--|
| Abrir un archivo o dispositivo de E/S     | <i>Manual de referencia técnica - RAPID Overview</i> |
| Gestión de archivos y dispositivos de E/S | <i>Application manual - Controller software IRC5</i> |



## 1.344 WriteAnyBin - Escribe datos en un archivo o dispositivo de E/S binario

### Utilización

`WriteAnyBin` (*Write Any Binary*) se utiliza para escribir cualquier tipo de dato en un dispositivo de E/S o un archivo binario.

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción `WriteAnyBin`:

#### Ejemplo 1

```
VAR iodev file1;
VAR orient quat1 := [1, 0, 0, 0];
...
Open "HOME:" \File:= "FILE1.DOC", file1 \Bin;
WriteAnyBin file1, quat1;
```

El dato de tipo `orient data quat1` se escribe en el canal al que se hace referencia con `channell`.

### Argumentos

`WriteAnyBin` IODevice Data

IODevice

Tipo de dato: `iodev`

El nombre (referencia) del archivo o dispositivo de E/S binario para la operación de escritura.

Data

Tipo de dato: `ANYTYPE`

Dato a escribir.

### Ejecución de programas

Se escriben en el dispositivo de E/S o el archivo binario especificado el número de bytes necesario para los datos especificados.

En caso de un reinicio tras una caída de alimentación, todos los archivos o dispositivos de E/S abiertos del sistema se cierran y el descriptor de E/S de la variable del tipo `iodev` se restablece.

### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                   | Causa del error                           |
|--------------------------|---|
| <code>ERR_FILEACC</code> | Se produjo un error durante la escritura. |

### Limitaciones

Esta instrucción sólo puede usarse con dispositivos de E/S o archivos que hayan sido abiertos para escritura binaria.

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.344 WriteAnyBin - Escribe datos en un archivo o dispositivo de E/S binario

RobotWare - OS

Continuación

Los datos a escribir con esta instrucción `WriteAnyBin` deben ser de los tipos de datos `num`, `bool` o `string`. También puede usarse un registro, un componente de registro, una matriz o un elemento de matriz de este tipo de dato de valor. No es posible usar datos enteros o datos parciales con `semivalor` ni tipos de datos sin valor.

Dado que `WriteAnyBin-ReadAnyBin` sólo se han diseñado para enviar datos internos de controlador entre controladores de robot no se hace público ningún protocolo de datos y no es posible interpretar estos datos en ningún PC.



### Nota

El desarrollo de software de control puede afectar a la compatibilidad, de forma que quizá no sea posible usar `WriteAnyBin-ReadAnyBin` entre versiones de software de RobotWare diferentes.

### Sintaxis

```
WriteAnyBin
  [ IODevice '[:=' ] <variable (VAR) of iodev> ','
  [ Data '[:=' ] <expression (IN) of ANYTYPE> ';'

```

### Información relacionada

| Para obtener más información sobre                       | Consulte   |
|--|--|
| Apertura de dispositivos de E/S o archivos               | <i>Manual de referencia técnica - RAPID Overview</i>   |
| Leer datos de un dispositivo de E/S o un archivo binario | <a href="#">ReadAnyBin - Leer datos de un dispositivo de E/S o un archivo binario en la página 577</a> |
| Gestión de archivos y dispositivos de E/S                | <i>Application manual - Controller software IRC5</i>   |

### 1.345 WriteBin - Escribe en un dispositivo de E/S binario

#### Utilización

`WriteBin` se usa para escribir un número de bytes en un dispositivo de E/S binario.

#### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción `WriteBin`:

##### Ejemplo 1

```
WriteBin channel2, text_buffer, 10;
```

Se escriben 10 caracteres de la lista `text_buffer` en el canal al que se hace referencia con `channel2`.

#### Argumentos

```
WriteBin IODevice Buffer NChar
```

IODevice

Tipo de dato: `iodev`

Nombre (referencia) del dispositivo de E/S actual.

Buffer

Tipo de dato: `array of num`

La lista (matriz) que contiene los números (caracteres) que se desean escribir.

NChar

*Number of Characters*

Tipo de dato: `num`

El número de caracteres a escribir desde `Buffer`.

#### Ejecución de programas

Se escribe en el dispositivo de E/S la cantidad especificada de números (caracteres) de la lista.

En caso de un reinicio tras una caída de alimentación, todos los archivos o dispositivos de E/S abiertos del sistema se cierran y el descriptor de E/S de la variable del tipo `iodev` se restablece.

#### Limitaciones

Esta instrucción sólo puede usarse con dispositivos de E/S que hayan sido abiertos para escritura binaria.

#### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                   | Causa del error                           |
|--------------------------|---|
| <code>ERR_FILEACC</code> | Se produjo un error durante la escritura. |

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.345 WriteBin - Escribe en un dispositivo de E/S binario

RobotWare - OS

Continuación

---

### Sintaxis

```
WriteBin
  [ IODevice ':='] <variable (VAR) of iodev> ','
  [ Buffer ':='] <array {*} (IN) of num> ','
  [ NChar ':='] <expression (IN) of num> ';'

```

---

### Información relacionada

| Para obtener más información sobre                     | Consulte  |
|--|---|
| Apertura, etc. de dispositivos de E/S                  | <i>Manual de referencia técnica - RAPID Overview</i>                          |
| Conversión de una cadena de caracteres a datos de byte | <a href="#">StrToByte - Convierte una cadena en un byte en la página 1509</a> |
| Datos de byte  | <a href="#">byte - Valores enteros 0-255 en la página 1616</a>                |
| Gestión de archivos y dispositivos de E/S              | <i>Application manual - Controller software IRC5</i>                          |

## 1.346 WriteBlock - Escribir un bloque de datos en un dispositivo

### Utilización

`WriteBlock` se utiliza para escribir un bloque de datos en un dispositivo que está conectado a la interfaz de sensores serie. Los datos se capturan de un archivo.

La interfaz de sensores se comunica con los sensores a través de canales serie, utilizando el protocolo de transporte RTP1.

Éste es un ejemplo de configuración de un canal de sensor.

```
COM_PHY_CHANNEL:
  Name "COM1:"
  Connector "COM1"
  Baudrate 19200
COM_TRP:
  Name "sen1:"
  Type "RTP1"
  PhyChannel "COM1"
```

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción `WriteBlock`:

#### Ejemplo 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;
```

### Argumentos

```
WriteBlock device BlockNo FileName [ \TaskName ]
```

device

**Tipo de dato:** string

El nombre del dispositivo de E/S configurado en `sio.cfg` para el sensor utilizado.

BlockNo

**Tipo de dato:** num

El argumento `BlockNo` se utiliza para seleccionar el bloque de datos de sensor que se desea escribir.

FileName

**Tipo de dato:** string

El argumento `FileName` se utiliza para seleccionar un archivo cuyos datos se escriben en el bloque de datos de sensor seleccionado en el argumento `BlockNo`.

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.346 WriteBlock - Escribir un bloque de datos en un dispositivo

### Sensor Interface

#### Continuación

[ \TaskName ]

Tipo de dato: string

El argumento TaskName hace posible el acceso a dispositivos de otras tareas de RAPID.

#### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema ERRNO cambiará a:

| Constante de error (valor de ERRNO) | Descripción                            |
|-------------------------------------|--|
| SEN_NO_MEAS                         | Fallo de medición                      |
| SEN_NOREADY                         | Sensor incapaz de gestionar el comando |
| SEN_GENERRO                         | Error general del sensor               |
| SEN_BUSY                            | Bus de sensor                          |
| SEN_UNKNOWN                         | Sensor desconocido                     |
| SEN_EXALARM                         | Error de sensor externo                |
| SEN_CAALARM                         | Error de sensor interno                |
| SEN_TEMP                            | Error de temperatura del sensor        |
| SEN_VALUE                           | Valor de comunicación no válido        |
| SEN_CAMCHECK                        | Fallo de comprobación de sensor        |
| SEN_TIMEOUT                         | Error de comunicación                  |

#### Sintaxis

```
WriteBlock
  [ device ':'= ' ] < expression(IN) of string > ','
  [ BlockNo ':'= ' ] < expression (IN) of num > ','
  [ FileName ':'= ' ] < expression (IN) of string > ','
  [ '\ ' TaskName ':'= ' < expression (IN) of string > ] ';' ;
```

#### Información relacionada

| Para obtener más información sobre                         | Consulte   |
|--|--|
| Establecimiento de una conexión a un dispositivo de sensor | <a href="#">SenDevice - Establece una conexión a un dispositivo de sensor en la página 687</a> |
| Escritura de una variable de sensor                        | <a href="#">WriteVar - Escribir una variable en la página 1116</a>                             |
| Lectura de un bloque de datos de sensor                    | <a href="#">ReadBlock - Lee un bloque de datos de un dispositivo en la página 580</a>          |
| Configuración de la comunicación del sensor                | <a href="#">Manual de referencia técnica - Parámetros del sistema</a>                          |

## 1.347 WriteCfgData - Escribe un atributo de un parámetro del sistema

### Utilización

`WriteCfgData` se utiliza para escribir un atributo de un parámetro del sistema (de los datos de configuración).

Además de escribir parámetros con nombre, también es posible buscar y actualizar parámetros sin nombre.

### Ejemplos básicos

Los siguientes ejemplos ilustran la instrucción `WriteCfgData`. Estos dos ejemplos muestran cómo escribir datos de parámetro con nombre.

#### Ejemplo 1

```
VAR num offset1 := 1.2;
...
WriteCfgData "/MOC/MOTOR_CALIB/rob1_1", "cal_offset", offset1;
```

Escribe en la variable de tipo `num` con nombre `offset1` el offset de calibración para el eje 1 de `rob_1`.

#### Ejemplo 2

```
VAR string io_device := "my_device";
...
WriteCfgData "/EIO/EIO_SIGNAL/process_error", "Device", io_device;
```

Escribe en la variable de tipo `string` con nombre `io_device`, el nombre del dispositivo de E/S en el que está definida la señal `process_error`.

### Argumentos

```
WriteCfgData InstancePath Attribute CfgData [\ListNo]
```

InstancePath

Tipo de dato: `string`

Especifica una ruta hasta el parámetro que debe leerse.

En el caso de los parámetros con nombre, el formato de esta cadena es `/DOMAIN/TYPE/ParameterName`.

En el caso de los parámetros sin nombre, el formato de esta cadena es `/DOMAIN/TYPE/Attribute/AttributeValue`.

Attribute

Tipo de dato: `string`

El nombre del atributo del parámetro que se desea escribir.

CfgData

Tipo de dato: `anytype`

El objeto de datos desde el que se leen los nuevos datos que se desean almacenar. En función del tipo de atributo, los tipos válidos son `bool`, `num`, `dnum` o `string`.

[\ListNo]

Tipo de dato: `num`

*Continúa en la página siguiente*

## 1 Instrucciones

---

### 1.347 WriteCfgData - Escribe un atributo de un parámetro del sistema

RobotWare - OS

Continuación

Una variable que contiene el número de instancia de atributo+ AttributeValue a encontrar y actualizar.

La primera vez que aparece Attribute + AttributeValue tiene el número de instancia 0. Si hay más instancias a buscar, el valor devuelto en \ListNo se incrementará en 1. De lo contrario, si no hay más instancias el valor de retorno será -1. La constante predefinida END\_OF\_LIST puede usarse para comprobar si hay más instancias a buscar.

---

### Ejecución de programas

El valor del atributo especificado por el argumento Attribute se define de acuerdo con el valor del objeto de dato especificado por el argumento CfgData.

Si se usa el formato /DOMAIN/TYPE/ParameterName en InstancePath, sólo están disponibles los parámetros con nombre, es decir, los parámetros cuyo primer atributo sea name, Name o NAME.

En el caso de los parámetros sin nombre, utilice el parámetro opcional \ListNo para especificar en cuál de las instancias debe escribirse el valor del atributo. Se actualiza tras cada escritura exitosa en la siguiente instancia disponible.

---

### Más ejemplos

A continuación aparecen más ejemplos de la instrucción WriteCfgdata. Estos dos ejemplos muestran cómo escribir en parámetros sin nombre.

#### Ejemplo 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;
```

Lee la señal resultante de la señal digital de actuación sin nombre do\_13 y guarda el nombre en la variable de cadena llamada read\_str. A continuación se actualiza el nombre a di\_13 con el prefijo "my".

En este ejemplo, el dominio EIO tiene el código cfg siguiente:

EIO\_CROSS:

```
-Name "Cross_di_1_do_2" -Res "di_1" -Act1 "do_2"
-Name "Cross_di_2_do_2" -Res "di_2" -Act1 "do_2"
-Name "Cross_di_13_do_13" -Res "di_13" -Act1 "do_13"
```

#### Ejemplo 2

```
VAR num read_index;
VAR num write_index;
VAR string read_str;
...
read_index:=0;
```

Continúa en la página siguiente



## 1.347 WriteCfgData - Escribe un atributo de un parámetro del sistema

RobotWare - OS

Continuación

```

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;
  ENDF
ENDWHILE

```

Leer los nombres de todas las señales definidas para el dispositivo de E/S USERIO. Cambiar los nombres de las señales al nombre leído, con el prefijo "my".

En este ejemplo, el dominio EIO tiene el código cfg siguiente:

```

EIO_SIGNAL:
  -Name "USERD01" -SignalType "DO" -Device "USERIO" -DeviceMap "0"
  -Name "USERD02" -SignalType "DO" -Device "USERIO" -DeviceMap "1"
  -Name "USERD03" -SignalType "DO" -Device "USERIO" -DeviceMap "2"

```

## Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema ERRNO cambiará a:

| Nombre              | Causa del error   |
|---------------------|---|
| ERR_CFG_NOTFND      | No fue posible encontrar los datos especificados con "InstancePath + Attribute" en la base de datos de configuración.   |
| ERR_CFG_ILLTYPE     | El tipo de dato del parámetro CfgData es distinto del tipo de dato real de los datos encontrados y especificados con "InstancePath + Attribute" en la base de datos de configuración. |
| ERR_CFG_LIMIT       | Los datos para el parámetro CfgData se encuentran fuera de los límites (valor máx./mín.).   |
| ERR_CFG_INTERNAL    | Intento de escribir datos protegidos escritos internamente.   |
| ERR_CFG_OUTOFBOUNDS | La variable del argumento \ListNo tiene un valor que está fuera del rango de instancias disponibles (0 ... n) al ejecutar la instrucción.   |

## Limitaciones

La conversión de las unidades de los programas de RAPID (mm, grados, segundos, etc.) a las unidades del parámetro del sistema (metros, radianes, segundos, etc.), que afecta a los CfgData del tipo de datos num y dnum debe ser realizada por el usuario en el programa de RAPID.

Para la mayoría de los parámetros del sistema, debe reiniciar manualmente el controlador o ejecutar la instrucción WarmStart para que el cambio tenga efecto. Los parámetros del sistema que pueden cambiarse de RobotStudio o FlexPendant sin un reinicio tampoco requieren un reinicio cuando se cambia de RAPID.

Si se usa el formato /DOMAIN/TYPE/ParameterName en InstancePath, sólo están disponibles los parámetros con nombre, es decir, los parámetros cuyo primer atributo sea name, Name o NAME.

Continúa en la página siguiente

# 1 Instrucciones

## 1.347 WriteCfgData - Escribe un atributo de un parámetro del sistema

RobotWare - OS

Continuación

Las cadenas de RAPID están limitadas a 80 caracteres. En algunos casos, puede ser en teoría una longitud demasiado reducida para la definición de InstancePath, Attribute o CfgData.

### Datos predefinidos

La constante predefinida END\_OF\_LIST, con valor -1, puede usarse para detener la escritura si no es posible encontrar más instancias.

### Sintaxis

```
WriteCfgData
  [ InstancePath ':=' ] < expression (IN) of string > ','
  [ Attribute ':=' ] < expression (IN) of string > ','
  [ CfgData ':=' ] < expression (IN) of anytype >
  [ '\ ' ListNo ':=' < variable (VAR) of num > ] ';' ;
```

### Información relacionada

| Para obtener más información sobre                 | Consulte   |
|--|--|
| Definición de cadena de caracteres                 | <a href="#">string - Cadenas en la página 1778</a>   |
| Lectura de un atributo de un parámetro del sistema | <a href="#">ReadCfgData - Lee un atributo de un parámetro del sistema en la página 582</a> |
| Obtención del nombre del robot de la tarea actual  | <a href="#">RobName - Obtiene el nombre del robot del TCP en la página 1458</a>            |
| Configuración                                      | <a href="#">Manual de referencia técnica - Parámetros del sistema</a>                      |
| Reinicio del sistema                               | <a href="#">WarmStart - Reinicio del controlador en la página 1093</a>                     |
| <i>Advanced RAPID</i>                              | <a href="#">Especificaciones del producto - Controller software IRC5</a>                   |

## 1.348 WriteRawBytes - Escribe un dato de tipo rawbytes

### Utilización

`WriteRawBytes` se usa para escribir datos de tipo `rawbytes` en un dispositivo abierto con `Open\Bin`.

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción `WriteRawBytes`:

#### Ejemplo 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;

```

En este ejemplo, `raw_data_out` se deja sin contenido y se empaqueta con el encabezado de `DeviceNet`, junto con un valor de coma flotante con el valor 0.2. Se abre un dispositivo, `"/FCI1:/dsqc328_1"`, y los datos válidos actuales de `raw_data_out` se escriben en el dispositivo. A continuación, el programa espera al menos 1 segundo antes de leer del dispositivo, almacenando en `raw_data_in` la información leída.

Después de cerrar el dispositivo `"/FCI1:/dsqc328_1"`, los datos leídos se desempaquetan dando lugar a una cadena de 10 caracteres que se almacena en `answer`.

### Argumentos

```
WriteRawBytes IODEV RawData [\NoOfBytes]
```

`IODEV`

**Tipo de dato:** `iODEV`

`IODEV` es el identificador del dispositivo en el que se debe escribir el dato `RawData`.

`RawData`

**Tipo de dato:** `rawbytes`

`RawData` el contenedor de datos que se debe escribir en `IODEV`.

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.348 WriteRawBytes - Escribe un dato de tipo rawbytes

RobotWare - OS

Continuación

[ \NoOfBytes ]

Tipo de dato: num

\NoOfBytes indica cuántos bytes de RawData deben escribirse en IODevice, empezando por el número de índice 1.

Si \NoOfBytes no está presente, se escribe en IODevice la longitud actual de bytes válidos de la variable RawData.

### Ejecución de programas

Durante la ejecución del programa, los datos se escriben en el dispositivo indicado por IODevice.

Si se utiliza WriteRawBytes junto con los comandos de bus de campo, por ejemplo DeviceNet, el bus de campo siempre envía una respuesta. La respuesta debe ser manejada en RAPID con la instrucción ReadRawBytes.

La longitud actual de los bytes válidos de la variable RawData no cambia.

En caso de un reinicio tras una caída de alimentación, todos los archivos o dispositivos de E/S abiertos del sistema se cierran y el descriptor de E/S de la variable del tipo iodev se restablece.

### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema ERRNO cambiará a:

| Nombre      | Causa del error                           |
|-------------|---|
| ERR_FILEACC | Se produjo un error durante la escritura. |

### Sintaxis

```
WriteRawBytes
  [ IODevice ':' ] < variable (VAR) of iodev> ', '
  [ RawData ':' ] < variable (VAR) of rawbytes>
  [ '\ ' NoOfBytes ':' ] < expression (IN) of num> ';'
```

### Información relacionada

| Para obtener más información sobre                              | Consulte   |
|---|--|
| rawbytes datos  | <a href="#">rawbytes - Datos sin formato en la página 1736</a>   |
| Obtención de la longitud de un dato rawbytes                    | <a href="#">RawBytesLen - Obtiene la longitud de un dato de tipo rawbytes en la página 1432</a>        |
| Borrado del contenido de un dato de tipo rawbytes               | <a href="#">ClearRawBytes - Borra el contenido de un dato de tipo rawbytes en la página 146</a>        |
| Copiado del contenido de un dato de tipo rawbytes               | <a href="#">CopyRawBytes - Copia el contenido de un dato de tipo rawbytes en la página 171</a>         |
| Empaquetamiento de un encabezado de DeviceNet en datos rawbytes | <a href="#">PackDNHeader - Empaqueta un encabezado de DeviceNet en datos rawbytes en la página 496</a> |
| Empaquetamiento de datos en datos rawbytes                      | <a href="#">PackRawBytes - Empaqueta datos en un dato de tipo rawbytes en la página 499</a>            |
| Lectura de un dato rawbytes                                     | <a href="#">ReadRawBytes - Lee datos de tipo rawbytes en la página 589</a>                             |

Continúa en la página siguiente

## 1.348 WriteRawBytes - Escribe un dato de tipo rawbytes

*RobotWare - OS*

*Continuación*

| Para obtener más información sobre              | Consulte  |
|---|---|
| Desempaquetamiento de datos de un dato rawbytes | <a href="#">UnpackRawBytes - Desempaqueta datos de un dato de tipo rawbytes en la página 1024</a> |
| Gestión de archivos y dispositivos de E/S       | <i>Application manual - Controller software IRC5</i>  |

# 1 Instrucciones

---

1.349 WriteStrBin - Escribe una cadena en un dispositivo de E/S binario  
RobotWare - OS

## 1.349 WriteStrBin - Escribe una cadena en un dispositivo de E/S binario

---

### Utilización

`WriteStrBin` (*Write String Binary*) se utiliza para escribir una cadena en un dispositivo de E/S o archivo binario.

---

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción `WriteStrBin`:

#### Ejemplo 1

```
WriteStrBin channel2, "Hello World\0A";
```

La cadena "Hello World\0A" se escribe en el canal al que se hace referencia con `channel2`. En este caso, la cadena termina con el carácter de salto de línea \0A. Todos los caracteres y valores hexadecimales escritos con `WriteStrBin` permanecen sin cambios en el sistema.

---

### Argumentos

```
WriteStrBin IODevice Str
```

IODevice

Tipo de dato: `iodev`

Nombre (referencia) del dispositivo de E/S actual.

Str

*String*

Tipo de dato: `string`

El texto a escribir.

---

### Ejecución de programas

La cadena de texto se escribe en el dispositivo de E/S o archivo especificado.

En caso de un reinicio tras una caída de alimentación, todos los archivos o dispositivos de E/S abiertos del sistema se cierran y el descriptor de E/S de la variable del tipo `iodev` se restablece.

---

### Limitaciones

Esta instrucción sólo puede usarse con dispositivos de E/S o archivos que hayan sido abiertos para lectura y escritura binaria.

---

### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                   | Causa del error                           |
|--------------------------|---|
| <code>ERR_FILEACC</code> | Se produjo un error durante la escritura. |

---

### Sintaxis

```
WriteStrBin  
[ IODevice '[:='] <variable (VAR) of iodev> ','
```

*Continúa en la página siguiente*

---

## 1.349 WriteStrBin - Escribe una cadena en un dispositivo de E/S binario

*RobotWare - OS*

*Continuación*

```
[ Str ':=' ] <expression (IN) of string> ';' 
```

### Información relacionada

| Para obtener más información sobre        | Consulte   |
|---|--|
| Apertura, etc. de dispositivos de E/S     | <i>Manual de referencia técnica - RAPID Overview</i>   |
| Lectura de cadenas binarias               | <a href="#">ReadStrBin - Lee una cadena de un dispositivo de E/S o archivo binario en la página 1449</a> |
| Gestión de archivos y dispositivos de E/S | <i>Application manual - Controller software IRC5</i>   |

# 1 Instrucciones

---

## 1.350 WriteVar - Escribir una variable

### Sensor Interface

## 1.350 WriteVar - Escribir una variable

---

### Utilización

`WriteVar` se utiliza para escribir una variable en un dispositivo que está conectado a la interfaz de sensores.

La interfaz de sensores se comunica con los sensores a través de dispositivos de E/S.

### Ejemplo de configuración

Éste es un ejemplo de configuración de un canal de sensor.

Estos parámetros corresponden al tipo *Transmission Protocol* del tema *Communication*.

| Name  | Type    | Remote Address  | Remote Port |
|-------|---------|-----------------|-------------|
| sen1: | SOCKDEV | 192.168.125.101 | 6344        |

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción `WriteVar`:

#### Ejemplo 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 measurements
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;
```

### Argumentos

```
WriteVar device VarNo VarData [ \TaskName ]
```

device

**Tipo de dato:** string

El nombre del dispositivo de E/S configurado en `sio.cfg` para el sensor utilizado.

VarNo

**Tipo de dato:** num

*Continúa en la página siguiente*



El argumento `VarNo` se utiliza para seleccionar la variable de sensor.

`VarData`

Tipo de dato: `num`

El argumento `VarData` define qué datos deben ser escritos en la variable seleccionada por el argumento `VarNo`.

[ `\TaskName` ]

Tipo de dato: `string`

El argumento `TaskName` hace posible el acceso a dispositivos de otras tareas de RAPID.

### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Constante de error (valor de ERRNO) | Descripción                            |
|-------------------------------------|--|
| <code>SEN_NO_MEAS</code>            | Fallo de medición                      |
| <code>SEN_NOREADY</code>            | Sensor incapaz de gestionar el comando |
| <code>SEN_GENERRO</code>            | Error general del sensor               |
| <code>SEN_BUSY</code>               | Bus de sensor                          |
| <code>SEN_UNKNOWN</code>            | Sensor desconocido                     |
| <code>SEN_EXALARM</code>            | Error de sensor externo                |
| <code>SEN_CAALARM</code>            | Error de sensor interno                |
| <code>SEN_TEMP</code>               | Error de temperatura del sensor        |
| <code>SEN_VALUE</code>              | Valor de comunicación no válido        |
| <code>SEN_CAMCHECK</code>           | Fallo de comprobación de sensor        |
| <code>SEN_TIMEOUT</code>            | Error de comunicación                  |

### Sintaxis

```
WriteVar
  [device ':='] <expression (IN) of string> ', '
  [VarNo ':='] <expression (IN) of num> ', '
  [VarData ':='] <expression (IN) of num> ', '
  ['\' TaskName ':='] <expression (IN) of string> ' ;'
```

### Información relacionada

| Para obtener más información sobre                         | Consulte   |
|--|--|
| Establecimiento de una conexión a un dispositivo de sensor | <a href="#">SenDevice - Establece una conexión a un dispositivo de sensor en la página 687</a> |
| Lectura de una variable de sensor                          | <a href="#">ReadVar - Lee una variable de un dispositivo en la página 1451</a>                 |
| Escritura de un bloque de datos de sensor                  | <a href="#">WriteBlock - Escribir un bloque de datos en un dispositivo en la página 1105</a>   |

Continúa en la página siguiente

# 1 Instrucciones

---

## 1.350 WriteVar - Escribir una variable

### *Sensor Interface*

#### *Continuación*

| Para obtener más información sobre          | Consulte  |
|---|---|
| Lectura de un bloque de datos de sensor     | <a href="#">ReadBlock - Lee un bloque de datos de un dispositivo en la página 580</a> |
| Configuración de la comunicación del sensor | <i>Manual de referencia técnica - Parámetros del sistema</i>                          |

## 1.351 WriteVarArr - Escribe múltiples variables en un dispositivo sensor

*Sensor Interface*

### 1.351 WriteVarArr - Escribe múltiples variables en un dispositivo sensor

#### Utilización

`WriteVarArr` Se utiliza para escribir hasta seis variables a la vez en un dispositivo sensor.

El sensor debe estar configurado y comunicarse a través de la opción *Sensor Interface* de RobotWare.

#### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción `WriteVarArr`:

##### Ejemplo 1

```
! Define variable numbers
CONST num jointno := 16;
CONST num unit := 19;
VAR sensorvardata writeData{3};

! Connect to the sensor device "sen1:" (defined in sio.cfg).
SenDevice "sen1:";

! Setup two variables to write
writeData{1}:=[jointno, 0, false, 1, 5];
writeData{2}:=[unit, 0, false, 1, 1];
! A varNumber of -1 will be ignored
writeData{3}:=[-1, 0, false, 1, 1];

WriteVarArr "sen1:", writeData;
```

El ejemplo muestra una solicitud de escritura de las variables `jointno` y `unit`.

#### Argumentos

```
WriteVarArr Device, Data, [\taskName]
```

Device

**Tipo de dato:** `string`

El nombre del dispositivo de E/S configurado en `sio.cfg` para el sensor utilizado.

Data

**Tipo de dato:** `sensorvardata`

Una variable matricial que hace referencia a la definición de datos de las variables que hay que escribir.

[ \TaskName ]

**Tipo de dato:** `string`

El argumento `TaskName` hace posible el acceso a dispositivos de otras tareas de RAPID.

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.351 WriteVarArr - Escribe múltiples variables en un dispositivo sensor

### Sensor Interface

#### Continuación

#### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre       | Causa del error                        |
|--------------|--|
| SEN_NO_MEAS  | Fallo de medición                      |
| SEN_NOREADY  | Sensor incapaz de gestionar el comando |
| SEN_GENERRO  | Error general del sensor               |
| SEN_BUSY     | Sensor ocupado                         |
| SEN_UNKNOWN  | Sensor desconocido                     |
| SEN_EXALARM  | Error de sensor externo                |
| SEN_CAALARM  | Error de sensor interno                |
| SEN_TEMP     | Error de temperatura del sensor        |
| SEN_VALUE    | Valor de comunicación no válido        |
| SEN_CAMCHECK | Fallo de comprobación de sensor        |
| SEN_TIMEOUT  | Error de comunicación                  |

#### Sintaxis

```
WriteVarArr
  [Device ':='] <expression(IN) of string> ','
  [Data ':='] < array variable {*} (INOUT) of sensorvardata > ','
  ['\ ' TaskName ':='] <expression (IN) of string> ] ';'

```

#### Información relacionada

| Para obtener más información sobre   | Consulte   |
|--|--|
| Establecimiento de una conexión a un dispositivo de sensor                 | <a href="#">SenDevice - Establece una conexión a un dispositivo de sensor en la página 687</a>                               |
| Leer múltiples variables de un dispositivo                                 | <a href="#">ReadVarArr - Lee múltiples variables de un dispositivo sensor en la página 592</a>                               |
| Escritura de una variable de sensor  | <a href="#">WriteVar - Escribir una variable en la página 1116</a>   |
| Lectura de una variable de sensor  | <a href="#">ReadVar - Lee una variable de un dispositivo en la página 1451</a>   |
| Escritura de un bloque de datos de sensor                                  | <a href="#">WriteBlock - Escribir un bloque de datos en un dispositivo en la página 1105</a>                                 |
| Lectura de un bloque de datos de sensor                                    | <a href="#">ReadBlock - Lee un bloque de datos de un dispositivo en la página 580</a>  |
| Configuración de múltiples variables de datos para la interfaz de sensores | <a href="#">sensorvardata - Configuración de múltiples variables de datos para la interfaz de sensores en la página 1757</a> |
| Configuración de la comunicación del sensor                                | <a href="#">Manual de referencia técnica - Parámetros del sistema</a>  |

## 1.352 WZBoxDef - Define una zona mundo con forma de prisma

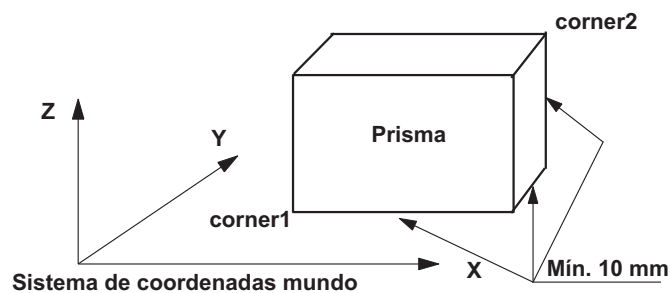
## Utilización

WZBoxDef (*World Zone Box Definition*) se usa para definir una zona mundo con forma de prisma recto cuyos ejes son paralelos a los ejes del sistema de coordenadas mundo.

## Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción WZBoxDef:

## Ejemplo 1



xx0500002205

```
VAR shapedata volume;
CONST pos corner1:=[200,100,100];
CONST pos corner2:=[600,400,400];
...
WZBoxDef \Inside, volume, corner1, corner2;
```

Define un prisma cuyas coordenadas son paralelas a los ejes del sistema de coordenadas mundo. Se define a partir de las esquinas opuestas `corner1` y `corner2`.

## Argumentos

WZBoxDef [`\Inside`] | [`\Outside`] Shape LowPoint HighPoint

[`\Inside`]

Tipo de dato: `switch`

Define el volumen que contiene el prisma.

[`\Outside`]

Tipo de dato: `switch`

Define el volumen que queda fuera del prisma (el volumen inverso).

Es imprescindible especificar uno de los argumentos `\Inside` o `\Outside`.

Shape

Tipo de dato: `shapedata`

Una variable para el almacenamiento del volumen definido (datos privados del sistema).

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.352 WZBoxDef - Define una zona mundo con forma de prisma

### World Zones

#### Continuación

LowPoint

Tipo de dato: pos

La posición (x,y,z) en mm que define una esquina inferior del prisma.

HighPoint

Tipo de dato: pos

La posición (x,y,z) en mm que define la esquina diagonalmente opuesta a la anterior.

### Ejecución de programas

La definición del prisma se almacena en la variable de tipo `shapedata` (el argumento `Shape`), para su uso futuro en instrucciones `WZLimSup` o `WZDOSet`.

### Limitaciones

Las posiciones de `LowPoint` y `HighPoint` deben ser válidas para las esquinas opuestas (con valores distintos de coordenadas x, y, z).

Si el robot se usa para apuntar hacia fuera de `LowPoint` o `HighPoint`, el objeto de trabajo `wobj0` debe estar activo (se usa el componente `trans` de `robtarget` por ejemplo `p1.trans` como argumento).

### Sintaxis

```
WZBoxDef
  [ ['\ ' Inside] | ['\ ' Outside] ', ' ]
  [ LowPoint ' := ' ] <expression (IN) of pos> ', '
  [ Shape ' := ' ] <variable (VAR) of shapedata> ', '
  [ HighPoint ' := ' ] <expression (IN) of pos> ';'
```

### Información relacionada

| Para obtener más información sobre                                     | Consulte   |
|--|--|
| World Zones  | <i>Manual de referencia técnica - RAPID Overview</i>   |
| Forma de las zonas mundo   | <a href="#">shapedata - Datos de forma de zonas mundo en la página 1759</a>  |
| Definición de zonas mundo esféricas                                    | <a href="#">WZSphDef - Define una zona mundo con forma esférica en la página 1148</a>                              |
| Definición de zonas mundo cilíndricas                                  | <a href="#">WZCylDef - Define una zona mundo con forma cilíndrica en la página 1123</a>                            |
| Definición de una zona mundo para las posiciones iniciales de los ejes | <a href="#">WZHomeJointDef - Define una zona mundo para las posiciones iniciales de los ejes en la página 1137</a> |
| Definición de una zona mundo para las posiciones límite de los ejes    | <a href="#">WZLimJointDef - Define una zona mundo para la limitación de los ejes en la página 1141</a>             |
| Activación de la supervisión de límites de las zonas mundo             | <a href="#">WZLimSup - Activa la supervisión de límites de las zonas mundo en la página 1145</a>                   |
| Activación de salidas digitales basadas en zonas mundo                 | <a href="#">WZDOSet - Activación de salidas digitales basadas en zonas mundo en la página 1128</a>                 |

## 1.353 WZCylDef - Define una zona mundo con forma cilíndrica

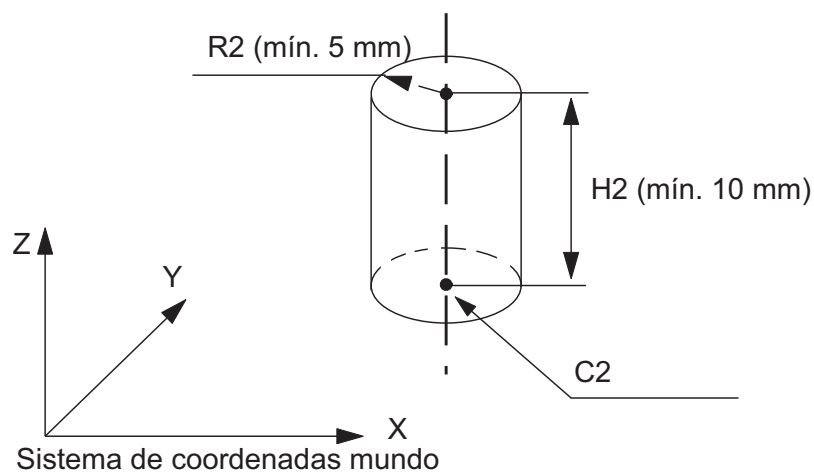
## Utilización

`WZCylDef` (*definición de zona mundo cilíndrica*) se usa para definir una zona mundo que tiene la forma de un cilindro con el eje del cilindro paralelo al eje z del sistema de coordenadas mundo.

## Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción `WZCylDef`:

## Ejemplo 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;
```

Define un cilindro con el centro de la circunferencia inferior en `C2`, el radio `R2` y la altura `H2`.

## Argumentos

`WZCylDef` [`\Inside`] | [`\Outside`] Shape CentrePoint Radius Height

[`\Inside`]

Tipo de dato: `switch`

Define el volumen que contiene el cilindro.

[`\Outside`]

Tipo de dato: `switch`

Define el volumen que queda fuera del cilindro (el volumen inverso).

Es imprescindible especificar uno de los argumentos `\Inside` o `\Outside`.

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.353 WZCylDef - Define una zona mundo con forma cilíndrica

### World Zones

#### Continuación

Shape

Tipo de dato: `shapedata`

Una variable para el almacenamiento del volumen definido (datos privados del sistema).

CentrePoint

Tipo de dato: `pos`

La posición (x,y,z) en mm que define el centro de un extremo circular del cilindro.

Radius

Tipo de dato: `num`

El radio del cilindro, en mm.

Height

Tipo de dato: `num`

La altura del cilindro, en mm. Si es positiva (dirección +z), el argumento `CentrePoint` es el centro del extremo inferior del cilindro (como en el ejemplo anterior). Si es negativa (dirección -z), el argumento `CentrePoint` es el centro del extremo superior del cilindro.

---

### Ejecución de programas

La definición del cilindro se almacena en la variable de tipo `shapedata` (el argumento `Shape`), para su uso futuro en instrucciones `WZLimSup` o `WZDOSet`.

---

### Limitaciones

Si el robot se usa para apuntar hacia fuera de `CentrePoint`, el objeto de trabajo `wobj0` debe estar activo (se usa el componente `trans` de `robtarget` por ejemplo `p1.trans` como argumento).

---

### Sintaxis

```
WZCylDef
  [ '\ ' Inside ] | [ '\ ' Outside ] ', '
  [ Shape := ] <variable (VAR) of shapedata> ', '
  [ CentrePoint := ] <expression (IN) of pos> ', '
  [ Radius := ] <expression (IN) of num> ', '
  [ Height := ] <expression (IN) of num> ';'
```

---

### Información relacionada

| Para obtener más información sobre           | Consulte   |
|--|--|
| World Zones                                  | <a href="#">Manual de referencia técnica - RAPID Overview</a>                          |
| Forma de las zonas mundo                     | <a href="#">shapedata - Datos de forma de zonas mundo en la página 1759</a>            |
| Definición de zonas mundo en forma de prisma | <a href="#">WZBoxDef - Define una zona mundo con forma de prisma en la página 1121</a> |
| Definición de zonas mundo esféricas          | <a href="#">WZSphDef - Define una zona mundo con forma esférica en la página 1148</a>  |

Continúa en la página siguiente



## 1.353 WZCylDef - Define una zona mundo con forma cilíndrica

*World Zones*

*Continuación*

| <b>Para obtener más información sobre</b>                              | <b>Consulte</b>   |
|--|---|
| Definición de una zona mundo para las posiciones iniciales de los ejes | <a href="#"><i>WZHomeJointDef - Define una zona mundo para las posiciones iniciales de los ejes en la página 1137</i></a> |
| Definición de una zona mundo para las posiciones límite de los ejes    | <a href="#"><i>WZLimJointDef - Define una zona mundo para la limitación de los ejes en la página 1141</i></a>             |
| Activación de la supervisión de límites de las zonas mundo             | <a href="#"><i>WZLimSup - Activa la supervisión de límites de las zonas mundo en la página 1145</i></a>                   |
| Activación de salidas digitales basadas en zonas mundo                 | <a href="#"><i>WZDOSet - Activación de salidas digitales basadas en zonas mundo en la página 1128</i></a>                 |

# 1 Instrucciones

---

## 1.354 WZDisable - Desactiva la supervisión de las zonas mundo temporales

*World Zones*

## 1.354 WZDisable - Desactiva la supervisión de las zonas mundo temporales

---

### Utilización

WZDisable (*World Zone Disable*) se utiliza para desactivar la supervisión de una zona mundo temporal, definida anteriormente para detener el movimiento o activar una salida.

---

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción WZDisable:

#### Ejemplo 1

```
VAR wztemporary wzone;
...
PROC...
  WZLimSup \Temp, wzone, volume;
  MoveL p_pick, v500, z40, tool1;
  WZDisable wzone;
  MoveL p_place, v200, z30, tool1;
ENDPROC
```

Al moverse hacia `p_pick`, se comprueba la posición del TCP del robot para que no entre dentro del volumen `wzone` especificado. Esta supervisión no se realiza cuando se va hacia `p_place`.

---

### Argumentos

WZDisable WorldZone

WorldZone

Tipo de dato: `wztemporary`

Una variable o una variable persistente de tipo `wztemporary` que contiene la identidad de la zona mundo que debe desactivarse.

---

### Ejecución de programas

La zona mundo temporal se desactiva. Esto significa que la supervisión del TCP del robot respecto del volumen correspondiente se detiene temporalmente. Puede reactivarse mediante la instrucción `WZEnable`.

---

### Limitaciones

Sólo es posible desactivar las zonas mundo temporales. Las zonas mundo estacionarias están siempre activadas.

---

### Sintaxis

```
WZDisable
  [ WorldZone '[:=' ] <variable or persistent (INOUT) of wztemporary>
  ;'
```

---

### Información relacionada

| Para obtener más información sobre | Consulte   |
|------------------------------------|--|
| World Zones                        | <i>Manual de referencia técnica - RAPID Overview</i> |

---

Continúa en la página siguiente

## 1.354 WZDisable - Desactiva la supervisión de las zonas mundo temporales

*World Zones*

*Continuación*

| Para obtener más información sobre                         | Consulte  |
|--|---|
| Forma de las zonas mundo                                   | <a href="#"><i>shapedata - Datos de forma de zonas mundo en la página 1759</i></a>                        |
| Datos de zonas mundo temporales                            | <a href="#"><i>wztemporary - Datos de zona mundo temporal en la página 1829</i></a>                       |
| Activación de la supervisión de límites de las zonas mundo | <a href="#"><i>WZLimSup - Activa la supervisión de límites de las zonas mundo en la página 1145</i></a>   |
| Activación de salidas digitales basadas en zonas mundo     | <a href="#"><i>WZDOSet - Activación de salidas digitales basadas en zonas mundo en la página 1128</i></a> |
| Activación de zonas mundo                                  | <a href="#"><i>WZEnable - Activa la supervisión de las zonas mundo temporales en la página 1133</i></a>   |
| Eliminación de zonas mundo                                 | <a href="#"><i>WZFree - Elimina la supervisión de las zonas mundo temporales en la página 1135</i></a>    |

# 1 Instrucciones

---

## 1.355 WZDOSet - Activación de salidas digitales basadas en zonas mundo *World Zones*

### 1.355 WZDOSet - Activación de salidas digitales basadas en zonas mundo

---

#### Utilización

WZDOSet (*World Zone Digital Output Set*) se utiliza para definir la acción y para activar una zona mundo para la supervisión de los movimientos del robot.

Después de la ejecución de esta instrucción, cuando el TCP del robot o los ejes del robot o los ejes externos (la zona de los ejes) se encuentra dentro de la zona mundo definida o se está acercando a ella, se establece una señal digital de salida con el valor especificado.

#### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción WZDOSet:  
Consulte también [Más ejemplos en la página 1130](#).

#### Ejemplo 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
```

Se define la zona mundo temporal *service* en el programa de aplicación. Dicha zona establece la señal *do\_service* cuando el TCP del robot se encuentra dentro de la esfera definida, durante la ejecución del programa o durante los desplazamientos.

#### Argumentos

```
WZDOSet [\Temp] | [\Stat] WorldZone [\Inside] | [\Before] Shape  
Signal SetValue
```

[\Temp]

#### *Temporary*

Tipo de dato: switch

La zona mundo a definir es una zona mundo temporal.

[\Stat]

#### *Stationary*

Tipo de dato: switch

La zona mundo a definir es una zona mundo estacionaria.

Es imprescindible especificar uno de los argumentos *\Temp* o *\Stat*.

WorldZone

Tipo de dato: wztemporary o wzstationary

*Continúa en la página siguiente*

## 1.355 WZDOSet - Activación de salidas digitales basadas en zonas mundo

*World Zones*

*Continuación*

Una variable o una variable persistente que se actualizará con la identidad (el valor numérico) de la zona mundo.

Si se usa el modificador `\Temp`, el tipo de dato debe ser `wztemporary`. Si se usa el modificador `\Stat`, el tipo de dato debe ser `wzstationary`.

`[\Inside]`

Tipo de dato: `switch`

La señal digital de salida se activa cuando el TCP del robot o los ejes especificados se encuentran dentro del volumen definido.

`[\Before]`

Tipo de dato: `switch`

La señal digital de salida se activa antes de que el TCP del robot o los ejes especificados alcancen el volumen definido (lo antes posible antes del volumen).

Es imprescindible especificar uno de los argumentos `\Inside` o `\Before`.

`Shape`

Tipo de dato: `shapedata`

La variable que define el volumen de la zona mundo.

`Signal`

Tipo de dato: `signaldo`

El nombre de la señal digital de salida que debe cambiar de valor.

Si se utiliza una zona mundo estacionaria, la señal debe tener el nivel de acceso *interno* correcto. En nivel de acceso se establece en la definición de señales en los parámetros del sistema de E/S. Estos conceptos se describen con más detalle en *Manual de referencia técnica - Parámetros del sistema*, consulte los tipos *Signal* y *Access Level*. El nivel de acceso debe estar protegido para el acceso del usuario (RAPID, FlexPendant); por lo tanto, puede utilizarse el nivel de acceso interno predefinido, o el usuario puede definir un nivel de acceso personalizado.

`SetValue`

Tipo de dato: `dionum`

El valor deseado para la señal (0 ó 1) cuando el TCP se encuentra dentro del volumen o sólo antes de que entre en el volumen.

Cuando está fuera del volumen o cerca de entrar en él, la señal cambia al valor opuesto.

---

### Ejecución de programas

Se activa la zona mundo definida. A partir de ese momento, se supervisa la posición del TCP del robot (o la posición de los ejes del robot o de los ejes externos). La salida se establece cuando la posición del TCP del robot (o la posición de los ejes del robot o de los ejes externos) se encuentra dentro del volumen (`\Inside`) o se acerca al borde del volumen (`\Before`).

Si se usa `WZHomeJointDef` o `WZLimJointDef` junto con `WZDOSet`, la señal digital se establece sólo si todos los ejes activos con supervisión de espacio de ejes se encuentran dentro del espacio de ejes o cerca de él.

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.355 WZDOSet - Activación de salidas digitales basadas en zonas mundo

### World Zones

#### Continuación

---

#### Más ejemplos

A continuación aparecen más ejemplos de cómo usar la instrucción WZDOSet.

#### Ejemplo 1

```
VAR wztemporary home;
VAR wztemporary service;
PERS wztemporary equip1:=[];

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;
...
! 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
```

Se definen las zonas mundo temporales `home` y `service` en el programa de aplicación para cambiar el valor de las señales `do_home` y `do_service`, cuando el robot está dentro de la esfera `home` o `service` respectivamente durante la ejecución del programa o durante los movimientos manuales.

También se define la zona mundo temporal `equip1`, que está activa sólo en la parte del programa durante la cual `equip1` se encuentra dentro del área de trabajo del robot. En ese momento el robot se para antes de entrar en el volumen `equip1`, tanto durante la ejecución del programa como durante el movimiento. `equip1` puede desactivarse o activarse desde otras tareas de programa usando el valor de la variable persistente `equip1`.

*Continúa en la página siguiente*

## Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                          | Causa del error   |
|---------------------------------|---|
| <code>ERR_NO_ALIASIO_DEF</code> | La variable de señal es una variable declarada en <code>RAPID</code> . No se ha conectado a una señal E/S definida en la configuración de E/S con la instrucción <code>AliasIO</code> .               |
| <code>ERR_NORUNUNIT</code>      | Se ha perdido el contacto con el dispositivo de E/S.  |
| <code>ERR_SIG_NOT_VALID</code>  | La señal de E/S no está disponible. Los motivos pueden ser que el dispositivo de E/S no está en funcionamiento o que exista un error en la configuración (válido solamente para el bus de campo ICI). |

## Limitaciones

Las zonas mundo no pueden redefinirse usando la misma variable en el argumento `WorldZone`.

Las zonas mundo estacionarias no pueden ser desactivadas, activadas de nuevo ni borrarse en el programa de `RAPID`.

Las zonas mundo temporales pueden ser desactivadas (`WZDisable`), activadas de nuevo (`WZEnable`) o borrarse (`WZFree`) en el programa de `RAPID`.

## Sintaxis

```
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> ' ; ' ]
```

## Información relacionada

| Para obtener más información sobre                                     | Consulte   |
|--|--|
| World Zones  | <i>Manual de referencia técnica - RAPID Overview</i>   |
| Forma de las zonas mundo   | <a href="#">shapedata - Datos de forma de zonas mundo en la página 1759</a>  |
| Zona mundo temporal  | <a href="#">wztemporary - Datos de zona mundo temporal en la página 1829</a>                                       |
| Zona mundo estacionaria  | <a href="#">wzstationary - Datos de zona mundo estacionaria en la página 1827</a>                                  |
| Definición de zonas mundo en forma de prisma recto                     | <a href="#">WZBoxDef - Define una zona mundo con forma de prisma en la página 1121</a>                             |
| Definición de zonas mundo esféricas                                    | <a href="#">WZSphDef - Define una zona mundo con forma esférica en la página 1148</a>                              |
| Definición de zonas mundo cilíndricas                                  | <a href="#">WZCylDef - Define una zona mundo con forma cilíndrica en la página 1123</a>                            |
| Definición de una zona mundo para las posiciones iniciales de los ejes | <a href="#">WZHomeJointDef - Define una zona mundo para las posiciones iniciales de los ejes en la página 1137</a> |

Continúa en la página siguiente

# 1 Instrucciones

---

## 1.355 WZDOSet - Activación de salidas digitales basadas en zonas mundo

*World Zones*

*Continuación*

| Para obtener más información sobre                         | Consulte   |
|--|--|
| Activación de la supervisión de límites de las zonas mundo | <a href="#">WZLimSup - Activa la supervisión de límites de las zonas mundo en la página 1145</a> |
| Nivel de acceso de las señales                             | <i>Manual de referencia técnica - Parámetros del sistema</i>                                     |



## 1.356 WZEnable - Activa la supervisión de las zonas mundo temporales *World Zones*

### 1.356 WZEnable - Activa la supervisión de las zonas mundo temporales

#### Utilización

WZEnable (*World Zone Enable*) se utiliza para reactivar la supervisión de una zona mundo temporal, definida anteriormente para detener el movimiento o activar una salida.

#### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción WZEnable:

#### Ejemplo 1

```
VAR wztemporary wzzone;  
...  
PROC ...  
  WZLimSup \Temp, wzzone, volume;  
  MoveL p_pick, v500, z40, tool1;  
  WZDisable wzzone;  
  MoveL p_place, v200, z30, tool1;  
  WZEnable wzzone;  
  MoveL p_home, v200, z30, tool1;  
ENDPROC
```

Al moverse hacia p\_pick, se comprueba la posición del TCP del robot para que no entre dentro del volumen wzzone especificado. Esta supervisión no se realiza cuando se va hacia p\_place, pero se reactiva antes del movimiento hacia p\_home.

#### Argumentos

WZEnable WorldZone

WorldZone

Tipo de dato: wztemporary

Una variable o una variable persistente de tipo wztemporary que contiene la identidad de la zona mundo que debe activarse.

#### Ejecución de programas

La zona mundo temporal se reactiva. Recuerde que las zonas mundo se activan automáticamente cuando se crean. Sólo es necesario reactivarlas cuando han sido desactivadas anteriormente mediante WZDisable.

#### Limitaciones

Sólo es posible desactivar y reactivar las zonas mundo temporales. Las zonas mundo estacionarias están siempre activadas.

#### Sintaxis

```
WZEnable  
  [ WorldZone ':='] <variable or persistent (INOUT) of wztemporary>  
  ;
```

*Continúa en la página siguiente*

# 1 Instrucciones

1.356 WZEnable - Activa la supervisión de las zonas mundo temporales

*World Zones*

*Continuación*

## Información relacionada

| Para obtener más información sobre                         | Consulte  |
|--|---|
| World Zones  | <i>Manual de referencia técnica - RAPID Overview</i>  |
| Forma de las zonas mundo                                   | <a href="#"><i>shapedata - Datos de forma de zonas mundo en la página 1759</i></a>                          |
| Datos de zonas mundo temporales                            | <a href="#"><i>wztemporary - Datos de zona mundo temporal en la página 1829</i></a>                         |
| Activación de la supervisión de límites de las zonas mundo | <a href="#"><i>WZLimSup - Activa la supervisión de límites de las zonas mundo en la página 1145</i></a>     |
| Activación de salidas digitales basadas en zonas mundo     | <a href="#"><i>WZDSet - Activación de salidas digitales basadas en zonas mundo en la página 1128</i></a>    |
| Desactivación de zonas mundo                               | <a href="#"><i>WZDisable - Desactiva la supervisión de las zonas mundo temporales en la página 1126</i></a> |
| Eliminación de zonas mundo                                 | <a href="#"><i>WZFree - Elimina la supervisión de las zonas mundo temporales en la página 1135</i></a>      |

---

## 1.357 WZFree - Elimina la supervisión de las zonas mundo temporales

---

### Utilización

WZFree (*World Zone Free*) se utiliza para eliminar la definición de una zona mundo temporal, definida anteriormente para detener el movimiento o activar una salida.

---

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción WZFree:

#### Ejemplo 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
```

Al moverse hacia p\_pick, se comprueba la posición del TCP del robot para que no entre dentro del volumen wzone especificado. Esta supervisión no se realiza cuando se va hacia p\_place, pero se reactiva antes del movimiento hacia p\_home. Cuando se alcanza esta posición, se elimina la definición de zona mundo.

---

### Argumentos

WZFree WorldZone

WorldZone

Tipo de dato: wztemporary

Una variable o una variable persistente de tipo wztemporary que contiene la identidad de la zona mundo que debe eliminarse.

---

### Ejecución de programas

La zona mundo temporal se desactiva primero, tras lo cual su definición se elimina. Después de eliminar una zona mundo temporal, no es posible reactivarla ni desactivarla.

---

### Limitaciones

Sólo es posible desactivar, reactivar o eliminar las zonas mundo temporales. Las zonas mundo estacionarias están siempre activadas.

---

### Sintaxis

```
WZFree
  [ WorldZone '[:=' ] <variable or persistent (INOUT) of wztemporary>
  ;'
```

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.357 WZFree - Elimina la supervisión de las zonas mundo temporales

*World Zones*

*Continuación*

### Información relacionada

| Para obtener más información sobre                         | Consulte  |
|--|---|
| World Zones  | <i>Manual de referencia técnica - RAPID Overview</i>  |
| Forma de las zonas mundo                                   | <a href="#"><i>shapedata - Datos de forma de zonas mundo en la página 1759</i></a>                          |
| Datos de zonas mundo temporales                            | <a href="#"><i>wztemporary - Datos de zona mundo temporal en la página 1829</i></a>                         |
| Activación de la supervisión de límites de las zonas mundo | <a href="#"><i>WZLimSup - Activa la supervisión de límites de las zonas mundo en la página 1145</i></a>     |
| Activación de salidas digitales basadas en zonas mundo     | <a href="#"><i>WZDSet - Activación de salidas digitales basadas en zonas mundo en la página 1128</i></a>    |
| Desactivación de zonas mundo                               | <a href="#"><i>WZDisable - Desactiva la supervisión de las zonas mundo temporales en la página 1126</i></a> |
| Activación de zonas mundo                                  | <a href="#"><i>WZEnable - Activa la supervisión de las zonas mundo temporales en la página 1133</i></a>     |

## 1.358 WZHomeJointDef - Define una zona mundo para las posiciones iniciales de los ejes *World Zones*

### 1.358 WZHomeJointDef - Define una zona mundo para las posiciones iniciales de los ejes

#### Utilización

WZHomeJointDef (*World Zone Home Joint Definition*) se utiliza para definir una zona mundo en las coordenadas de los ejes del robot y de los ejes externos para usarlas como posición INICIO o de SERVICIO.

#### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción WZHomeJointDef:

#### Ejemplo 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
```

**Definición y activación de la zona mundo estacionaria home, que cambia la señal do\_home to 1, cuando todos los ejes del robot y el eje externo extax.eax\_a se encuentran en la posición de ejes home\_pos (dentro de +/- delta\_pos para cada eje) durante la ejecución del programa y durante los desplazamientos. La variable joint\_space del tipo de dato shapedata se usa para transferir datos de la instrucción WZHomeJointDef a la instrucción WZDOSet.**

#### Argumentos

```
WZHomeJointDef [\Inside] | [\Outside] Shape MiddleJointVal
DeltaJointVal
```

[\Inside]

**Tipo de dato:** switch

**Define el espacio de ejes que existe dentro de MiddleJointVal +/- DeltaJointVal.**

[\Outside]

**Tipo de dato:** switch

**Define el espacio de ejes que existe fuera de MiddleJointVal +/- DeltaJointVal (espacio de ejes inverso).**

Shape

**Tipo de dato:** shapedata

**Una variable para el almacenamiento del espacio de ejes (datos privados del sistema).**

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.358 WZHomeJointDef - Define una zona mundo para las posiciones iniciales de los ejes

### World Zones

#### Continuación

MiddleJointVal

Tipo de dato: jointtarget

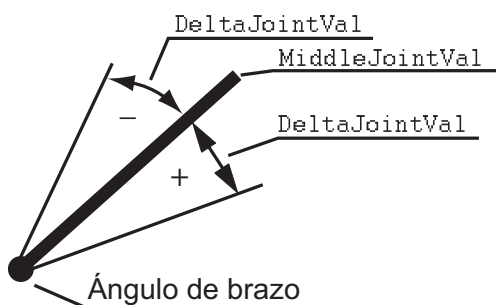
La posición en coordenadas de ejes para el centro del espacio de ejes que se desea definir. Especifica la posición para cada eje del robot y cada eje externo (grados en el caso de los ejes de rotación y mm en el caso de los ejes lineales). Especifica la posición en posiciones absolutas de los ejes (no dentro del sistema de coordenadas de desplazamiento `EOffsSet-EOffsOn` en el caso de los ejes externos). El valor 9E9 en un eje significa que el eje no se supervisará. Los ejes externos no activos también usan 9E9 en el momento de la programación.

DeltaJointVal

Tipo de dato: jointtarget

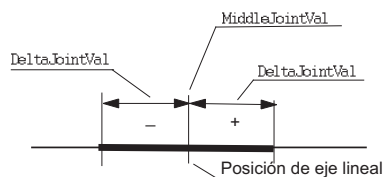
La variación de posición delta +/- en coordenadas de ejes desde el centro del espacio de ejes. El valor debe ser mayor que 0 en todos los ejes que se desee supervisar.

En la figura siguiente se muestra la definición de espacios de ejes para los ejes de rotación.



xx0500002208

En la figura siguiente se muestra la definición de espacios de ejes para los ejes lineales.



xx0500002209

## Ejecución de programas

La definición del espacio de ejes se almacena en la variable de tipo `shapedata` (el argumento `Shape`), para su uso futuro en instrucciones `WZLimSup` o `WZDOSet`.

Si se usa `WZHomeJointDef` junto con `WZDOSet`, la señal digital se establece, pero sólo si todos los ejes activos con supervisión de espacio de ejes se encuentran dentro del espacio de ejes o cerca de él.

Continúa en la página siguiente

## 1.358 WZHomeJointDef - Define una zona mundo para las posiciones iniciales de los ejes

*World Zones*  
*Continuación*

Si se usa `WZHomeJointDef` con un espacio exterior de ejes (argumento `\Outside`) junto con `WZLimSup`, se detiene el robot tan pronto como uno de los ejes activos con supervisión de espacio de ejes alcanza el espacio de ejes.

Si se usa `WZHomeJointDef` con un espacio interior de ejes (argumento `\Inside`) junto con `WZLimSup`, se detiene el robot tan pronto como el último eje activo con supervisión de espacio de ejes alcanza el espacio de ejes. Esto significa que uno o varios ejes, pero no todos los ejes activos y supervisados, se encontrarán dentro del espacio de ejes en un momento determinado.

En el momento de la ejecución de la instrucción `ActUnit` o `DeactUnit` para la activación o desactivación de unidades mecánicas, se actualiza el estado de supervisión de la posición HOME o la limitación del área de trabajo.

### Limitaciones



xx010000002

Sólo las unidades mecánicas activas y sus ejes activos en el momento de la activación de la zona mundo (con la instrucción `WZDOSet` o `WZLimSup`), se incluyen en la supervisión de la posición HOME respecto de la limitación del área de trabajo. Además, para que sean supervisados, la unidad mecánica y sus ejes deben seguir estando activos durante el movimiento del programa o mediante un movimiento especial.

Por ejemplo, si un eje con supervisión está fuera de la posición de ejes HOME pero está desactivado, esto no impide que la señal digital de salida de la posición de ejes HOME se establezca si todos los demás ejes activos con supervisión de espacio se encuentran dentro de la posición de ejes HOME. Cuando se activa de nuevo el eje, éste se incluye en la supervisión y el sistema de robot se encuentra fuera de la posición de ejes HOME. La salida digital se restablece.

### Sintaxis

```
WZHomeJointDef
[ ['\ ' Inside] | ['\ ' Outside] ', ' ]
[ Shape ' := ' ] <variable (VAR) of shapedata> ', '
[ MiddleJointVal ' := ' ] <expression (IN) of jointtarget> ', '
[ DeltaJointVal ' := ' ] <expression (IN) of jointtarget> ';'
```

### Información relacionada

| Para obtener más información sobre           | Consulte  |
|--|---|
| World Zones                                  | <i>Manual de referencia técnica - RAPID Overview</i>                                    |
| Forma de las zonas mundo                     | <a href="#">shapedata - Datos de forma de zonas mundo en la página 1759</a>             |
| Definición de zonas mundo en forma de prisma | <a href="#">WZBoxDef - Define una zona mundo con forma de prisma en la página 1121</a>  |
| Definición de zonas mundo cilíndricas        | <a href="#">WZCylDef - Define una zona mundo con forma cilíndrica en la página 1123</a> |

*Continúa en la página siguiente*

# 1 Instrucciones

---

1.358 WZHomeJointDef - Define una zona mundo para las posiciones iniciales de los ejes

*World Zones*

*Continuación*

| <b>Para obtener más información sobre</b>                           | <b>Consulte</b>   |
|---|---|
| Definición de zonas mundo esféricas                                 | <a href="#"><i>WZSphDef - Define una zona mundo con forma esférica en la página 1148</i></a>                  |
| Definición de una zona mundo para las posiciones límite de los ejes | <a href="#"><i>WZLimJointDef - Define una zona mundo para la limitación de los ejes en la página 1141</i></a> |
| Activación de la supervisión de límites de las zonas mundo          | <a href="#"><i>WZLimSup - Activa la supervisión de límites de las zonas mundo en la página 1145</i></a>       |
| Activación de salidas digitales basadas en zonas mundo              | <a href="#"><i>WZDOSet - Activación de salidas digitales basadas en zonas mundo en la página 1128</i></a>     |



## 1.359 WZLimJointDef - Define una zona mundo para la limitación de los ejes

### Utilización

WZLimJointDef (*World Zone Limit Joint Definition*) se utiliza para definir una zona mundo en las coordenadas de los ejes del robot y de los ejes externos para usarlas para la limitación del área de trabajo.

Con WZLimJointDef es posible limitar el área de trabajo de cada eje del robot y cada eje externo en el programa RAPID, aparte de la limitación que puede realizar con los parámetros del sistema *Motion - Arm - robx\_y - Upper Joint Bound ... Lower Joint Bound*.

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción WZLimJointDef:

#### Ejemplo 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
```

Definición y activación de la zona mundo estacionaria *work\_limit*, que limita el área de trabajo del eje del robot 1 a -90 y +90 grados y de los ejes externos *extax.eax\_a* a -1.000 mm durante la ejecución del programa y los movimientos. La variable *joint\_space* del tipo de dato *shapedata* se usa para transferir datos de la instrucción WZLimJointDef a la instrucción WZLimSup.

### Argumentos

WZLimJointDef [\Inside] | [\Outside] Shape LowJointVal HighJointVal

[\Inside]

Tipo de dato: switch

Define el espacio de ejes que existe dentro de LowJointVal ... HighJointVal.

[\Outside]

Tipo de dato: switch

Define el espacio de ejes que existe fuera de LowJointVal ... HighJointVal (espacio de ejes inverso).

Shape

Tipo de dato: shapedata

*Continúa en la página siguiente*

# 1 Instrucciones

## 1.359 WZLimJointDef - Define una zona mundo para la limitación de los ejes

### World Zones

#### Continuación

Una variable para el almacenamiento del espacio de ejes (datos privados del sistema).

LowJointVal

Tipo de dato: jointtarget

La posición en coordenadas de ejes para el límite inferior del espacio de ejes que se desea definir. Especifica la posición para cada eje del robot y cada eje externo (grados en el caso de los ejes de rotación y mm en el caso de los ejes lineales). Especifica la posición en posiciones absolutas de los ejes (no dentro del sistema de coordenadas de desplazamiento `EOffsSet` o `EOffsOn` en el caso de los ejes externos). El valor `9E9` en un eje significa que el eje no se supervisará en cuanto al límite inferior. Los ejes externos no activos también usan `9E9` en el momento de la programación.

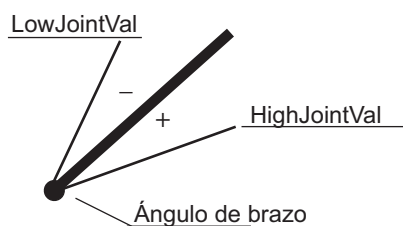
HighJointVal

Tipo de dato: jointtarget

La posición en coordenadas de ejes para el límite superior del espacio de ejes que se desea definir. Especifica la posición para cada eje del robot y cada eje externo (grados en el caso de los ejes de rotación y mm en el caso de los ejes lineales). Especifica la posición en posiciones absolutas de los ejes (no dentro del sistema de coordenadas de desplazamiento `EOffsSet` o `EOffsOn` en el caso de los ejes externos). El valor `9E9` en un eje significa que el eje no se supervisará en cuanto al límite superior. Los ejes externos no activos también usan `9E9` en el momento de la programación.

El valor de `HighJointVal` menos `LowJointVal` en cada eje debe ser mayor que 0 en todos los ejes que se desee supervisar.

En la figura siguiente se muestra la definición de espacios de ejes para los ejes de rotación.



xx0500002281

En la figura siguiente se muestra la definición de espacios de ejes para los ejes lineales.



xx0100000002

Continúa en la página siguiente

## 1.359 WZLimJointDef - Define una zona mundo para la limitación de los ejes World Zones Continuación

### Ejecución de programas

La definición del espacio de ejes se almacena en la variable de tipo `shapedata` (el argumento `Shape`), para su uso futuro en instrucciones `WZLimSup` o `WZDOSet`.

Si se usa `WZLimJointDef` junto con `WZDOSet`, la señal digital se establece sólo si todos los ejes activos con supervisión de espacio de ejes se encuentran dentro del espacio de ejes o cerca de él.

Si se usa `WZLimJointDef` con un espacio exterior de ejes (argumento `\Outside`) junto con `WZLimSup`, se detiene el robot tan pronto como uno de los ejes activos con supervisión de espacio de ejes alcanza el espacio de ejes.

Si se usa `WZLimJointDef` con un espacio interior de ejes (argumento `\Inside`) junto con `WZLimSup`, se detiene el robot tan pronto como el último eje activo con supervisión de espacio de ejes alcanza el espacio de ejes. Esto significa que uno o varios ejes, pero no todos los ejes activos y supervisados, se encontrarán dentro del espacio de ejes en un momento determinado.

En el momento de la ejecución de la instrucción `ActUnit` o `DeactUnit`, se actualiza el estado de la supervisión.

### Limitaciones



xx0100000002

#### ¡AVISO!

Sólo las unidades mecánicas activas y sus ejes activos en el momento de la activación de la zona mundo (con la instrucción `WZDOSet` o `WZLimSup`), se incluyen en la supervisión de la posición HOME respecto de la limitación del área de trabajo. Además, para que sean supervisados, la unidad mecánica y sus ejes deben seguir estando activos durante el movimiento del programa o mediante un movimiento especial.

Por ejemplo, si un eje con supervisión está fuera de la posición de ejes INICIO pero está desactivado, esto no impide que la señal digital de salida de la posición de ejes INICIO se establezca si todos los demás ejes activos con supervisión de espacio se encuentran dentro de la posición de ejes INICIO. Cuando se activa de nuevo el eje, éste se incluye en la supervisión y el sistema de robot se encuentra fuera de la posición de ejes INICIO. La salida digital se restablece.

### Sintaxis

```
WZLimJointDef
[ ['\ ' Inside] | ['\ ' Outside] ', ' ]
[ Shape ' := ' ] <variable (VAR) of shapedata> ', '
[ LowJointVal ' := ' ] <expression (IN) of jointtarget> ', '
[ HighJointVal ' := ' ] <expression (IN) of jointtarget> ';'
```

### Información relacionada

| Para obtener más información sobre | Consulte                                      |
|------------------------------------|---|
| World Zones                        | Manual de referencia técnica - RAPID Overview |

Continúa en la página siguiente

# 1 Instrucciones

## 1.359 WZLimJointDef - Define una zona mundo para la limitación de los ejes

### World Zones

#### Continuación

| Para obtener más información sobre                                     | Consulte   |
|--|--|
| Forma de las zonas mundo   | <a href="#">shapedata - Datos de forma de zonas mundo en la página 1759</a>  |
| Definición de zonas mundo en forma de prisma                           | <a href="#">WZBoxDef - Define una zona mundo con forma de prisma en la página 1121</a>                             |
| Definición de zonas mundo cilíndricas                                  | <a href="#">WZCylDef - Define una zona mundo con forma cilíndrica en la página 1123</a>                            |
| Definición de zonas mundo esféricas                                    | <a href="#">WZSphDef - Define una zona mundo con forma esférica en la página 1148</a>                              |
| Definición de una zona mundo para las posiciones iniciales de los ejes | <a href="#">WZHomeJointDef - Define una zona mundo para las posiciones iniciales de los ejes en la página 1137</a> |
| Activación de la supervisión de límites de las zonas mundo             | <a href="#">WZLimSup - Activa la supervisión de límites de las zonas mundo en la página 1145</a>                   |
| Activación de salidas digitales basadas en zonas mundo                 | <a href="#">WZDOSet - Activación de salidas digitales basadas en zonas mundo en la página 1128</a>                 |

## 1.360 WZLimSup - Activa la supervisión de límites de las zonas mundo

### Utilización

WZLimSup (*World Zone Limit Supervision*) se usa para definir la acción y para activar una zona mundo para la supervisión del área de trabajo del robot o de los ejes externos.

Después de la ejecución de esta instrucción, cuando el TCP llega a la zona mundo definida o cuando los ejes del robot o los ejes externos alcanzan la zona mundo definida para los ejes, el movimiento se detiene tanto durante la ejecución del programa como durante los movimientos.

### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción WZLimSup:  
Consulte también [Más ejemplos en la página 1146](#).

#### Ejemplo 1

```
VAR wzstationary max_workarea;
...
PROC POWER_ON()
  VAR shapedata volume;
  ...
  WZBoxDef \Outside, volume, corner1, corner2;
  WZLimSup \Stat, max_workarea, volume;
ENDPROC
```

Definición y activación de la zona mundo estacionaria max\_workarea, con la forma del área que queda fuera de un prisma (almacenado temporalmente en volume) y la supervisión del área de trabajo para la realización de acciones. El robot se detiene y muestra un mensaje de error antes de entrar en el área que queda fuera del prisma.

### Argumentos

WZLimSup [\Temp] | [\Stat] WorldZone Shape

[\Temp]

#### *Temporary*

Tipo de dato: switch

La zona mundo a definir es una zona mundo temporal.

[\Stat]

#### *Stationary*

Tipo de dato: switch

La zona mundo a definir es una zona mundo estacionaria.

Es imprescindible especificar uno de los argumentos \Temp o \Stat.

WorldZone

Tipo de dato: wztemporary o wzstationary

*Continúa en la página siguiente*

# 1 Instrucciones

---

## 1.360 WZLimSup - Activa la supervisión de límites de las zonas mundo

### World Zones

#### Continuación

Una variable o una variable persistente que se actualizará con la identidad (el valor numérico) de la zona mundo.

Si se usa el modificador `\Temp`, el tipo de dato debe ser `wztemporary`. Si se usa el modificador `\Stat`, el tipo de dato debe ser `wzstationary`.

Shape

Tipo de dato: `shapedata`

La variable que define el volumen de la zona mundo.

---

### Ejecución de programas

Se activa la zona mundo definida. A partir de ese momento, se supervisa la posición del TCP del robot o la posición de los ejes del robot o de los ejes externos. Si se alcanza el área definida, se detiene el movimiento.

Si se usa `WZLimJointDef` o `WZHomeJointDef` con un espacio exterior de ejes (argumento `\Outside`) junto con `WZLimSup`, se detiene el robot tan pronto como uno de los ejes activos con supervisión de espacio de ejes alcanza el espacio de ejes.

Si se usa `WZLimJointDef` o `WZHomeJointDef` con un espacio interior de ejes (argumento `\Inside`) junto con `WZLimSup`, se detiene el robot tan pronto como el último eje activo con supervisión de espacio de ejes alcanza el espacio de ejes. Esto significa que uno o varios ejes, pero no todos los ejes activos y supervisados, se encontrarán dentro del espacio de ejes en un momento determinado.

En el momento de la ejecución de la instrucción `ActUnit` o `DeactUnit`, se actualiza el estado de la supervisión.

---

### Más ejemplos

A continuación aparecen más ejemplos de cómo usar la instrucción `WZLimSup`.

#### Ejemplo 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
```

Limitación del área de trabajo para el robot con las siguientes zonas mundo estacionarias:

- Fuera del área de trabajo cuando se está fuera de `box1_invers`

*Continúa en la página siguiente*

## 1.360 WZLimSup - Activa la supervisión de límites de las zonas mundo

*World Zones*

*Continuación*

- Fuera del área de trabajo cuando se está dentro de `box2`

Si la rutina está conectada al evento de sistema POWER ON (ARRANQUE), estas zonas mundo estarán siempre activas en el sistema, tanto para los movimientos del programa como para el movimiento manual.

### Limitaciones

Las zonas mundo no pueden redefinirse usando la misma variable en el argumento `WorldZone`.

Las zonas mundo estacionarias no pueden ser desactivadas, activadas de nuevo ni borrarse en el programa de RAPID.

Las zonas mundo temporales pueden ser desactivadas (`WZDisable`), activadas de nuevo (`WZEnable`) o borrarse (`WZFree`) en el programa de RAPID.

### Sintaxis

```
WZLimSup
[ ['\ ' Temp] | ['\Stat] ', ' ]
[ WorldZone ' := ' ] <variable or persistent (INOUT) of wztemporary>
', '
[ Shape ' := ' ] <variable (VAR) of shapedata> ';'
```

### Información relacionada

| Para obtener más información sobre                                     | Consulte   |
|--|--|
| World Zones  | <i>Manual de referencia técnica - RAPID Overview</i>   |
| Forma de las zonas mundo   | <a href="#">shapedata - Datos de forma de zonas mundo en la página 1759</a>  |
| Zona mundo temporal  | <a href="#">wztemporary - Datos de zona mundo temporal en la página 1829</a>                                       |
| Zona mundo estacionaria  | <a href="#">wzstationary - Datos de zona mundo estacionaria en la página 1827</a>                                  |
| Definición de zonas mundo en forma de prisma recto                     | <a href="#">WZBoxDef - Define una zona mundo con forma de prisma en la página 1121</a>                             |
| Definición de zonas mundo esféricas                                    | <a href="#">WZSphDef - Define una zona mundo con forma esférica en la página 1148</a>                              |
| Definición de zonas mundo cilíndricas                                  | <a href="#">WZCylDef - Define una zona mundo con forma cilíndrica en la página 1123</a>                            |
| Definición de una zona mundo para las posiciones iniciales de los ejes | <a href="#">WZHomeJointDef - Define una zona mundo para las posiciones iniciales de los ejes en la página 1137</a> |
| Definición de una zona mundo para las posiciones límite de los ejes    | <a href="#">WZLimJointDef - Define una zona mundo para la limitación de los ejes en la página 1141</a>             |
| Activación de salidas digitales basadas en zonas mundo                 | <a href="#">WZDOSet - Activación de salidas digitales basadas en zonas mundo en la página 1128</a>                 |

# 1 Instrucciones

## 1.361 WZSphDef - Define una zona mundo con forma esférica *World Zones*

### 1.361 WZSphDef - Define una zona mundo con forma esférica

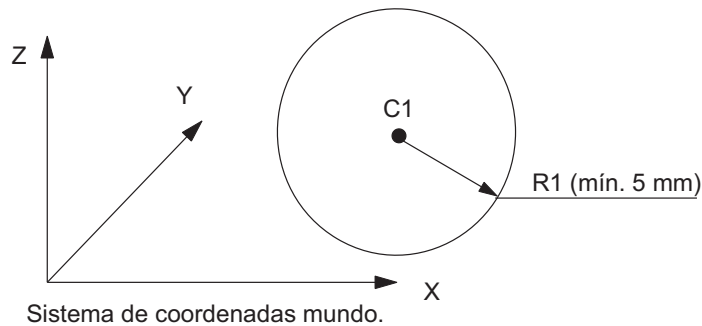
#### Utilización

WZSphDef (*World Zone Sphere Definition*) se usa para definir una zona mundo con la forma de una esfera.

#### Ejemplos básicos

El ejemplo que aparece a continuación ilustra la instrucción WZSphDef:

#### Ejemplo 1



xx0500002207

```
VAR shapedata volume;  
CONST pos C1:=[300,300,200];  
CONST num R1:=200;  
...  
WZSphDef \Inside, volume, C1, R1;
```

Se define una esfera con el nombre `volume`, con el centro `C1` y el radio `R1`.

#### Argumentos

WZSphDef [`\Inside`] | [`\Outside`] Shape CentrePoint Radius

[`\Inside`]

Tipo de dato: `switch`

Define el volumen que contiene la esfera.

[`\Outside`]

Tipo de dato: `switch`

Define el volumen que queda fuera de la esfera (el volumen inverso).

Es imprescindible especificar uno de los argumentos `\Inside` o `\Outside`.

Shape

Tipo de dato: `shapedata`

Una variable para el almacenamiento del volumen definido (datos privados del sistema).

CentrePoint

Tipo de dato: `pos`

*Continúa en la página siguiente*



## 1.361 WZSphDef - Define una zona mundo con forma esférica

World Zones

Continuación

La posición (x,y,z) en mm que define el centro de la esfera.

Radius

Tipo de dato: num

El radio de la esfera, en mm.

## Ejecución de programas

La definición de la esfera se almacena en la variable de tipo `shapedata` (el argumento `Shape`), para su uso futuro en instrucciones `WZLimSup` o `WZDOSet`.

## Limitaciones

Si el robot se usa para apuntar hacia fuera de `CentrePoint`, el objeto de trabajo `wobj0` debe estar activo (se usa el componente `trans` de `robtarget` por ejemplo `p1.trans` como argumento).

## Sintaxis

```
WZSphDef
  [ ['\ ' Inside] | ['\ ' Outside] ', ' ]
  [ Shape ' := ' ] <variable (VAR) of shapedata> ', '
  [ CentrePoint ' := ' ] <expression (IN) of pos> ', '
  [ Radius ' := ' ] <expression (IN) of num> ' ; '
```

## Información relacionada

| Para obtener más información sobre                                     | Consulte   |
|--|--|
| World Zones  | <a href="#">Manual de referencia técnica - RAPID Overview</a>  |
| Forma de las zonas mundo   | <a href="#">shapedata - Datos de forma de zonas mundo en la página 1759</a>  |
| Definición de zonas mundo en forma de prisma                           | <a href="#">WZBoxDef - Define una zona mundo con forma de prisma en la página 1121</a>                             |
| Definición de zonas mundo cilíndricas                                  | <a href="#">WZCylDef - Define una zona mundo con forma cilíndrica en la página 1123</a>                            |
| Definición de una zona mundo para las posiciones iniciales de los ejes | <a href="#">WZHomeJointDef - Define una zona mundo para las posiciones iniciales de los ejes en la página 1137</a> |
| Definición de una zona mundo para las posiciones límite de los ejes    | <a href="#">WZLimJointDef - Define una zona mundo para la limitación de los ejes en la página 1141</a>             |
| Activación de la supervisión de límites de las zonas mundo             | <a href="#">WZLimSup - Activa la supervisión de límites de las zonas mundo en la página 1145</a>                   |
| Activación de salidas digitales basadas en zonas mundo                 | <a href="#">WZDOSet - Activación de salidas digitales basadas en zonas mundo en la página 1128</a>                 |

**Esta página se ha dejado vacía intencionadamente**

## 2 Funciones

### 2.1 Abs - Obtiene el valor absoluto

#### Utilización

Abs se utiliza para obtener el valor absoluto, es decir, un valor positivo a partir de un dato numérico.

#### Ejemplos básicos

El ejemplo siguiente ilustra la función Abs.

Consulte también [Más ejemplos en la página 1151](#).

#### Ejemplo 1

```
reg1 := Abs(reg2);
```

Se asigna a Reg1 el valor absoluto de reg2.

#### Valor de retorno

Tipo de dato: num

El valor absoluto, es decir, un valor numérico positivo. Por ejemplo:

| Valor de entrada | Valor devuelto |
|------------------|----------------|
| 3                | 3              |
| -3               | 3              |
| -2.53            | 2.53           |

#### Argumentos

Abs (Value)

Value

Tipo de dato: num

El valor de entrada.

#### Más ejemplos

A continuación aparecen más ejemplos de la función Abs.

#### Ejemplo 1

```
TPReadNum no_of_parts, "How many parts should be produced? ";
no_of_parts := Abs(no_of_parts);
```

Se pregunta al operador cuántas piezas deben producirse. Para garantizar que el valor sea mayor que cero, se convierte en positivo el valor introducido por el operador.

#### Sintaxis

```
Abs '('
  [ Value ':= ' ] < expression (IN) of num > ')'
```

Una función con un valor de retorno del tipo de dato num.

*Continúa en la página siguiente*

## 2 Funciones

---

### 2.1 Abs - Obtiene el valor absoluto

*RobotWare - OS*

*Continuación*

---

#### Información relacionada

| Para obtener más información sobre    | Consulte   |
|---------------------------------------|--|
| Instrucciones y funciones matemáticas | <i>Manual de referencia técnica - RAPID Overview</i> |

## 2.2 AbsDnum - Obtiene el valor absoluto de un dnum

### Utilización

AbsDnum se utiliza para obtener el valor absoluto, es decir, un valor positivo a partir de un valor numérico dnum.

### Ejemplos básicos

El ejemplo siguiente ilustra la función AbsDnum.

Consulte también [Más ejemplos en la página 1153](#).

#### Ejemplo 1

```
VAR dnum value1;
VAR dnum value2:=-20000000;
value1 := AbsDnum(value2);
```

Se asigna a Value1 el valor absoluto de value2.

### Valor de retorno

Tipo de dato: dnum

El valor absoluto, es decir, un valor numérico positivo. Por ejemplo:

| Valor de entrada  | Valor devuelto   |
|-------------------|------------------|
| 3                 | 3                |
| -3                | -3               |
| -2.53             | 2.53             |
| -4503599627370496 | 4503599627370496 |

### Argumentos

AbsDnum (Value)

Value

Tipo de dato: dnum

El valor de entrada.

### Más ejemplos

A continuación aparecen más ejemplos de la función AbsDnum.

#### Ejemplo 1

```
TPReadDnum no_of_parts, "How many parts should be produced? ";
no_of_parts := AbsDnum(no_of_parts);
```

Se pregunta al operador cuántas piezas deben producirse. Para garantizar que el valor sea mayor que cero, se convierte en positivo el valor introducido por el operador.

### Sintaxis

```
AbsDnum '('
[ Value ':=' ] < expression (IN) of dnum > ')'
```

Una función con un valor de retorno del tipo de dato dnum.

*Continúa en la página siguiente*

## 2 Funciones

---

### 2.2 AbsDnum - Obtiene el valor absoluto de un dnum

*RobotWare - OS*

*Continuación*

---

#### Información relacionada

| Para obtener más información sobre    | Consulte   |
|---------------------------------------|--|
| Instrucciones y funciones matemáticas | <i>Manual de referencia técnica - RAPID Overview</i> |

## 2.3 ACos - Calcula el valor de arco coseno

### Utilización

ACos (*Arc Cosine*) se utiliza para calcular el valor de arco coseno en tipos de datos num.

### Ejemplos básicos

El ejemplo siguiente ilustra la función ACos.

#### Ejemplo 1

```
VAR num angle;
VAR num value;
...
...
angle := ACos(value);
```

angle obtiene el valor de arco coseno de value.

### Valor de retorno

Tipo de dato: num

El valor del arco coseno, expresado en grados, en el rango [0, 180].

### Argumentos

ACos (Value)

Value

Tipo de dato: num

El valor del argumento debe estar en el rango [-1, 1].

### Limitaciones

La ejecución de la función  $\text{ACos}(x)$  genera un error si  $x$  está fuera del rango [-1, 1].

### Sintaxis

```
ACos '('
  [Value ':=' ] <expression (IN) of num> ')'
```

Una función con un valor de retorno del tipo de dato num.

### Información relacionada

| Para obtener más información sobre    | Consulte   |
|---------------------------------------|--|
| Instrucciones y funciones matemáticas | <i>Manual de referencia técnica - RAPID Overview</i> |

## 2 Funciones

---

### 2.4 ACosDnum - Calcula el valor de arco coseno

RobotWare - OS

### 2.4 ACosDnum - Calcula el valor de arco coseno

---

#### Utilización

ACosDnum (*Arc Cosine dnum*) se utiliza para calcular el valor de arco coseno en tipos de datos dnum.

---

#### Ejemplos básicos

El ejemplo siguiente ilustra la función ACosDnum.

#### Ejemplo 1

```
VAR dnum angle;  
VAR dnum value;  
...  
...  
angle := ACosDnum(value);  
angle obtiene el valor de arco coseno de value.
```

---

#### Valor de retorno

Tipo de dato: dnum

El valor del arco coseno, expresado en grados, en el rango [0, 180].

---

#### Argumentos

ACosDnum (Value)

Value

Tipo de dato: dnum

El valor del argumento debe estar en el rango [-1, 1].

---

#### Limitaciones

La ejecución de la función ACosDnum(x) genera un error si x está fuera del rango [-1, 1].

---

#### Sintaxis

```
ACosDnum '('  
  [Value ':='] <expression (IN) of dnum>')'
```

Una función con un valor de retorno del tipo de dato dnum.

---

#### Información relacionada

| Para obtener más información sobre    | Consulte   |
|---------------------------------------|--|
| Instrucciones y funciones matemáticas | <i>Manual de referencia técnica - RAPID Overview</i> |



## 2.5 AInput - Lee el valor de una señal analógica de entrada

### Utilización

AInput se utiliza para leer el valor actual de una señal analógica de entrada.



#### Nota

Recuerde que la función AInput es una función antigua que ya no es necesario utilizar. Consulte los ejemplos relativos a una forma alternativa y recomendada de programar.

### Ejemplos básicos

El ejemplo siguiente ilustra la función AInput.

Consulte también [Más ejemplos en la página 1158](#).

#### Ejemplo 1

```
IF AInput(ail) < 1.5 THEN ...
...
IF ail < 1.5 THEN ...
```

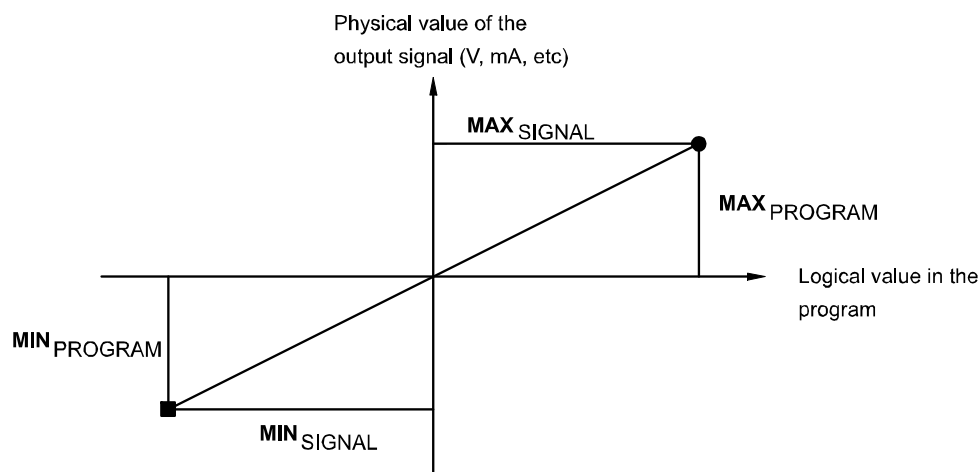
Si el valor actual de la señal ail es menor que 1,5, entonces ...

### Valor de retorno

Tipo de dato: num

El valor actual de la señal.

Al valor actual se le aplica una escala (acorde con los parámetros del sistema) antes de leerlo desde el programa de RAPID. En la figura que aparece a continuación, se muestra un diagrama de cómo se ajustan los valores de las señales analógicas a una escala.



xx0500002408

### Argumentos

AInput (Signal)

*Continúa en la página siguiente*

## 2 Funciones

---

### 2.5 AInput - Lee el valor de una señal analógica de entrada

RobotWare - OS

Continuación

Signal

Tipo de dato: signalai

El nombre de la entrada analógica que debe leerse.

---

#### Más ejemplos

A continuación aparecen más ejemplos de cómo usar la función AInput.

##### Ejemplo 1

```
WHILE AInput(current) > 35 DO ...  
...  
WHILE current > 35 DO ...
```

Siempre y cuando el valor actual de la señal `current` sea mayor que 35, ejecutar

...

##### Ejemplo 2

```
deviation := 3 * AInput(sensor) + 10;  
...  
deviation := 3 * sensor + 10;
```

La desviación se calcula a partir del valor de la señal `sensor` y se almacena en la variable `deviation`.

---

#### Sintaxis

```
AInput '('  
  [ Signal ':= ' ] < variable (VAR) of signalai > ')'
```

Una función con un valor de retorno del tipo de dato `num`.

---

#### Información relacionada

| Para obtener más información sobre         | Consulte   |
|--|--|
| Instrucciones de entrada/salida            | <i>Manual de referencia técnica - RAPID Overview, sección Resumen sobre RAPID - Señales de entrada y salida</i>  |
| Funcionalidad de entrada/salida en general | <i>Manual de referencia técnica - RAPID Overview, sección Principios de movimiento y E/S - Principios de E/S</i> |
| Configuración de E/S                       | <i>Manual de referencia técnica - Parámetros del sistema</i>   |

## 2.6 AND - Evalúa un valor lógico

### Utilización

AND es una función que se utiliza para evaluar dos expresiones condicionales (verdadero/falso).

### Ejemplos básicos

Los siguientes ejemplos ilustran la función AND.

#### Ejemplo 1

```
VAR num a;
VAR num b;
VAR bool c;
...
c := a>5 AND b=3;
```

El valor de retorno de `c` es TRUE si `a` es mayor de 5 y `b` es igual a 3. De lo contrario, el valor de retorno es FALSE.

#### Ejemplo 2

```
VAR num mynum;
VAR string mystring;
VAR bool mybool;
VAR bool result;
...
result := mystring="Hello" AND mynum<15 OR mybool;
```

El valor de retorno de `result` es TRUE si `mystring` es "Hello" y `mynum` es menor de 15. O bien si `mybool` es TRUE. De lo contrario, el valor de retorno es FALSE.

En primer lugar se evalúa la sentencia AND y, a continuación, la sentencia OR. Esto se ilustra mediante los paréntesis de la fila que aparece a continuación.

```
result := (mystring="Hello" AND mynum<15) OR mybool;
```

### Valor de retorno

Tipo de dato: `bool`

El valor de retorno es TRUE si ambas expresiones condicionales son correctas; de lo contrario, el valor de retorno es FALSE.

### Sintaxis

```
<expression of bool> AND <expression of bool>
```

Una función con un valor de retorno del tipo de dato `bool`.

### Información relacionada

| Para obtener más información sobre                        | Consulte  |
|---|---|
| Operación lógica AND bit a bit en datos <code>byte</code> | <a href="#">BitAnd - AND lógico bit a bit - Operación con datos de byte en la página 1173</a> |
| Operación lógica AND bit a bit en datos <code>dnum</code> | <a href="#">BitAndDnum - Operación lógica AND bit a bit en un dato dnum en la página 1175</a> |
| OR  | <a href="#">OR - Evalúa un valor lógico en la página 1394</a>                                 |

Continúa en la página siguiente

## 2 Funciones

---

### 2.6 AND - Evalúa un valor lógico

RobotWare - OS

Continuación

| Para obtener más información sobre | Consulte   |
|------------------------------------|--|
| XOR                                | <a href="#">XOR - Evalúa un valor lógico en la página 1605</a>   |
| NOT                                | <a href="#">NOT - Invierte un valor lógico en la página 1385</a> |
| Expresiones                        | <a href="#">Manual de referencia técnica - RAPID Overview</a>    |

## 2.7 AOutput - Lee el valor de una señal analógica de salida

### Utilización

AOutput se utiliza para obtener el valor actual de una señal analógica de salida.

### Ejemplos básicos

El ejemplo siguiente ilustra la función AOutput.

#### Ejemplo 1

```
IF AOutput(ao4) > 5 THEN ...
```

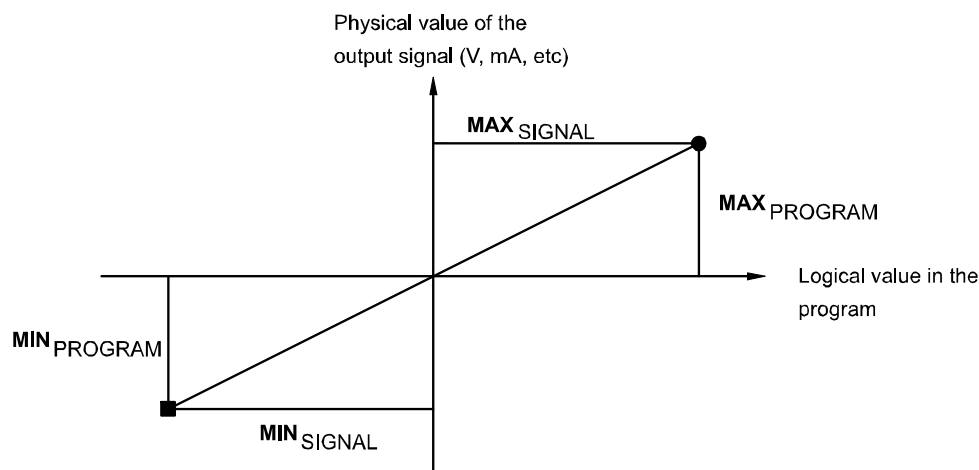
Si el valor actual de la señal ao4 es mayor que 5, entonces ...

### Valor de retorno

Tipo de dato: num

El valor actual de la señal.

Al valor actual se le aplica una escala (acorde con los parámetros del sistema) antes de leerlo desde el programa de RAPID. En la figura que aparece a continuación, se muestra un diagrama de cómo se ajustan los valores de las señales analógicas a una escala.



xx0500002408

### Argumentos

AOutput (Signal)

Signal

Tipo de dato: signalao

El nombre de la salida analógica que debe leerse.

*Continúa en la página siguiente*

## 2 Funciones

### 2.7 AOutput - Lee el valor de una señal analógica de salida

RobotWare - OS

Continuación

#### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                          | Causa del error   |
|---------------------------------|---|
| <code>ERR_NO_ALIASIO_DEF</code> | La variable de señal es una variable declarada en RAPID. No se ha conectado a una señal E/S definida en la configuración de E/S con la instrucción <code>AliasIO</code> .                             |
| <code>ERR_NORUNUNIT</code>      | Se ha perdido el contacto con el dispositivo de E/S.  |
| <code>ERR_SIG_NOT_VALID</code>  | La señal de E/S no está disponible. Los motivos pueden ser que el dispositivo de E/S no está en funcionamiento o que exista un error en la configuración (válido solamente para el bus de campo ICI). |

#### Sintaxis

```
AOutput '('  
  [ Signal ':'= ' ] < variable (VAR) of signalao > ')'
```

Una función con un valor de retorno del tipo de dato `num`.

#### Información relacionada

| Para obtener más información sobre               | Consulte   |
|--|--|
| Establecimiento de una señal analógica de salida | <a href="#">SetAO - Cambia el valor de una señal analógica de salida en la página 694</a>                        |
| Instrucciones de entrada/salida                  | <i>Manual de referencia técnica - RAPID Overview, sección Resumen sobre RAPID - Señales de entrada y salida</i>  |
| Funcionalidad de entrada/salida en general       | <i>Manual de referencia técnica - RAPID Overview, sección Principios de movimiento y E/S - Principios de E/S</i> |
| Configuración de E/S                             | <i>Manual de referencia técnica - Parámetros del sistema</i>   |

---

## 2.8 ArgName - Obtiene el nombre de un argumento

---

### Utilización

`ArgName` (*Argument Name*) se utiliza para obtener el nombre del objeto de datos original del argumento o el dato actual.

---

### Ejemplos básicos

El ejemplo siguiente ilustra la función `ArgName`.

Consulte también [Más ejemplos en la página 1164](#).

#### Ejemplo 1

```
VAR num chales :=5;
...
procl chales;
PROC procl (num parl)
  VAR string name;
  ...
  name:=ArgName(parl);
  TPWrite "Argument name "+name+" with value "\Num:=parl;
ENDPROC
```

La variable `name` se asigna al valor de cadena "chales" y en FlexPendant se escribe la siguiente cadena: "Argument name chales with value 5".

---

### Valor de retorno

Tipo de dato: `string`

El nombre del objeto de datos original.

---

### Argumentos

`ArgName` (Parameter [`\ErrorNumber`])

#### Parameter

Tipo de dato: `anytype`

El identificador formal del parámetro (en la rutina en la que se encuentra `ArgName`) o la identidad de los datos.

Puede usar todos los tipos de datos con estructura atómica o de registro, componente de registro, matriz o elemento de matriz.

#### ErrorNumber

Tipo de dato: `errnum`

Una variable (antes de usarla es cambiada a 0 por el sistema) que contendrá el código de error si el argumento es un valor de expresión, el argumento no está presente o el argumento es del tipo switch. Si se omite esta variable opcional, se ejecuta el gestor de errores.

*Continúa en la página siguiente*

## 2 Funciones

---

### 2.8 ArgName - Obtiene el nombre de un argumento

RobotWare - OS

Continuación

---

#### Ejecución de programas

La función devuelve el nombre del objeto de datos original de todo un objeto de tipo constante, variable o variable persistente. El objeto de datos original puede ser global, local del módulo de programa o local de una rutina (se aplican las reglas de ámbito habituales de RAPID).

Si forma parte de un objeto de datos, se devuelve el nombre del objeto de datos en su conjunto.

---

#### Más ejemplos

A continuación aparecen más ejemplos de la función ArgName.

#### Conversión de un identificador en una cadena

Esta función también puede usarse para convertir un identificador en un valor string, especificando el identificador en el argumento `Parameter` de cualquier objeto de datos global, local del módulo o local del ámbito de la rutina:

```
VAR num chales :=5;
...
proc1;

PROC proc1 ()
  VAR string name;
  ...
  name:=ArgName(chales);
  TPWrite "Global data object "+name+" has value "\Num:=chales;
ENDPROC
```

La variable `name` recibe el valor de cadena "chales". En el FlexPendant se escribe la cadena siguiente: "El objeto de datos global chales tiene el valor 5".

#### Llamada a la rutina en varios pasos

Recuerde que la función devuelve el nombre original del objeto de datos:

```
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
```

*Continúa en la página siguiente*



La variable `name` recibe el valor de cadena "chales". En el FlexPendant se escribe la cadena siguiente: "Nombre original del objeto de datos chales con el valor 5".

#### Suprimir la ejecución en el gestor de errores

```
PROC main()
  VAR string mystring:="DUMMY";
  procl mystring;
  procl "This is a test";
  ...
ENDPROC

PROC procl (string parl)
  VAR string name;
  VAR errnum myerrnum;

  name := ArgName(parl \ErrorNumber:=myerrnum);
  IF myerrnum=ERR_ARGNAME THEN
    TPWrite "The argument parl 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
```

La variable `name` recibe el valor de cadena "mystring" al realizarse la primera llamada a `procl`. Al realizarse la segunda llamada a `procl`, se asigna una cadena vacía al nombre. En el FlexPendant se escribe la siguiente cadena: "The argument parl is an expression value" y "The name of the argument can not be evaluated".

#### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre      | Causa del error  |
|-------------|--|
| ERR_ARGNAME | <ul style="list-style-type: none"> <li>El argumento es un valor de expresión.</li> <li>El argumento no está presente.</li> <li>El argumento es del tipo switch.</li> </ul> |

#### Sintaxis

```
ArgName '('
  [ Parameter ':=' ] < reference (REF) of any type>
  ['\' ErrorNumber ':=' <var or pers (INOUT) of errnum> ] ')'
```

Una función con un valor de retorno del tipo de dato `string`.

#### Información relacionada

| Para obtener más información sobre   | Consulte  |
|--------------------------------------|---|
| Funciones para cadenas de caracteres | <a href="#">Manual de referencia técnica - RAPID Overview</a> |
| Definición de cadena de caracteres   | <a href="#">string - Cadenas en la página 1778</a>            |

Continúa en la página siguiente

## 2 Funciones

---

### 2.8 ArgName - Obtiene el nombre de un argumento

*RobotWare - OS*

*Continuación*

| Para obtener más información sobre | Consulte   |
|------------------------------------|--|
| Valores de cadena de caracteres    | <i>Manual de referencia técnica - RAPID Overview</i> |
| <i>Advanced RAPID</i>              | <i>Advanced RAPID</i>                                |

## 2.9 ASin - Calcula el valor de arco seno

### Utilización

ASin (*Arc Sine*) se utiliza para calcular el valor de arco seno en tipos de datos num.

### Ejemplos básicos

El ejemplo siguiente ilustra la función ASin.

#### Ejemplo 1

```
VAR num angle;
VAR num value;
...
...
angle := ASin(value);
angle will get the arc sine value of value
```

### Valor de retorno

Tipo de dato: num

El valor del arco seno, expresado en grados, en el rango [-90, 90].

### Argumentos

ASin (Value)

Value

Tipo de dato: num

El valor del argumento debe estar en el rango [-1, 1].

### Limitaciones

La ejecución de la función ASin(x) genera un error si x está fuera del rango [-1, 1].

### Sintaxis

```
ASin '('
  [Value ':=' ] <expression (IN) of num> ')'
```

Una función con un valor de retorno del tipo de dato num.

### Información relacionada

| Para obtener más información sobre    | Consulte   |
|---------------------------------------|--|
| Instrucciones y funciones matemáticas | <i>Manual de referencia técnica - RAPID Overview</i> |

## 2 Funciones

---

### 2.10 ASinDnum - Calcula el valor de arco seno

RobotWare - OS

### 2.10 ASinDnum - Calcula el valor de arco seno

---

#### Utilización

ASinDnum (*Arc Sine dnum*) se utiliza para calcular el valor de arco seno en tipos de datos dnum.

---

#### Ejemplos básicos

El ejemplo siguiente ilustra la función ASinDnum

##### Ejemplo 1

```
VAR dnum angle;  
VAR dnum value;  
...  
...  
angle := ASinDnum(value);  
angle will get the arc sine value of value
```

---

#### Valor de retorno

Tipo de dato: dnum

El valor del arco seno, expresado en grados, en el rango [-90, 90].

---

#### Argumentos

ASinDnum (Value)

Value

Tipo de dato: dnum

El valor del argumento debe estar en el rango [-1, 1].

---

#### Limitaciones

La ejecución de la función ASinDnum(x) genera un error si x está fuera del rango [-1, 1].

---

#### Sintaxis

```
ASinDnum '('  
[Value ':=' ] <expression (IN) of dnum> ')'
```

Una función con un valor de retorno del tipo de dato dnum.

---

#### Información relacionada

| Para obtener más información sobre    | Consulte   |
|---------------------------------------|--|
| Instrucciones y funciones matemáticas | <i>Manual de referencia técnica - RAPID Overview</i> |

## 2.11 ATan - Calcula el valor de arco tangente

### Utilización

ATan (*Arc Tangent*) se utiliza para calcular el valor de arco tangente en tipos de datos num.

### Ejemplos básicos

El ejemplo siguiente ilustra la función ATan.

#### Ejemplo 1

```
VAR num angle;
VAR num value;
...
...
angle := ATan(value);
angle obtiene el valor de arco tangente de value.
```

### Valor de retorno

Tipo de dato: num

El valor del arco tangente, expresado en grados, en el rango [-90, 90].

### Argumentos

ATan (Value)

Value

Tipo de dato: num

El valor del argumento.

### Sintaxis

```
ATan '('
  [Value ':='] <expression (IN) of num> ')'
```

Una función con un valor de retorno del tipo de dato num.

### Información relacionada

| Para obtener más información sobre                       | Consulte  |
|--|---|
| Instrucciones y funciones matemáticas                    | <i>Manual de referencia técnica - RAPID Overview</i>                          |
| Arco tangente con un valor devuelto en el rango [°, 180] | <a href="#">ATan2 - Calcula el valor de arco tangente 2 en la página 1171</a> |

## 2 Funciones

---

### 2.12 ATanDnum - Calcula el valor de arco tangente

RobotWare - OS

### 2.12 ATanDnum - Calcula el valor de arco tangente

---

#### Utilización

ATanDnum (*Arc Tangent dnum*) se utiliza para calcular el valor de arco tangente en tipos de datos dnum.

---

#### Ejemplos básicos

El ejemplo siguiente ilustra la función ATanDnum.

#### Ejemplo 1

```
VAR dnum angle;  
VAR dnum value;  
...  
...  
angle := ATanDnum(value);  
angle obtiene el valor de arco tangente de value.
```

---

#### Valor de retorno

Tipo de dato: dnum

El valor del arco tangente, expresado en grados, en el rango [-90, 90].

---

#### Argumentos

ATanDnum (Value)

Value

Tipo de dato: dnum

El valor del argumento.

---

#### Sintaxis

```
ATanDnum '('  
[Value ':=' ] <expression (IN) of dnum> ')'
```

Una función con un valor de retorno del tipo de dato dnum.

---

#### Información relacionada

| Para obtener más información sobre                        | Consulte  |
|---|---|
| Instrucciones y funciones matemáticas                     | <i>Manual de referencia técnica - RAPID Overview</i>                          |
| Arco tangente con un valor devuelto en el rango [° , 180] | <a href="#">ATan2 - Calcula el valor de arco tangente 2 en la página 1171</a> |

## 2.13 ATan2 - Calcula el valor de arco tangente 2

### Utilización

ATan2 (*Arc Tangent2*) se utiliza para calcular el valor de arco tangente 2 en tipos de datos num.

### Ejemplos básicos

El ejemplo siguiente ilustra la función ATan2.

#### Ejemplo 1

```
VAR num angle;
VAR num x_value;
VAR num y_value;
...
...
angle := ATan2(y_value, x_value);
```

angle **obtiene el valor de arco tangente de** y\_value/x\_value.

### Valor de retorno

Tipo de dato: num

El valor del arco tangente, expresado en grados, en el rango [-180, 180]. El valor será igual a ATan(y/x), pero en el rango [-180, 180], ya que la función utiliza el signo de los dos argumentos para determinar el cuadrante del valor de retorno.

### Argumentos

ATan2 (Y X)

Y

Tipo de dato: num

El valor del argumento numerador.

X

Tipo de dato: num

El valor del argumento denominador.

### Sintaxis

```
ATan2 '('
  [Y ':=' ] <expression (IN) of num> ','
  [X ':=' ] <expression (IN) of num> ')'
```

Una función con un valor de retorno del tipo de dato num.

### Información relacionada

| Para obtener más información sobre    | Consulte   |
|---------------------------------------|--|
| Instrucciones y funciones matemáticas | <i>Manual de referencia técnica - RAPID Overview</i>                       |
| Arco tangente con un solo argumento   | <a href="#">ATan - Calcula el valor de arco tangente en la página 1169</a> |

## 2 Funciones

### 2.14 ATan2Dnum - Calcula el valor de arco tangente 2

RobotWare - OS

### 2.14 ATan2Dnum - Calcula el valor de arco tangente 2

#### Utilización

ATan2Dnum (*Arc Tangent2 dnum*) se utiliza para calcular el valor de arco tangente 2 en tipos de datos dnum.

#### Ejemplos básicos

El ejemplo siguiente ilustra la función ATan2Dnum.

#### Ejemplo 1

```
VAR dnum angle;  
VAR dnum x_value;  
VAR dnum y_value;  
...  
...  
angle := ATan2Dnum(y_value, x_value);  
angle obtiene el valor de arco tangente de y_value/x_value.
```

#### Valor de retorno

Tipo de dato: dnum

El valor del arco tangente, expresado en grados, en el rango [-180, 180]. El valor será igual a ATanDnum(y/x), pero en el rango [-180, 180], ya que la función utiliza el signo de los dos argumentos para determinar el cuadrante del valor de retorno.

#### Argumentos

ATan2Dnum (Y X)

Y

Tipo de dato: dnum

El valor del argumento numerador.

X

Tipo de dato: dnum

El valor del argumento denominador.

#### Sintaxis

```
ATan2Dnum '(' '  
  [Y ':=' ] <expression (IN) of dnum> ',' '  
  [X ':=' ] <expression (IN) of dnum> ')' '
```

Una función con un valor de retorno del tipo de dato dnum.

#### Información relacionada

| Para obtener más información sobre    | Consulte   |
|---------------------------------------|--|
| Instrucciones y funciones matemáticas | <a href="#">Manual de referencia técnica - RAPID Overview</a>              |
| Arco tangente con un solo argumento   | <a href="#">ATan - Calcula el valor de arco tangente en la página 1169</a> |



## 2.15 BitAnd - AND lógico bit a bit - Operación con datos de byte

### Utilización

`BitAnd` se utiliza para ejecutar una operación lógica bit a bit `AND` en tipos de datos `byte`.

### Ejemplos básicos

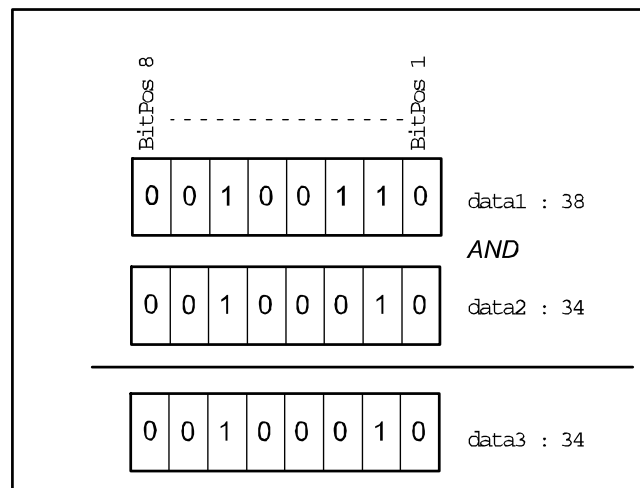
El ejemplo siguiente ilustra la función `BitAnd`.

#### Ejemplo 1

```
VAR byte data1 := 38;
VAR byte data2 := 34;
VAR byte data3;

data3 := BitAnd(data1, data2);
```

Se ejecuta la operación lógica `AND` bit a bit (consulte la figura que aparece a continuación) en `data1` y `data2`. El resultado se devuelve en `data3` (representación entera).



xx0500002454

### Valor de retorno

Tipo de dato: `byte`

El resultado de la operación lógica `AND` bit a bit en representación entera.

### Argumentos

`BitAnd (BitData1 BitData2)`

`BitData1`

Tipo de dato: `byte`

El dato de bit 1, en representación entera.

`BitData2`

Tipo de dato: `byte`

*Continúa en la página siguiente*

## 2 Funciones

### 2.15 BitAnd - AND lógico bit a bit - Operación con datos de byte

RobotWare - OS

Continuación

El dato de bit 2, en representación entera.

#### Limitaciones

El rango de los tipos de datos `byte` es de 0 a 255.

#### Sintaxis

```
BitAnd '('  
  [BitData1 ':=' ] <expression (IN) of byte> ','  
  [BitData2 ':=' ] <expression (IN) of byte> ')'
```

Una función con un valor de retorno del tipo de dato `byte`.

#### Información relacionada

| Para obtener más información sobre                      | Consulte   |
|---|--|
| OR lógico bit a bit - Operación con datos de byte       | <a href="#">BitOr - OR lógico bit a bit - Operación con datos de byte en la página 1190</a>        |
| XOR lógico bit a bit - Operación con datos de byte      | <a href="#">BitXOr - XOR lógico bit a bit - Operación con datos de byte en la página 1198</a>      |
| NEGACIÓN lógica bit a bit - Operación con datos de byte | <a href="#">BitNeg - NEGACIÓN lógica bit a bit - Operación con datos de byte en la página 1186</a> |
| Otras funciones de bits                                 | <i>Manual de referencia técnica - RAPID Overview</i>   |
| <i>Advanced RAPID</i>                                   | <i>Especificaciones del producto - Controller software IRC5</i>                                    |

## 2.16 BitAndDnum - Operación lógica AND bit a bit en un dato dnum

### Utilización

`BitAndDnum` se utiliza para ejecutar una operación lógica AND bit a bit en los tipos de datos `dnum`.

### Ejemplos básicos

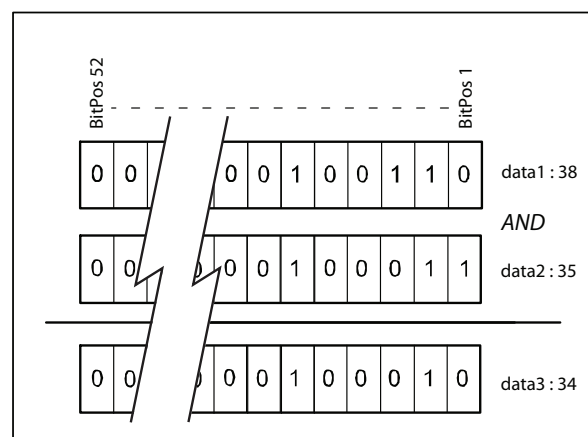
El ejemplo siguiente ilustra la función `BitAndDnum`.

#### Ejemplo 1

```
VAR dnum data1 := 38;
VAR dnum data2 := 35;
VAR dnum data3;

data3 := BitAndDnum(data1, data2);
```

Se ejecuta la operación lógica AND bit a bit (consulte la figura siguiente) con los datos `data1` y `data2`. El resultado se devuelve en `data3` (representación entera).



xx120000007

### Valor de retorno

Tipo de dato: `dnum`

El resultado de la operación lógica AND bit a bit en representación entera.

### Argumentos

`BitAndDnum (Value1 Value2)`

Value1

Tipo de dato: `dnum`

El primer valor de datos de bits, en representación entera.

Value2

Tipo de dato: `dnum`

El segundo valor de datos de bits, en representación entera.

*Continúa en la página siguiente*

## 2 Funciones

### 2.16 BitAndDnum - Operación lógica AND bit a bit en un dato dnum

RobotWare - OS

Continuación

#### Limitaciones

El rango de los tipos de datos `dnum` es de 0 - 4503599627370495.

#### Sintaxis

```
BitAndDnum '('  
  [Value1 ':=' ] <expression (IN) of dnum> ','  
  [Value2 ':=' ] <expression (IN) of dnum> ')'
```

Una función con un valor de retorno del tipo de dato `dnum`.

#### Información relacionada

| Para obtener más información sobre   | Consulte   |
|--|--|
| Operación lógica AND bit a bit en datos <code>byte</code>                  | <a href="#">BitAnd - AND lógico bit a bit - Operación con datos de byte en la página 1173</a>                |
| Tipo de dato <code>dnum</code>   | <a href="#">dnum - Valores numéricos dobles en la página 1660</a>  |
| Operación lógica OR bit a bit en datos <code>dnum</code>                   | <a href="#">BitOrDnum - Operación lógica OR bit a bit en datos dnum en la página 1192</a>                    |
| Operación lógica XOR bit a bit en datos <code>dnum</code>                  | <a href="#">BitXOrDnum - Operación lógica XOR bit a bit en datos dnum en la página 1200</a>                  |
| Operación lógica NEGACIÓN bit a bit - operación en datos <code>dnum</code> | <a href="#">BitNegDnum - Operación lógica NEGACIÓN bit a bit - operación en datos dnum en la página 1188</a> |
| Otras funciones de bits  | <a href="#">Manual de referencia técnica - RAPID Overview</a>  |

## 2.17 BitCheck - Comprueba si un bit especificado de un dato de byte está activado

## Utilización

BitCheck se utiliza para comprobar si un bit especificado de un dato de `byte` definido tiene el valor 1.

## Ejemplos básicos

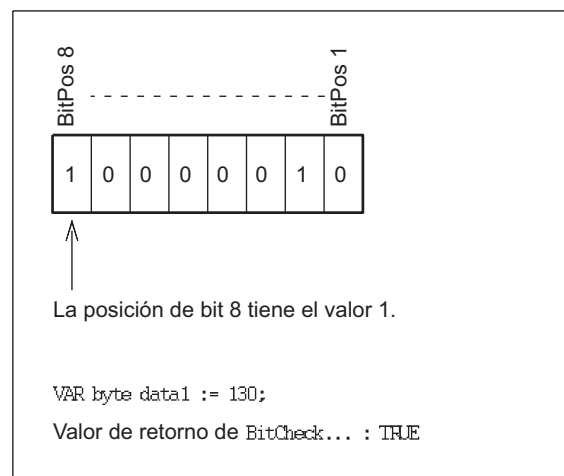
El ejemplo siguiente ilustra la función BitCheck.

## Ejemplo 1

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

IF BitCheck(data1, parity_bit) = TRUE THEN
    ...
ELSE
    ...
ENDIF
```

Se comprueba el número de bit 8 (`parity_bit`) de la variable `data1`; por ejemplo, si el bit especificado es 1 en la variable `data1`, esta función volverá a `TRUE`. La comprobación de bits de un dato de tipo `byte` se muestra en la figura que aparece a continuación.



xx0500002442

## Valor de retorno

Tipo de dato: `bool`

`TRUE` si el bit especificado tiene el valor 1, `FALSE` si el bit especificado tiene el valor 0.

## Argumentos

BitCheck (BitData BitPos)

BitData

Tipo de dato: `byte`

*Continúa en la página siguiente*

## 2 Funciones

---

### 2.17 BitCheck - Comprueba si un bit especificado de un dato de byte está activado

RobotWare - OS

Continuación

Los datos de bits a comprobar, en representación entera.

BitPos

*Bit Position*

Tipo de dato: num

El valor de la posición de bit (de 1 a 8) de BitData a comprobar.

---

#### Limitaciones

El rango de los tipos de datos byte es de 0 a 255 en decimal.

Las posiciones de bit válidas son de la 1 a la 8.

---

#### Sintaxis

```
BitCheck '('  
    [BitData ':=' ] <expression (IN) of byte> ','  
    [BitPos ':=' ] <expression (IN) of num> ')'
```

Una función con un valor de retorno del tipo de dato bool.

---

#### Información relacionada

| Para obtener más información sobre                    | Consulte  |
|---|---|
| Activación de un bit específico de un dato de byte    | <a href="#">BitSet - Activa un bit específico de un dato byte o dnum en la página 47</a>      |
| Desactivación de un bit específico de un dato de byte | <a href="#">BitClear - Desactiva un bit específico de un dato byte o dnum en la página 44</a> |
| Otras funciones de bits                               | <i>Manual de referencia técnica - RAPID Overview</i>  |
| <i>Advanced RAPID</i>                                 | <i>Especificaciones del producto - Controller software IRC5</i>                               |

## 2.18 BitCheckDnum - Comprueba si un bit especificado de un dato dnum está activado

### Utilización

BitCheckDnum se utiliza para comprobar si un bit especificado de un dato de dnum definido tiene el valor 1.

### Ejemplos básicos

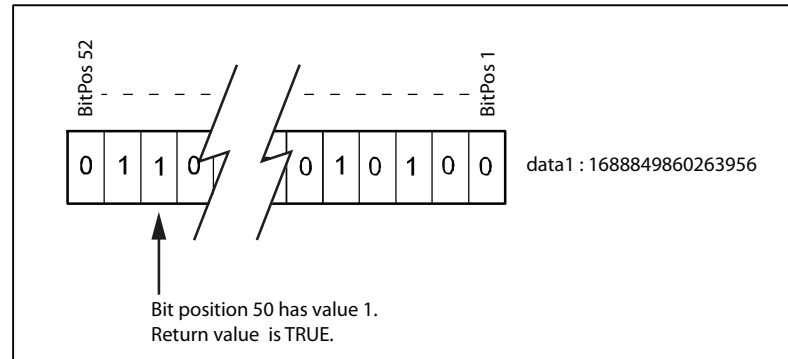
El ejemplo siguiente ilustra la función BitCheckDnum.

#### Ejemplo 1

```
CONST num check_bit := 50;
VAR dnum data1 := 1688849860263956;

IF BitCheckDnum(data1, check_bit) = TRUE THEN
    ...
ELSE
    ...
ENDIF
```

Se comprueba el número de bit 50 (check\_bit) de la variable data1; por ejemplo, si el bit especificado es 1 en la variable, data1 esta función vuelve a TRUE. La comprobación de bits de un dato de tipo dnum se muestra en la figura que aparece a continuación.



xx1200000016

### Valor de retorno

Tipo de dato: `bool`

TRUE si el bit especificado tiene el valor 1, FALSE si el bit especificado tiene el valor 0.

### Argumentos

BitCheckDnum (Value BitPos)

Value

Tipo de dato: `dnum`

Los datos de bits a comprobar, en representación entera.

*Continúa en la página siguiente*

## 2 Funciones

### 2.18 BitCheckDnum - Comprueba si un bit especificado de un dato dnum está activado

RobotWare - OS

Continuación

BitPos

**Bit Position**

Tipo de dato: num

El valor de la posición de bit (1-52) de Value a comprobar.

#### Limitaciones

El rango de los tipos de datos dnum es de 0 - 4503599627370495 en decimal.

Las posiciones de bit válidas son de la 1 a la 52.

#### Sintaxis

```
BitCheckDnum '('  
  [Value ':=' ] <expression (IN) of dnum> ','  
  [BitPos ':=' ] <expression (IN) of num> ')'
```

Una función con un valor de retorno del tipo de dato bool.

#### Información relacionada

| Para obtener más información sobre                                 | Consulte   |
|--|--|
| Comprobación de si un bit específico de un dato byte está activado | <a href="#">BitCheck - Comprueba si un bit especificado de un dato de byte está activado en la página 1177</a> |
| Tipo de dato dnum  | <a href="#">dnum - Valores numéricos dobles en la página 1660</a>  |
| Activación de un bit específico de un dato byte o dnum             | <a href="#">BitSet - Activa un bit específico de un dato byte o dnum en la página 47</a>                       |
| Desactivación de un bit específico de un dato byte o dnum          | <a href="#">BitClear - Desactiva un bit específico de un dato byte o dnum en la página 44</a>                  |
| Otras funciones de bits  | <a href="#">Manual de referencia técnica - RAPID Overview</a>  |



## 2.19 BitLSh - DESPLAZAMIENTO A LA IZQUIERDA lógico bit a bit - Operación de byte

### Utilización

BitLSh (*Bit Left Shift*) se utiliza para ejecutar una operación lógica de DESPLAZAMIENTO A LA IZQUIERDA bit a bit en tipos de datos `byte`.

### Ejemplos básicos

El ejemplo siguiente ilustra la función BitLSh.

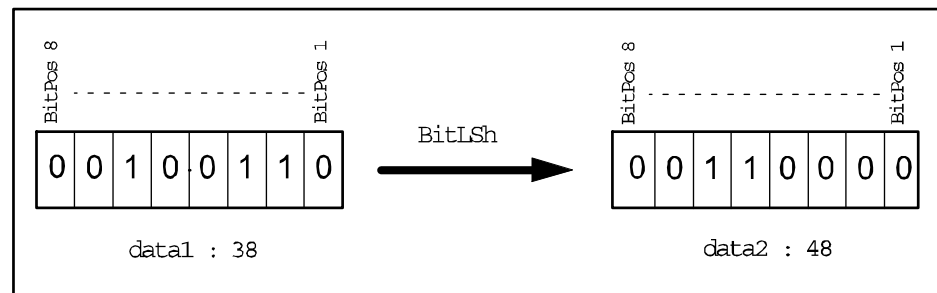
#### Ejemplo 1

```
VAR num left_shift := 3;
VAR byte data1 := 38;
VAR byte data2;

data2 := BitLSh(data1, left_shift);
```

Se ejecuta la operación lógica DESPLAZAMIENTO A LA IZQUIERDA bit a bit con el dato `data1`, con 3 pasos (`left_shift`) de desplazamiento hacia la izquierda. El resultado se devuelve en `data2` (representación entera).

En la figura siguiente se muestra la operación DESPLAZAMIENTO A LA IZQUIERDA lógico bit a bit.



xx0500002457

### Valor de retorno

Tipo de dato: `byte`

El resultado de la operación lógica DESPLAZAMIENTO A LA IZQUIERDA bit a bit en representación entera.

Las posiciones de bit de la derecha se rellenan con bits 0.

### Argumentos

BitLSh (BitData ShiftSteps)

BitData

Tipo de dato: `byte`

Los datos de bits a desplazar, en representación entera.

ShiftSteps

Tipo de dato: `num`

El número de desplazamientos lógicos (de 1 a 8) que se desea ejecutar.

*Continúa en la página siguiente*

## 2 Funciones

### 2.19 BitLSh - DESPLAZAMIENTO A LA IZQUIERDA lógico bit a bit - Operación de byte

RobotWare - OS

Continuación

#### Limitaciones

El rango de los tipos de datos `byte` es de 0 a 255.

El valor válido del argumento `ShiftSteps` es un número de 1 a 8, lo que corresponde a un byte.

#### Sintaxis

```
BitLSh '('  
  [BitData ':='] <expression (IN) of byte> ','  
  [ShiftSteps ':='] <expression (IN) of num> ')'
```

Una función con un valor de retorno del tipo de dato `byte`.

#### Información relacionada

| Para obtener más información sobre  | Consulte  |
|---|---|
| DESPLAZAMIENTO A LA DERECHA lógico bit a bit - Operación con datos de bit | <a href="#">BitRSh - DESPLAZAMIENTO A LA DERECHA lógico bit a bit - Operación de byte en la página 1194</a> |
| Otras funciones de bits   | <i>Manual de referencia técnica - RAPID Overview</i>  |
| <i>Advanced RAPID</i>   | <i>Especificaciones del producto - Controller software IRC5</i>   |

## 2.20 BitLShDnum - Operación lógica DESPLAZAMIENTO A LA IZQUIERDA bit a bit en dato dnum

### Utilización

BitLShDnum (*Bit Left Shift dnum*) se utiliza para ejecutar una operación lógica de DESPLAZAMIENTO A LA IZQUIERDA bit a bit en tipos de datos dnum.

### Ejemplos básicos

El ejemplo siguiente ilustra la función BitLShDnum.

Consulte también [Más ejemplos en la página 1184](#).

#### Ejemplo 1

```

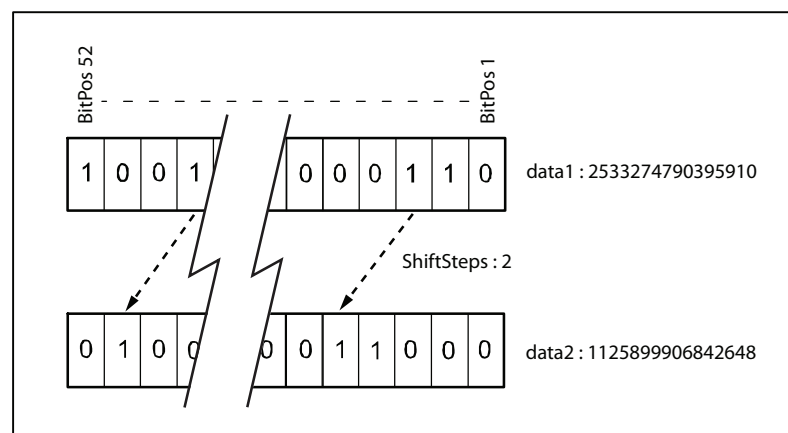
VAR num left_shift := 2;
VAR dnum data1 := 2533274790395910;
VAR dnum data2;

data2 := BitLShDnum(data1, left_shift);

```

Se ejecuta la operación lógica DESPLAZAMIENTO A LA IZQUIERDA bit a bit con el dato data1, con 2 pasos (left\_shift) de desplazamiento hacia la izquierda. El resultado se devuelve en data2 (representación entera).

En la figura siguiente se muestra la operación DESPLAZAMIENTO A LA IZQUIERDA lógico bit a bit.



xx120000008

### Valor de retorno

Tipo de dato: dnum

El resultado de la operación lógica DESPLAZAMIENTO A LA IZQUIERDA bit a bit en representación entera.

Las posiciones de bit de la derecha se rellenan con bits 0.

### Argumentos

BitLShDnum (Value ShiftSteps [\Size])

Value

Tipo de dato: dnum

*Continúa en la página siguiente*

## 2 Funciones

### 2.20 BitLShDnum - Operación lógica DESPLAZAMIENTO A LA IZQUIERDA bit a bit en dato dnum

RobotWare - OS

Continuación

Los datos de bits a desplazar, en representación entera.

ShiftSteps

Tipo de dato: num

El número de desplazamientos lógicos (de 1 a 52) que se desea ejecutar.

Size

Tipo de dato: num

El tamaño (número de bits) que deben considerarse al realizar la operación lógica DESPLAZAMIENTO A LA IZQUIERDA bit a bit en el argumento Value. Los tamaños válidos son de 1 a 52.

#### Limitaciones

El rango de los tipos de datos dnum es de 0 - 4503599627370495.

Los valores válidos del argumento ShiftSteps son del 1 al 52 dado que un dnum tiene 52 bits.

#### Más ejemplos

A continuación aparecen más ejemplos de la función BitLshDnum.

#### Ejemplo 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;
```

Se ejecuta la operación lógica DESPLAZAMIENTO A LA IZQUIERDA bit a bit con el data1 y el resultado se devuelve en result (representación entera). El primer valor escrito en el FlexPendant es 208. El segundo valor escrito en el FlexPendant es 3536.

#### Sintaxis

```
BitLshDnum '('  
  [Value ':=' ] <expression (IN) of dnum> ','  
  [ShiftSteps ':=' ] <expression (IN) of num>  
  ['\ ' Size ':=' < expression (IN) of num>  
  ')'
```

Una función con un valor de retorno del tipo de dato dnum.

#### Información relacionada

| Para obtener más información sobre   | Consulte  |
|--|---|
| DESPLAZAMIENTO A LA IZQUIERDA lógico bit a bit - Operación con datos de byte | <a href="#">BitLsh - DESPLAZAMIENTO A LA IZQUIERDA lógico bit a bit - Operación de byte en la página 1181</a> |
| Tipo de dato dnum  | <a href="#">dnum - Valores numéricos dobles en la página 1660</a>   |

Continúa en la página siguiente

### 2.20 BitLShDnum - Operación lógica DESPLAZAMIENTO A LA IZQUIERDA bit a bit en dato dnum

*RobotWare - OS*

*Continuación*

| Para obtener más información sobre   | Consulte   |
|--|--|
| DESPLAZAMIENTO A LA DERECHA lógico bit a bit - Operación con datos de dnum | <a href="#">BitRShDnum - Operación lógica DESPLAZAMIENTO A LA DERECHA bit a bit en un dnum en la página 1196</a> |
| Otras funciones de bits  | <i>Manual de referencia técnica - RAPID Overview</i>   |

## 2 Funciones

### 2.21 BitNeg - NEGACIÓN lógica bit a bit - Operación con datos de byte

RobotWare - OS

### 2.21 BitNeg - NEGACIÓN lógica bit a bit - Operación con datos de byte

#### Utilización

BitNeg (*Bit Negation*) se utiliza para ejecutar una operación lógica NEGATION - bit a bit (complemento de uno) en los tipos de datos `byte`.

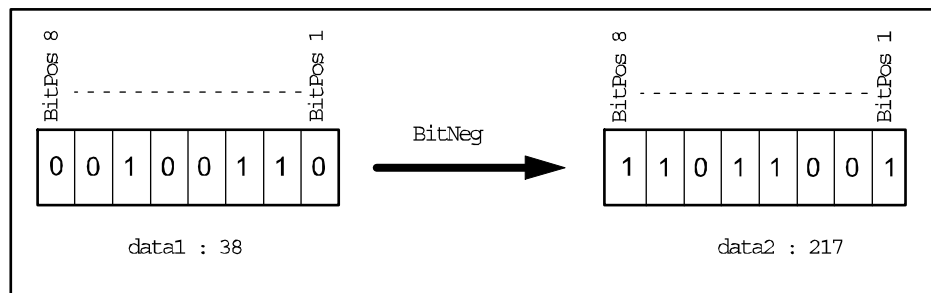
#### Ejemplos básicos

El ejemplo siguiente ilustra la función BitNeg.

#### Ejemplo 1

```
VAR byte data1 := 38;  
VAR byte data2;  
  
data2 := BitNeg(data1);
```

Se ejecuta la operación lógica NEGATION bit a bit (consulte la figura siguiente) con los datos `data1`. El resultado se devuelve en `data2` (representación entera).



xx0500002456

#### Valor de retorno

Tipo de dato: `byte`

El resultado de la operación lógica NEGATION bit a bit en representación entera.

#### Argumentos

BitNeg (BitData)

BitData

Tipo de dato: `byte`

El dato de byte, en representación entera.

#### Limitaciones

El rango de los tipos de datos `byte` es de 0 a 255.

#### Sintaxis

```
BitNeg '('  
  [BitData ':='] <expression (IN) of byte>  
  ')'
```

Una función con un valor de retorno del tipo de dato `byte`.

Continúa en la página siguiente

## Información relacionada

| Para obtener más información sobre                 | Consulte  |
|--|---|
| AND lógico bit a bit - Operación con datos de byte | <a href="#">BitAnd - AND lógico bit a bit - Operación con datos de byte en la página 1173</a> |
| OR lógico bit a bit - Operación con datos de byte  | <a href="#">BitOr - OR lógico bit a bit - Operación con datos de byte en la página 1190</a>   |
| XOR lógico bit a bit - Operación con datos de byte | <a href="#">BitXOr - XOR lógico bit a bit - Operación con datos de byte en la página 1198</a> |
| Otras funciones de bits                            | Manual de referencia técnica - <i>RAPID Overview</i>  |
| <i>Advanced RAPID</i>                              | Especificaciones del producto - <i>Controller software IRC5</i>                               |

## 2 Funciones

### 2.22 BitNegDnum - Operación lógica NEGACIÓN bit a bit - operación en datos dnum RobotWare - OS

### 2.22 BitNegDnum - Operación lógica NEGACIÓN bit a bit - operación en datos dnum

#### Utilización

BitNegDnum (*Bit Negation dnum*) se utiliza para ejecutar una operación lógica NEGATION - bit a bit (complemento de uno) en los tipos de datos dnum.

#### Ejemplos básicos

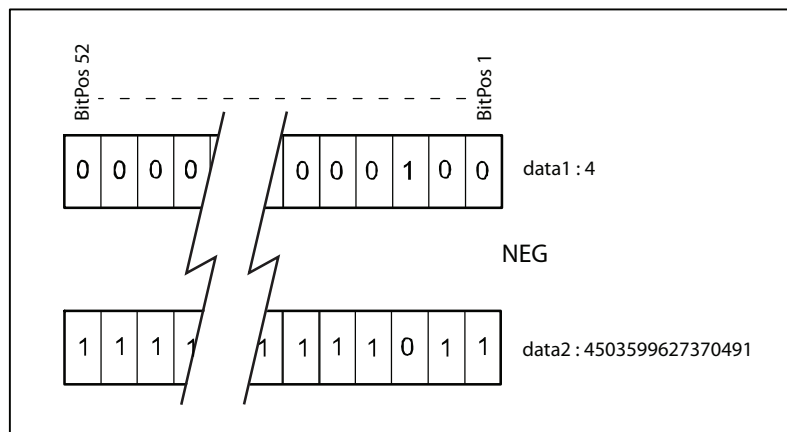
El ejemplo siguiente ilustra la función BitNegDnum.

Consulte también [Más ejemplos en la página 1189](#).

#### Ejemplo 1

```
VAR dnum data1 := 4;  
VAR dnum data2;  
  
data2 := BitNegDnum(data1);
```

Se ejecuta la operación lógica NEGATION bit a bit (consulte la figura siguiente) con los datos data1. El resultado se devuelve en data2 (representación entera).



xx1200000012

#### Valor de retorno

Tipo de dato: dnum

El resultado de la operación lógica NEGATION bit a bit en representación entera.

#### Argumentos

BitNegDnum (Value [\Size])

Value

Tipo de dato: dnum

El dato dnum, en representación entera.

Size

Tipo de dato: num

El tamaño (número de bits) que deben considerarse al realizar la operación lógica NEGATION bit a bit en el argumento Value. Los tamaños válidos son de 1 a 52.

*Continúa en la página siguiente*



**Limitaciones**

El rango de los tipos de datos `dnum` es de 0 - 4503599627370495.

**Más ejemplos**

A continuación aparecen más ejemplos de la función `BitNegDnum`.

**Ejemplo 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;
```

Se ejecuta la operación lógica NEGACIÓN bit a bit con los datos `data1` y el resultado se devuelve en `result` (representación entera). El primer valor escrito en el FlexPendant es 65497. El segundo valor escrito en el FlexPendant es 4503599627370457.

**Sintaxis**

```
BitNegDnum '('
  [Value ':=' ] <expression (IN) of dnum>
  ['\'Size ':=' < expression (IN) of num>]
  ')'
```

Una función con un valor de retorno del tipo de dato `dnum`.

**Información relacionada**

| Para obtener más información sobre                                | Consulte   |
|---|--|
| NEGACIÓN lógica bit a bit - Operación con datos <code>byte</code> | <a href="#">BitNeg - NEGACIÓN lógica bit a bit - Operación con datos de byte en la página 1186</a> |
| Tipo de dato <code>dnum</code>                                    | <a href="#">dnum - Valores numéricos dobles en la página 1660</a>                                  |
| Operación lógica AND bit a bit en un dato <code>dnum</code>       | <a href="#">BitAndDnum - Operación lógica AND bit a bit en un dato dnum en la página 1175</a>      |
| Operación lógica OR bit a bit en datos <code>dnum</code>          | <a href="#">BitOrDnum - Operación lógica OR bit a bit en datos dnum en la página 1192</a>          |
| Operación lógica XOR bit a bit en datos <code>dnum</code>         | <a href="#">BitXorDnum - Operación lógica XOR bit a bit en datos dnum en la página 1200</a>        |
| Otras funciones de bits   | <a href="#">Manual de referencia técnica - RAPID Overview</a>                                      |

## 2 Funciones

### 2.23 BitOr - OR lógico bit a bit - Operación con datos de byte RobotWare - OS

### 2.23 BitOr - OR lógico bit a bit - Operación con datos de byte

#### Utilización

BitOr (*Bit inclusive Or*) se utiliza para ejecutar una operación lógica OR bit a bit en los tipos de datos byte.

#### Ejemplos básicos

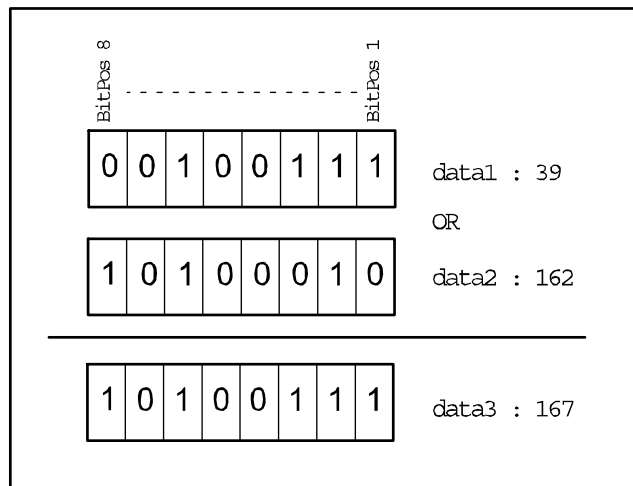
El ejemplo siguiente ilustra la función BitOr.

#### Ejemplo 1

```
VAR byte data1 := 39;  
VAR byte data2 := 162;  
VAR byte data3;  
  
data3 := BitOr(data1, data2);
```

Se ejecuta la operación lógica OR bit a bit con los datos data1 y data2. El resultado se devuelve en data3 (representación entera).

En la figura siguiente se muestra la operación OR lógico bit a bit.



xx0500002458

#### Valor de retorno

Tipo de dato: byte

El resultado de la operación lógica OR bit a bit en representación entera.

#### Argumentos

```
BitOr (BitData1 BitData2)
```

BitData1

Tipo de dato: byte

El dato de bit 1, en representación entera.

BitData2

Tipo de dato: byte

*Continúa en la página siguiente*

El dato de bit 2, en representación entera.

### Limitaciones

El rango de los tipos de datos `byte` es de 0 a 255.

### Sintaxis

```
BitOr '('
  [BitData1 ':='] <expression (IN) of byte> ','
  [BitData2 ':='] <expression (IN) of byte>
  ')'
```

Una función con un valor de retorno del tipo de dato `byte`.

### Información relacionada

| Para obtener más información sobre                      | Consulte   |
|---|--|
| AND lógico bit a bit - Operación con datos de byte      | <a href="#">BitAnd - AND lógico bit a bit - Operación con datos de byte en la página 1173</a>      |
| XOR lógico bit a bit - Operación con datos de byte      | <a href="#">BitXOr - XOR lógico bit a bit - Operación con datos de byte en la página 1198</a>      |
| NEGACIÓN lógica bit a bit - Operación con datos de byte | <a href="#">BitNeg - NEGACIÓN lógica bit a bit - Operación con datos de byte en la página 1186</a> |
| Otras funciones de bits                                 | <i>Manual de referencia técnica - RAPID Overview</i>   |
| <i>Advanced RAPID</i>                                   | <i>Especificaciones del producto - Controller software IRC5</i>                                    |

## 2 Funciones

### 2.24 BitOrDnum - Operación lógica OR bit a bit en datos dnum

RobotWare - OS

### 2.24 BitOrDnum - Operación lógica OR bit a bit en datos dnum

#### Utilización

BitOrDnum (*Bit inclusive Or dnum*) se utiliza para ejecutar una operación lógica OR bit a bit en los tipos de datos dnum.

#### Ejemplos básicos

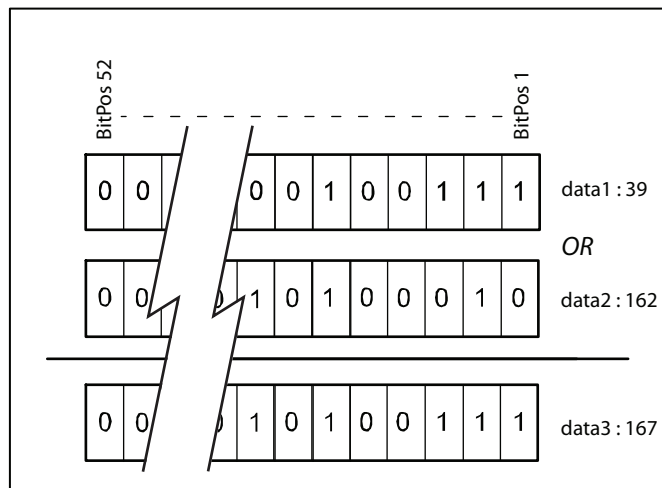
El ejemplo siguiente ilustra la función BitOrDnum.

#### Ejemplo 1

```
VAR dnum data1 := 39;  
VAR dnum data2 := 162;  
VAR dnum data3;  
  
data3 := BitOrDnum(data1, data2);
```

Se ejecuta la operación lógica OR bit a bit con los datos data1 y data2. El resultado se devuelve en data3 (representación entera).

En la figura siguiente se muestra la operación OR lógico bit a bit.



xx120000011

#### Valor de retorno

Tipo de dato: dnum

El resultado de la operación lógica OR bit a bit en representación entera.

#### Argumentos

BitOrDnum (Value1 Value2)

Value1

Tipo de dato: dnum

El primer valor de datos de bits, en representación entera.

Value2

Tipo de dato: dnum

El segundo valor de datos de bits, en representación entera.

Continúa en la página siguiente

**Limitaciones**

El rango de los tipos de datos `dnum` es de 0 - 4503599627370495.

**Sintaxis**

```
BitOrDnum '('
  [Value1 ':=' ] <expression (IN) of dnum> ','
  [Value2 ':=' ] <expression (IN) of dnum>
  ')'
```

Una función con un valor de retorno del tipo de dato `dnum`.

**Información relacionada**

| Para obtener más información sobre   | Consulte  |
|--|---|
| OR lógico bit a bit - Operación con datos byte                             | <a href="#">BitOr - OR lógico bit a bit - Operación con datos de byte en la página 1190</a>                               |
| Tipo de dato <code>dnum</code>   | <a href="#">dnum - Valores numéricos dobles en la página 1660</a>   |
| Operación lógica AND bit a bit en datos <code>dnum</code>                  | <a href="#">BitAndDnum - Operación lógica AND bit a bit en un dato <code>dnum</code> en la página 1175</a>                |
| Operación lógica XOR bit a bit en datos <code>dnum</code>                  | <a href="#">BitXOrDnum - Operación lógica XOR bit a bit en datos <code>dnum</code> en la página 1200</a>                  |
| Operación lógica NEGACIÓN bit a bit - operación en datos <code>dnum</code> | <a href="#">BitNegDnum - Operación lógica NEGACIÓN bit a bit - operación en datos <code>dnum</code> en la página 1188</a> |
| Otras funciones de bits  | <a href="#">Manual de referencia técnica - RAPID Overview</a>   |

## 2 Funciones

### 2.25 BitRSh - DESPLAZAMIENTO A LA DERECHA lógico bit a bit - Operación de byte RobotWare - OS

### 2.25 BitRSh - DESPLAZAMIENTO A LA DERECHA lógico bit a bit - Operación de byte

#### Utilización

BitRSh (*Bit Right Shift*) se utiliza para ejecutar una operación lógica de DESPLAZAMIENTO A LA DERECHA bit a bit en tipos de datos `byte`.

#### Ejemplos básicos

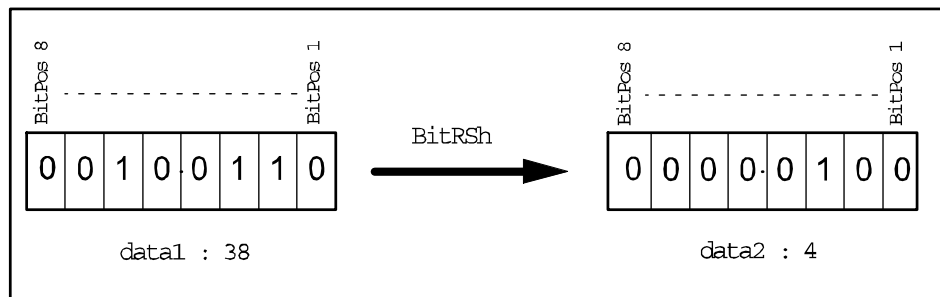
El ejemplo siguiente ilustra la función BitRSh.

#### Ejemplo 1

```
VAR num right_shift := 3;  
VAR byte data1 := 38;  
VAR byte data2;  
  
data2 := BitRSh(data1, right_shift);
```

Se ejecuta la operación lógica DESPLAZAMIENTO A LA DERECHA bit a bit con el dato `data1`, con 3 pasos (`right_shift`) de desplazamiento hacia la derecha. El resultado se devuelve en `data2` (representación entera)

En la figura siguiente se muestra la operación DESPLAZAMIENTO A LA DERECHA lógico bit a bit.



xx0500002455

#### Valor de retorno

Tipo de dato: `byte`

El resultado de la operación lógica DESPLAZAMIENTO A LA DERECHA bit a bit en representación entera.

Las posiciones de bit de la izquierda se rellenan con bits 0.

#### Argumentos

BitRSh (BitData ShiftSteps)

BitData

Tipo de dato: `byte`

Los datos de bits a desplazar, en representación entera.

ShiftSteps

Tipo de dato: `num`

*Continúa en la página siguiente*

## 2.25 BitRSh - DESPLAZAMIENTO A LA DERECHA lógico bit a bit - Operación de byte

*RobotWare - OS*  
*Continuación*

El número de desplazamientos lógicos (de 1 a 8) que se desea ejecutar.

### Limitaciones

El rango de los tipos de datos `byte` es de 0 a 255.

El valor válido del argumento `ShiftSteps` es un número de 1 a 8, lo que corresponde a un byte.

### Sintaxis

```
BitRSh '('
  [BitData ':='] <expression (IN) of byte> ','
  [ShiftSteps ':='] <expression (IN) of num>
  ')'
```

Una función con un valor de retorno del tipo de dato `byte`.

### Información relacionada

| Para obtener más información sobre  | Consulte  |
|---|---|
| DESPLAZAMIENTO A LA IZQUIERDA lógico bit a bit - Operación con datos de bit | <a href="#">BitLSh - DESPLAZAMIENTO A LA IZQUIERDA lógico bit a bit - Operación de byte en la página 1181</a> |
| Otras funciones de bits   | <i>Manual de referencia técnica - RAPID Overview</i>  |
| <i>Advanced RAPID</i>   | <i>Especificaciones del producto - Controller software IRC5</i>   |

## 2 Funciones

### 2.26 BitRShDnum - Operación lógica DESPLAZAMIENTO A LA DERECHA bit a bit en un dnum RobotWare - OS

### 2.26 BitRShDnum - Operación lógica DESPLAZAMIENTO A LA DERECHA bit a bit en un dnum

#### Utilización

BitRShDnum (*Bit Right Shift dnum*) se utiliza para ejecutar una operación lógica de DESPLAZAMIENTO A LA DERECHA bit a bit en tipos de datos dnum.

#### Ejemplos básicos

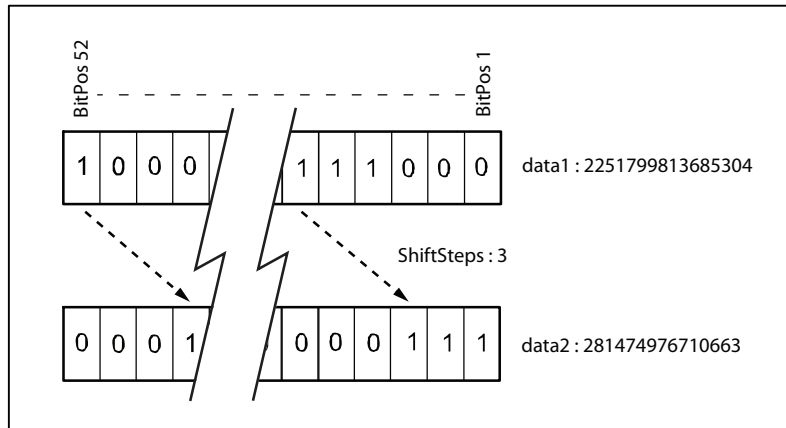
El ejemplo siguiente ilustra la función BitRShDnum.

##### Ejemplo 1

```
VAR num right_shift := 3;  
VAR dnum data1 := 2251799813685304;  
VAR dnum data2;  
  
data2 := BitRShDnum(data1, right_shift);
```

Se ejecuta la operación lógica DESPLAZAMIENTO A LA DERECHA bit a bit con el dato data1, con 3 pasos (right\_shift) de desplazamiento hacia la derecha. El resultado se devuelve en data2 (representación entera)

En la figura siguiente se muestra la operación DESPLAZAMIENTO A LA DERECHA lógico bit a bit.



xx120000009

#### Valor de retorno

Tipo de dato: dnum

El resultado de la operación lógica DESPLAZAMIENTO A LA DERECHA bit a bit en representación entera.

Las posiciones de bit de la izquierda se rellenan con bits 0.

#### Argumentos

BitRShDnum (Value ShiftSteps)

Value

Tipo de dato: dnum

Los datos de bits a desplazar, en representación entera.

*Continúa en la página siguiente*



## 2.26 BitRShDnum - Operación lógica DESPLAZAMIENTO A LA DERECHA bit a bit en un dnum

RobotWare - OS

Continuación

ShiftSteps

Tipo de dato: num

El número de desplazamientos lógicos (de 1 a 52) que se desea ejecutar.

**Limitaciones**El rango de los tipos de datos `dnum` es de 0 - 4503599627370495.Los valores válidos del argumento `ShiftSteps` son del 1 al 52 dado que un `dnum` tiene 52 bits.**Sintaxis**

```
BitRShDnum '('
  [Value ':='] <expression (IN) of dnum> ','
  [ShiftSteps ':='] <expression (IN) of num>
  ')'
```

Una función con un valor de retorno del tipo de dato `dnum`.**Información relacionada**

| Para obtener más información sobre   | Consulte  |
|--|---|
| DESPLAZAMIENTO A LA DERECHA lógico bit a bit - Operación con datos byte                | <a href="#">BitRSh - DESPLAZAMIENTO A LA DERECHA lógico bit a bit - Operación de byte en la página 1194</a>                       |
| Tipo de dato <code>dnum</code>   | <a href="#">dnum - Valores numéricos dobles en la página 1660</a>   |
| DESPLAZAMIENTO A LA IZQUIERDA lógico bit a bit - Operación con datos <code>dnum</code> | <a href="#">BitLShDnum - Operación lógica DESPLAZAMIENTO A LA IZQUIERDA bit a bit en dato <code>dnum</code> en la página 1183</a> |
| Otras funciones de bits  | <a href="#">Manual de referencia técnica - RAPID Overview</a>   |

## 2 Funciones

### 2.27 BitXOr - XOR lógico bit a bit - Operación con datos de byte

RobotWare - OS

### 2.27 BitXOr - XOR lógico bit a bit - Operación con datos de byte

#### Utilización

`BitXOr` (*Bit exclusive Or*) se utiliza para ejecutar una operación lógica bit a bit XOR en tipos de datos `byte`.

#### Ejemplos básicos

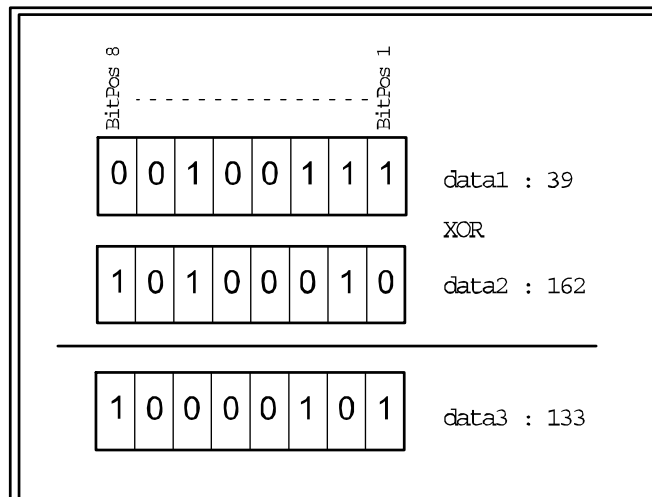
El ejemplo siguiente ilustra la función `BitXOr`.

#### Ejemplo 1

```
VAR byte data1 := 39;  
VAR byte data2 := 162;  
VAR byte data3;  
  
data3 := BitXOr(data1, data2);
```

Se ejecuta la operación lógica XOR bit a bit con los datos `data1` y `data2`. El resultado se devuelve en `data3` (representación entera).

En la figura siguiente se muestra la operación XOR lógico bit a bit.



xx0500002459

#### Valor de retorno

Tipo de dato: `byte`

El resultado de la operación lógica XOR bit a bit en representación entera.

#### Argumentos

`BitXOr` (`BitData1` `BitData2`)

`BitData1`

Tipo de dato: `byte`

El dato de bit 1, en representación entera.

`BitData2`

Tipo de dato: `byte`

*Continúa en la página siguiente*

El dato de bit 2, en representación entera.

### Limitaciones

El rango de los tipos de datos `byte` es de 0 a 255.

### Sintaxis

```
BitXOr '('
  [BitData1 ':='] <expression (IN) of byte> ','
  [BitData2 ':='] <expression (IN) of byte>
  ')'
```

Una función con un valor de retorno del tipo de dato `byte`.

### Información relacionada

| Para obtener más información sobre                      | Consulte   |
|---|--|
| AND lógico bit a bit - Operación con datos de byte      | <a href="#">BitAnd - AND lógico bit a bit - Operación con datos de byte en la página 1173</a>      |
| OR lógico bit a bit - Operación con datos de byte       | <a href="#">BitOr - OR lógico bit a bit - Operación con datos de byte en la página 1190</a>        |
| NEGACIÓN lógica bit a bit - Operación con datos de byte | <a href="#">BitNeg - NEGACIÓN lógica bit a bit - Operación con datos de byte en la página 1186</a> |
| Otras funciones de bits                                 | <i>Manual de referencia técnica - RAPID Overview</i>   |
| <i>Advanced RAPID</i>                                   | <i>Especificaciones del producto - Controller software IRC5</i>                                    |

## 2 Funciones

### 2.28 BitXOrDnum - Operación lógica XOR bit a bit en datos dnum

RobotWare - OS

### 2.28 BitXOrDnum - Operación lógica XOR bit a bit en datos dnum

#### Utilización

`BitXOrDnum` (*Bit eXclusive Or dnum*) se utiliza para ejecutar una operación lógica bit a bit XOR en tipos de datos `dnum`.

#### Ejemplos básicos

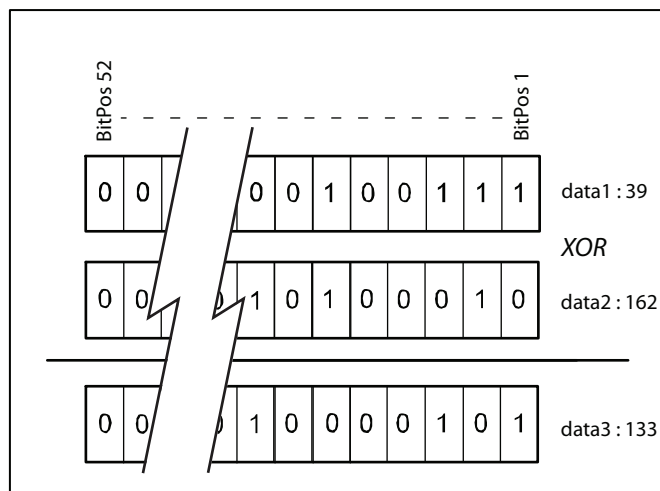
El ejemplo siguiente ilustra la función `BitXOrDnum`.

#### Ejemplo 1

```
VAR dnum data1 := 39;  
VAR dnum data2 := 162;  
VAR dnum data3;  
  
data3 := BitXOrDnum(data1, data2);
```

Se ejecuta la operación lógica XOR bit a bit con los datos `data1` y `data2`. El resultado se devuelve en `data3` (representación entera).

En la figura siguiente se muestra la operación XOR lógico bit a bit.



xx120000010

#### Valor de retorno

Tipo de dato: `dnum`

El resultado de la operación lógica XOR bit a bit en representación entera.

#### Argumentos

`BitXOrDnum (Value1 Value2)`

Value1

Tipo de dato: `dnum`

El primer valor de datos de bits, en representación entera.

Value2

Tipo de dato: `dnum`

Continúa en la página siguiente

El segundo valor de datos de bits, en representación entera.

### Limitaciones

El rango de los tipos de datos `dnum` es de 0 - 4503599627370495.

### Sintaxis

```
BitXOrDnum '('
  [Value1 ':=' ] <expression (IN) of dnum> ', '
  [Value2 ':=' ] <expression (IN) of dnum>
  ')'
```

Una función con un valor de retorno del tipo de dato `dnum`.

### Información relacionada

| Para obtener más información sobre   | Consulte   |
|--|--|
| XOR lógico bit a bit - Operación con datos byte                            | <a href="#">BitXOr - XOR lógico bit a bit - Operación con datos de byte en la página 1198</a>                |
| Tipo de dato <code>dnum</code>   | <a href="#">dnum - Valores numéricos dobles en la página 1660</a>  |
| Operación lógica AND bit a bit en un dato <code>dnum</code>                | <a href="#">BitAndDnum - Operación lógica AND bit a bit en un dato dnum en la página 1175</a>                |
| Operación lógica OR bit a bit en datos <code>dnum</code>                   | <a href="#">BitOrDnum - Operación lógica OR bit a bit en datos dnum en la página 1192</a>                    |
| Operación lógica NEGACIÓN bit a bit - operación en datos <code>dnum</code> | <a href="#">BitNegDnum - Operación lógica NEGACIÓN bit a bit - operación en datos dnum en la página 1188</a> |
| Otras funciones de bits  | <a href="#">Manual de referencia técnica - RAPID Overview</a>  |

## 2 Funciones

### 2.29 ByteToStr - Convierte un byte en un dato de cadena de caracteres

RobotWare - OS

### 2.29 ByteToStr - Convierte un byte en un dato de cadena de caracteres

#### Utilización

ByteToStr (*Byte To String*) se utiliza para convertir un byte en un dato de tipo cadena con un formato de byte de datos definido.

#### Ejemplos básicos

El ejemplo siguiente ilustra la función ByteToStr.

#### Ejemplo 1

```
VAR string con_data_buffer{5};
VAR byte data1 := 122;
```

```
con_data_buffer{1} := ByteToStr(data1);
```

El contenido del componente con\_data\_buffer{1} de la matriz será "122" tras la función ByteToStr ....

```
con_data_buffer{2} := ByteToStr(data1\Hex);
```

El contenido del componente con\_data\_buffer{2} de la matriz será "7A" tras la función ByteToStr ....

```
con_data_buffer{3} := ByteToStr(data1\Okt);
```

El contenido del componente con\_data\_buffer{3} de la matriz será "172" tras la función ByteToStr ....

```
con_data_buffer{4} := ByteToStr(data1\Bin);
```

El contenido del componente con\_data\_buffer{4} de la matriz será "01111010" después de la función ByteToStr ....

```
con_data_buffer{5} := ByteToStr(data1\Char);
```

El contenido del componente con\_data\_buffer{5} de la matriz será "z" tras la función ByteToStr ....

#### Valor de retorno

Tipo de dato: string

El resultado de la operación de conversión, con el formato siguiente:

| Formato    | Caracteres                   | Longitud de cadena | Rango                   |
|------------|------------------------------|--------------------|-------------------------|
| Dec .....  | '0' - '9'                    | 1-3                | "0" - "255"             |
| Hex .....  | '0' - '9', 'A' - 'F'         | 2                  | "00" - "FF"             |
| Oct .....  | '0' - '7'                    | 3                  | "000" - "377"           |
| Bin .....  | '0' - '1'                    | 8                  | "00000000" - "11111111" |
| Char ..... | Cualquier carácter ASCII (*) | 1                  | Un carácter ASCII       |

(\*) Si es un carácter ASCII no imprimible, el formato de retorno será el formato de código de carácter de RAPID (por ejemplo, "\07" para el carácter de control BEL).

#### Argumentos

```
ByteToStr (BitData [\Hex] | [\Okt] | [\Bin] | [\Char])
```

Continúa en la página siguiente

## 2.29 ByteToStr - Convierte un byte en un dato de cadena de caracteres

RobotWare - OS

Continuación

BitData

Tipo de dato: `byte`

Los datos de bits a convertir.

Si se omite el argumento modificador opcional, los datos se convierten al formato decimal (**Dec**).

[\Hex]

**Hexadecimal**Tipo de dato: `switch`

Los datos se convierten al formato hexadecimal.

[\Okt]

**Octal**Tipo de dato: `switch`

Los datos se convierten al formato octal.

[\Bin]

**Binary**Tipo de dato: `switch`Los datos se convierten al formato `binary`.

[\Char]

**Character**Tipo de dato: `switch`Los datos se convierten al formato de carácter `ASCII`.**Limitaciones**El rango de los tipos de datos `byte` es de 0 a 255 en decimal.**Sintaxis**

```
ByteToStr '('
  [BitData ':='] <expression (IN) of byte>
  ['\ Hex ] | ['\ Okt] | ['\ Bin] | ['\ Char]
  ')'
```

Una función con un valor de retorno del tipo de dato `string`.**Información relacionada**

| Para obtener más información sobre                     | Consulte  |
|--|---|
| Conversión de una cadena de caracteres a datos de byte | <a href="#">StrToByte - Convierte una cadena en un byte en la página 1509</a> |
| Otras funciones de bits (byte)                         | <i>Manual de referencia técnica - RAPID Overview</i>                          |
| Otras funciones de cadenas de caracteres               | <i>Manual de referencia técnica - RAPID Overview</i>                          |

## 2 Funciones

---

### 2.30 CalcJointT - Calcula los ángulos de las articulaciones a partir de un robtarget RobotWare - OS

### 2.30 CalcJointT - Calcula los ángulos de las articulaciones a partir de un robtarget

---

#### Utilización

CalcJointT (*Calculate Joint Target*) se utiliza para calcular ángulos de los ejes del robot y de los ejes externos a partir de datos `robtarget` especificados.

Los datos de entrada `robtarget` deben especificarse con el mismo sistema de coordenadas que se especifique con argumento en `Tool`, `WObj`, en el desplazamiento de programa activo en el momento de la ejecución (`ProgDisp`) y en el offset de los ejes externos (`EOffs`).. Los datos `jointtarget` devueltos se expresan en el sistema de coordenadas de calibración.

Si el tipo de aplicación es `MultiMove` con el modo semicoordinado o el modo coordinado sincronizado con el objeto de trabajo coordinado, y éste es movido por alguna unidad mecánica situada en otra tarea de programa, la función `CalcJointT` puede usarse si:

- Es adecuado que la posición actual del objeto de trabajo coordinado movido por la unidad mecánica se use en el cálculo (base de coordenadas del usuario actual). Todos los demás datos serán capturados desde el programa de RAPID.
- La unidad mecánica situada en otra tarea de programa está en reposo.
- Se utiliza el argumento `\UseCurWObjPos`.

---

#### Ejemplos básicos

Los siguientes ejemplos ilustran la función `CalcJointT`.

##### Ejemplo 1

```
VAR jointtarget jointpos1;  
CONST robtarget p1 := [...];  
jointpos1 := CalcJointT(p1, tool1 \WObj:=wobj1);
```

El valor de `jointtarget` correspondiente al valor de `robtarget p1` se almacena en `jointpos1`. La herramienta `tool1` y el objeto de trabajo `wobj1` se utilizan para calcular los ángulos de los ejes `jointpos1`.

##### Ejemplo 2

```
VAR jointtarget jointpos2;  
CONST robtarget p2 := [...];  
jointpos2 := CalcJointT(\UseCurWObjPos, p2, tool2 \WObj:=orb1);
```

El valor de `jointtarget` correspondiente al valor de `robtarget p2` se almacena en `jointpos2`. La herramienta `tool2` y el objeto de trabajo `orb1` se utilizan para calcular los ángulos de eje `jointpos2`. Se usa para el cálculo la posición actual del manipulador `orb1` en reposo, no situado en la misma tarea de programa que el robot del TCP.

##### Ejemplo 3

```
VAR jointtarget jointpos3;  
CONST robtarget p3 := [...];  
VAR errnum myerrnum;  
jointpos3 := CalcJointT(p3, tool2 \WObj:=orb1  
  \ErrorNumber:=myerrnum);
```

*Continúa en la página siguiente*



### 2.30 CalcJointT - Calcula los ángulos de las articulaciones a partir de un robtarget RobotWare - OS Continuación

```

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);
  ..
  ..
  TPWrite "jointpos3.extax.eax_f"+ValToStr(jointpos3.extax.eax_f);
ELSE
  MoveAbsJ jointpos3, v100, fine, tool2 \Wobj:=orbl;
ENDIF

```

El valor de `jointtarget` correspondiente al valor de `robtarget p3` se almacena en `jointpos3`. Si la posición puede alcanzarse, se utiliza; de lo contrario, el valor `jointtarget` se escribe en el FlexPendant.

---

#### Valor de retorno

Tipo de dato: `jointtarget`

Los ángulos en grados de los ejes del robot en el lado del brazo.

Los valores de los ejes externos, en mm para los ejes lineales y en grados para los ejes de rotación.

Los valores devueltos están siempre relacionados con la posición de calibración.

---

#### Argumentos

```

CalcJointT ( [\UseCurWObjPos] Rob_target Tool [\Wobj]
             [\ErrorNumber])

```

`[\UseCurWObjPos]`

Tipo de dato: `switch`

Se usa en el cálculo la posición actual del objeto de trabajo coordinado movido por la unidad mecánica en otra tarea (base de coordenadas del usuario actual). Todos los demás datos serán capturados desde el programa de RAPID.

`Rob_target`

Tipo de dato: `robtarget`

La posición del robot y de los ejes externos en el sistema de coordenadas más externo respecto de la herramienta y el objeto de trabajo especificados y respecto del desplazamiento de programa activo en el momento de la ejecución (`ProgDisp`) y/o el offset de los ejes externos (`EOffs`).

`Tool`

Tipo de dato: `tooldata`

La herramienta utilizada para el cálculo de los ángulos de los ejes del robot.

`[\Wobj]`

*Work Object*

*Continúa en la página siguiente*

## 2 Funciones

---

### 2.30 CalcJointT - Calcula los ángulos de las articulaciones a partir de un robtarget

RobotWare - OS

Continuación

Tipo de dato: wobjdata

El objeto de trabajo (sistema de coordenadas) con el que está relacionada la posición del robot.

Si se omite este argumento, se utiliza el objeto de trabajo `wobj0`. Este argumento debe especificarse si se utiliza una herramienta estacionaria, ejes externos coordinados o un transportador.

[ \ErrorNumber ]

*Error number*

Tipo de dato: errnum

Una variable (VAR o PERS) que contendrá la constante de error `ERR_ROBLIMIT` si al menos un eje está fuera de los límites de los ejes o los límites se rebasan en al menos un eje acoplado, o `ERR_OUTSIDE_REACH` si la posición (robtarget) está fuera del área de trabajo del robot. Si se utiliza este argumento opcional y la variable tiene el valor `ERR_ROBLIMIT` o `ERR_OUTSIDE_REACH` tras la ejecución de la función, la variable de retorno será un valor `jointtarget` que corresponde al `robtarget` utilizado.

Si se omite esta variable opcional, se ejecuta el gestor de errores y el `jointtarget` devuelto no se actualizará si un eje se encuentra fuera del área de trabajo o se rebasan los límites.

---

#### Ejecución de programas

El valor `jointtarget` devuelto se calcula a partir del valor de entrada `robtarget`. Si se usa el argumento `\UseCurWObjPos`, la posición que se utiliza proviene de la posición actual de la unidad mecánica que controla la base de coordenadas del usuario. Para calcular los ángulos de los ejes del robot, se tienen en cuenta los `Tool`, `WObj` especificados (incluida la base coordinada de coordenadas del usuario) y el `ProgDisp` activo en el momento de la ejecución. Para calcular la posición de los ejes externos en el momento de la ejecución, se tiene en cuenta el `EOffs` activo.

El cálculo selecciona siempre la configuración del robot acorde con los datos de configuración especificados en los datos de entrada `robtarget`. Ni la instrucción `ConfL` ni `ConfJ` afectan a este principio de cálculo. Si se utiliza la singularidad de la muñeca, se cambia el eje 4 a 0 grados.

Si hay cualquier desplazamiento de programa activo (`ProgDisp`) y/o se almacena un desplazamiento de eje externo (`EOffs`) en el momento de almacenar los datos `robtarget`, es necesario que el mismo desplazamiento de programa y/o el mismo offset de eje externo esté activo cuando se ejecuta `CalcJointT`.

---

#### Limitación

Si se utiliza una base de coordenadas, la unidad coordinada debe estar activada antes de usar `CalcJointT`.

La unidad mecánica que controla la base de coordenadas del usuario en el objeto de trabajo debe estar disponible normalmente en la misma tarea de programa que el robot del TCP que ejecuta `CalcJointT`.

Continúa en la página siguiente

### 2.30 CalcJointT - Calcula los ángulos de las articulaciones a partir de un robtarget RobotWare - OS Continuación

Normalmente, CalcJointT utiliza robtarget, tooldata y wobjdata desde el programa de RAPID para calcular jointtarget. En el caso de los objetos de trabajo coordinados, la posición de la unidad mecánica se indica como una posición de ejes externos en el robtarget. Éste no es el caso si la unidad mecánica es controlada por otra tarea de programa (sistema MultiMove) o si la unidad mecánica no está controlada por el sistema de control (transportador). En el caso del sistema MultiMove, pero no en el caso del transportador, es posible usar el argumento \UseCurWObjPos si la unidad mecánica está en reposo en el momento de la ejecución de CalcJointT.

#### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema ERRNO cambiará a:

| Nombre            | Causa del error  |
|-------------------|--|
| ERR_ROBLIMIT      | La posición puede alcanzarse, aunque al menos uno de los ejes está fuera de los límites de los ejes o se superan los límites en al menos un eje acoplado.            |
| ERR_OUTSIDE_REACH | La posición (robtarget) está fuera del rango de trabajo del robot.   |
| ERR_WOBJ_MOVING   | La unidad mecánica que controla el objeto de trabajo (base de coordenadas del usuario) no está en reposo en el momento de la ejecución de CalcJointT \UseCurWObjPos. |

#### Sintaxis

```
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>
  ')'
```

Una función con un valor de retorno del tipo de dato jointtarget.

#### Información relacionada

| Para obtener más información sobre           | Consulte   |
|--|--|
| Cálculo de robtarget a partir de jointtarget | <a href="#">CalcRobT - Calcula el valor de robtarget a partir de jointtarget en la página 1209</a> |
| Definición de posición                       | <a href="#">robtarget - Datos de posición en la página 1750</a>                                    |
| Definición de posición de los ejes           | <a href="#">jointtarget - Datos de posición de eje en la página 1691</a>                           |
| Definición de herramientas                   | <a href="#">tooldata - Datos de herramienta en la página 1795</a>                                  |
| Definición de objetos de trabajo             | <a href="#">wobjdata - Datos del objeto de trabajo en la página 1823</a>                           |
| Sistemas de coordenadas                      | <a href="#">Manual de referencia técnica - RAPID Overview</a>                                      |

Continúa en la página siguiente

## 2 Funciones

---

2.30 CalcJointT - Calcula los ángulos de las articulaciones a partir de un robtarget

RobotWare - OS

Continuación

| Para obtener más información sobre                   | Consulte  |
|--|---|
| Sistema de coordenadas de desplazamiento de programa | <a href="#">PDispOn - Activa el desplazamiento de programa en la página 533</a> |
| Sistema de coordenadas de offset de ejes externos    | <a href="#">EOffsOn - Activa un offset de ejes adicionales en la página 195</a> |

## 2.31 CalcRobT - Calcula el valor de robtarget a partir de jointtarget

### Utilización

CalcRobT (*Calculate Robot Target*) se utiliza para calcular los datos de robtarget a partir de datos jointtarget determinados.

Esta función devuelve un valor de robtarget con la posición (x, y, z), la orientación (q1 ... q4), la configuración de los ejes del robot y la posición de los ejes externos. Los datos de entrada de jointtarget deben especificarse en el sistema de coordenadas de calibración.

Los datos robtarget devueltos se expresan en el sistema de coordenadas más exterior. Tiene en cuenta la herramienta, el objeto de trabajo y el desplazamiento de programa activo en el momento de la ejecución (ProgDisp) y el offset de eje externo (EOffs).

### Ejemplos básicos

El ejemplo siguiente ilustra la función CalcRobT.

#### Ejemplo 1

```
VAR robtarget p1;
CONST jointtarget jointpos1 := [...];

p1 := CalcRobT(jointpos1, tool1 \WObj:=wobj1);
```

El valor de robtarget correspondiente al valor de jointtarget jointpos1 se almacena en p1. La herramienta tool1 y el objeto de trabajo wobj1 se utilizan para calcular la posición p1.

### Valor de retorno

Tipo de dato: robtarget

El robot y la posición de los ejes externos se devuelven en el tipo de dato robtarget y se expresan en el sistema de coordenadas más exterior. Tiene en cuenta la herramienta, el objeto de trabajo y el desplazamiento de programa activo en el momento de la ejecución (ProgDisp) y el offset de eje externo (EOffs).

Si no hay ningún ProgDisp activo, la posición del robot se expresa en el sistema de coordenadas del objeto. Si no hay ningún EOffs activo, la posición de los ejes externos se expresa en el sistema de coordenadas de calibración.

### Argumentos

```
CalcRobT( Joint_target Tool [\WObj] )
```

Joint\_target

Tipo de dato: jointtarget

La posición de los ejes del robot y de los ejes externos, respecto del sistema de coordenadas de calibración.

Tool

Tipo de dato: tooldata

La herramienta utilizada para el cálculo de la posición del robot.

*Continúa en la página siguiente*

## 2 Funciones

### 2.31 CalcRobT - Calcula el valor de robtarget a partir de jointtarget

RobotWare - OS

Continuación

[ \WObj ]

#### Work Object

Tipo de dato: wobjdata

El objeto de trabajo (sistema de coordenadas) con el que está relacionada la posición del robot devuelta por la función.

Si se omite este argumento, se utiliza el objeto de trabajo wobj0. Este argumento debe especificarse si se utiliza una herramienta estacionaria, ejes externos coordinados o un transportador.

#### Ejecución de programas

El valor `robtarget` devuelto se calcula a partir del valor de entrada `jointtarget`. Para calcular la posición cartesiana del robot, se tienen en cuenta los valores especificados de `Tool`, `WObj` (incluida la base de coordenadas de usuario coordinada) y el `ProgDisp` activo en el momento de la ejecución.

Para calcular la posición de los ejes externos, se tiene en cuenta además el `EOffs` activo en el momento de la ejecución.

#### Limitación

Si se utiliza una base de coordenadas, la unidad coordinada debe estar activada antes de usar `CalcRobT`. La unidad coordinada también debe estar situada en la misma tarea que el robot.

#### Sintaxis

```
CalcRobT '('  
  [Joint_target ':='] <expression (IN) of jointtarget> ','  
  [Tool ':='] <persistent (PERS) of tooldata>  
  ['\ ' WObj ':='] <persistent (PERS) of wobjdata> ')'
```

Una función con un valor de retorno del tipo de dato `robtarget`.

#### Información relacionada

| Para obtener más información sobre                                     | Consulte  |
|--|---|
| Cálculo de <code>robtarget</code> a partir de <code>jointtarget</code> | <a href="#">CalcJointT - Calcula los ángulos de las articulaciones a partir de un robtarget en la página 1204</a> |
| Definición de posición   | <a href="#">robtarget - Datos de posición en la página 1750</a>   |
| Definición de posición de los ejes                                     | <a href="#">jointtarget - Datos de posición de eje en la página 1691</a>  |
| Definición de herramientas   | <a href="#">tooldata - Datos de herramienta en la página 1795</a>   |
| Definición de objetos de trabajo                                       | <a href="#">wobjdata - Datos del objeto de trabajo en la página 1823</a>  |
| Sistemas de coordenadas  | <a href="#">Manual de referencia técnica - RAPID Overview</a>   |
| Sistema de coordenadas de desplazamiento de programa                   | <a href="#">PDispOn - Activa el desplazamiento de programa en la página 533</a>                                   |
| Sistema de coordenadas de offset de ejes externos                      | <a href="#">EOffsOn - Activa un offset de ejes adicionales en la página 195</a>                                   |

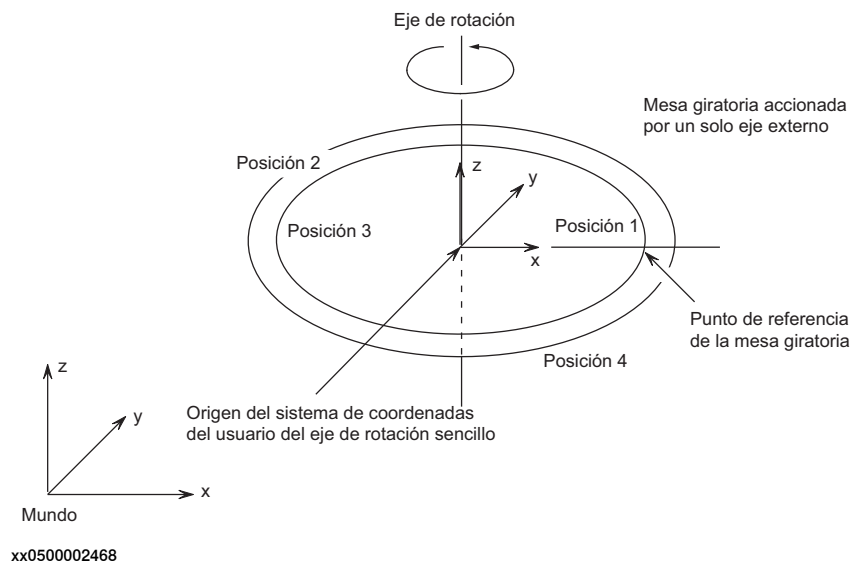
## 2.32 CalcRotAxFrameZ - Calcular la base de coordenadas de un eje de rotación

## Utilización

CalcRotAxFrameZ (*Calculate Rotational Axis Frame with positive Z-point*) se utiliza para calcular el sistema de coordenadas de usuario de una unidad mecánica de eje de rotación. Esta función debe utilizarse cuando el robot maestro y el eje adicional se encuentran en tareas de RAPID diferentes. Si se encuentran en la misma tarea, se debe utilizar la función CalcRotAxisFrame.

## Descripción

La definición de una base de coordenadas de usuario para un eje de rotación externo requiere que la mesa giratoria (o la estructura mecánica similar) del eje externo tenga un punto de referencia marcado. Además, es necesario calibrar la base de coordenadas del robot del TCP y del TCP. El procedimiento de calibración se basa en un conjunto de posicionamiento del TCP del robot sobre el punto de referencia, mientras la mesa giratoria gira hasta distintos ángulos. También se requiere el posicionamiento del TCP del robot en la dirección Z positiva. Para ver una definición de puntos de un eje de rotación, consulte la figura siguiente.



El sistema de coordenadas de usuario del eje de rotación tiene su origen en el centro de la mesa giratoria. La dirección Z coincide con el eje de rotación y el eje X atraviesa el punto de referencia.

Continúa en la página siguiente

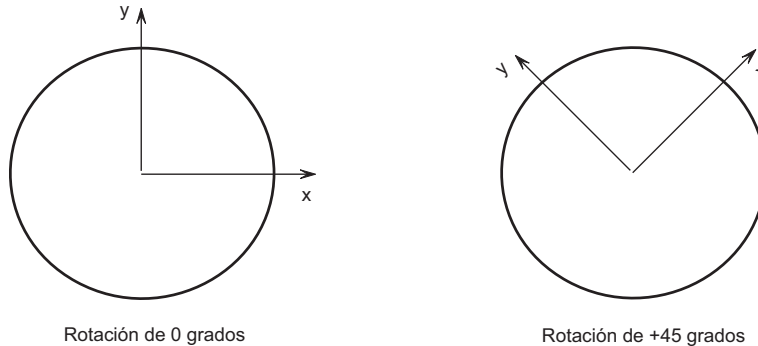
## 2 Funciones

### 2.32 CalcRotAxFrameZ - Calcular la base de coordenadas de un eje de rotación

RobotWare - OS

Continuación

La figura siguiente muestra el sistema de coordenadas de usuario para dos posicionamientos diferentes de la mesa giratoria (con la mesa giratoria vista desde arriba).



xx0500002469

### Ejemplos básicos

El ejemplo siguiente ilustra la función CalcRotAxFrameZ.

#### Ejemplo 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.
```

Continúa en la página siguiente



## 2.32 CalcRotAxFrameZ - Calcular la base de coordenadas de un eje de rotación

RobotWare - OS

Continuación

```

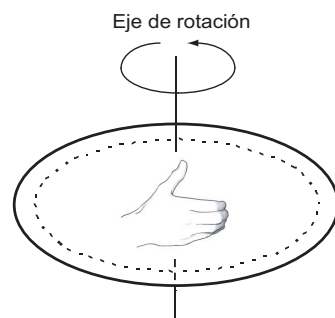
IF (max_err < 1.0) AND (mean_err < 0.5) THEN
  WriteCfgData "/MOC/SINGLE/STN_1", "base_frame_pos_x",
    resFr.trans.x/1000;
  WriteCfgData "/MOC/SINGLE/STN_1", "base_frame_pos_y",
    resFr.trans.y/1000;
  WriteCfgData "/MOC/SINGLE/STN_1", "base_frame_pos_z",
    resFr.trans.z/1000;
  WriteCfgData "/MOC/SINGLE/STN_1", "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

```

Se crean cuatro posiciones, `pos1 - pos4`, y se les aplica `ModPos` de forma que la herramienta del robot `tMyTool` apunte al mismo punto de referencia del eje externo `STN_1`, pero con otra rotación de eje externo. La posición, `zpos`, se crea o modifica con `ModPos` de forma que la herramienta del robot `tMyTool` apunte en la dirección Z positiva de acuerdo con la definición de la dirección Z positiva de una unidad mecánica externa de rotación. Con la definición de la dirección Z positiva de una unidad mecánica externa de rotación. Consulte [Descripción en la página 1211](#). A continuación, los puntos se utilizan para calcular la base de coordenadas de la base del eje externo, `resFr`, respecto del sistema de coordenadas mundo. Por último, se escribe la base de coordenadas en el archivo de configuración y se ejecuta una instrucción `WarmStart` para aplicar el cambio.

**Nota**

Definición en la dirección Z positiva de una unidad mecánica externa de rotación: Haga coincidir los dedos de la mano derecha con el eje de rotación positivo del eje de rotación. En esta posición, la dirección del pulgar define la dirección Z positiva. Consulte la figura que aparece a continuación.



xx0500002472

Continúa en la página siguiente

## 2 Funciones

### 2.32 CalcRotAxFrameZ - Calcular la base de coordenadas de un eje de rotación

RobotWare - OS

Continuación

---

#### Valor de retorno

Tipo de dato: pose

La base de coordenadas calculada.

---

#### Argumentos

CalcRotAxFrameZ (TargetList TargetsInList PositiveZPoint  
MaxErrMeanErr)

TargetList

Tipo de dato: robtarget

Una matriz de robtargets que contienen las posiciones definidas al apuntar la mesa giratoria hacia fuera. El número mínimo de robtargets es 4 y el máximo es 10.

TargetsInList

Tipo de dato: num

Número de robtargets de una matriz.

PositiveZPoint

Tipo de dato: robtarget

robtarget contiene la posición definida apuntando hacia un punto en la dirección Z positiva. Con la definición en la dirección Z positiva de una unidad mecánica externa de rotación. Consulte [Descripción en la página 1211](#).

MaxErr

**Maximum Error**

Tipo de dato: num

El error máximo estimado, en mm.

MeanErr

**Mean Error**

Tipo de dato: num

El error medio estimado, en mm.

---

#### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema ERRNO cambiará a:

| Nombre    | Causa del error   |
|-----------|---|
| ERR_FRAME | Las posiciones no tienen la relación requerida o no están especificadas con suficiente precisión. |

---

#### Sintaxis

```
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> ')'
```

Continúa en la página siguiente

### 2.32 CalcRotAxFrameZ - Calcular la base de coordenadas de un eje de rotación

*RobotWare - OS*

*Continuación*

Una función con un valor de retorno del tipo de dato `pose`.

---

#### Información relacionada

| Para obtener más información sobre    | Consulte   |
|---------------------------------------|--|
| Instrucciones y funciones matemáticas | <i>Manual de referencia técnica - RAPID Overview</i> |

## 2 Funciones

### 2.33 CalcRotAxisFrame - Calcular la base de coordenadas de un eje de rotación RobotWare - OS

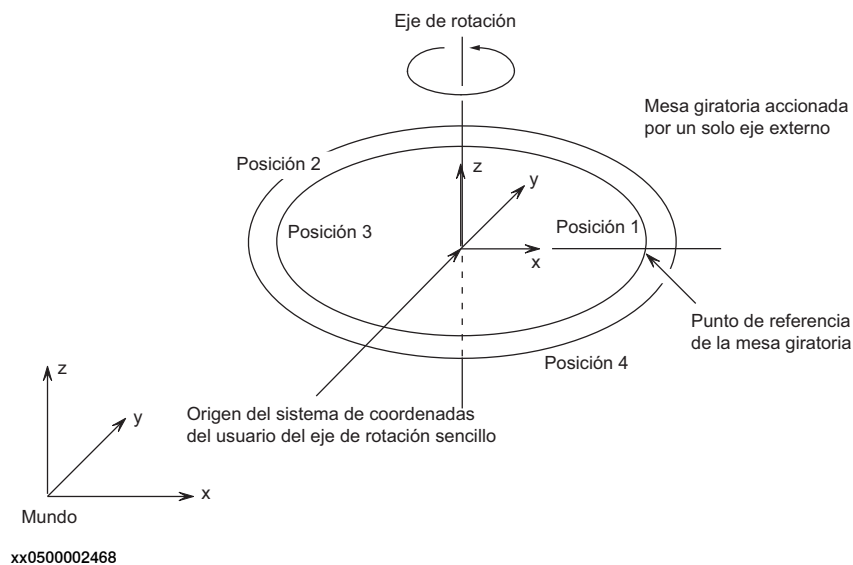
### 2.33 CalcRotAxisFrame - Calcular la base de coordenadas de un eje de rotación

#### Utilización

`CalcRotAxisFrame` (*Calculate Rotational Axis Frame*) se utiliza para calcular el sistema de coordenadas de usuario de una unidad mecánica de eje de rotación. Esta función debe utilizarse cuando el robot maestro y el eje adicional se encuentran en la misma tarea de RAPID. Si se encuentran en tareas diferentes, se debe utilizar la función `CalcRotAxFrameZ`.

#### Descripción

La definición de una base de coordenadas de usuario para un eje de rotación externo requiere que la mesa giratoria (o la estructura mecánica similar) del eje externo tenga un punto de referencia marcado. Además, es necesario calibrar la base de coordenadas principal del robot y el TCP. El procedimiento de calibración se basa en un conjunto de posicionamiento del TCP del robot sobre el punto de referencia, mientras la mesa giratoria gira hasta distintos ángulos. La definición de puntos de un eje de rotación se muestra en la figura siguiente.



El sistema de coordenadas de usuario del eje de rotación tiene su origen en el centro de la mesa giratoria. La dirección Z coincide con el eje de rotación y el eje X atraviesa el punto de referencia.

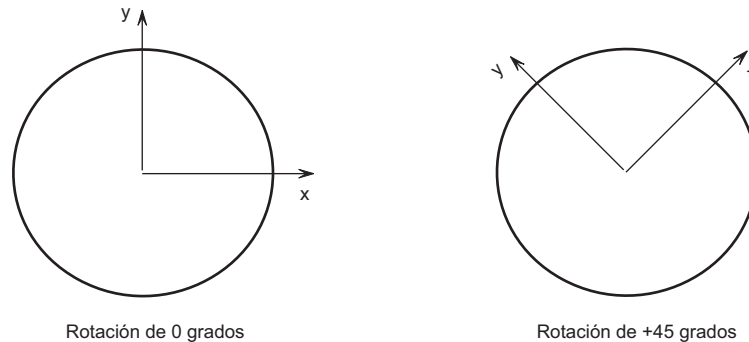
Continúa en la página siguiente

## 2.33 CalcRotAxisFrame - Calcular la base de coordenadas de un eje de rotación

RobotWare - OS

Continuación

La figura siguiente muestra el sistema de coordenadas de usuario para dos posicionamientos diferentes de la mesa giratoria (con la mesa giratoria vista desde arriba).



Rotación de 0 grados

Rotación de +45 grados

xx0500002469

## Ejemplos básicos

El ejemplo siguiente ilustra la función CalcRotAxisFrame.

## Ejemplo 1

```

CONST robtargt pos1 := [...];
CONST robtargt pos2 := [...];
CONST robtargt pos3 := [...];
CONST robtargt pos4 := [...];
VAR robtargt 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;

```

*Continúa en la página siguiente*

## 2 Funciones

---

### 2.33 CalcRotAxisFrame - Calcular la base de coordenadas de un eje de rotación

RobotWare - OS

Continuación

```
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
```

Se crean cuatro posiciones, pos1 - pos4, y se les aplica ModPos de forma que la herramienta del robot `tMyTool` apunte al mismo punto de referencia del eje externo STN\_1, pero con otra rotación de eje externo. A continuación, los puntos se utilizan para calcular la base de coordenadas de la base del eje externo, `resFr`, respecto del sistema de coordenadas mundo. Por último, se escribe la base de coordenadas en el archivo de configuración y se ejecuta una instrucción `WarmStart` para aplicar el cambio.

---

#### Valor de retorno

Tipo de dato: pose

La base de coordenadas calculada.

---

#### Argumentos

```
CalcRotAxisFrame (MechUnit [\AxisNo] TargetList TargetsInList MaxErr
    MeanErr)
```

MechUnit

*Mechanical Unit*

Tipo de dato: mecunit

El nombre de la unidad mecánica a calibrar.

[\AxisNo]

Tipo de dato: num

Un argumento opcional que define el número de eje cuya base de coordenadas debe determinarse. Si sólo existe un eje de rotación, éste recibe el valor 1. En el caso de las unidades mecánicas con varios ejes, el número del eje debe indicarse con este argumento.

TargetList

Tipo de dato: robtarget

Una matriz de robtargets que contienen las posiciones definidas al apuntar la mesa giratoria hacia fuera. El número mínimo de robtargets es 4 y el máximo es 10.

TargetsInList

Tipo de dato: num

*Continúa en la página siguiente*

## 2.33 CalcRotAxisFrame - Calcular la base de coordenadas de un eje de rotación

RobotWare - OS

Continuación

Número de robtargets de una matriz.

MaxErr

**Maximum Error**

Tipo de dato: num

El error máximo estimado, en mm.

MeanErr

**Mean Error**

Tipo de dato: num

El error medio estimado, en mm.

**Gestión de errores**

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema ERRNO cambiará a:

| Nombre    | Causa del error   |
|-----------|---|
| ERR_FRAME | Las posiciones no tienen la relación requerida o no están especificadas con suficiente precisión. |

**Sintaxis**

```
CalcRotAxisFrame '('
  [MechUnit ':=' ] <variable (VAR) of mecunit>
  [\AxisNo ':=' <expression (IN) of num> ] ', '
  [TargetList ':=' ] <array {*} (IN) of robtarget> ', '
  [TargetsInList ':=' ] <expression (IN) of num> ', '
  [MaxErr ':=' ] <variable (VAR) of num> ', '
  [MeanErr ':=' ] <variable (VAR) of num> ')'
```

Una función con un valor de retorno del tipo de dato pose.

**Información relacionada**

| Para obtener más información sobre    | Consulte   |
|---------------------------------------|--|
| Instrucciones y funciones matemáticas | <i>Manual de referencia técnica - RAPID Overview</i> |

## 2 Funciones

---

### 2.34 CamGetExposure - Obtiene parámetros específicos de la cámara *Integrated Vision*

### 2.34 CamGetExposure - Obtiene parámetros específicos de la cámara

---

#### Utilización

CamGetExposure (Camera Get Exposure) es una función que lee la configuración actual de una cámara. Con esta función y la instrucción CamSetExposure es posible adaptar las imágenes de la cámara en función del entorno durante el tiempo de ejecución.

---

#### Ejemplos básicos

El ejemplo siguiente ilustra la función CamGetExposure.

##### Ejemplo 1

```
VAR num exposuretime;
...
exposuretime:=CamGetExposure(mycamera \ExposureTime);
IF exposuretime = 10 THEN
  CamSetExposure mycamera \ExposureTime:=9.5;
ENDIF
```

Ordenar a la cámara mycamera que cambie el tiempo de exposición a 9,5 ms si el ajuste actual es de 10 ms.

---

#### Valor de retorno

Tipo de dato: num

Se devuelve uno de los parámetros de la cámara, ya sea tiempo de exposición, claridad o contraste, en forma de un valor numérico.

---

#### Argumentos

```
CamGetExposure (Camera [\ExposureTime] | [\Brightness] |
[\Contrast])
```

Camera

Tipo de dato: cameradev

El nombre de la cámara.

[\ExposureTime]

Tipo de dato: num

Devuelve el tiempo de exposición de la cámara. El valor se indica en milisegundos (ms).

[\Brightness]

Tipo de dato: num

Devuelve el ajuste de claridad de la cámara.

[\Contrast]

Tipo de dato: num

Devuelve el ajuste de contraste de la cámara.

---

*Continúa en la página siguiente*



**Gestión de errores**

Pueden generarse los errores recuperables enumerados a continuación. Estos errores pueden ser gestionados en un gestor de errores. El valor de la variable de sistema `ERRNO` cambia a:

| Nombre                              | Causa del error   |
|-------------------------------------|---|
| <code>ERR_CAM_BUSY</code>           | La cámara está ocupada con otra petición y no puede ejecutar la orden actual.     |
| <code>ERR_CAM_COM_TIMEOUT</code>    | Error de comunicación con la cámara. Es probable que la cámara esté desconectada. |
| <code>ERR_CAM_NOT_ON_NETWORK</code> | La cámara no está conectada.  |

**Sintaxis**

```
CamGetExposure '('
  [ Camera ':=' ] < variable (VAR) of cameradev >
  ['\ExposureTime]
  | ['\Brightness]
  | ['\Contrast] ')'
```

Una función con un valor de retorno del tipo de dato `num`.

**Información relacionada**

| Para obtener más información sobre | Consulte  |
|------------------------------------|---|
| Integrated Vision                  | <i>Manual de aplicaciones - Integrated Vision</i> |

## 2 Funciones

---

### 2.35 CamGetLoadedJob - Obtiene el nombre de la tarea de cámara cargada *Integrated Vision*

### 2.35 CamGetLoadedJob - Obtiene el nombre de la tarea de cámara cargada

---

#### Utilización

CamGetLoadedJob (*Camera Get Loaded Job*) es una función que lee de la cámara el nombre del trabajo cargado actualmente y lo devuelve en una cadena.

---

#### Ejemplos básicos

El ejemplo siguiente ilustra la función CamGetLoadedJob.

#### Ejemplo 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
```

Escribir en el FlexPendant el nombre del trabajo cargado.

---

#### Valor de retorno

Tipo de dato: string

El nombre del trabajo cargado actualmente en la cámara especificada.

---

#### Argumentos

CamGetLoadedJob (Camera)

Camera

Tipo de dato: cameradev

El nombre de la cámara.

---

#### Ejecución de programas

La función CamGetLoadedJob obtiene de la cámara el nombre del trabajo cargado actualmente. Si no hay ningún trabajo cargado en la cámara, se devuelve una cadena vacía.

---

#### Gestión de errores

Pueden generarse los errores recuperables enumerados a continuación. Estos errores pueden ser gestionados en un gestor de errores. El valor de la variable de sistema ERRNO cambia a:

| Nombre                 | Causa del error   |
|------------------------|---|
| ERR_CAM_BUSY           | La cámara está ocupada con otra petición y no puede ejecutar la orden actual.     |
| ERR_CAM_COM_TIMEOUT    | Error de comunicación con la cámara. Es probable que la cámara esté desconectada. |
| ERR_CAM_NOT_ON_NETWORK | La cámara no está conectada.  |

*Continúa en la página siguiente*

### 2.35 CamGetLoadedJob - Obtiene el nombre de la tarea de cámara cargada

*Integrated Vision*

*Continuación*

#### Sintaxis

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

Una función con un valor de retorno del tipo de dato `string`.

#### Información relacionada

| Para obtener más información sobre | Consulte  |
|------------------------------------|---|
| Integrated Vision                  | <i>Manual de aplicaciones - Integrated Vision</i> |

## 2 Funciones

---

### 2.36 CamGetMode: obtener el modo actual de la cámara *Integrated Vision*

### 2.36 CamGetMode: obtener el modo actual de la cámara

---

#### Utilización

CamGetMode devuelve el modo actual de una cámara.

---

#### Ejemplos básicos

El ejemplo siguiente ilustra la función CamGetMode.

##### Ejemplo 1

```
VAR camerastatus curr_camerastatus;  
...  
curr_camerastatus:=CamGetMode(mycamera);  
IF curr_camerastatus = CAMERA_DISCON THEN  
    TPWrite "The camera is disconnected. Check cabling for camera  
           "+CamGetName(mycamera);  
ENDIF
```

Obtener el modo actual de la cámara. Si la cámara está desconectada, escribir esto en el FlexPendant.

---

#### Valor de retorno

Tipo de dato: camerastatus

El estado actual de la cámara.

Sólo es posible usar constantes simbólicas predefinidas del tipo camerastatus para comprobar el estado.

---

#### Argumentos

CamGetMode( Camera )

Camera

Tipo de dato: cameradev

La cámara cuyo estado se desea averiguar.

---

#### Ejecución de programas

Esta función devuelve uno de los siguientes estados predefinidos de la cámara:

---

#### Sintaxis

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

Una función con un valor de retorno del tipo de dato camerastatus.

---

#### Información relacionada

| Para obtener más información sobre | Consulte  |
|------------------------------------|---|
| Integrated Vision                  | <i>Manual de aplicaciones - Integrated Vision</i>                                   |
| Tipo de dato camerastatus          | <a href="#">camerastatus: estado de comunicación de la cámara en la página 1618</a> |
| Tipo de dato cameradev             | <a href="#">cameradev - dispositivo de cámara en la página 1617</a>                 |

---

## 2.37 CamGetName - Obtiene el nombre de la cámara utilizada

### Utilización

CamGetName (Camera Get Name) se utiliza para obtener el nombre configurado de la cámara.

### Ejemplos básicos

El ejemplo siguiente ilustra la función CamGetName.

#### Ejemplo 1

```

...
logcameraname camera1;
CamReqImage camera1;
...
logcameraname camera2;
CamReqImage camera2;
...
PROC logcameraname(VAR cameradev camdev)
  TPWrite "Now using camera: "+CamGetName(camdev);
ENDPROC

```

Este procedimiento registra en el FlexPendant el nombre de la cámara utilizada actualmente.

### Valor de retorno

Tipo de dato: string

Este procedimiento registra en el FlexPendant el nombre de la cámara utilizada actualmente.

### Argumentos

CamGetName (Camera)

Camera

Tipo de dato: cameradev

El nombre de la cámara.

### Sintaxis

```

CamGetName( '('
  [ Camera ::= ] < variable (VAR) of cameradev > ') '

```

Una función con un valor de retorno del tipo de dato string.

### Información relacionada

| Para obtener más información sobre | Consulte  |
|------------------------------------|---|
| Integrated Vision                  | <i>Manual de aplicaciones - Integrated Vision</i> |

## 2 Funciones

---

### 2.38 CamNumberOfResults - Obtiene el número de resultados disponibles

*Integrated Vision*

### 2.38 CamNumberOfResults - Obtiene el número de resultados disponibles

---

#### Utilización

`CamNumberOfResults` (Camera Number of Results) es una función que lee el número de resultados de visión disponibles y lo devuelve en forma de un valor numérico.

---

#### Ejemplos básicos

El ejemplo siguiente ilustra la función `CamNumberOfResults`.

#### Ejemplo 1

```
VAR num foundparts;  
...  
CamReqImage mycamera;  
WaitTime 1;  
FoundParts := CamNumberOfResults(mycamera);  
TPWrite "Number of identified parts in the camera image:  
        "\Num:=foundparts;
```

Adquirir una imagen. Esperar a que se complete el procesamiento de la imagen, en este caso 1 segundo. Leer el número de piezas identificadas y escribirlo en el FlexPendant.

---

#### Valor de retorno

Tipo de dato: num

Devuelve el número de resultados de la colección de la cámara especificada.

---

#### Argumentos

```
CamNumberOfResults (Camera [\SceneId])
```

Camera

Tipo de dato: cameradev

El nombre de la cámara.

[\SceneId]

*Identificación de escena*

Tipo de dato: num

El `SceneId` es un identificador que especifica de qué imagen debe leerse el número de piezas identificadas.

---

#### Ejecución de programas

`CamNumberOfResults` es una función que lee el número de resultados de visión disponibles y lo devuelve en forma de un valor numérico. Puede usarse para recorrer en un bucle todos los resultados disponibles.

La función devuelve directamente el nivel de la cola al ejecutar la función. Si la función se ejecuta directamente tras solicitar una imagen, el resultado es con frecuencia 0 porque la cámara aún no ha terminado de procesar la imagen.

---

*Continúa en la página siguiente*

**Gestión de errores**

Pueden generarse los errores recuperables enumerados a continuación. Estos errores pueden ser gestionados en un gestor de errores. El valor de la variable de sistema `ERRNO` cambia a:

| Nombre                    | Causa del error   |
|---------------------------|---|
| <code>ERR_CAM_BUSY</code> | La cámara está ocupada con otra petición y no puede ejecutar la orden actual. |

**Sintaxis**

```
CamNumberOfResults '('
  [ Camera ':= ' ] < variable (VAR) of cameradev >
  [ '\SceneId ':= ' < expression (IN) of num > ] ')'
```

Una función con un valor de retorno del tipo de dato `num`.

**Información relacionada**

| Para obtener más información sobre | Consulte  |
|------------------------------------|---|
| Integrated Vision                  | <i>Manual de aplicaciones - Integrated Vision</i> |

## 2 Funciones

### 2.39 CapGetFailSigs - Obtener señales de E/S con fallos *Continuous Application Platform (CAP)*

### 2.39 CapGetFailSigs - Obtener señales de E/S con fallos

#### Utilización

CapGetFailSigs se utiliza para devolver los nombres de la señal o señales que fallaron durante la supervisión de CapL o CapC.

Si la supervisión de una o varias señales falla durante el proceso se devuelve un error recuperable desde la instrucción CapL/CapC. Para determinar qué señal o señales fallaron, puede utilizarse CapGetFailSigs en un gestor de errores para todos los casos de errores de supervisión.

#### Ejemplo básico

```
Stringcopied := CapGetFailSigs(Failstring);
```

Stringcopied se asigna al valor TRUE si la copia es correcta y FALSE si la copia falla.

Failstring contiene las señales que fallaron, en forma de texto. Si no pudo copiarse la cadena, se devuelve la cadena EMPTY.

#### Valor de retorno

Tipo de dato: bool

TRUE o FALSE en función de si la cadena de fallo se modifica.

#### Argumentos

```
CapGetFailSigs (ErrorNames)
```

#### ErrorNames

Tipo de dato: string

CapGetFailSigs requiere una variable de cadena como parámetro de entrada.

#### Limitaciones

Si fallan muchas señales de una lista de supervisión a la vez, solo tres de ellas se indican con CapGetFailSigs.

#### Sintaxis

```
CapGetFailSigs '('  
[ErrorNames ':=' ] < variable (INOUT) of string >')'
```

Una función con un valor de retorno del tipo de dato bool.

#### Información relacionada

| Para obtener más información sobre     | Consulte  |
|--|---|
| <i>Continuous Application Platform</i> | <i>Application manual - Continuous Application Platform</i>   |
| Instrucción InitSuperv                 | <a href="#">InitSuperv - Restablecer toda la supervisión para CAP en la página 286</a>                          |
| Instrucción SetupSuperv                | <a href="#">SetupSuperv - Configurar las condiciones para la supervisión de señales en CAP en la página 719</a> |

Continúa en la página siguiente



### 2.39 CapGetFailSigs - Obtener señales de E/S con fallos *Continuous Application Platform (CAP)* Continuación

| Para obtener más información sobre    | Consulte   |
|---------------------------------------|--|
| Instrucción <code>RemoveSuperv</code> | <a href="#">RemoveSuperv - Eliminar la condición de una señal en la página 602</a> |

## 2 Funciones

---

### 2.40 CDate - Lee la fecha actual como una cadena RobotWare-OS

### 2.40 CDate - Lee la fecha actual como una cadena

---

#### Utilización

CDate (*Current Date*) se utiliza para leer la fecha actual del sistema.

Esta función puede usarse para mostrar la fecha actual al operador en la pantalla del FlexPendant o para pegar la fecha actual en un archivo de texto en el que se escribe desde el programa.

---

#### Ejemplos básicos

El ejemplo siguiente ilustra la función CDate.

Consulte también [Más ejemplos en la página 1230](#).

#### Ejemplo 1

```
VAR string date;  
date := CDate();
```

La fecha actual se almacena en la variable `date`.

---

#### Valor de retorno

Tipo de dato: `string`

La fecha actual en una cadena.

El formato de fecha estándar es “año-mes-día”, por ejemplo “1998-01-29”.

---

#### Más ejemplos

A continuación aparecen más ejemplos de la función CDate.

#### Ejemplo 1

```
VAR string date;  
date := CDate();  
TPWrite "The current date is: "+date;  
Write logfile, date;
```

Se escribe la fecha actual en la pantalla del FlexPendant y dentro de un archivo de texto.

---

#### Sintaxis

```
CDate '(' ' ' )'
```

Una función con un valor de retorno del tipo de dato `string`.

---

#### Información relacionada

| Para obtener más información sobre   | Consulte   |
|--------------------------------------|--|
| Instrucciones de tiempo              | <i>Manual de referencia técnica - RAPID Overview</i> |
| Cambio de hora del reloj del sistema | <i>Manual del operador - IRC5 con FlexPendant</i>    |

## 2.41 CJointT - Lee los ángulos actuales de los ejes

### Utilización

CJointT (*Current Joint Target*) se utiliza para leer los ángulos actuales de los ejes del robot y los ejes externos.

### Ejemplos básicos

El ejemplo siguiente ilustra la función CJointT.

Consulte también [Más ejemplos en la página 1231](#).

#### Ejemplo 1

```
VAR jointtarget joints;
joints := CJointT();
```

Los ángulos actuales de los ejes de un robot y de los ejes externos se almacenan en joints.

### Valor de retorno

Tipo de dato: jointtarget

Los ángulos actuales en grados de los ejes del robot en el lado del brazo.

Los valores actuales de los ejes externos, en mm para los ejes lineales y en grados para los ejes de rotación.

Los valores devueltos están relacionados con la posición de calibración.

### Argumentos

```
CJointT ([\TaskRef][\TaskName])
```

[\TaskRef]

*Task Reference*

Tipo de dato: taskid

La identidad de tarea de programa desde la cual debe leerse jointtarget.

Existen variables predefinidas con el tipo de dato taskid para todas las tareas de programa del sistema. La identificación de la variable será "nombre\_tarea"+"ID". Por ejemplo, para la tarea T\_ROB1 la identificación de la variable es T\_ROB1Id.

[\TaskName]

Tipo de dato: string

El nombre de tarea de programa desde la cual debe leerse jointtarget.

Si no se especifica ninguno de los argumentos, ni \TaskRef ni \TaskName, se usa la tarea actual.

### Más ejemplos

A continuación aparecen más ejemplos de la función CJointT.

#### Ejemplo 1

```
! In task T_ROB1
VAR jointtarget joints;
joints := CJointT(\TaskRef:=T_ROB2Id);
```

*Continúa en la página siguiente*

## 2 Funciones

### 2.41 CJointT - Lee los ángulos actuales de los ejes

RobotWare - OS

Continuación

La posición actual de los ejes del robot y los ejes externos de la tarea T\_ROB2 se almacena en `joints` en la tarea T\_ROB1.

Recuerde que el robot de la tarea T\_ROB2 puede estar moviéndose al leer la posición. Para asegurarse de que el robot esté en reposo, puede programarse un punto de paro `fine` en la instrucción de movimiento precedente de la tarea T\_ROB2 y usar la instrucción `WaitSyncTask` para sincronizar las instrucciones de la tarea T\_ROB1.

#### Ejemplo 2

```
! In task T_ROB1
VAR jointtarget joints;
joints := CJointT(\TaskName:="T_ROB2");
```

El mismo efecto que en el ejemplo 1 anterior.

#### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                         | Causa del error   |
|--------------------------------|---|
| <code>ERR_TASKNAME</code>      | El nombre de tarea de programa en el argumento <code>\TaskName</code> no puede encontrarse en el sistema. |
| <code>ERR_NOT_MOVE TASK</code> | Argumento <code>\TaskRef</code> o <code>\TaskName</code> especifica alguna tarea sin movimiento.          |

No se generará ningún error si el argumento `\TaskRef` o `\TaskName` especifica la tarea sin movimiento que ejecuta esta función `CJointT` (referencia a mi propia tarea sin movimiento). En este caso, la posición se captura de la tarea de movimiento conectada.

#### Sintaxis

```
CJointT '('
  ['\TaskRef ':<variable (VAR) of taskid>]
  ['\TaskName ':<expression (IN) of string>] ')'
```

Una función con un valor de retorno del tipo de dato `jointtarget`.

#### Información relacionada

| Para obtener más información sobre  | Consulte  |
|-------------------------------------|---|
| Definición de ejes                  | <a href="#">jointtarget - Datos de posición de eje en la página 1691</a>              |
| Lectura del ángulo actual del motor | <a href="#">ReadMotor - Lee los ángulos actuales de los motores en la página 1440</a> |

---

## 2.42 ClkRead - Lee un reloj utilizado para la temporización

---

### Utilización

ClkRead se utiliza para leer un reloj que funciona como un cronómetro para funciones de temporización.

---

### Ejemplos básicos

Los siguientes ejemplos ilustran la función ClkRead.

#### Ejemplo 1

```
reg1:=ClkRead(clock1);
```

Se lee el reloj `clock1` y su tiempo en segundos se almacena en la variable `reg1`.

#### Ejemplo 2

```
reg1:=ClkRead(clock1 \HighRes);
```

Se lee el reloj `clock1` y su tiempo en segundos se almacena con alta resolución en la variable `reg1`.

---

### Valor de retorno

Tipo de dato: num

El tiempo almacenado en el reloj, en segundos. La resolución es normalmente de 0,001 segundos. Si se utiliza el modificador `HighRes`, es posible obtener una resolución de 0,000001 segundos.

---

### Argumento

```
ClkRead (Clock \HighRes)
```

Reloj

Tipo de dato: clock

El nombre del reloj a leer.

[ \HighRes ]

*High Resolution*

Tipo de dato: switch

Especifica que el tiempo debe leerse con una resolución mayor. Si se utiliza este modificador, es posible leer el tiempo con una resolución de 0,000001 segundos.

Debido a la precisión del tipo de dato num, sólo puede obtener la resolución en microsegundos mientras el valor leído sea inferior a 1 segundo.

---

### Ejecución de programas

El reloj puede leerse tanto cuando está parado como cuando está en marcha.

Después de leer el reloj, es posible leerlo de nuevo, ponerlo otra vez en marcha, pararlo o ponerlo a cero.

*Continúa en la página siguiente*

## 2 Funciones

---

### 2.42 ClkRead - Lee un reloj utilizado para la temporización

RobotWare-OS

Continuación

---

#### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                    | Causa del error  |
|---------------------------|--|
| <code>ERR_OVERFLOW</code> | El reloj funciona durante 4.294.967 segundos (49 días, 17 horas, 2 minutos y 47 segundos), tras lo cual se desborda. |

Si se utiliza el modificador `HighRes`, el error `ERR_OVERFLOW` no puede producirse, pero el reloj comenzará de nuevo por el principio tras aproximadamente 49700 días.

---

#### Sintaxis

```
ClkRead '('  
  [ Clock ':= ' ] < variable (VAR) of clock >  
  [ '\ ' HighRes ] ')'
```

Una función con un valor de retorno del tipo de dato `num`.

---

#### Información relacionada

| Para obtener más información sobre | Consulte  |
|------------------------------------|---|
| Instrucciones de reloj             | <i>Manual de referencia técnica - RAPID Overview</i>  |
| Más ejemplos                       | <a href="#">ClkStart - Pone en marcha un reloj utilizado para la temporización en la página 149</a> |

## 2.43 CorrRead - Lee los offsets totales actuales

### Utilización

`CorrRead` se utiliza para leer todas las correcciones entregadas por todos los generadores de correcciones conectados.

`CorrRead` puede usarse para:

- Determinar hasta qué punto son distintas la trayectoria actual y la trayectoria original.
- Tomar las acciones oportunas para reducir la diferencia.

### Ejemplos básicos

El ejemplo siguiente ilustra la función `CorrRead`.

Consulte también [Más ejemplos en la página 1235](#).

#### Ejemplo 1

```
VAR pos offset;
...
offset := CorrRead();
```

Los offsets actuales proporcionados por todos los generadores de correcciones conectados están disponibles en la variable `offset`.

### Valor de retorno

Tipo de dato: `pos`

Los offsets totales absolutos proporcionados hasta el momento por todos los generadores de correcciones conectados.

### Más ejemplos

Para ver más ejemplos de la función `CorrRead`, consulte [Instrucciones - CorrCon](#).

### Sintaxis

```
CorrRead '(' '')
```

Una función con un valor de retorno del tipo de dato `pos`.

### Información relacionada

| Para obtener más información sobre                   | Consulte   |
|--|--|
| Conexión con un generador de correcciones            | <a href="#">CorrCon - Establece una conexión con un generador de correcciones en la página 177</a> |
| Desconexión de un generador de correcciones          | <a href="#">CorrDiscon - Cierra la conexión con un generador de correcciones en la página 182</a>  |
| Escritura en un generador de correcciones            | <a href="#">CorrWrite - Escribe en un generador de correcciones en la página 183</a>               |
| Eliminación de todos los generadores de correcciones | <a href="#">CorrClear - Elimina todos los generadores de correcciones en la página 176</a>         |
| Descriptor de corrección                             | <a href="#">corrdescr - Descriptor de generador de correcciones en la página 1655</a>              |

## 2 Funciones

---

### 2.44 Cos - Calcula el valor de coseno

RobotWare - OS

### 2.44 Cos - Calcula el valor de coseno

---

#### Utilización

`Cos` (*Cosine*) se utiliza para calcular el valor de coseno de un valor de ángulo de los tipos de datos `num`.

---

#### Ejemplos básicos

El ejemplo siguiente ilustra la función `Cos`.

#### Ejemplo 1

```
VAR num angle;  
VAR num value;  
...  
...  
value := Cos(angle);  
value obtiene el valor de coseno de angle.
```

---

#### Valor de retorno

**Tipo de dato:** `num`  
El valor del coseno en el rango `[-1, 1]`.

---

#### Argumentos

`Cos (Angle)`

Angle

**Tipo de dato:** `num`  
El valor del ángulo, expresado en grados.

---

#### Sintaxis

```
Cos '('  
  [Angle ':=' ] <expression (IN) of num> ')'  
Una función con un valor de retorno del tipo de dato num.
```

---

#### Información relacionada

| Para obtener más información sobre    | Consulte   |
|---------------------------------------|--|
| Instrucciones y funciones matemáticas | <i>Manual de referencia técnica - RAPID Overview</i> |

---



## 2.45 CosDnum - Calcula el valor de coseno

### Utilización

CosDnum (*Cosine dnum*) se utiliza para calcular el valor de coseno de un valor de ángulo de los tipos de datos dnum.

### Ejemplos básicos

El ejemplo siguiente ilustra la función CosDnum.

#### Ejemplo 1

```
VAR dnum angle;
VAR dnum value;
...
...
value := CosDnum(angle);
```

value **obtiene el valor de coseno de** angle.

### Valor de retorno

Tipo de dato: dnum  
El valor del coseno en el rango [-1, 1] .

### Argumentos

CosDnum (Angle)

Angle

Tipo de dato: dnum  
El valor del ángulo, expresado en grados.

### Sintaxis

```
CosDnum '('
  [Angle ':=' ] <expression (IN) of dnum> ')'
```

Una función con un valor de retorno del tipo de dato dnum.

### Información relacionada

| Para obtener más información sobre    | Consulte   |
|---------------------------------------|--|
| Instrucciones y funciones matemáticas | <i>Manual de referencia técnica - RAPID Overview</i> |

## 2 Funciones

---

### 2.46 CPos - Lee los datos de posición (pos) actuales

RobotWare - OS

### 2.46 CPos - Lee los datos de posición (pos) actuales

---

#### Utilización

CPos (*Current Position*) se utiliza para leer la posición actual del robot.

Esta función devuelve los valores x, y, z del TCP del robot, en un dato del tipo pos. Si es necesario leer la posición completa del robot (robtarget), utilice en su lugar la función CRobT.

---

#### Ejemplos básicos

El ejemplo siguiente ilustra la función CPos.

Consulte también [Más ejemplos en la página 1239](#).

#### Ejemplo 1

```
VAR pos pos1;

MoveL *, v500, fine \Inpos := inpos50, tool1;
pos1 := CPos(\Tool:=tool1 \WObj:=wobj0);
```

La posición actual del TCP del robot se almacena en la variable pos1. La herramienta tool1 y el objeto de trabajo wobj0 se utilizan para calcular la posición.

Recuerde que el robot está en reposo antes de que se lea y calcule la posición. Esto se consigue utilizando el punto de paro fine con la exactitud de posición inpos50 en la instrucción de movimiento anterior.

---

#### Valor de retorno

Tipo de dato: pos

La posición actual (pos) del robot con x, y, z en el sistema de coordenadas más externo, teniendo en cuenta la herramienta especificada, el objeto de trabajo y el sistema de coordenadas ProgDisp activo.

---

#### Argumentos

```
CPos([\Tool] [\WObj])
```

[\Tool]

Tipo de dato: tooldata

La herramienta utilizada para el cálculo de la posición actual del robot.

Si se omite este argumento, se utiliza la herramienta activa actualmente.

[\WObj]

*Work Object*

Tipo de dato: wobjdata

El objeto de trabajo (sistema de coordenadas) con el que está relacionada la posición actual del robot devuelta por la función.

Si se omite este argumento, se utiliza el objeto de trabajo activo actualmente.

---

*Continúa en la página siguiente*

**¡AVISO!**

Resulta aconsejable especificar siempre los argumentos `\Tool` y `\WObj` durante la programación. De esta forma, la posición devolverá siempre la posición deseada, incluso si se ha activado otra herramienta u otro objeto de trabajo.

**Ejecución de programas**

Las coordenadas devueltas representan la posición del TCP en el sistema de coordenadas ProgDisp.

**Más ejemplos**

A continuación aparecen más ejemplos de la función `CPos`.

```
VAR pos pos2;
VAR pos pos3;
VAR pos pos4;

pos2 := CPos(\Tool:=grip3 \WObj:=fixture);
...
pos3 := CPos(\Tool:=grip3 \WObj:=fixture);
pos4 := pos3-pos2;
```

La posición `x`, `y`, `z` del robot se captura en dos puntos dentro del programa, mediante la función `CPos`. La herramienta `grip3` y el objeto de trabajo `fixture` se utilizan para calcular la posición. Se calculan las distancias `x`, `y`, `z` recorridas entre estas posiciones y se almacenan dichas distancias en la variable `pos4`.

**Sintaxis**

```
CPos '('
  ['\' Tool :=' <persistent (PERS) of tooldata>]
  ['\' WObj :=' <persistent (PERS) of wobjdata>] ')'
```

Una función con un valor de retorno del tipo de dato `pos`.

**Información relacionada**

| Para obtener más información sobre | Consulte   |
|------------------------------------|--|
| Definición de posición             | <a href="#">pos - Posiciones (sólo X, Y y Z) en la página 1729</a>                     |
| Definición de herramientas         | <a href="#">tooldata - Datos de herramienta en la página 1795</a>                      |
| Definición de objetos de trabajo   | <a href="#">wobjdata - Datos del objeto de trabajo en la página 1823</a>               |
| Sistema de coordenadas Prog-Disp   | <a href="#">PDispOn - Activa el desplazamiento de programa en la página 533</a>        |
| Sistemas de coordenadas            | <a href="#">Manual de referencia técnica - RAPID Overview</a>                          |
| Lectura del <code>robtarg</code>   | <a href="#">CRobT - Lee los datos de posición (robtarg) actuales en la página 1240</a> |

## 2 Funciones

---

### 2.47 CRobT - Lee los datos de posición (robtargt) actuales RobotWare - OS

### 2.47 CRobT - Lee los datos de posición (robtargt) actuales

---

#### Utilización

CRobT(*Current Robot Target*) se utiliza para leer la posición actual de los ejes del robot y los ejes externos.

Esta función devuelve un valor de `robtargt` con la posición (x, y, z), la orientación (q1 ... q4), la configuración de los ejes del robot y la posición de los ejes externos. Si sólo es necesario leer los valores x, y, z del TCP del robot (`pos`), utilice en su lugar la función `CPos`.

---

#### Ejemplos básicos

El ejemplo siguiente ilustra la función CRobT.

Consulte también [Más ejemplos en la página 1241](#).

#### Ejemplo 1

```
VAR robtarget p1;  
MoveL *, v500, fine \Inpos := inpos50, tool1;  
p1 := CRobT(\Tool:=tool1 \Wobj:=wobj0);
```

La posición actual de los ejes del robot y los ejes externos se almacena en `p1`. La herramienta `tool1` y el objeto de trabajo `wobj0` se utilizan para calcular la posición.

Recuerde que el robot está en reposo antes de que se lea y calcule la posición. Esto se consigue utilizando el punto de paro `fine` con la exactitud de posición `inpos50` en la instrucción de movimiento anterior.

---

#### Valor de retorno

Tipo de dato: `robtargt`

La posición actual de los ejes de un robot y de los ejes externos en el sistema de coordenadas más externo, teniendo en cuenta la herramienta especificada, el objeto de trabajo y el sistema de coordenadas `ProgDisp/ExtOffs` activo.

---

#### Argumentos

```
CRobT ([\TaskRef][\TaskName] [\Tool] [\Wobj])
```

`[\TaskRef]`

##### *Task Reference*

Tipo de dato: `taskid`

La identidad de tarea de programa desde la cual debe leerse `robtargt`.

Existen variables predefinidas con el tipo de dato `taskid` para todas las tareas de programa del sistema. La identificación de la variable será "nombre\_tarea"+"ID". Por ejemplo, para la tarea `T_ROB1` la identificación de la variable es `T_ROB1Id`.

`[\TaskName]`

Tipo de dato: `string`

El nombre de tarea de programa desde la cual debe leerse `robtargt`.

Si no se especifica ninguno de los argumentos, ni `\TaskRef` ni `\TaskName`, se usa la tarea actual.

---

*Continúa en la página siguiente*

[\Tool]

Tipo de dato: tooldata

La variable persistente de la herramienta utilizada para calcular la posición actual del robot.

Si se omite este argumento, se utiliza la herramienta activa actualmente.

[\WObj]

*Work Object*

Tipo de dato: wobjdata

La variable persistente del objeto de trabajo (sistema de coordenadas) con el que está relacionada la posición actual del robot devuelta por la función.

Si se omite este argumento, se utiliza el objeto de trabajo activo actualmente.

**¡AVISO!**

Resulta aconsejable especificar siempre los argumentos `\Tool` y `\WObj` durante la programación. De esta forma, la posición devolverá siempre la posición deseada, incluso si se ha activado otra herramienta u otro objeto de trabajo.

### Ejecución de programas

Las coordenadas devueltas representan la posición del TCP en el sistema de coordenadas `ProgDisp`. Los ejes externos se representan en el sistema de coordenadas `ExtOffs`.

Si uno de los argumentos `\TaskRef` o `\TaskName` se usa pero no así los argumentos `Tool` y `WObj`, se usarán la herramienta actual y el objeto de trabajo de la tarea específica.

### Más ejemplos

A continuación aparecen más ejemplos de la función `CRobT`.

#### Ejemplo 1

```
VAR robtargt p2;
p2 := ORobT( CRobT(\Tool:=grip3 \WObj:=fixture) );
```

La posición actual de los ejes del robot y de los ejes externos en el sistema de coordenadas del objeto (sin `ProgDisp` ni `ExtOffs`) se almacena en `p2`. La herramienta `grip3` y el objeto de trabajo `fixture` se utilizan para calcular la posición.

#### Ejemplo 2

```
! In task T_ROB1
VAR robtargt p3;
p3 := CRobT(\TaskRef:=T_ROB2Id \Tool:=tool1 \WObj:=wobj0);
```

La posición actual de los ejes del robot y los ejes externos de la tarea `T_ROB2` se almacena en `p3` en la tarea `T_ROB1`. La herramienta `tool1` y el objeto de trabajo `wobj0` se utilizan para calcular la posición.

Recuerde que el robot de la tarea `T_ROB2` puede estar moviéndose al leer y calcular la posición. Para asegurarse de que el robot esté en reposo, puede programarse

*Continúa en la página siguiente*

## 2 Funciones

### 2.47 CRobT - Lee los datos de posición (robtargt) actuales

RobotWare - OS

Continuación

un punto de paro `fine` en la instrucción de movimiento precedente de la tarea `T_ROB2` y usar la instrucción `WaitSyncTask` para sincronizar las instrucciones de la tarea `T_ROB1`.

#### Ejemplo 3

```
! In task T_ROB1
VAR robtarget p4;
p4 := CRobT(\TaskName:="T_ROB2");
```

La posición actual de los ejes del robot y los ejes externos de la tarea `T_ROB2` se almacena en `p4` en la tarea `T_ROB1`. La herramienta actual y el objeto de trabajo de la tarea `T_ROB2` se utilizan para calcular la posición.

#### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                        | Causa del error   |
|-------------------------------|---|
| <code>ERR_TASKNAME</code>     | El nombre de tarea de programa en el argumento <code>\TaskName</code> no puede encontrarse en el sistema. |
| <code>ERR_NOT_MOVETASK</code> | Argumento <code>\TaskRef</code> o <code>\TaskName</code> especifica alguna tarea sin movimiento.          |

No se generará ningún error si los argumentos `\TaskRef` o `\TaskName` especifican la tarea sin movimiento que ejecuta esta función `CRobT` (referencia a mi propia tarea sin movimiento). En este caso, la posición se captura de la tarea de movimiento conectada.

#### Sintaxis

```
CRobT '('  
  ['\' TaskRef ':=' <variable (VAR) of taskid>]  
  |['\' TaskName ':=' <expression (IN) of string>]  
  |['\' Tool ':=' <persistent (PERS) of tooldata>]  
  |['\' WObj ':=' <persistent (PERS) of wobjdata>] ')'
```

Una función con un valor de retorno del tipo de dato `robtargt`.

#### Información relacionada

| Para obtener más información sobre                 | Consulte  |
|--|---|
| Definición de posición                             | <a href="#">robtargt - Datos de posición en la página 1750</a>                    |
| Definición de herramientas                         | <a href="#">tooldata - Datos de herramienta en la página 1795</a>                 |
| Definición de objetos de trabajo                   | <a href="#">wobjdata - Datos del objeto de trabajo en la página 1823</a>          |
| Sistemas de coordenadas                            | <a href="#">Manual de referencia técnica - RAPID Overview</a>                     |
| Sistema de coordenadas ProgDisp                    | <a href="#">PDispOn - Activa el desplazamiento de programa en la página 533</a>   |
| Sistema de coordenadas ExtOffs                     | <a href="#">EOffsOn - Activa un offset de ejes adicionales en la página 195</a>   |
| Lectura del <code>pos</code> actual (sólo x, y, z) | <a href="#">CPos - Lee los datos de posición (pos) actuales en la página 1238</a> |

## 2.48 CrossProd - Producto cruzado de dos vectores pos

### Utilización

CrossProd (*Cross Product*) se usa para calcular el producto cruzado (o producto de vector) de dos vectores pos.

El producto cruzado de dos vectores **A** y **B** es un vector perpendicular al argumento de ambos vectores. La longitud del vector resultante es igual a los productos de la longitud de **A** y **B** y el seno del ángulo entre ellos  $\theta_{AB}$ .

$$|A \times B| = |A||B| \sin \theta_{AB}$$



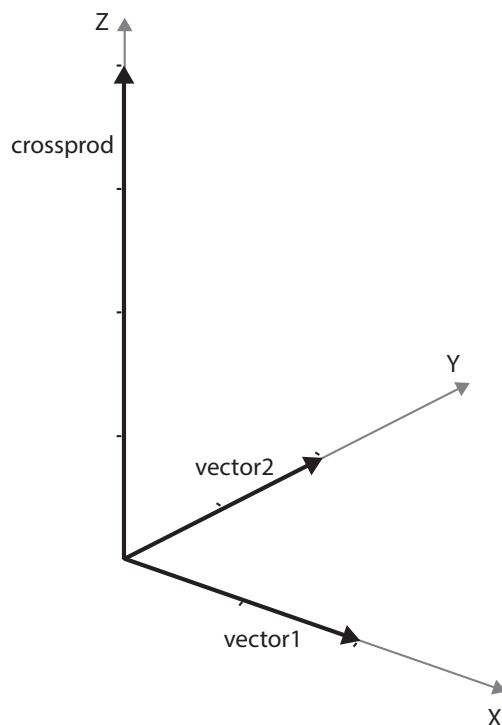
#### Nota

- La magnitud del producto cruzado es igual al área de un paralelogramo con los vectores como lados.
- El producto cruzado de dos vectores en paralelo es cero.
- $A \times B = - B \times A$

### Ejemplos básicos

El ejemplo siguiente ilustra la función CrossProd con vectores perpendiculares. Para ver otros ejemplos, consulte [Más ejemplos en la página 1244](#).

#### Ejemplo 1



xx1700001570

```
VAR pos crossprod_1;
VAR pos vector1;
```

*Continúa en la página siguiente*

## 2 Funciones

### 2.48 CrossProd - Producto cruzado de dos vectores pos

RobotWare - OS

Continuación

```
VAR pos vector2;  
...  
...  
vector1 := [2,0,0];  
vector2 := [0,2,0];  
crossprod_1 := CrossProd(vector1, vector2);
```

En este ejemplo, `vector1` está en paralelo con el eje `x`, `vector2` está en paralelo con el eje `y`. El producto cruzado es perpendicular a ambos, es decir, en paralelo con el eje `z`.

Ya que el ángulo entre `vector1` y `vector2` es  $90^\circ$ , la magnitud del producto cruzado es:  $2 * 2 * \sin 90^\circ = 4$

---

#### Valor de retorno

Tipo de dato: `pos`

Un vector que es el resultado del producto cruzado de dos vectores.

---

#### Argumentos

```
CrossProd (Vector1 Vector2)
```

`Vector1`

Tipo de dato: `pos`

El primer vector descrito con el tipo de dato `pos`.

`Vector2`

Tipo de dato: `pos`

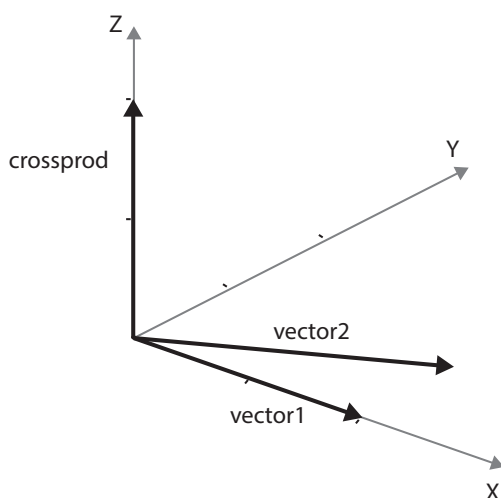
El segundo vector descrito con el tipo de dato `pos`.

---

#### Más ejemplos

A continuación aparecen más ejemplos de la función `CrossProd`.

#### Ejemplo 2



xx1700001571

```
VAR pos crossprod_1;  
VAR pos vector1;
```

Continúa en la página siguiente



### 2.48 CrossProd - Producto cruzado de dos vectores pos RobotWare - OS Continuación

```
VAR pos vector2;
...
...
vector1 := [2,0,0];
vector2 := [2,1,0];
crossprod_1 := CrossProd(vector1, vector2);
```

En este ejemplo, `vector1` y `vector2` se encuentran en el plano xy. El producto cruzado es perpendicular a ambos, es decir, paralelo al eje z.

La magnitud de `vector1` es 2. La magnitud de `vector2` es  $\sqrt{5}$ . El ángulo entre `vector1` y `vector2` es  $26,565^\circ$ . La magnitud del producto cruzado es:

$$2 * \sqrt{5} * \sin 26.565^\circ = 2$$

#### Sintaxis

```
CrossProd '('
  [Vector1 ':=' ] <expression (IN) of pos> ','
  [Vector2 ':=' ] <expression (IN) of pos>
  ')'
```

Una función con un valor de retorno del tipo de dato `pos`.

#### Información relacionada

| Para obtener más información sobre    | Consulte  |
|---------------------------------------|---|
| Instrucciones y funciones matemáticas | <i>Manual de referencia técnica - RAPID Overview, sección Resumen sobre RAPID - Matemáticas</i> |

## 2 Funciones

---

### 2.49 CSpeedOverride - Lee el ajuste de velocidad actual RobotWare - OS

### 2.49 CSpeedOverride - Lee el ajuste de velocidad actual

---

#### Utilización

`CSpeedOverride` se utiliza para leer el ajuste de velocidad definido por el operador desde el FlexPendant. El valor de retorno se muestra como un porcentaje, en el que el 100% corresponde a la velocidad programada.

En aplicaciones que utilizan la instrucción `SpeedRefresh`, esta función también puede usarse para leer el valor de ajuste de velocidad actual para esta tarea de programa de movimiento o la tarea de movimiento conectada.

¡Atención! No debe confundirlo con el argumento `Override` de la instrucción `VelSet` de RAPID.

---

#### Ejemplos básicos

El ejemplo siguiente ilustra la función `CSpeedOverride`.

#### Ejemplo 1

```
VAR num myspeed;  
myspeed := CSpeedOverride();
```

El ajuste de velocidad actual se almacena en la variable `myspeed`. Por ejemplo, si el valor es 100, equivale al 100%.

---

#### Valor de retorno

Tipo de dato: `num`

El valor del ajuste de velocidad, como porcentaje de la velocidad programada. Será un valor numérico en el rango de 0 a 100.

---

#### Argumentos

```
CSpeedOverride ( [ \CTask ] )
```

[ \CTask ]

Tipo de dato: `switch`

Se obtiene el valor de ajuste de velocidad actual para esta tarea de programa de movimiento o la conectada. Se utiliza junto con la instrucción `SpeedRefresh`.

Si no se utiliza este argumento, la función devuelve el ajuste de velocidad actual de todo el sistema (todas las tareas de programa de movimiento). Esto significa el ajuste de velocidad manual, definido desde el FlexPendant.

---

#### Sintaxis

```
CSpeedOverride '('  
  [ '\' CTask ] ')'
```

Una función con un valor de retorno del tipo de dato `num`.

---

#### Información relacionada

| Para obtener más información sobre | Consulte   |
|------------------------------------|--|
| Cambio del ajuste de velocidad     | <i>Manual del operador - IRC5 con FlexPendant, sección Programación y testing de ciclos de producción - Menú de configuración rápida - Velocidad</i> |

Continúa en la página siguiente

### 2.49 CSpeedOverride - Lee el ajuste de velocidad actual

*RobotWare - OS*

*Continuación*

|   |  |
|---|--|
| Para obtener más información sobre                | Consulte   |
| Actualización del ajuste de velocidad desde RAPID | <a href="#">SpeedRefresh - La redefinición de velocidad para el movimiento en curso en la página 785</a> |

## 2 Funciones

---

2.50 CTime - Lee la hora actual en forma de una cadena  
RobotWare-OS

### 2.50 CTime - Lee la hora actual en forma de una cadena

---

#### Utilización

CTime se utiliza para leer la hora actual del sistema.

Esta función puede usarse para mostrar la hora actual al operador en la pantalla del FlexPendant o para pegar la hora actual en un archivo de texto en el que se escribe desde el programa.

---

#### Ejemplos básicos

El ejemplo siguiente ilustra la función CTime.

Consulte también [Más ejemplos en la página 1248](#).

#### Ejemplo 1

```
VAR string time;  
time := CTime();
```

La hora actual se almacena en la variable `time`.

---

#### Valor de retorno

Tipo de dato: `string`

La hora actual en una cadena.

El formato de hora estándar es "horas:minutos:segundos", por ejemplo, "18:20:46".

---

#### Más ejemplos

A continuación aparecen más ejemplos de la función CTime.

#### Ejemplo 1

```
VAR string time;  
time := CTime();  
TPWrite "The current time is: "+time;  
Write logfile, time;
```

Se escribe la hora actual en la pantalla del FlexPendant y dentro de un archivo de texto.

---

#### Sintaxis

```
CTime '(' '')
```

Una función con un valor de retorno del tipo de dato `string`.

---

#### Información relacionada

| Para obtener más información sobre   | Consulte   |
|--------------------------------------|--|
| Instrucciones de hora y fecha        | <i>Manual de referencia técnica - RAPID Overview, sección Resumen sobre RAPID - Sistema &amp; tiempo</i> |
| Cambio de hora del reloj del sistema | <i>Manual del operador - IRC5 con FlexPendant, sección Cómo cambiar la configuración del FlexPendant</i> |

## 2.51 CTool - Lee los datos actuales de la herramienta

### Utilización

`CTool` (*Current Tool*) se utiliza para leer los datos de la herramienta actual.

### Ejemplos básicos

El ejemplo siguiente ilustra la función `CTool`.

#### Ejemplo 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();
```

El valor de la herramienta actual se almacena en la variable `temp_tool`.

### Valor de retorno

Tipo de dato: `tooldata`

Esta función devuelve un valor de tipo `tooldata` que contiene el valor de la herramienta actual, es decir, la última herramienta utilizada en una instrucción de movimiento.

El valor devuelto representa la posición y la orientación del TCP en el sistema de coordenadas del centro de la muñeca. Consulte `tooldata`.

### Argumentos

```
CTool ([\TaskRef][\TaskName])
```

`[\TaskRef]`

#### *Task Reference*

Tipo de dato: `taskid`

La identidad de tarea de programa desde la cual debe leerse la herramienta actual.

Existen variables predefinidas con el tipo de dato `taskid` para todas las tareas de programa del sistema. La identificación de la variable será "nombre\_tarea"+"ID". Por ejemplo, para la tarea `T_ROB1` la identificación de la variable es `T_ROB1Id`.

`[\TaskName]`

Tipo de dato: `string`

El nombre de tarea de programa desde el cual debe leerse la herramienta actual.

Si no se especifica ninguno de los argumentos, ni `\TaskRef` ni `\TaskName`, se usa la tarea actual.

### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                    | Causa del error   |
|---------------------------|---|
| <code>ERR_TASKNAME</code> | El nombre de tarea de programa en el argumento <code>\TaskName</code> no puede encontrarse en el sistema. |

*Continúa en la página siguiente*

## 2 Funciones

---

### 2.51 CTool - Lee los datos actuales de la herramienta

RobotWare - OS

Continuación

| Nombre           | Causa del error  |
|------------------|--|
| ERR_NOT_MOVETASK | Argumento <code>\TaskRef</code> o <code>\TaskName</code> especifica alguna tarea sin movimiento. |

No se generará ningún error si los argumentos `\TaskRef` o `\TaskName` especifican la tarea sin movimiento que ejecuta esta función `CTool` (referencia a mi propia tarea sin movimiento). En este caso, los datos de la herramienta se capturan de la tarea de movimiento conectada.

---

### Sintaxis

```
CTool '('  
  ['\' TaskRef ':'] <variable (VAR) of taskid>  
  |['\' TaskName ':'] <expression (IN) of string> ] ')'
```

Una función con un valor de retorno del tipo de dato `tooldata`.

---

### Información relacionada

| Para obtener más información sobre | Consulte   |
|------------------------------------|--|
| Definición de herramientas         | <a href="#">tooldata - Datos de herramienta en la página 1795</a>  |
| Sistemas de coordenadas            | <i>Manual de referencia técnica - RAPID Overview, sección Principios de movimiento y E/S - Sistemas de coordenadas</i> |

---

## 2.52 CWOBJ - Lee los datos del objeto de trabajo actual

---

### Utilización

CWOBJ (*Current Work Object*) se utiliza para leer los datos del objeto de trabajo actual.

---

### Ejemplos básicos

El ejemplo siguiente ilustra la función CWOBJ.

#### Ejemplo 1

```
PERS wobjdata temp_wobj:= [FALSE, TRUE, "", [[0,0,0], [1,0,0,0]],
  [[0,0,0], [1,0,0,0]]];
temp_wobj := CWOBJ();
```

El valor del objeto de trabajo actual se almacena en la variable temp\_wobj.

---

### Valor de retorno

Tipo de dato: wobjdata

Esta función devuelve un valor de tipo wobjdata que contiene el valor del objeto de trabajo actual, es decir, el último objeto de trabajo utilizado en una instrucción de movimiento.

El valor devuelto representa la posición y la orientación del objeto de trabajo en el sistema de coordenadas mundo. Consulte wobjdata.

---

### Argumentos

CWOBJ ([\TaskRef]|\TaskName)

[\TaskRef]

#### Task Reference

Tipo de dato: taskid

La identidad de tarea de programa desde la cual debe leerse el objeto de trabajo actual.

Existen variables predefinidas con el tipo de dato taskid para todas las tareas de programa del sistema. La identificación de la variable será "nombre\_tarea"+"ID". Por ejemplo, para la tarea T\_ROB1 la identificación de la variable es T\_ROB1Id.

[\TaskName]

Tipo de dato: string

El nombre de tarea de programa desde el cual debe leerse el objeto de trabajo actual.

Si no se especifica ninguno de los argumentos, ni \TaskRef ni \TaskName, se usa la tarea actual.

*Continúa en la página siguiente*

## 2 Funciones

### 2.52 CWObj - Lee los datos del objeto de trabajo actual

RobotWare - OS

Continuación

#### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                         | Causa del error   |
|--------------------------------|---|
| <code>ERR_TASKNAME</code>      | El nombre de tarea de programa en el argumento <code>\TaskName</code> no puede encontrarse en el sistema. |
| <code>ERR_NOT_MOVE TASK</code> | Argumento <code>\TaskRef</code> o <code>\TaskName</code> especifica alguna tarea sin movimiento.          |

No se generará ningún error si los argumentos `\TaskRef` o `\TaskName` especifican la tarea sin movimiento que ejecuta esta función `CWobj` (referencia a mi propia tarea sin movimiento). En este caso, los datos del objeto de trabajo se capturan de la tarea de movimiento conectada.

#### Sintaxis

```
CWobj '('  
    ['\TaskRef :=' <variable (VAR) of taskid>]  
    |['\TaskName :=' <expression (IN) of string>] )'
```

Una función con un valor de retorno del tipo de dato `wobjdata`.

#### Información relacionada

| Para obtener más información sobre | Consulte   |
|------------------------------------|--|
| Definición de objetos de trabajo   | <a href="#">wobjdata - Datos del objeto de trabajo en la página 1823</a>   |
| Sistemas de coordenadas            | <i>Manual de referencia técnica - RAPID Overview, sección Principios de movimiento y E/S - Sistemas de coordenadas</i> |



## 2.53 DecToHex - Convierte de decimal a hexadecimal

### Utilización

DecToHex se usa para convertir un número especificado en una cadena que admita lectura de la base 10 a la base 16.

La cadena resultante se construye con el conjunto de caracteres [0-9,A-F,a-f].

Esta rutina admite los números del 0 al 9223372036854775807 en decimal o 7FFFFFFFFFFFFFFF en hexadecimal.

### Ejemplos básicos

El ejemplo siguiente ilustra la función DecToHex.

#### Ejemplo 1

```
VAR string str;
```

```
str := DecToHex("99999999");
```

Se asigna a la variable `str` el valor "5F5E0FF".

### Valor de retorno

Tipo de dato: `string`

La cadena convertida a la representación hexadecimal a partir del número indicado en la cadena del parámetro de entrada.

### Argumentos

```
DecToHex ( Str )
```

Str

*String*

Tipo de dato: `string`

La cadena a convertir.

### Sintaxis

```
DecToHex '('  
  [ Str ':=' ] <expression (IN) of string> ')'
```

Una función con un valor de retorno del tipo de dato `string`.

### Información relacionada

| Para obtener más información sobre   | Consulte   |
|--------------------------------------|--|
| Funciones para cadenas de caracteres | <i>Manual de referencia técnica - RAPID Overview, sección Resumen sobre RAPID - Funciones para cadenas de caracteres</i> |
| Definición de cadena de caracteres   | <a href="#">string - Cadenas en la página 1778</a>   |
| Valores de cadena de caracteres      | <i>Manual de referencia técnica - RAPID Overview, sección Características básicas - Elementos básicos</i>                |

## 2 Funciones

### 2.54 DefAccFrame - Define una base de coordenadas exacta RobotWare - OS

### 2.54 DefAccFrame - Define una base de coordenadas exacta

#### Utilización

DefAccFrame (*Define Accurate Frame*) se utiliza para definir una base de coordenadas de tres a diez posiciones originales y un número igual de posiciones desplazadas.

#### Descripción

Es posible definir una base de coordenadas cuando se conoce un conjunto de objetivos en dos posiciones diferentes. Por tanto, se utilizan *las mismas posiciones físicas*, pero expresadas de forma diferente.

Debe considerarse con dos enfoques diferentes:

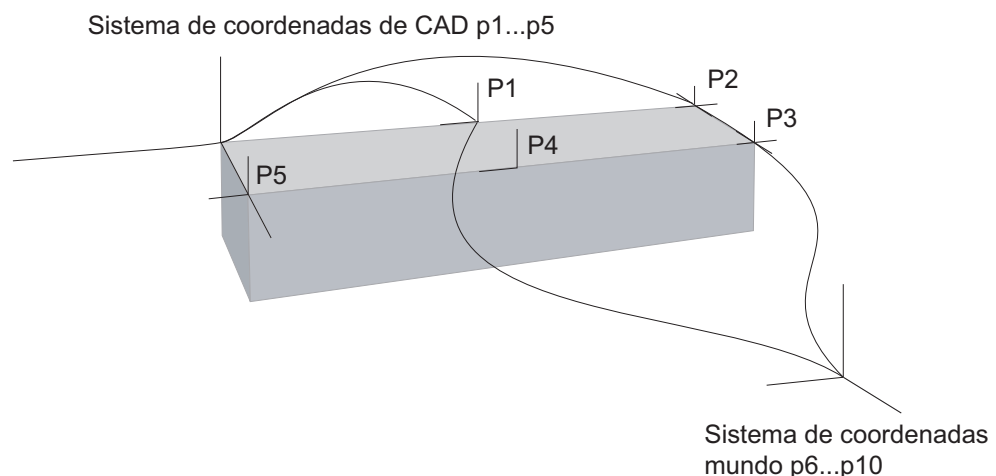
- 1 Se expresan las mismas posiciones físicas respecto de sistemas de coordenadas diferentes. Por ejemplo, se obtienen varias posiciones de un plano de CAD, por lo que las posiciones se expresan en el sistema de coordenadas local del CAD. A continuación, las mismas posiciones se expresan en el sistema de coordenadas mundo del robot. A partir de estos dos conjuntos de posiciones, se calcula la base de coordenadas existente entre el sistema de coordenadas del CAD y el sistema de coordenadas mundo.
- 2 Existe un conjunto de posiciones relacionadas con un objeto en una posición original. Después de un desplazamiento del objeto, se determinan de nuevo las posiciones (buscadas con frecuencia). A partir de estos dos conjuntos de posiciones (posiciones anteriores y nuevas posiciones) se calcula la base de coordenadas de desplazamiento.

Basta con tres objetivos para definir una base de coordenadas, pero deben utilizarse varios puntos para aumentar la exactitud.

#### Ejemplos básicos

El ejemplo siguiente ilustra la función DefAccFrame.

#### Ejemplo 1



Continúa en la página siguiente

## 2.54 DefAccFrame - Define una base de coordenadas exacta

RobotWare - OS

Continuación

```

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 := [...];
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);

```

Se han almacenado cinco posiciones, p1- p5, relacionadas con un objeto. Las cinco posiciones también se almacenan en relación con el sistema de coordenadas mundo como p6-p10. A partir de estas 10 posiciones se calcula la base de coordenadas frame1 existente entre el objeto y el sistema de coordenadas mundo. La base de coordenadas será la base de coordenadas del CAD, expresada en el sistema de coordenadas mundo. Si se intercambia el orden de entrada de las listas de objetivos, por ejemplo, DefAccFrame (pWCS, pCAD...), la base de coordenadas mundo se expresa en el sistema de coordenadas del CAD.

**Valor de retorno**

Tipo de dato: pose

La base de coordenadas TargetListOne calculada se expresa en el sistema de coordenadas TargetListTwo.

**Argumentos**

```

DefAccFrame (TargetListOne TargetListTwo TargetsInList
             MaxErrMeanErr)

```

TargetListOne

Tipo de dato: robtarget

Una matriz de robtargets que contienen las posiciones definidas en el primer sistema de coordenadas. El número mínimo de robtargets es 3 y el máximo es 10.

*Continúa en la página siguiente*

## 2 Funciones

### 2.54 DefAccFrame - Define una base de coordenadas exacta

RobotWare - OS

Continuación

TargetListTwo

Tipo de dato: robtarget

Una matriz de robtargets que contienen las posiciones definidas en el segundo sistema de coordenadas. El número mínimo de robtargets es 3 y el máximo es 10.

TargetsInList

Tipo de dato: num

Número de robtargets de una matriz.

MaxErr

Tipo de dato: num

El error máximo estimado, en mm.

MeanErr

Tipo de dato: num

El error medio estimado, en mm.

### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema ERRNO cambiará a:

| Nombre    | Causa del error   |
|-----------|---|
| ERR_FRAME | Las posiciones no tienen la relación requerida o no están especificadas con suficiente precisión. |

### Sintaxis

```
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> ')'
```

Una función con un valor de retorno del tipo de dato pose.

### Información relacionada

| Para obtener más información sobre                             | Consulte   |
|--|--|
| Cálculo de una base de coordenadas a partir de tres posiciones | <a href="#">DefFrame - Define una base de coordenadas en la página 1260</a>                    |
| Cálculo de una base de coordenadas a partir de 6 posiciones    | <a href="#">DefDFrame - Define una base de coordenadas de desplazamiento en la página 1257</a> |

## 2.55 DefDFrame - Define una base de coordenadas de desplazamiento

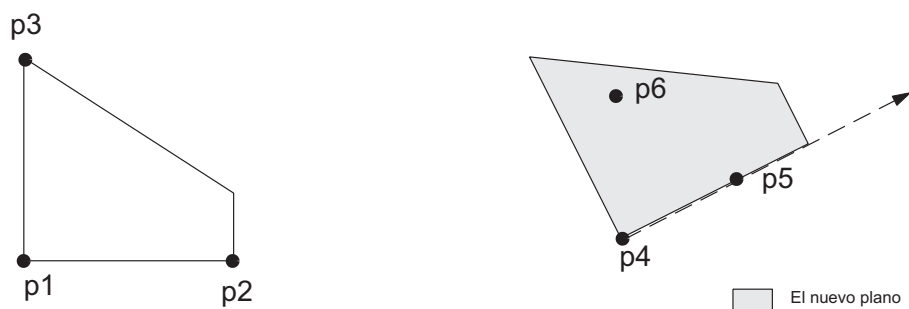
## Utilización

DefDFrame(*Define Displacement Frame*) se utiliza para calcular una base de coordenadas de desplazamiento a partir de tres posiciones originales y tres posiciones desplazadas.

## Ejemplos básicos

El ejemplo siguiente ilustra la función DefDFrame.

## Ejemplo 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;

```

Se han almacenado tres posiciones,  $p1-p3$ , relacionadas con un objeto en una posición original. Después de un desplazamiento del objeto, se buscan tres posiciones nuevas, que se almacenan como  $p4-p6$ . La base de coordenadas de desplazamiento se calcula a partir de estas seis posiciones. A continuación se utiliza la base de coordenadas calculada para desplazar todas las posiciones almacenadas en el programa.

*Continúa en la página siguiente*

## 2 Funciones

### 2.55 DefDFrame - Define una base de coordenadas de desplazamiento

RobotWare - OS

Continuación

---

#### Valor de retorno

Tipo de dato: pose

La base de coordenadas de desplazamiento.

---

#### Argumentos

DefDFrame (OldP1 OldP2 OldP3 NewP1 NewP2 NewP3)

OldP1

Tipo de dato: robtarget

La primera posición original.

OldP2

Tipo de dato: robtarget

La segunda posición original.

OldP3

Tipo de dato: robtarget

La tercera posición original.

NewP1

Tipo de dato: robtarget

La primera posición desplazada. La diferencia entre OldP1 y NewP1 define la parte de traslación de la base de coordenadas y debe medirse y determinarse con una gran exactitud.

NewP2

Tipo de dato: robtarget

La segunda posición desplazada. La línea NewP1 ... NewP2 define la rotación de la línea anterior OldP1 ... OldP2.

NewP3

Tipo de dato: robtarget

La tercera posición desplazada. Esta posición definirá la rotación del plano. Por ejemplo, debe estar situada en el nuevo plano formado por NewP1, NewP2 y NewP3.

---

#### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre    | Causa del error  |
|-----------|--|
| ERR_FRAME | No se puede calcular la base de coordenadas debido a una baja precisión de las posiciones. |

---

#### Sintaxis

```
DefDFrame '('  
  [OldP1 ':=' ] <expression (IN) of robtarget> ','  
  [OldP2 ':=' ] <expression (IN) of robtarget> ','  
  [OldP3 ':=' ] <expression (IN) of robtarget> ','  
  [NewP1 ':=' ] <expression (IN) of robtarget> ','
```

Continúa en la página siguiente

### 2.55 DefDFrame - Define una base de coordenadas de desplazamiento

RobotWare - OS

Continuación

```
[NewP2 ':='] <expression (IN) of robtarget> ','  
[NewP3 ':='] <expression (IN) of robtarget> ')'
```

Una función con un valor de retorno del tipo de dato pose.

#### Información relacionada

| Para obtener más información sobre                            | Consulte  |
|---|---|
| Activación de una base de coordenadas de desplazamiento       | <a href="#">PDispSet - Activa un desplazamiento de programa a partir de una base de coordenadas conocida en la página 538</a> |
| Definición manual de la base de coordenadas de desplazamiento | <i>Manual del operador - IRC5 con FlexPendant, sección Calibración</i>  |

## 2 Funciones

### 2.56 DefFrame - Define una base de coordenadas

RobotWare - OS

### 2.56 DefFrame - Define una base de coordenadas

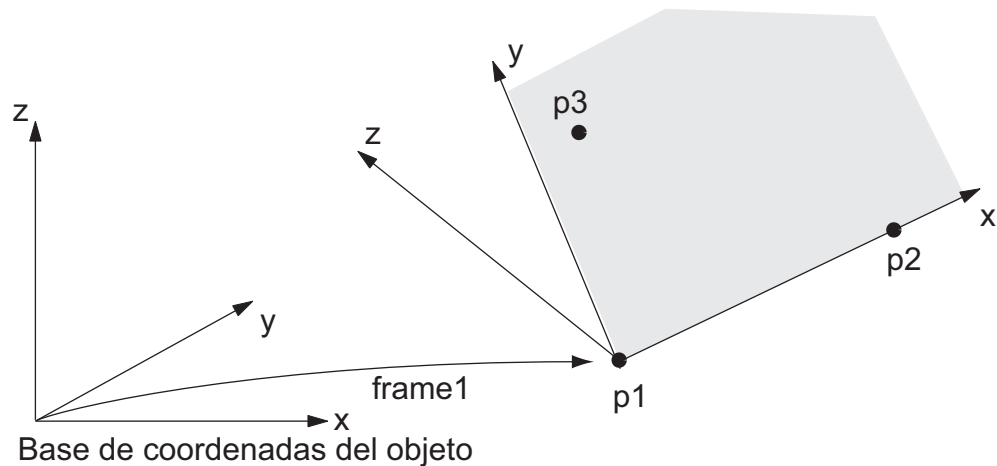
#### Utilización

DefFrame (*Define Frame*) se utiliza para calcular una base de coordenadas a partir de tres posiciones que la definen.

#### Ejemplos básicos

El ejemplo siguiente ilustra la función DefFrame.

#### Ejemplo 1



xx0500002181

Se utilizan tres posiciones, p1- p3, en relación con el sistema de coordenadas del objeto, para definir el nuevo sistema de coordenadas, frame1. La primera posición, p1, define el origen del nuevo sistema de coordenadas. La segunda posición, p2, define la dirección del eje x. La tercera posición, p3, define la ubicación del plano xy. La base de coordenadas frame1 definida puede usarse como base de coordenadas de desplazamiento, como se muestra en el ejemplo siguiente:

```
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;
```

#### Valor de retorno

Tipo de dato: pose

La base de coordenadas calculada.

El cálculo se realiza respecto del sistema de coordenadas del objeto activo.

Continúa en la página siguiente



**Argumentos**

```
DefFrame (NewP1 NewP2 NewP3 [\Origin])
```

NewP1

**Tipo de dato:** robtarget

La primera posición, que definirá el origen del nuevo sistema de coordenadas.

NewP2

**Tipo de dato:** robtarget

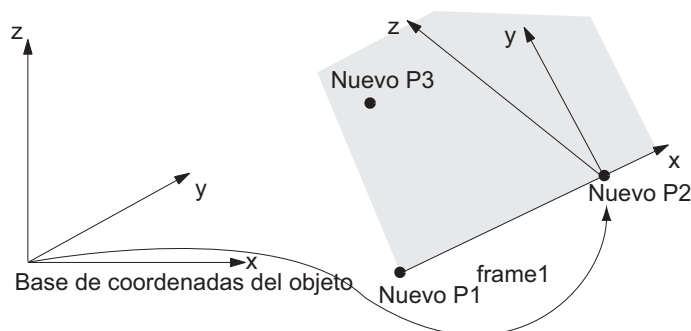
La segunda posición, que definirá la dirección del eje x de la nueva base de coordenadas.

NewP3

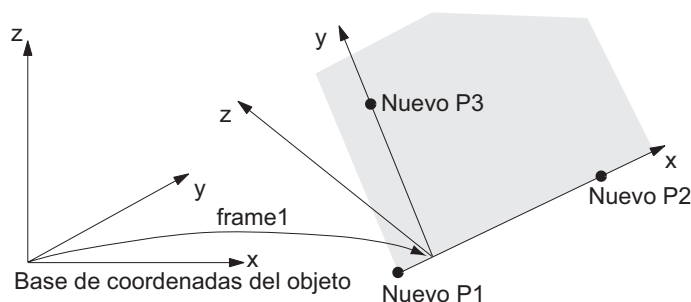
**Tipo de dato:** robtarget

La tercera posición, que definirá el plano xy del nuevo sistema de coordenadas. La posición del punto 3 estará en el lado y positivo, como se muestra en la figura anterior.

[\Origin]

**Tipo de dato:** numUn argumento opcional que definirá cómo se posiciona el origen del nuevo sistema de coordenadas. El valor `Origin = 1` significa que el origen se sitúa en `NewP1`, es decir, el mismo que si se omite este argumento. El valor `Origin = 2` significa que el origen se sitúa en `NewP2`. Consulte la figura que aparece a continuación.

xx0500002178

El valor `Origin = 3` significa que el origen se sitúa en la línea que va desde `NewP1` a `NewP2` y por tanto `NewP3` se sitúa en el eje y. Consulte la figura siguiente.

xx0500002180

Cualquier otro valor, o la omisión de `Origin`, situará el origen en `NewP1`.*Continúa en la página siguiente*

## 2 Funciones

### 2.56 DefFrame - Define una base de coordenadas

RobotWare - OS

Continuación

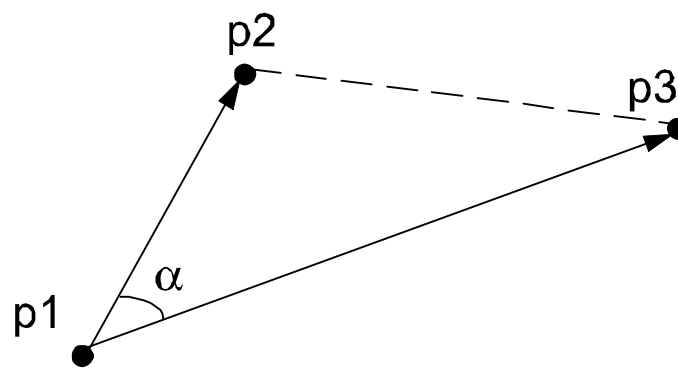
#### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                 | Causa del error   |
|------------------------|---|
| <code>ERR_FRAME</code> | No se puede calcular la base de coordenadas debido a las limitaciones siguientes. |

#### Limitaciones

Las tres posiciones de `p1` - `p3` que definen la base de coordenadas deben formar un triángulo bien definido. El tipo de triángulo mejor definido es aquél en el que todos los lados tienen la misma longitud.



xx0500002182

Se considera que este triángulo no está bien definido si el ángulo  $\alpha$  es demasiado pequeño. El ángulo  $\alpha$  es demasiado pequeño si se cumple la expresión siguiente:

$$|\cos \alpha| < 1 - 10^{-4}$$

El triángulo `p1`, `p2`, `p3` no debe ser demasiado pequeño, es decir, las posiciones no deben estar demasiado cercanas entre sí. Las distancias existentes entre `p1` - `p2` y entre `p1` - `p3` no deben ser inferiores a 0,1 mm.

#### Sintaxis

```
DefFrame '('  
    [NewP1 ':=' ] <expression (IN) of robtarget> ','  
    [NewP2 ':=' ] <expression (IN) of robtarget> ','  
    [NewP3 ':=' ] <expression (IN) of robtarget>  
    ['\ ' Origin ':=' <expression (IN) of num > ' ' )'
```

Una función con un valor de retorno del tipo de dato `pose`.

#### Información relacionada

| Para obtener más información sobre                      | Consulte  |
|---|---|
| Instrucciones y funciones matemáticas                   | <i>Manual de referencia técnica - RAPID Overview</i> , sección <i>Resumen sobre RAPID - Matemáticas</i>                       |
| Activación de una base de coordenadas de desplazamiento | <a href="#">PDispSet - Activa un desplazamiento de programa a partir de una base de coordenadas conocida en la página 538</a> |

---

## 2.57 Dim - Obtiene las dimensiones de una matriz

---

### Utilización

`Dim` (*Dimension*) se utiliza para obtener el número de elementos de una matriz.

---

### Ejemplos básicos

El ejemplo siguiente ilustra la función `Dim`.

Consulte también [Más ejemplos en la página 1263](#).

#### Ejemplo 1

```
PROC arrmul(VAR num array{*}, num factor)
  FOR index FROM 1 TO Dim(array, 1) DO
    array{index} := array{index} * factor;
  ENDFOR
ENDPROC
```

Se multiplican por un factor todos los elementos de una matriz de elementos de tipo `num`. Este procedimiento acepta como entrada cualquier matriz de una sola dimensión y compuesta de elementos de tipo `num`.

---

### Valor de retorno

Tipo de dato: `num`

El número de elementos de la matriz en la dimensión especificada.

---

### Argumentos

`Dim` (`ArrPar` `DimNo`)

`ArrPar`

*Array Parameter*

Tipo de dato: cualquier tipo

El nombre de la matriz.

`DimNo`

*Dimension Number*

Tipo de dato: `num`

La dimensión deseada de la matriz:

1 = Primera dimensión

2 = Segunda dimensión

3 = Tercera dimensión

---

### Más ejemplos

A continuación aparecen más ejemplos de cómo usar la función `Dim`.

#### Ejemplo 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";
```

*Continúa en la página siguiente*

## 2 Funciones

---

### 2.57 Dim - Obtiene las dimensiones de una matriz

RobotWare - OS

Continuación

```
        Stop;
    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
```

Se suman dos matrices. Si las matrices son de distinto tamaño, el programa se detiene y aparece un mensaje de error.

Este procedimiento acepta cualquier matriz tridimensional y compuesta de elementos de tipo `num`.

---

#### Sintaxis

```
Dim '('
    [ArrPar ':='] <reference (REF) of any type> ','
    [DimNo ':='] <expression (IN) of num> ')'
```

Los parámetros **REF** exigen que el argumento correspondiente sea una constante, una variable o una variable persistente entera. El argumento también puede ser un parámetro **IN**, un parámetro **VAR** o un parámetro **PERS** entero.

Una función con un valor de retorno del tipo de dato `num`.

---

#### Información relacionada

| Para obtener más información sobre | Consulte  |
|------------------------------------|---|
| Parámetros de matriz               | <i>Manual de referencia técnica - RAPID Overview, sección Características básicas - Rutinas</i> |
| Declaración de matriz              | <i>Manual de referencia técnica - RAPID Overview, sección Características básicas - Datos</i>   |

---

## 2.58 DInput - Lee el valor de una señal digital de entrada

---

### Utilización

DInput se utiliza para leer el valor actual de una señal digital de entrada.



#### Nota

Recuerde que la función DInput es una función antigua que ya no es necesario utilizar. Consulte los ejemplos relativos a una forma alternativa y recomendada de programar.

---

### Ejemplos básicos

El ejemplo siguiente ilustra la función DInput.

Consulte también [Más ejemplos en la página 1265](#).

#### Ejemplo 1

```
IF DInput(di2) = 1 THEN ...  
...  
IF di2 = 1 THEN ...
```

Si el valor actual de la señal di2 es igual a 1, entonces ....

---

### Valor de retorno

Tipo de dato: num

El valor actual de la señal (0 ó 1).

---

### Argumentos

DInput (Signal)

Signal

Tipo de dato: signaldi

El nombre de la entrada digital que debe leerse.

---

### Ejecución de programas

El valor leído depende de la configuración de la señal. Si la señal está invertida en los parámetros de sistema, el valor devuelto por esta función es lo opuesto al valor real del canal físico.

---

### Más ejemplos

A continuación aparecen más ejemplos de cómo usar la función DInput.

#### Ejemplo 1

```
weld_flag := DInput(weld);  
...  
weld_flag := weld;
```

*Continúa en la página siguiente*

## 2 Funciones

---

### 2.58 DInput - Lee el valor de una señal digital de entrada

RobotWare - OS

Continuación

Se asigna a la variable `weld_flag` el mismo valor que el valor actual de la señal `weld`.



#### Nota

Recuerde que, en este caso, `weld_flag` refleja el valor actual de la señal. Por tanto, si se usa `weld_flag` más adelante en el programa, no existen garantías de que refleje el valor actual de la señal.

---

### Sintaxis

```
DInput '('  
  [Signal ':='] <variable (VAR) of signaladi>')'
```

Una función con un valor de retorno del tipo de dato `dionum`.

---

### Información relacionada

| Para obtener más información sobre         | Consulte   |
|--|--|
| Instrucciones de entrada/salida            | <i>Manual de referencia técnica - RAPID Overview, sección Resumen sobre RAPID - Señales de entrada y salida</i>  |
| Funcionalidad de entrada/salida en general | <i>Manual de referencia técnica - RAPID Overview, sección Principios de movimiento y E/S - Principios de E/S</i> |
| Configuración de E/S                       | <i>Manual de referencia técnica - Parámetros del sistema</i>   |

## 2.59 Distance - Distancia entre dos puntos

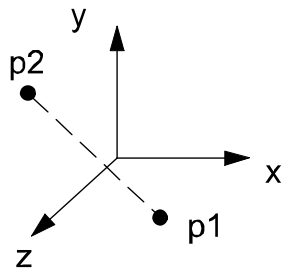
### Utilización

`Distance` se utiliza para calcular la distancia entre dos puntos del espacio.

### Ejemplos básicos

El ejemplo siguiente ilustra la función `Distance`.

#### Ejemplo 1



xx0500002321

```
VAR num dist;
CONST pos p1 := [4,0,4];
CONST pos p2 := [-4,4,4];
...
dist := Distance(p1, p2);
```

Se calcula la distancia en el espacio existente entre los puntos `p1` y `p2`. El resultado se asigna a la variable `dist`.

### Valor de retorno

Tipo de dato: `num`

La distancia (siempre positiva) en mm entre los puntos.

### Argumentos

`Distance (Point1 Point2)`

`Point1`

Tipo de dato: `pos`

El primer punto descrito con el tipo de dato `pos`.

`Point2`

Tipo de dato: `pos`

El segundo punto descrito con el tipo de dato `pos`.

*Continúa en la página siguiente*

## 2 Funciones

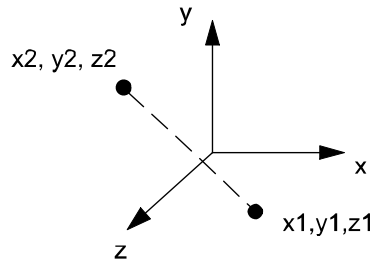
### 2.59 Distance - Distancia entre dos puntos

RobotWare - OS

Continuación

#### Ejecución de programas

Cálculo de la distancia existente entre dos puntos:



xx0500002322

$$\text{Distancia} = \sqrt{\hat{E}(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2}$$

xx0500002323

#### Sintaxis

```
Distance '('  
  [Point1 ':=' ] <expression (IN) of pos> ','  
  [Point2 ':=' ] <expression (IN) of pos> ')'
```

Una función con un valor de retorno del tipo de dato num.

#### Información relacionada

| Para obtener más información sobre    | Consulte  |
|---------------------------------------|---|
| Instrucciones y funciones matemáticas | <i>Manual de referencia técnica - RAPID Overview, sección Resumen sobre RAPID - Matemáticas</i> |
| Definición de una posición con pos    | <a href="#">pos - Posiciones (sólo X, Y y Z) en la página 1729</a>                              |



## 2.60 DIV - Evalúa una división entera

### Utilización

DIV es una expresión condicional utilizada para evaluar una división de enteros.

### Ejemplos básicos

Los siguientes ejemplos ilustran la función DIV.

#### Ejemplo 1

```
reg1 := 14 DIV 4;
```

El valor de retorno es 3, dado que 14 es divisible entre 4 un número de 3 veces.

#### Ejemplo 2

```
VAR dnum mydnum1 := 10;
VAR dnum mydnum2 := 5;
VAR dnum mydnum3;
...
mydnum3 := mydnum1 DIV mydnum2;
```

El valor de retorno es 2, dado que 10 es divisible entre 5 un número de 2 veces.

### Valor de retorno

Tipo de dato: num, dnum

Devuelve el número entero de una división de enteros.

### Sintaxis

```
<expression of num> DIV <expression of num>
```

Una función con un valor de retorno del tipo de dato num.

```
<expression of dnum> DIV <expression of dnum>
```

Una función con un valor de retorno del tipo de dato dnum.

### Información relacionada

| Para obtener más información sobre | Consulte   |
|------------------------------------|--|
| num - Valores numéricos            | <a href="#">num - Valores numéricos en la página 1712</a>          |
| dnum - Valores numéricos dobles    | <a href="#">dnum - Valores numéricos dobles en la página 1660</a>  |
| MOD                                | <a href="#">MOD - Evalúa un módulo de entero en la página 1377</a> |
| Expresiones                        | <a href="#">Manual de referencia técnica - RAPID Overview</a>      |

## 2 Funciones

---

### 2.61 DnumToNum - Convierte dnum a num

RobotWare - OS

### 2.61 DnumToNum - Convierte dnum a num

---

#### Utilización

DnumToNum **convierte un valor dnum a un valor num si es posible. De lo contrario, genera un error recuperable.**

---

#### Ejemplos básicos

El ejemplo siguiente ilustra la función DnumToNum.

#### Ejemplo 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);
```

El valor dnum 8388607 es devuelto por la función como el valor num 8388607.

El valor dnum 8388609 es devuelto por la función como el valor num 8.38861E+06.

El valor dnum 4294967295 genera el error recuperable ERR\_ARGVALERR.

---

#### Valor de retorno

Tipo de dato: num

El valor dnum de entrada puede estar en el rango de -8388607 a 8388608 y devuelve el mismo valor como un num. Si no se utiliza el modificador `\Integer`, el valor dnum de entrada puede estar en el rango de -3,40282347E+38 a 3,40282347E+38 y el valor de retorno puede convertirse en un valor de coma flotante.

---

#### Argumentos

DnumToNum (Value [`\Integer`])

Value

Tipo de dato: dnum

El valor numérico a convertir.

[`\Integer`]

Tipo de dato: switch

Solo valores enteros.

Si no se utiliza el modificador `\Integer`, se realiza un cambio al tipo menor incluso si el valor pasa a ser de punto flotante. Si se utiliza, se realiza una comprobación de si el valor es un entero que esté entre -8388607 y 8388608. Si el valor no está en dicho intervalo, se genera un error recuperable.

---

*Continúa en la página siguiente*

**Gestión de errores**

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                      | Causa del error   |
|-----------------------------|---|
| <code>ERR_ARGVALERR</code>  | Si el valor es superior a 8388608 o inferior a -8388607 o no es un entero (si se utiliza el argumento opcional <code>Integer</code> ) |
| <code>ERR_NUM_LIMIT</code>  | El valor es superior a 3,40282347E+38 o inferior a -3,40282347E+38  |
| <code>ERR_INT_NOTVAL</code> | El valor no es un entero  |

**Sintaxis**

```
DnumToNum '( '
  [ Value ':= ' ] < expression (IN) of dnum >
  [ '\ ' Integer ')'
```

Una función con un valor de retorno del tipo de dato `num`.

**Información relacionada**

| Para obtener más información sobre | Consulte   |
|------------------------------------|--|
| <code>Dnum</code> tipo de dato     | <a href="#">dnum - Valores numéricos dobles en la página 1660.</a> |
| <code>Num</code> tipo de dato      | <a href="#">num - Valores numéricos en la página 1712.</a>         |

## 2 Funciones

---

### 2.62 DnumToStr - Convierte un valor numérico en una cadena de caracteres

RobotWare - OS

### 2.62 DnumToStr - Convierte un valor numérico en una cadena de caracteres

---

#### Utilización

DnumToStr (*Numeric To String*) se utiliza para convertir un valor numérico en una cadena.

---

#### Ejemplos básicos

Los siguientes ejemplos ilustran la función DnumToStr.

##### Ejemplo 1

```
VAR string str;  
str := DnumToStr(0.3852138754655357,3);
```

Se asigna a la variable `str` el valor "0.385".

##### Ejemplo 2

```
VAR dnum val;  
val:= 0.3852138754655357;  
str := DnumToStr(val, 2\Exp);
```

Se asigna a la variable `str` el valor "3.85E-01".

##### Ejemplo 3

```
VAR dnum val;  
val := 0.3852138754655357;  
str := DnumToStr(val, 15);
```

Se asigna a la variable `str` el valor "0.385213875465536".

##### Ejemplo 4

```
VAR dnum val;  
val:=4294967295.385215;  
str := DnumToStr(val, 4);
```

Se asigna a la variable `str` el valor "4294967295.3852".

---

#### Valor de retorno

Tipo de dato: `str`

El valor numérico, convertido en una cadena con el número especificado de decimales, en notación científica si así se solicita. Si es necesario, el valor numérico se redondea. Si no se incluye ningún decimal, se suprime el punto decimal.

---

#### Argumentos

DnumToStr (Val Dec [\Exp])

Val

*Value*

Tipo de dato: `dnum`

El valor numérico a convertir.

Dec

*Decimals*

Tipo de dato: `num`

---

Continúa en la página siguiente

### 2.62 DnumToStr - Convierte un valor numérico en una cadena de caracteres

RobotWare - OS

Continuación

Número de decimales. El número de decimales no debe ser negativo ni mayor que la precisión disponible para los valores numéricos.

El número máximo de decimales que pueden usarse es 15.

[ \Exp ]

**Exponent**

Tipo de dato: `switch`

Para usar exponente en el valor de retorno.

#### Sintaxis

```
DnumToStr '('
  [ Val ':=' ] <expression (IN) of dnum>
  [ Dec ':=' ] <expression (IN) of num>
  [ '\ ' Exp ' )'
```

Una función con un valor de retorno del tipo de dato `string`.

#### Información relacionada

| Para obtener más información sobre                          | Consulte   |
|---|--|
| Funciones para cadenas de caracteres                        | <i>Manual de referencia técnica - RAPID Overview, sección Resumen sobre RAPID - Funciones para cadenas de caracteres</i> |
| Definición de cadena de caracteres                          | <a href="#">string - Cadenas en la página 1778</a>   |
| Valores de cadena de caracteres                             | <i>Manual de referencia técnica - RAPID Overview, sección Características básicas - Elementos básicos</i>                |
| Convertir un valor numérico num en una cadena de caracteres | <a href="#">NumToStr - Convierte un valor numérico en una cadena de caracteres en la página 1389</a>                     |

## 2 Funciones

### 2.63 DotProd - Producto escalar de dos vectores pos RobotWare - OS

### 2.63 DotProd - Producto escalar de dos vectores pos

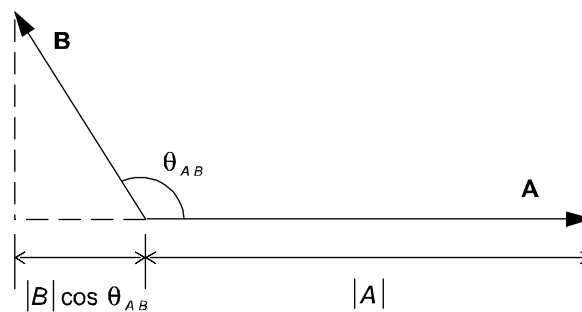
#### Utilización

`DotProd` (*Dot Product*) se utiliza para calcular el producto escalar de dos vectores pos. La aplicación típica de esta función es calcular la proyección de un vector sobre el otro o calcular el ángulo existente entre dos vectores.

#### Ejemplos básicos

El ejemplo siguiente ilustra la función `DotProd`.

#### Ejemplo 1



xx0500002449

El producto escalar de dos vectores **A** y **B** es un escalar, lo que equivale a los productos de las magnitudes de **A** y **B** y el coseno del ángulo que forman.

$$A \cdot B = |A||B| \cos \theta_{AB}$$

El producto escalar:

- Es menor o igual que el producto de las magnitudes.
- Puede ser un valor positivo o negativo, en función de si el ángulo formado por los vectores es superior o inferior a 90 grados.
- Es igual al producto de la magnitud de un vector y la proyección del segundo vector sobre el primero.
- Es cero si los vectores son perpendiculares entre sí.

Los vectores se describen con el tipo de dato `pos` y el producto escalar se describe con el tipo de dato `num`:

```
VAR num dotprod;  
VAR pos vector1;  
VAR pos vector2;  
...  
...  
vector1 := [1,1,1];  
vector2 := [1,2,3];  
dotprod := DotProd(vector1, vector2);
```

#### Valor de retorno

Tipo de dato: `num`

*Continúa en la página siguiente*

El valor del producto escalar de los dos vectores.

**Argumentos**

DotProd (Vector1 Vector2)

Vector1

**Tipo de dato:** pos

El primer vector descrito con el tipo de dato pos.

Vector2

**Tipo de dato:** pos

El segundo vector descrito con el tipo de dato pos.

**Sintaxis**

```
DotProd '('
  [Vector1 ':=' ] <expression (IN) of pos> ','
  [Vector2 ':=' ] <expression (IN) of pos>
  ')'
```

Una función con un valor de retorno del tipo de dato num.

**Información relacionada**

| Para obtener más información sobre    | Consulte  |
|---------------------------------------|---|
| Instrucciones y funciones matemáticas | <i>Manual de referencia técnica - RAPID Overview, sección Resumen sobre RAPID - Matemáticas</i> |

## 2 Funciones

---

### 2.64 DOutput - Lee el valor de una señal digital de salida

RobotWare - OS

### 2.64 DOutput - Lee el valor de una señal digital de salida

---

#### Utilización

DOutput se utiliza para leer el valor actual de una señal digital de salida.



#### Nota

Recuerde que la función DOutput es una función antigua que ya no es necesario utilizar. Consulte los ejemplos relativos a una forma alternativa y recomendada de programar.

#### Ejemplos básicos

El ejemplo siguiente ilustra la función DOutput.

Consulte también [Más ejemplos en la página 1277](#).

#### Ejemplo 1

```
IF DOutput(do2) = 1 THEN ...  
...  
IF do2 = 1 THEN ...
```

Si el valor actual de la señal do2 es igual a 1, entonces . . .

#### Valor de retorno

Tipo de dato: dionum

El valor actual de la señal (0 ó 1).

#### Argumentos

DOutput (Signal)

Signal

Tipo de dato: signaldo

El nombre de la señal a leer.

#### Ejecución de programas

El valor leído depende de la configuración de la señal. Si la señal está invertida en los parámetros de sistema, el valor devuelto por esta función es lo opuesto al valor real del canal físico.

#### Gestión de errores

Pueden generarse los errores recuperables enumerados a continuación. Estos errores pueden ser gestionados en un gestor de errores. El valor de la variable de sistema ERRNO cambia a:

| Nombre             | Causa del error   |
|--------------------|---|
| ERR_NO_ALIASIO_DEF | La variable de señal es una variable declarada en RAPID. No se ha conectado a una señal E/S definida en la configuración de E/S con la instrucción AliasIO. |
| ERR_NORUNUNIT      | Si se ha perdido el contacto con el dispositivo de E/S.   |

*Continúa en la página siguiente*



| Nombre            | Causa del error   |
|-------------------|---|
| ERR_SIG_NOT_VALID | La señal de E/S no está disponible. Los motivos pueden ser que el dispositivo de E/S no está en funcionamiento o que exista un error en la configuración (válido solamente para el bus de campo ICI). |

#### Más ejemplos

A continuación aparecen más ejemplos de la función DOutput.

#### Ejemplo 1

```
IF DOutput(auto_on) <> active THEN ...
...
IF auto_on <> active THEN ...
```

Si el valor actual de la señal de sistema `auto_on` es `not active`, entonces ..., es decir, si el robot está en el modo de funcionamiento manual, entonces ...



#### Nota

La señal debe estar definida previamente como salida de sistema en los parámetros de sistema.

#### Sintaxis

```
DOutput '('
  [ Signal '[:=' ] < variable (VAR) of signaldo > ')'
```

Una función con un valor de retorno del tipo de dato `dionum`.

#### Información relacionada

| Para obtener más información sobre         | Consulte   |
|--|--|
| Activación de una señal digital de salida  | <a href="#">SetDO - Cambia el valor de una señal digital de salida en la página 704</a>                                  |
| Instrucciones de entrada/salida            | <i>Manual de referencia técnica - RAPID Overview</i> , sección <i>Resumen sobre RAPID - Señales de entrada y salida</i>  |
| Funcionalidad de entrada/salida en general | <i>Manual de referencia técnica - RAPID Overview</i> , sección <i>Principios de movimiento y E/S - Principios de E/S</i> |
| Configuración de E/S                       | <i>Manual de referencia técnica - Parámetros del sistema</i>   |

## 2 Funciones

---

### 2.65 EulerZYX - Obtiene ángulos Euler a partir de una orientación *RobotWare - OS*

### 2.65 EulerZYX - Obtiene ángulos Euler a partir de una orientación

---

#### Utilización

`EulerZYX` (*Euler ZYX rotations*) se utiliza para obtener un componente de ángulo Euler a partir de una variable de tipo `orient`.

---

#### Ejemplos básicos

El ejemplo siguiente ilustra la función `EulerZYX`.

#### Ejemplo 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);
```

---

#### Valor de retorno

Tipo de dato: `num`

El ángulo Euler correspondiente, expresado en grados, en el rango [-180, 180].

---

#### Argumentos

`EulerZYX ([\X] | [\Y] | [\Z] Rotation)`

`[\X]`

Tipo de dato: `switch`

Obtiene la rotación alrededor del eje X.

`[\Y]`

Tipo de dato: `switch`

Obtiene la rotación alrededor del eje Y.

`[\Z]`

Tipo de dato: `switch`

Obtiene la rotación alrededor del eje Z.

**¡Atención!**

Los argumentos `\X`, `\Y` y `\Z` son excluyentes entre sí. Si no se especifica ninguno de estos argumentos, se genera un error en tiempo de ejecución.

`Rotation`

Tipo de dato: `orient`

La rotación representada en forma de cuaternio.

---

*Continúa en la página siguiente*

#### Sintaxis

```
EulerZYX '('  
  ['\ ' X ','] | ['\ ' Y ','] | ['\ ' Z ',']  
  [Rotation ':='] <expression (IN) of orient> ')'
```

Una función con un valor de retorno del tipo de dato num.

#### Información relacionada

| Para obtener más información sobre    | Consulte  |
|---------------------------------------|---|
| Instrucciones y funciones matemáticas | <i>Manual de referencia técnica - RAPID Overview</i> , sección <i>Resumen sobre RAPID - Matemáticas</i> |

## 2 Funciones

---

2.66 EventType - Obtiene el tipo de evento actual dentro de cualquier rutina de evento  
RobotWare - OS

### 2.66 EventType - Obtiene el tipo de evento actual dentro de cualquier rutina de evento

---

#### Utilización

EventType puede usarse en cualquier rutina de evento y devuelve a continuación el tipo de evento ejecutado actualmente.

Si la llamada a EventType se realiza desde cualquier rutina de tarea de programa, EventType devuelve siempre 0, para indicar EVENT\_NONE.

---

#### Ejemplos básicos

El ejemplo siguiente ilustra la función EventType.

##### Ejemplo 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
```

Uso de la función EventType dentro de cualquier rutina de evento para determinar qué eventos de sistema, si los hay, se están ejecutando en ese momento.

---

#### Valor de retorno

Tipo de dato: event\_type

El tipo de evento ejecutado actualmente 1 ... 7, ó 0 si no se ejecuta ninguna rutina de evento.

---

#### Datos predefinidos

Pueden utilizarse las siguientes constantes simbólicas predefinidas del tipo event\_type para comprobar el valor de retorno.

```
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;
```

*Continúa en la página siguiente*

### 2.66 EventType - Obtiene el tipo de evento actual dentro de cualquier rutina de evento

*RobotWare - OS*

*Continuación*

```
CONST event_type EVENT_RESET := 6;  
CONST event_type EVENT_STEP := 7;
```

#### Sintaxis

```
EventType '(' ')
```

Una función con un valor de retorno del tipo de dato `event_type`.

#### Información relacionada

| Para obtener más información sobre                               | Consulte   |
|--|--|
| Rutinas de evento en general                                     | <i>Manual de referencia técnica - Parámetros del sistema, sección Controller - Event Routine</i> |
| Tipo de datos <code>event_type</code> , constantes pre-definidas | <a href="#">event_type - Tipo de rutina de evento en la página 1675</a>                          |

## 2 Funciones

---

### 2.67 ExecHandler - Obtener el tipo de gestor de ejecución RobotWare - OS

## 2.67 ExecHandler - Obtener el tipo de gestor de ejecución

---

### Utilización

`ExecHandler` puede usarse para determinar si el código de `RAPID` actual se está ejecutando en algún gestor de rutina de programa de `RAPID`.

---

### Ejemplos básicos

El ejemplo siguiente ilustra la función `ExecHandler`.

#### Ejemplo 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
```

Uso de la función `ExecHandler` para determinar si el código se está ejecutando dentro de algún tipo de gestor de rutina o no.

Se devolverá `HANDLER_ERR` incluso si la llamada es ejecutada dentro de un método secundario del gestor de errores.

---

### Valor de retorno

Tipo de dato: `handler_type`

El tipo de gestor actual ejecutado de 1 a 3, ó 0 si no se ejecuta dentro de ningún gestor de rutina.

---

### Datos predefinidos

Pueden utilizarse las siguientes constantes simbólicas predefinidas del tipo `handler_type` para comprobar el valor de retorno.

```
CONST handler_type HANDLER_NONE := 0;  
CONST handler_type HANDLER_BWD := 1;  
CONST handler_type HANDLER_ERR := 2;  
CONST handler_type HANDLER_UNDO := 3;
```

---

### Sintaxis

```
ExecHandler '(' ' ')
```

Una función con un valor de retorno del tipo de dato `handler_type`.

---

### Información relacionada

| Para obtener más información sobre | Consulte   |
|------------------------------------|--|
| Tipo de gestor de ejecución        | <a href="#">handler_type - Tipo de gestor de ejecución en la página 1682</a> |

---

## 2.68 ExecLevel - Obtener el nivel de ejecución

### Utilización

ExecLevel puede usarse para determinar el nivel de ejecución actual del código de RAPID que se está ejecutando.

### Ejemplos básicos

El ejemplo siguiente ilustra la función ExecLevel.

#### Ejemplo 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
```

Uso de la función ExecLevel para determinar el nivel de ejecución actual.

### Valor de retorno

Tipo de dato: exec\_level

El nivel de ejecución actual, de 0 a 2.

### Datos predefinidos

Pueden utilizarse las siguientes constantes simbólicas predefinidas del tipo exec\_level para comprobar el valor de retorno:

```
CONST exec_level LEVEL_NORMAL := 0;
CONST exec_level LEVEL_TRAP := 1;
CONST exec_level LEVEL_SERVICE := 2;
```

### Sintaxis

```
ExecLevel '(' ')'
```

Una función con un valor de retorno del tipo de dato exec\_level.

### Información relacionada

| Para obtener más información sobre  | Consulte  |
|-------------------------------------|---|
| Tipo de dato del nivel de ejecución | <a href="#">exec_level - Nivel de ejecución en la página 1676</a> |

## 2 Funciones

---

### 2.69 Exp - Calcula el valor exponencial

RobotWare - OS

### 2.69 Exp - Calcula el valor exponencial

---

#### Utilización

Exp (*Exponential*) se utiliza para calcular el valor exponencial,  $e^x$ .

---

#### Ejemplos básicos

El ejemplo siguiente ilustra la función Exp.

#### Ejemplo 1

```
VAR num x;  
VAR num value;  
...  
value:= Exp( x);
```

value obtiene el valor exponencial de x.

---

#### Valor de retorno

Tipo de dato: num

El valor exponencial,  $e^x$ .

---

#### Argumentos

Exp (Exponent)

Exponent

Tipo de dato: num

El valor del argumento de exponente.

---

#### Sintaxis

```
Exp '('  
  [Exponent ':=' ] <expression (IN) of num> ')'
```

Una función con un valor de retorno del tipo de dato num.

---

#### Información relacionada

| Para obtener más información sobre    | Consulte  |
|---------------------------------------|---|
| Instrucciones y funciones matemáticas | <i>Manual de referencia técnica - RAPID Overview, sección Resumen sobre RAPID - Matemáticas</i> |



## 2.70 FileSize - Obtiene el tamaño de un archivo

### Utilización

FileSize se utiliza para obtener el tamaño del archivo especificado.

### Ejemplos básicos

El ejemplo siguiente ilustra la función FileSize.

Consulte también [Más ejemplos en la página 1285](#).

#### Ejemplo 1

```
PROC listfile(string filename)
  VAR num size;
  size := FileSize(filename);
  TPWrite filename+" size: "+NumToStr(size,0)+" Bytes";
ENDPROC
```

Este procedimiento imprime el nombre del archivo especificado, junto con la especificación de su tamaño.

### Valor de retorno

Tipo de dato: num

El tamaño en bytes.

### Argumentos

FileSize (Path)

Path

Tipo de dato: string

El nombre del archivo, especificado con una ruta completa o relativa.

### Ejecución de programas

Esta función devuelve un valor numérico que especifica el tamaño en bytes del archivo especificado.

También es posible obtener esta misma información acerca de un directorio.

### Más ejemplos

A continuación aparece un ejemplo básico de la función.

#### Ejemplo 1

En este ejemplo se enumeran todos los archivos que tengan un tamaño mayor de 1 KByte dentro de la estructura de directorios de "HOME:", incluidos todos los subdirectorios.

```
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
```

*Continúa en la página siguiente*

## 2 Funciones

---

### 2.70 FileSize - Obtiene el tamaño de un archivo

RobotWare - OS

Continuación

```
        IF filename <> ".." AND filename <> "." THEN
            searchdir dirname+"/"+filename, actionproc;
        ENDIF
    ENDWHILE
    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
```

Este programa recorre la estructura de directorios que existe dentro de "HOME:" y con cada archivo encontrado, ejecuta el procedimiento `listfile`. `searchdir` es una parte genérica, que no tiene ninguna información sobre el inicio de la búsqueda ni sobre a qué rutina se debe llamar con cada archivo. Utiliza `IsFile` para comprobar si se ha encontrado un subdirectorio o un archivo y utiliza el mecanismo de enlazamiento en tiempo de ejecución para llamar al procedimiento especificado en `actionproc` con todos los archivos encontrados. La rutina `actionproclistfile` comprueba si el archivo tiene un tamaño mayor que 1 KByte.

---

### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                   | Causa del error       |
|--------------------------|-----------------------|
| <code>ERR_FILEACC</code> | El archivo no existe. |

---

### Sintaxis

```
FileSize '('
    [ Path ':=' ] < expression (IN) of string> ')'
```

Una función con un valor de retorno del tipo de dato `num`.

---

### Información relacionada

| Para obtener más información sobre | Consulte  |
|------------------------------------|---|
| Creación de un directorio          | <a href="#">MakeDir - Crea un nuevo directorio en la página 358</a> |

Continúa en la página siguiente

| Para obtener más información sobre              | Consulte   |
|---|--|
| Eliminación de un directorio                    | <a href="#">RemoveDir - Elimina un directorio en la página 598</a>                     |
| Cambio del nombre de un archivo                 | <a href="#">RenameFile - Cambia el nombre de un archivo en la página 604</a>           |
| Eliminación de un archivo                       | <a href="#">RemoveFile - Elimina un archivo en la página 600</a>                       |
| Copia de un archivo                             | <a href="#">CopyFile - Copia un archivo en la página 169</a>                           |
| Comprobación del tipo del archivo               | <a href="#">IsFile - Comprobar el tipo de un archivo en la página 1351</a>             |
| Comprobación del tamaño del sistema de archivos | <a href="#">FSSize - Obtiene el tamaño de un sistema de archivos en la página 1291</a> |
| Gestión de archivos y dispositivos de E/S       | <a href="#">Application manual - Controller software IRC5</a>                          |

## 2 Funciones

---

### 2.71 FileTimeDnum - Obtener información de tiempo sobre un archivo

### 2.71 FileTimeDnum - Obtener información de tiempo sobre un archivo

---

#### Utilización

`FileTimeDnum` se utiliza para obtener la última hora de modificación, acceso o cambio de estado de un archivo. La hora se indica en segundos a partir de las 00:00:00 horas, hora de Greenwich, del 1 de enero de 1970. La hora se devuelve como un valor de tipo `dnum` y opcionalmente también en un valor de tipo `stringdig`.

---

#### Ejemplo básico

El ejemplo siguiente ilustra la función `FileTimeDnum`.

Consulte también [Más ejemplos en la página 1289](#).

#### Ejemplo 1

```
IF FileTimeDnum ("HOME:/mymod.mod" \ModifyTime) > ModTimeDnum
  ("mymod") THEN
  UnLoad "HOME:/mymod.mod";
  Load \Dynamic, "HOME:/mymod.mod";
ENDIF
```

Este programa recarga un módulo si el archivo de origen es más reciente. Utiliza `ModTimeDnum` para obtener la hora de la última modificación del módulo especificado y para compararlo con los valores de `FileTimeDnum` ("HOME:/mymod.mod" \ModifyTime) del origen. A continuación, si el origen es más reciente, el programa descarga y carga de nuevo el módulo.

---

#### Valor de retorno

Tipo de dato: `dnum`

El tiempo, medido en segundos, desde las 00:00:00 horas, horario de Greenwich, del 1 de enero de 1970.

---

#### Argumentos

```
FileTimeDnum ( Path [\ModifyTime] | [\AccessTime] | [\StatCTime]
              [\StrDig])
```

Path

Tipo de dato: `string`

El archivo especificado con una ruta completa o relativa.

[\ModifyTime]

Tipo de dato: `switch`

Última hora de modificación.

[\AccessTime]

Tipo de dato: `switch`

Hora del último acceso (lectura, ejecución o modificación).

[\StatCTime]

Tipo de dato: `switch`

---

*Continúa en la página siguiente*

## 2.71 FileTimeDnum - Obtener información de tiempo sobre un archivo

*Continuación*

Hora del último cambio de estado (cualificación de acceso) del archivo.

[\StrDig]

### *String Digit*

Tipo de dato: stringdig

Para obtener la hora del archivo en una representación de tipo stringdig.

### Ejecución de programas

Para el archivo o directorio especificado, esta función devuelve un valor numérico que especifica el tiempo desde la última operación de:

- Modificación
- Acceso
- Cambio de estado

### Más ejemplos

A continuación aparecen más ejemplos de la función FileTimeDnum.

A continuación aparece un ejemplo completo en el que se implementa un servicio de alerta para un máximo de 10 archivos.

```

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;
  
```

*Continúa en la página siguiente*

## 2 Funciones

### 2.71 FileTimeDnum - Obtener información de tiempo sobre un archivo

#### Continuación

```
        myfiles{currentpos}.ftime := FileTimeDnum (filename
        \ModifyTime);
    ENDIF
ENDPROC

PROC alertFree()
    IDelete timeint;
ENDPROC
```

#### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre      | Causa del error       |
|-------------|-----------------------|
| ERR_FILEACC | El archivo no existe. |

#### Sintaxis

```
FileTimeDnum '('
[Path ':' ] <expression (IN) of string>
['\ ' ModifyTime] |
['\ ' AccessTime] |
['\ ' StatCTime]
['\ ' StrDig ':' :=' <variable (VAR) of stringdig>'] )'
```

Una función con un valor de retorno del tipo de dato `dnum`.

#### Información relacionada

| Para obtener más información sobre               | Consulte   |
|--|--|
| Última hora de modificación de un módulo cargado | <a href="#">ModTimeDnum - Obtener la hora de modificación del módulo cargado en la página 1379</a> |
| Cadena de caracteres con sólo dígitos            | <a href="#">stringdig - Cadena de caracteres con sólo dos dígitos en la página 1780</a>            |
| Comparar dos cadenas que sólo contienen dígitos  | <a href="#">StrDigCmp - Comparar dos cadenas que sólo contienen dígitos en la página 1494</a>      |

## 2.72 FSSize - Obtiene el tamaño de un sistema de archivos

### Utilización

`FSSize` (*File System Size*) se utiliza para obtener el tamaño del sistema de archivos en el que se encuentra el archivo especificado. El tamaño en bytes, kilobytes o megabytes se devuelve con el tipo `num`.

### Ejemplo básico

El ejemplo siguiente ilustra la función `FSSize`.

Consulte también [Más ejemplos en la página 1292](#).

### Ejemplo 1

```
PROC main()
  VAR num totalsyssize;
  VAR num freesyssize;
  freesyssize := FSSize("HOME:/spy.log" \Free);
  totalsyssize := FSSize("HOME:/spy.log" \Total);
  TPWrite NumToStr(((totalsyssize -
    freesyssize)/totalsyssize)*100,0) + " percent used";
ENDPROC
```

Este procedimiento imprime la cantidad de espacio de disco utilizada en `HOME:` archivo de sistema como porcentaje.

### Valor de retorno

Tipo de dato: `num`  
El tamaño en bytes.

### Argumentos

```
FSSize (Name [\Total] | [\Free] [\Kbyte] [\Mbyte])
```

Name

Tipo de dato: `string`

El nombre de un archivo en el sistema de archivos, especificado con una ruta completa o relativa.

[ \Total ]

Tipo de dato: `switch`

Obtiene la cantidad total de espacio del sistema de archivos.

[ \Free ]

Tipo de dato: `switch`

Obtiene la cantidad de espacio libre del sistema de archivos.

[ \Kbyte ]

Tipo de dato: `switch`

Convertir a kilobytes el número de bytes leídos, por ejemplo, dividir el tamaño entre 1024.

*Continúa en la página siguiente*

## 2 Funciones

---

### 2.72 FSSize - Obtiene el tamaño de un sistema de archivos

RobotWare - OS

Continuación

[ \Mbyte ]

Tipo de dato: switch

Convertir a megabytes el número de bytes leídos, por ejemplo, dividir el tamaño entre 1 048 576 (1024\*1024).

---

### Ejecución de programas

Esta función devuelve un valor numérico que especifica el tamaño del sistema de archivos en el que se encuentra el archivo especificado.

---

### Más ejemplos

A continuación aparecen más ejemplos de la función FSSize.

#### Ejemplo 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
```

A continuación aparece un ejemplo completo de la implementación de un servicio de alerta que imprime una advertencia en el FlexPendant cuando el espacio libre que queda en el sistema de archivos "HOME:" es inferior al 10%.

---

### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema ERRNO cambiará a:

| Nombre       | Causa del error  |
|--------------|--|
| ERR_FILEACC  | El sistema de archivos no existe                                       |
| ERR_FILESIZE | El tamaño sobrepasa el valor de entero máximo de un valor num, 8388608 |

---

### Sintaxis

```
FSSize '('
  [ Name ':'= ] < expression (IN) of string>
  [ '\ Total ] | [ '\ Free ]
  [ '\ Kbyte ]
```

Continúa en la página siguiente

---



```
[ '\ ' Mbyte ] ' )'
```

Una función con un valor de retorno del tipo de dato num.

### Información relacionada

| Para obtener más información sobre        | Consulte   |
|---|--|
| Creación de un directorio                 | <a href="#">MakeDir - Crea un nuevo directorio en la página 358</a>          |
| Eliminación de un directorio              | <a href="#">RemoveDir - Elimina un directorio en la página 598</a>           |
| Cambio del nombre de un archivo           | <a href="#">RenameFile - Cambia el nombre de un archivo en la página 604</a> |
| Eliminación de un archivo                 | <a href="#">RemoveFile - Elimina un archivo en la página 600</a>             |
| Copia de un archivo                       | <a href="#">CopyFile - Copia un archivo en la página 169</a>                 |
| Comprobación del tipo del archivo         | <a href="#">IsFile - Comprobar el tipo de un archivo en la página 1351</a>   |
| Comprobación del tamaño del archivo       | <a href="#">FileSize - Obtiene el tamaño de un archivo en la página 1285</a> |
| Gestión de archivos y dispositivos de E/S | <a href="#">Application manual - Controller software IRC5</a>                |

## 2 Funciones

---

### 2.73 GetAxisDistance - Proporciona la información de distancia recorrida por el eje

### 2.73 GetAxisDistance - Proporciona la información de distancia recorrida por el eje

---

#### Utilización

`GetAxisDistance` Se utiliza para leer la distancia recorrida por el eje desde el último restablecimiento. Si el eje es giratorio, la distancia se verá en grados y si el eje es lineal, la distancia se dará en metros.

---

#### Ejemplos básicos

Los siguientes ejemplos ilustran la función `GetAxisDistance`.

##### Ejemplo 1

```
PERS dnum distance;  
distance := GetAxisDistance(Track,1);
```

La distancia total del eje 1 en una unidad mecánica `Track` recorrida desde el último restablecimiento se guarda en `distance`.

##### Ejemplo 2

```
PERS dnum distanceLimit := 1000;  
PERS dnum remaining;  
remaining := distanceLimit - GetAxisDistance(Track,1);
```

La distancia restante del eje 1 en una unidad mecánica `Track` se guarda en `remaining`.

---

#### Valor de retorno

Tipo de dato: `dnum`

El valor obtenido es la distancia, en metros o grados, que el eje ha recorrido desde el último restablecimiento.

---

#### Argumentos

```
GetAxisDistance (MechUnit AxisNo)
```

`MechUnit`

***Mechanical Unit***

Tipo de dato: `mecunit`

El nombre de la unidad mecánica.

`AxisNo`

Tipo de dato: `num`

El número del eje del que se obtendrá la distancia recorrida.

---

#### Sintaxis

```
GetAxisDistance '('  
  [ MechUnit ':= ' ] < variable (VAR) of mecunit > ','  
  [ AxisNo ':= ' ] < variable (VAR) of num > ')'
```

Una función con un valor de retorno del tipo de dato `dnum`.

---

*Continúa en la página siguiente*

**2.73 GetAxisDistance - Proporciona la información de distancia recorrida por el eje**  
*Continuación***Información relacionada**

| <b>Para obtener más información sobre</b> | <b>Consulte</b>  |
|---|--|
| ResetAxisDistance                         | <a href="#"><i>ResetAxisDistance - Restablece la información de distancia recorrida para el eje en la página 608</i></a> |
| ResetAxisMoveTime                         | <a href="#"><i>ResetAxisMoveTime - Restablece el cronómetro de movimiento del eje en la página 610</i></a>               |
| GetAxisMoveTime                           | <a href="#"><i>GetAxisMoveTime - Obtiene el valor del cronómetro de movimiento del eje en la página 1296</i></a>         |

## 2 Funciones

---

### 2.74 GetAxisMoveTime - Obtiene el valor del cronómetro de movimiento del eje

### 2.74 GetAxisMoveTime - Obtiene el valor del cronómetro de movimiento del eje

---

#### Utilización

`GetAxisMoveTime` Se utiliza para leer durante cuánto tiempo se ha movido el eje desde el último restablecimiento.

---

#### Ejemplos básicos

Los siguientes ejemplos ilustran la función `GetAxisMoveTime`.

##### Ejemplo 1

```
PERS dnum movetime;  
movetime := GetAxisMoveTime(Track,1);
```

La cantidad de tiempo que el eje 1 de la unidad mecánica `Track` se ha movido desde el último restablecimiento se guarda en `movetime`.

##### Ejemplo 2

```
PERS dnum timeLimit := 1000;  
PERS dnum remaining;  
remaining := timeLimit - GetAxisMoveTime(Track,1);
```

El tiempo restante del eje 1 de la unidad mecánica `Track` se guarda en `remaining`.

---

#### Valor de retorno

Tipo de dato: `dnum`

El valor obtenido es el tiempo total en horas que el eje se ha movido desde el último restablecimiento.

---

#### Argumentos

```
GetAxisMoveTime (MechUnit AxisNo)
```

`MechUnit`

*Mechanical Unit*

Tipo de dato: `mecunit`

El nombre de la unidad mecánica.

`AxisNo`

Tipo de dato: `num`

El número del eje en el que se leerá el tiempo de movimiento.

---

#### Sintaxis

```
GetAxisMoveTime '('  
  [ MechUnit ':='] < variable (VAR) of mecunit > ','  
  [ AxisNo ':='] < variable (VAR) of num > ')'
```

Una función con un valor de retorno del tipo de dato `dnum`.

---

#### Información relacionada

| Para obtener más información sobre | Consulte  |
|------------------------------------|---|
| <code>ResetAxisDistance</code>     | <a href="#">ResetAxisDistance - Restablece la información de distancia recorrida para el eje en la página 608</a> |

---

Continúa en la página siguiente

### 2.74 GetAxisMoveTime - Obtiene el valor del cronómetro de movimiento del eje

*Continuación*

| Para obtener más información sobre | Consulte  |
|------------------------------------|---|
| ResetAxisMoveTime                  | <a href="#"><i>ResetAxisMoveTime - Restablece el cronómetro de movimiento del eje en la página 610</i></a>              |
| GetAxisDistance                    | <a href="#"><i>GetAxisDistance - Proporciona la información de distancia recorrida por el eje en la página 1294</i></a> |

## 2 Funciones

---

2.75 GetMaxNumberOfCyclicBool - Obtener el número máximo de condiciones de Cyclic bool.  
RobotWare - OS

### 2.75 GetMaxNumberOfCyclicBool - Obtener el número máximo de condiciones de Cyclic bool.

---

#### Utilización

GetMaxNumberOfCyclicBool se utiliza para recuperar el número máximo de condiciones de Cyclic bool que pueden conectarse al mismo tiempo.

---

#### Ejemplos básicos

El ejemplo siguiente ilustra la función GetMaxNumberOfCyclicBool.

##### Ejemplo 1

```
VAR num maxno := 0;  
maxno := GetMaxNumberOfCyclicBool();  
TPWrite "Maximum cyclic bool: " \Num:=maxno;
```

El número máximo de condiciones de Cyclic bool se muestra en FlexPendant.

---

#### Valor de retorno

Tipo de dato: num

---

#### Sintaxis

```
GetMaxNumberOfCyclicBool '(' ' ')
```

Una función con un valor de retorno del tipo de dato num.

---

#### Información relacionada

| Para obtener más información sobre                             | Consulte   |
|--|--|
| Configurar una condición de Cyclic bool                        | <a href="#">SetupCyclicBool - Configurar una condición de Cyclic bool en la página 716</a>           |
| Eliminar una condición de Cyclic bool                          | <a href="#">RemoveCyclicBool - Eliminar una condición de Cyclic bool en la página 596</a>            |
| Eliminar todas las condiciones de Cyclic bool                  | <a href="#">RemoveAllCyclicBool - Eliminar todas las condiciones de Cyclic bool en la página 594</a> |
| Condiciones lógicas evaluadas cíclicamente, <i>Cyclic bool</i> | <i>Application manual - Controller software IRC5</i>   |
| Configuración de <i>Cyclic bool</i>                            | <i>Manual de referencia técnica - Parámetros del sistema</i>   |

## 2.76 GetMecUnitName - Obtener el nombre de la unidad mecánica

### Utilización

GetMecUnitName se utiliza para obtener el nombre de una unidad mecánica usando como argumento una de las unidades mecánicas instaladas. Esta función devuelve el nombre de las unidades mecánicas en forma de `string`.

### Ejemplos básicos

El ejemplo siguiente ilustra la función GetMecUnitName.

#### Ejemplo 1

```
VAR string mecname;
mecname:= GetMecUnitName(ROB1);
```

Se asigna a `mecname` el valor "ROB1" como un valor `string`. Todas las unidades mecánicas (tipo de dato `mecunit`), como por ejemplo ROB1, están predefinidas en el sistema.

### Valor de retorno

Tipo de dato: `string`

El valor de retorno será el nombre de la unidad mecánica como un valor `string`.

### Argumentos

```
GetMecUnitName ( MechUnit )
```

MechUnit

*Mechanical Unit*

Tipo de dato: `mecunit`

MechUnit acepta una de las unidades mecánicas predefinidas existentes en la configuración.

### Sintaxis

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

Una función con un valor de retorno del tipo de dato `string`.

### Información relacionada

| Para obtener más información sobre | Consulte  |
|------------------------------------|---|
| Unidad mecánica                    | <a href="#">mecunit - Unidad mecánica en la página 1703</a> |

## 2 Funciones

---

### 2.77 GetModalPayloadMode - Obtener el valor de ModalPayloadMode

RobotWare - OS

### 2.77 GetModalPayloadMode - Obtener el valor de ModalPayloadMode

---

#### Utilización

GetModalPayloadMode se utiliza para obtener el valor de ModalPayloadMode.

---

#### Ejemplos básicos

El ejemplo siguiente ilustra la función GetModalPayloadMode.

#### Ejemplo 1

```
IF GetModalPayloadMode() = 1 THEN
  GripLoad piece1;
  MoveL p1, v1000, fine, gripper;
ELSE
  MoveL p1, v1000, fine, tool2 \TLoad:=gripperpiece1;
ENDIF
```

Leer el valor de ModalPayloadMode del sistema y, en función del valor, utilizar un código diferente para especificar la carga utilizada en la instrucción de movimiento.

---

#### Valor de retorno

Tipo de dato: num

El valor de retorno será el valor de ModalPayloadMode en forma de num.

---

#### Sintaxis

```
GetModalPayloadMode '(' ' ')
```

Una función con un valor de retorno del tipo de dato num.

---

#### Información relacionada

| Para obtener más información sobre   | Consulte   |
|--|--|
| Parámetro de sistema <i>ModalPayloadMode</i> para la activación y la desactivación de la carga útil. | <a href="#">Manual de referencia técnica - Parámetros del sistema</a>                    |
| Uso de la carga útil en instrucciones de movimiento.   | <a href="#">MoveL - Mueve el robot siguiendo una trayectoria lineal en la página 447</a> |



---

## 2.78 GetMotorTorque - Lee el par motor actual

---

### Utilización

`GetMotorTorque` se utiliza para leer el par actual de los motores del robot y los ejes externos.

`GetMotorTorque` se utiliza principalmente para detectar si una pinza servo sostiene o no una carga.

---

### Ejemplos básicos

El ejemplo siguiente ilustra la función `GetMotorTorque`.

Consulte también [Más ejemplos en la página 1302](#).

#### Ejemplo 1

```
VAR num motor_torque2;  
motor_torque2 := GetMotorTorque(2);
```

El par motor actual del segundo eje del robot se almacena en `motor_torque2`.

---

### Valor de retorno

Tipo de dato: `num`

El par motor actual en newtons metro (Nm) del eje indicado del robot o de ejes externos.

---

### Argumentos

```
GetMotorTorque [ \MecUnit ] AxisNo
```

`MecUnit`

*Mechanical Unit*

Tipo de dato: `mecunit`

El nombre de la unidad mecánica cuyos valores de eje se desea comprobar. Si se omite este argumento, se obtiene el valor de un eje del robot conectado.

`AxisNo`

Tipo de dato: `num`

El número del eje cuyo valor se desea obtener (de 1 a 6).

---

### Ejecución de programas

Esta función lee el par motor filtrado actual aplicado sobre los motores del robot y los ejes externos.

El valor de par motor también puede verse como el número de señal de prueba 2000 al usar *TuneMaster*.

---

### Limitaciones

El resultado de `GetMotorTorque` variará en función de la fricción de la caja reductora, la temperatura del motor, etc. Como ejemplo, la temperatura de la caja reductora puede cambiar la fricción y, por consiguiente, el resultado.

---

*Continúa en la página siguiente*

## 2 Funciones

---

### 2.78 GetMotorTorque - Lee el par motor actual

RobotWare - OS

Continuación

Las limitaciones descritas arriba pueden impedir la detección de cambios muy pequeños en el par.

Solo es posible leer el par actual de las unidades mecánicas controladas desde la tarea de programa actual. En el caso de las tareas sin movimiento, es posible leer el par de las unidades mecánicas controladas por la tarea de movimiento conectada.

---

### Más ejemplos

Los siguientes ejemplos ilustran la función `GetMotorTorque`.

#### Ejemplo 1

```
VAR num torque_value;
torque_value := GetMotorTorque(\MecUnit:=STN1, 1);
```

El par motor actual del primer eje del robot de `STN1` se almacena en `torque_value`.

#### Ejemplo 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
..
```

El par motor actual del eje 5 del robot se lee antes de sujetar la pieza. A continuación, se sujeta la pieza. El par se lee de nuevo y se comparan los pares para detectar si existe una carga adicional actual en la pinza.

---

### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                    | Causa del error                            |
|---------------------------|--|
| <code>ERR_AXIS_PAR</code> | Parámetro de eje incorrecto en la función. |

Continúa en la página siguiente

#### Sintaxis

```
GetMotorTorque '('  
  ['\' MecUnit ':=' < variable (VAR) of mecunit> ',']  
  [AxisNo ':=' ] < expression (IN) of num> ')'
```

Una función con un valor de retorno del tipo de dato num.

#### Información relacionada

| Para obtener más información sobre      | Consulte  |
|---|---|
| Lee los ángulos actuales de los motores | <a href="#">ReadMotor - Lee los ángulos actuales de los motores en la página 1440</a> |

## 2 Funciones

---

### 2.79 GetNextCyclicBool - Obtener los nombres de todos los Cyclic bools

RobotWare - OS

### 2.79 GetNextCyclicBool - Obtener los nombres de todos los Cyclic bools

---

#### Utilización

GetNextCyclicBool se utiliza para recuperar los nombres de todos los Cyclic bools.

---

#### Ejemplos básicos

Los siguientes ejemplos ilustran la función GetNextCyclicBool.

##### Ejemplo 1

```
VAR num listno := 0;
VAR string name;
...
WHILE GetNextCyclicBool(listno, name) DO
  TPWrite "Cyclic bool: "+name;
  ! listno := listno + 1 is done by GetNextCyclicBool
ENDWHILE
```

Los nombres de todos los Cyclic bools conectados en el sistema se mostrarán en FlexPendant.

##### Ejemplo 2

```
PERS bool cyclicflag1;
TASK PERS bool cyclicflag2;

PROC main()
  SetupCyclicBool cyclicflag1, di1=1 AND do2=1;
  SetupCyclicBool cyclicflag2, di3=1 AND do4=0;
  WHILE GetNextCyclicBool(listno, name) DO
    TPWrite name;
    ! listno := listno + 1 is done by GetNextCyclicBool
  ENDWHILE
  ...
```

cyclicflag1 y T\_ROB1/cyclicflag1 se mostrarán en FlexPendant si se ejecuta el código RAPID en la tarea de RAPID T\_ROB1.

---

#### Valor de retorno

Tipo de dato: bool

El valor de retorno es TRUE si se encontrara un nombre de Cyclic bool, de lo contrario FALSE.

---

#### Argumentos

GetNextCyclicBool(ListNumber Name)

ListNumber

Tipo de dato: num

Este parámetro especifica qué Cyclic bool de la lista interna del sistema de nombres de Cyclic bool debe recuperarse. En el momento del retorno, el sistema siempre incrementa la variable en una unidad para facilitar el acceso al siguiente nombre

---

Continúa en la página siguiente

## 2.79 GetNextCyclicBool - Obtener los nombres de todos los Cyclic bools

RobotWare - OS

Continuación

de Cyclic bool de la lista. El primer nombre de Cyclic bool de la lista tiene el índice 0.

Name

Tipo de dato: `string`

El nombre de la variable persistente de Cyclic bool. Si la variable persistente se definiera como `TASK PERS`, el nombre recuperado será "Nombre de variable booleana persistente/nombre TASK".

## Sintaxis

```
GetNextCyclicBool '('
  [ ListNumber ':' ] < variable (VAR) of num> ','
  [ Name ':' ] < variable (VAR) of string>
  ')'
```

Una función con un valor de retorno del tipo de dato `bool`.

## Información relacionada

| Para obtener más información sobre                             | Consulte   |
|--|--|
| Comprobar si una variable persistente es un Cyclic bool        | <a href="#">IsCyclicBool - Comprueba si una variable persistente es un Cyclic bool en la página 1348</a> |
| Configurar una condición de Cyclic bool                        | <a href="#">SetupCyclicBool - Configurar una condición de Cyclic bool en la página 716</a>               |
| Eliminar una condición de Cyclic bool                          | <a href="#">RemoveCyclicBool - Eliminar una condición de Cyclic bool en la página 596</a>                |
| Eliminar todas las condiciones de Cyclic bool                  | <a href="#">RemoveAllCyclicBool - Eliminar todas las condiciones de Cyclic bool en la página 594</a>     |
| Condiciones lógicas evaluadas cíclicamente, <i>Cyclic bool</i> | <i>Application manual - Controller software IRC5</i>   |
| Configuración de <i>Cyclic bool</i>                            | <i>Manual de referencia técnica - Parámetros del sistema</i>   |

## 2 Funciones

---

### 2.80 GetNextMechUnit - Obtener el nombre y los datos de las unidades mecánicas RobotWare - OS

### 2.80 GetNextMechUnit - Obtener el nombre y los datos de las unidades mecánicas

---

#### Utilización

GetNextMechUnit (*Get Next Mechanical Unit*) se utiliza para obtener los nombres de las unidades mecánicas del sistema de robot. Aparte del nombre de la unidad mecánica, es posible obtener distintas propiedades opcionales de la unidad mecánica.

---

#### Ejemplos básicos

El ejemplo siguiente ilustra la función GetNextMechUnit.

Consulte también [Más ejemplos en la página 1307](#).

#### Ejemplo 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
```

Se muestran en el FlexPendant los nombres de todas las unidades mecánicas disponibles en el sistema.

---

#### Valor de retorno

Tipo de dato: bool

TRUE si se encuentra alguna unidad mecánica. De lo contrario, FALSE.

---

#### Argumentos

```
GetNextMechUnit( ListNumber UnitName [\MecRef] [\TCPRob] [\NoOfAxes]
                [\MecTaskNo] [\MotPlanNo] [\Active] [\DriveModule]
                [\OKToDeact])
```

ListNumber

Tipo de dato: num

Este parámetro especifica qué elementos de la lista de unidades mecánicas interna del sistema se desea obtener. En el momento del retorno, la variable es siempre incrementada por el sistema en una unidad, para facilitar el acceso a la siguiente unidad de la lista. La primera unidad mecánica de la lista tiene el número de índice 0.

UnitName

Tipo de dato: string

El nombre de la unidad mecánica.

[\MecRef]

Tipo de dato: mecunit

La referencia del sistema a la unidad mecánica.

---

*Continúa en la página siguiente*

### 2.80 GetNextMechUnit - Obtener el nombre y los datos de las unidades mecánicas *RobotWare - OS* *Continuación*

[ \TCPRob ]

Tipo de dato: bool

TRUE si la unidad mecánica es un robot de TCP. De lo contrario, FALSE.

[ \NoOfAxes ]

Tipo de dato: num

Número de ejes de la unidad mecánica. Valor entero.

[ \MecTaskNo ]

Tipo de dato: num

El número de la tarea de programa que controla a la unidad mecánica. Valor entero en el rango 1-20. Si no está controlada por ninguna tarea de programa, se devuelve -1.

Esta conexión real se define en los parámetros del sistema del dominio del controlador (en alguna aplicación puede ser definida en tiempo de ejecución).

[ \MotPlanNo ]

Tipo de dato: num

El número del planificador de movimientos que controla la unidad mecánica. Valor entero en el rango 1-6. Si no está controlada por ningún planificador de movimientos, se devuelve -1.

Esta conexión se define en los parámetros del sistema del dominio del controlador.

[ \Active ]

Tipo de dato: bool

TRUE si la unidad mecánica está activada. De lo contrario, FALSE.

[ \DriveModule ]

Tipo de dato: num

El número de Drive Module 1 - 4 utilizado por esta unidad mecánica.

[ \OKToDeact ]

Tipo de dato: bool

Devuelve TRUE si se permite la desactivación de la unidad mecánica desde el programa de RAPID.

---

### Más ejemplos

A continuación aparecen más ejemplos de la instrucción `GetNextMechUnit`.

#### Ejemplo 1

```
VAR num listno := 4;
VAR string name := "";
VAR bool found := FALSE;

found := GetNextMechUnit (listno, name);
```

Si se cambia `found` a TRUE, el nombre de la unidad mecánica con número 4 aparecerá en la variable `name`. De lo contrario, `name` sólo contiene una cadena vacía.

*Continúa en la página siguiente*

## 2 Funciones

### 2.80 GetNextMechUnit - Obtener el nombre y los datos de las unidades mecánicas

RobotWare - OS

Continuación

#### Sintaxis

```
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> ]  
  [ '\ ' OKToDeact ':= ' < variable (VAR) of bool> ]  
' )'
```

Una función con un valor de retorno del tipo de dato bool.

#### Información relacionada

| Para obtener más información sobre               | Consulte   |
|--|--|
| Unidad mecánica                                  | <a href="#">mecunit - Unidad mecánica en la página 1703</a>  |
| Activación y desactivación de unidades mecánicas | <a href="#">ActUnit - Activa una unidad mecánica en la página 30</a><br><a href="#">DeactUnit - Desactiva una unidad mecánica en la página 187</a> |
| Características de los tipos de datos sin valor  | <a href="#">Manual de referencia técnica - RAPID Overview, sección Características básicas - Tipos de datos</a>                                    |



## 2.81 GetNextSym - Obtiene el siguiente símbolo coincidente

### Utilización

GetNextSym (*Get Next Symbol*) se utiliza junto con SetDataSearch para obtener objetos de datos del sistema.

### Ejemplos básicos

El ejemplo siguiente ilustra la función GetNextSym.

#### Ejemplo 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
```

Esta sesión cambia a TRUE todos los objetos de datos locales de tipo bool cuyo nombre comience con my en el módulo mymod.

### Valor de retorno

Tipo de dato: bool

TRUE si se ha obtenido un nuevo objeto. El nombre del objeto y el bloque que contiene se devuelven a través de los argumentos.

FALSE si no se han encontrado más objetos coincidentes.

### Argumentos

```
GetNextSym (Object Block [\Recursive])
```

Object

Tipo de dato: string

Una variable (VAR o PERS) para almacenar el nombre del objeto de datos que se obtendrá.

Block

Tipo de dato: datapos

El bloque que contiene el objeto.

[ \Recursive ]

Tipo de dato: switch

Este modificador obliga a que la búsqueda entre en el bloque inferior. Por ejemplo, si la sesión de búsqueda ha comenzado en el nivel de tarea, también buscará en los módulos y las rutinas que se encuentran por debajo de la tarea.

### Sintaxis

```
GetNextSym '('
    [ Object ':=' ] < variable or persistent (INOUT) of string > ','
```

*Continúa en la página siguiente*

## 2 Funciones

### 2.81 GetNextSym - Obtiene el siguiente símbolo coincidente

RobotWare - OS

Continuación

```
[ Block '[:=' ] <variable (VAR) of datapos>  
[ '\ Recursive ] )'
```

Una función con un valor de retorno del tipo de dato `bool`.

#### Información relacionada

| Para obtener más información sobre                              | Consulte  |
|---|---|
| Definición de un conjunto de símbolos en una sesión de búsqueda | <a href="#">SetDataSearch - Definir el conjunto de símbolos de una secuencia de búsqueda en la página 696</a>             |
| Obtención del valor de un objeto de datos                       | <a href="#">GetDataVal - Obtiene el valor de un objeto de datos en la página 228</a>                                      |
| Asignación del valor de un objeto de datos                      | <a href="#">SetDataVal - Establece el valor de un objeto de datos en la página 701</a>                                    |
| Asignación del valor de varios objetos de datos                 | <a href="#">SetAllDataVal - Establece un valor en todos los objetos de datos de un conjunto definido en la página 691</a> |
| El tipo de datos relacionado <code>datapos</code>               | <a href="#">datapos - Inclusión de un bloque para un objeto de datos en la página 1657</a>                                |
| <i>Advanced RAPID</i>   | <i>Especificaciones del producto - Controller software IRC5</i>   |

## 2.82 GetNumberOfCyclicBool - Obtener el número de condiciones de Cyclic bool

### Utilización

`GetNumberOfCyclicBool` se utiliza para recuperar el número de condiciones conectadas de Cyclic bool.

### Ejemplos básicos

El ejemplo siguiente ilustra la función `GetNumberOfCyclicBool`.

#### Ejemplo 1

```
VAR num listno := 0;
listno := GetNumberOfCyclicBool();
TPWrite "Connected Cyclic bool: " \Num:=listno;
```

El número de condiciones conectadas de Cyclic bool se muestra en FlexPendant.

### Valor de retorno

Tipo de dato: num

### Sintaxis

```
GetNumberOfCyclicBool '(' ' ')
```

Una función con un valor de retorno del tipo de dato num.

### Información relacionada

| Para obtener más información sobre                             | Consulte   |
|--|--|
| Configurar una condición de Cyclic bool                        | <a href="#">SetupCyclicBool - Configurar una condición de Cyclic bool en la página 716</a>           |
| Eliminar una condición de Cyclic bool                          | <a href="#">RemoveCyclicBool - Eliminar una condición de Cyclic bool en la página 596</a>            |
| Eliminar todas las condiciones de Cyclic bool                  | <a href="#">RemoveAllCyclicBool - Eliminar todas las condiciones de Cyclic bool en la página 594</a> |
| Condiciones lógicas evaluadas cíclicamente, <i>Cyclic bool</i> | <i>Application manual - Controller software IRC5</i>   |
| Configuración de <i>Cyclic bool</i>                            | <i>Manual de referencia técnica - Parámetros del sistema</i>   |

## 2 Funciones

---

### 2.83 GetServiceInfo - Obtener información de servicio del sistema RobotWare - OS

### 2.83 GetServiceInfo - Obtener información de servicio del sistema

---

#### Utilización

`GetServiceInfo` se utiliza para leer información de servicio del sistema. Esta función devuelve la información de servicio en forma de `string`.

---

#### Ejemplos básicos

El ejemplo siguiente ilustra la función `GetServiceInfo`.  
Consulte también [Más ejemplos en la página 1313](#).

#### Ejemplo 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
```

Si la tarea controla un robot, utilice la variable predefinida `ROB_ID` para leer el contador de tiempo de funcionamiento. A continuación, convierta el valor de cadena en un valor numérico.

---

#### Valor de retorno

Tipo de dato: `string`

El valor de la unidad de servicio de la unidad mecánica especificada. Para obtener más información acerca de los valores de retorno, consulte *Argumentos a continuación*.

---

#### Argumentos

```
GetServiceInfo (MechUnit [\DutyTimeCnt])
```

`MechUnit`

##### *Mechanical Unit*

Tipo de dato: `mecunit`

El nombre de la unidad mecánica cuya información se desea obtener.

`[\DutyTimeCnt]`

##### *Duty Time Counter*

Tipo de dato: `switch`

Devuelve el contador de tiempo de funcionamiento de la unidad mecánica utilizada en el argumento `MechUnit`. Se devuelve una cadena que contiene "0" si esta opción se usa en el controlador virtual.

El contador de tiempo de funcionamiento es el valor en horas que la unidad mecánica ha estado en Motores ON y los frenos se han liberado.

---

*Continúa en la página siguiente*

**Ejecución de programas**

Se lee la información de servicio correspondiente al parámetro opcional utilizado.

**Más ejemplos**

A continuación aparecen más ejemplos de cómo usar la función `GetServiceInfo`.

**Ejemplo 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;
```

Obtiene información acerca del contador de tiempo de funcionamiento de todas las unidades mecánicas de un sistema MultiMove y escribe los valores en el FlexPendant.

**Sintaxis**

```
GetServiceInfo '('
  [MechUnit ':' ] <variable (VAR) of mecunit> ','
  ['\' DutyTimeCnt ] ')'
```

Una función con un valor de retorno del tipo de dato `string`.

**Información relacionada**

| Para obtener más información sobre | Consulte   |
|------------------------------------|--|
| Unidad mecánica                    | <a href="#">mecunit - Unidad mecánica en la página 1703.</a> |

## 2 Funciones

### 2.84 GetSignalOrigin - Obtención de información acerca del origen de una señal de E/S RobotWare - OS

### 2.84 GetSignalOrigin - Obtención de información acerca del origen de una señal de E/S

#### Utilización

GetSignalOrigin se utiliza para obtener información acerca del origen de la señal de E/S.

#### Ejemplos básicos

Los siguientes ejemplos ilustran la función GetSignalOrigin:

##### Ejemplo 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
```

El código anterior puede usarse para determinar el origen de la señal con el nombre mysignal.

#### Valor de retorno

Tipo de dato: signalorigin

signalorigin se describe en la tabla que aparece a continuación.

| Valor de retorno | Constante simbólica | Comentario   |
|------------------|---------------------|--|
| 0                | SIGORIG_NONE        | La variable de señal de E/S es declarada en RAPID y no tiene ningún alias asociado.  |
| 1                | SIGORIG_CFG         | La señal se configura en la configuración de E/S.  |
| 2                | SIGORIG_ALIAS       | La variable de señal de E/S es declarada en RAPID y tiene un alias asociado a una señal de E/S configurada en la configuración de E/S. |

#### Argumentos

GetSignalOrigin Signal SignalName

Signal

Tipo de dato: signalxx

El nombre de la señal. Debe ser del tipo de dato signaldo, signaldi, signalgo, signalgi, signalao o signalai.

SignalName

Tipo de dato: string

Continúa en la página siguiente

## 2.84 GetSignalOrigin - Obtención de información acerca del origen de una señal de E/S

RobotWare - OS  
Continuación

El nombre de la señal de acuerdo con la configuración de E/S o la cadena vacía.

### Ejecución de programas

La función devuelve uno de los orígenes de señal predefinidos siguientes:

SIGORIG\_NONE, SIGORIG\_CFG o SIGORIG\_ALIAS.

Si devuelve SIGORIG\_NONE, SignalName consta de una cadena vacía.

Si devuelve SIGORIG\_CFG o SIGORIG\_ALIAS, el argumento SignalName contiene el nombre de señal de E/S de acuerdo con la configuración de E/S.

GetSignalOrigin puede utilizarse en programas genéricos para comprobar si una señal tiene un alias asociado y si es una asociación a la señal de E/S física adecuada.

### Sintaxis

```
GetSignalOrigin
  [Signal ':='] <variable (VAR) of anytype>', '
  [SignalName ':='] <variable (VAR) of string>';'
```

### Información relacionada

| Para obtener más información sobre                      | Consulte   |
|---|--|
| Instrucciones de entrada/salida                         | <i>Manual de referencia técnica - RAPID Overview</i>   |
| Funcionalidad de entrada/salida en general              | <i>Manual de referencia técnica - RAPID Overview</i>   |
| Configuración de E/S                                    | <i>Manual de referencia técnica - Parámetros del sistema</i>                                       |
| Definición de señales de E/S con un nombre de alias     | <a href="#">AliasIO - Define una señal de E/S con un nombre de alias en la página 37</a>           |
| Restablecimiento de señal de E/S con un nombre de alias | <a href="#">AliasIOReset - Restablecer una señal de E/S con un nombre de alias en la página 40</a> |
| Tipo de dato signalorigin                               | <a href="#">signalorigin - Describe el origen de la señal de E/S en la página 1761</a>             |

## 2 Funciones

### 2.85 GetSysInfo - Obtener información acerca del sistema

RobotWare - OS

### 2.85 GetSysInfo - Obtener información acerca del sistema

#### Utilización

`GetSysInfo` se utiliza para leer información acerca del sistema. Los datos disponibles son el número de serie, la versión del software, el nombre de versión del software, el tipo de robot, la ID de controlador, la dirección IP de WAN, el idioma de controlador y el nombre del sistema.

#### Ejemplos básicos

El ejemplo siguiente ilustra la función `GetSysInfo`.

#### Ejemplo 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);
```

#### Ejemplos de cadenas devueltas:

| Descripción                   | Variable    | Valor de retorno       |
|-------------------------------|-------------|------------------------|
| Número de serie               | serial      | 24-12345               |
| Versión de software           | version     | ROBOTWARE_6.03.xxxx    |
| Nombre de versión de software | versionname | 6.03.00.00             |
| Nº de robot                   | rtype       | IRB 2400-16/1.5 Type A |
| ID de controlador             | cid         | MyRobot                |
| Dirección IP de WAN           | lanip       | 192.168.8.103          |
| Idioma del controlador        | clang       | en                     |
| Sistema activo                | sysname     | MySystem               |

#### Valor de retorno

Tipo de dato: `string`

El valor de retorno es una cadena con el número de serie, versión del software, nombre de versión del software, tipo de robot, ID de controlador, dirección IP de WAN, idioma del controlador y nombre del sistema. Encontrará más información acerca de los valores de retorno en [Argumentos en la página 1317](#), a continuación.

*Continúa en la página siguiente*



**Argumentos**

```
GetSysInfo ([\SerialNo] | [\SWVersion] | [\SWVersionName] |
[\RobotType] | [\CtrlId] | [\LanIp] | [\CtrlLang] |
[\SystemName])
```

Al menos uno de los argumentos debe estar presente.

[\SerialNo]

**Serial Number**

Tipo de dato: switch

Devuelve el número de serie.

[\SWVersion]

**Software Version**

Tipo de dato: switch

Devuelve la versión de soporte de datos de RobotWare, tal como está instalada en la carpeta *PRODUCTS*.

[\SWVersionName]

**Software Version Name**

Tipo de dato: switch

Devuelve el nombre visible de la versión de soporte de datos de RobotWare.

[\RobotType]

Tipo de dato: switch

Devuelve el tipo de robot en la tarea actual o conectada. Si la unidad mecánica no es un robot de TCP, se devuelve un guión -.

[\CtrlId]

**Controller ID**

Tipo de dato: switch

Devuelve una ID de controlador. Devuelve una cadena vacía si no se ha especificado ninguna ID de controlador. Se devuelve la cadena que contiene VC si se usa en el controlador virtual.

[\LanIp]

**Lan Ip address**

Tipo de dato: switch

Devuelve la dirección IP de WAN del controlador. Se devuelve la cadena que contiene VC si se usa en el controlador virtual. Se devuelve una cadena vacía si no hay ninguna dirección IP de WAN configurada en el sistema.

[\CtrlLang]

**Controller Language**

Tipo de dato: switch

Devuelve el idioma utilizado en el controlador.

| Valor de retorno | Idioma |
|------------------|--------|
| cs               | Checo  |

Continúa en la página siguiente

## 2 Funciones

### 2.85 GetSysInfo - Obtener información acerca del sistema

RobotWare - OS

Continuación

| Valor de retorno | Idioma  |
|------------------|---|
| zh               | Chino (chino simplificado, chino continental) |
| da               | Danés   |
| nl               | Holandés                                      |
| en               | Inglés  |
| fi               | Finlandés                                     |
| fr               | Francés                                       |
| de               | Alemán  |
| hu               | Húngaro                                       |
| it               | Italiano                                      |
| ja               | Japonés                                       |
| ko               | Coreano                                       |
| pl               | Polaco  |
| pt               | Portugués (portugués de Brasil)               |
| ro               | Rumano  |
| ru               | Ruso  |
| sl               | Esloveno                                      |
| es               | Español                                       |
| sv               | Sueco   |
| tr               | Turco   |

[ \SystemName ]

Tipo de dato: switch

Devuelve el nombre del sistema activo.

#### Sintaxis

```
GetSysInfo '('  
    ['\'SerialNo]  
    |['\' SWVersion]  
    |['\' SWVersionName]  
    |['\' RobotType]  
    |['\' CtrlId]  
    |['\' LanIp]  
    |['\' CtrlLang]  
    |['\' SystemName]')'
```

Una función con un valor de retorno del tipo de dato string.

#### Información relacionada

| Para obtener más información sobre | Consulte   |
|------------------------------------|--|
| Comprobar la identidad del sistema | <a href="#">IsSysID - Comprobar la identidad del sistema en la página 1366</a> |

---

## 2.86 GetTaskName - Obtiene el nombre y el número de la tarea actual

### Utilización

`GetTaskName` se utiliza para obtener la identidad de la tarea de programa actual, con su nombre y su número.

Desde algunas *tareas sin movimiento*, también es posible obtener el nombre y el número de su *tarea de movimiento* conectada. En el caso de los sistemas *MultiMove*, el parámetro del sistema *Controller/Tasks/Use Mechanical Unit Group* define la *tarea de movimiento* conectada, mientras que en un sistema básico la tarea principal siempre es la *tarea de movimiento* conectada desde cualquier otra tarea.

---

### Ejemplos básicos

Los siguientes ejemplos ilustran la función `GetTaskName`.

#### Ejemplo 1

```
VAR string taskname;
...
taskname := GetTaskName();
```

El nombre de la tarea actual se devuelve en la variable `taskname`.

#### Ejemplo 2

```
VAR string taskname;
VAR num taskno;
...
taskname := GetTaskName(\TaskNo:=taskno);
```

El nombre de la tarea actual se devuelve en la variable `taskname`. La identidad entera de la tarea se almacena en la variable `taskno`.

#### Ejemplo 3

```
VAR string taskname;
VAR num taskno;
...
taskname := GetTaskName(\MecTaskNo:=taskno);
```

Si la tarea actual es una *tarea sin movimiento*, el nombre de la tarea de movimiento conectada se devuelve en la variable `taskname`. La identidad numérica de la tarea de movimiento conectada se almacena en la variable `taskno`.

Si la tarea actual controla varias unidades mecánicas, el nombre de la tarea actual se devuelve en la variable `taskname`. La identidad numérica de la tarea se almacena en la variable `taskno`.

---

### Valor de retorno

Tipo de dato: `string`

El nombre de la tarea en la que se ejecuta la función, o bien el nombre de la tarea de movimiento conectada.

---

### Argumentos

```
GetTaskName ( [\TaskNo] | [\MecTaskNo] )
```

*Continúa en la página siguiente*

## 2 Funciones

---

### 2.86 GetTaskName - Obtiene el nombre y el número de la tarea actual

RobotWare - OS

Continuación

[ \TaskNo ]

Tipo de dato: num

Devuelve el nombre de la tarea actual (con la misma funcionalidad que si no se usa ninguno de los modificadores \TaskNo ni \MecTaskNo). Obtiene también la identidad de la tarea actual, representada como un valor entero. Los números devueltos estarán en el rango entre 1 y 20.

[ \MecTaskNo ]

Tipo de dato: num

Devuelve el nombre de la tarea de movimiento conectada o el nombre de la tarea de movimiento actual. También obtiene la identidad de la tarea de movimiento conectada o actual, representada como un valor entero. Los números devueltos estarán en el rango entre 1 y 20.

---

### Sintaxis

```
GetTaskName '('  
  [ \TaskNo ':=' ] < variable (VAR) of num >  
  [ \MecTaskNo ':=' ] < variable (VAR) of num > ')'
```

Una función con un valor de retorno del tipo de dato string.

---

### Información relacionada

| Para obtener más información sobre                      | Consulte   |
|---|--|
| Comprobación de si la tarea controla algún robot de TCP | <a href="#">TaskRunRob - Comprueba si una tarea controla algún robot en la página 1516</a>   |
| Multitarea  | <i>Manual de referencia técnica - RAPID Overview, sección Resumen sobre RAPID - Multitarea</i><br><i>Manual de referencia técnica - RAPID Overview, sección Características básicas - Multitarea</i> |

---

## 2.87 GetTime - Lee la hora actual como un valor numérico

---

### Utilización

`GetTime` se utiliza para leer un componente concreto del sistema actual como un valor numérico.

`GetTime` puede usarse para:

- Hacer que el programa realice una acción a una hora determinada
- Realizar determinadas actividades en un día laborable
- No realizar determinadas actividades durante el fin de semana
- Responder de una forma distinta ante los errores en función de la hora del día

---

### Ejemplos básicos

El ejemplo siguiente ilustra la función `GetTime`.

Consulte también [Más ejemplos en la página 1322](#).

#### Ejemplo 1

```
hour := GetTime(\Hour);
```

La hora actual se almacena en la variable `hour`.

---

### Valor de retorno

Tipo de dato: `num`

Uno de los cuatro componentes de hora especificados a continuación.

---

### Argumento

```
GetTime ( [\WDay] | [\Hour] | [\Min] | [\Sec] )
```

`[\WDay]`

Tipo de dato: `switch`

Devuelve el día de la semana. Rango: de 1 a 7 (de lunes a domingo).

`[\Hour]`

Tipo de dato: `switch`

Devuelve la hora actual. Rango: de 0 a 23.

`[\Min]`

Tipo de dato: `switch`

Devuelve el minuto actual. Rango: de 0 a 59.

`[\Sec]`

Tipo de dato: `switch`

Devuelve el segundo actual. Rango: de 0 a 59.

Es necesario especificar al menos uno de los argumentos. De lo contrario, la ejecución del programa se detiene y se genera un mensaje de error.

*Continúa en la página siguiente*

## 2 Funciones

---

### 2.87 GetTime - Lee la hora actual como un valor numérico

RobotWare - OS

Continuación

---

#### Más ejemplos

A continuación aparecen más ejemplos de la función `GetTime`.

#### Ejemplo 1

```
weekday := GetTime(\WDay);
hour := GetTime(\Hour);
IF weekday < 6 AND hour >6 AND hour < 16 THEN
  production;
ELSE
  maintenance;
ENDIF
```

Si el día actual es un día laborable y la hora está entre las 7 y las 15:49 horas, el robot realiza tareas de producción. En cualquier otro momento, el robot se encuentra en el modo de mantenimiento.

---

#### Sintaxis

```
GetTime '('
  ['\ ' WDay ]
  | [ '\ ' Hour ]
  | [ '\ ' Min ]
  | [ '\ ' Sec ] ')'
```

Una función con un valor de retorno del tipo de dato `num`.

---

#### Información relacionada

| Para obtener más información sobre   | Consulte   |
|--------------------------------------|--|
| Instrucciones de hora y fecha        | <i>Manual de referencia técnica - RAPID Overview</i> , sección <i>Resumen sobre RAPID - Sistema &amp; tiempo</i> |
| Cambio de hora del reloj del sistema | <i>Manual del operador - IRC5 con FlexPendant</i> , sección <i>Cómo cambiar la configuración del FlexPendant</i> |

---

## 2.88 GetTSPStatus: Obtener el estado del panel de selección de tareas actuales

---

### Utilización

GetTSPStatus se utiliza para comprobar si una tarea está activada o desactivada en el **Panel de selección de tareas de FlexPendant**.

---

### Ejemplos básicos

El ejemplo siguiente ilustra la función GetTSPStatus.

#### Ejemplo 1

```
VAR tsp_status tspstatus;
...
tspstatus:=GetTSPStatus("MYTASK");
IF tspstatus >= TSP_NORMAL_UNCHECKED AND tspstatus <=
    TSP_SEMISTATIC_UNCHECKED THEN
    TPWrite "Task MYTASK is unchecked in the Task Selection Panel";
ELSEIF tspstatus >= TSP_NORMAL_CHECKED THEN
    TPWrite "Task MYTASK is checked in the Task Selection Panel";
ELSE
    TPWrite "Task MYTASK is unchecked in TSP due to execution in
        service routine";
ENDIF
```

Compruebe si la tarea MYTASK del programa está activada o desactivada en el **Panel de selección de tareas de FlexPendant**.

---

### Valor de retorno

Tipo de dato: `tsp_status`

El estado del panel de selección de tareas actuales.

---

### Datos predefinidos

Pueden utilizarse las siguientes constantes simbólicas predefinidas del tipo `tsp_status` para comprobar el valor de retorno:

```
CONST tsp_status TSP_UNCHECKED_RUN_SERV_ROUT := 10;
CONST tsp_status TSP_NORMAL_UNCHECKED := 11;
CONST tsp_status TSP_STATIC_UNCHECKED := 12;
CONST tsp_status TSP_SEMISTATIC_UNCHECKED := 13;
CONST tsp_status TSP_NORMAL_CHECKED := 14;
CONST tsp_status TSP_STATIC_CHECKED := 15;
CONST tsp_status TSP_SEMISTATIC_CHECKED := 16;
```

---

### Argumentos

GetTSPStatus ( TaskRef | TaskName )

TaskRef

Tipo de dato: `taskid`

La identidad de la tarea de programa que debe comprobarse.

Las variables predefinidas del tipo de datos `taskid` está disponible para todas las tareas de programa del sistema.

---

*Continúa en la página siguiente*

## 2 Funciones

---

### 2.88 GetTSPStatus: Obtener el estado del panel de selección de tareas actuales

RobotWare - OS

Continuación

La identidad variable es "taskname"+"Id", por ejemplo, la identidad variable de la tarea T\_ROB1 es T\_ROB1Id.

TaskName

Tipo de dato: string

El nombre de la tarea de programa que debe comprobarse.

---

### Gestión de errores

Se generan los siguientes errores recuperables que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` se establecerá en:

|              |  |
|--------------|--|
| ERR_TASKNAME | El nombre de tarea de programa en el argumento \TaskName no puede encontrarse en el sistema. |
|--------------|--|

---

### Sintaxis

```
GetTSPStatus '('  
  [ TaskRef ':= ' ] <variable (VAR) of taskid>  
  |[ TaskName ':= ' ] <expression (IN) of string> ')'
```

Una función con un valor de retorno del tipo de dato `tsp_status`.

---

### Información relacionada

| Para obtener más información sobre        | Consulte   |
|---|--|
| Estado de panel de selección de tareas    | <a href="#">tsp_status: Estado de panel de selección de tareas en la página 1817</a>     |
| Comprobar si una tarea normal está activa | <a href="#">TasksActive: Comprobar si una tarea normal está activa en la página 1519</a> |



## 2.89 GetUASUserName - Obtener el nombre de usuario del usuario

### Utilización

GetUASUserName se utiliza para obtener el nombre de usuario del usuario actualmente conectado en FlexPendant.

### Ejemplos básicos

El ejemplo siguiente ilustra la función GetUASUserName.

#### Ejemplo 1

```
VAR string strUser;
...
strUser := GetUASUserName();
```

El nombre de usuario del usuario actualmente conectado se devuelve en la variable strUser.

### Valor de retorno

Tipo de dato: string

El nombre de usuario del usuario actualmente conectado.

### Más ejemplos

El ejemplo siguiente ilustra la función GetUASUserName.

Consulte también [Más ejemplos en la página 1325](#).

#### Ejemplo 1

```
VAR string user_name:="No user";
...
IF UIClientExist() = TRUE THEN
    user_name:=GetUASUserName();
ENDIF
```

El nombre de usuario del usuario actualmente conectado se devuelve en la variable user\_name. Si el sistema está en funcionamiento son ningún FlexPendant, el user\_name contiene la cadena "No user".

### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema ERRNO cambiará a:

| Nombre           | Causa del error  |
|------------------|--|
| ERR_TP_NO_CLIENT | No hay ningún usuario registrado en el FlexPendant, es decir no hay ningún FlexPendant en uso actualmente. |

### Sintaxis

```
GetUASUserName(' ')
```

Una función con un valor de retorno del tipo de dato string.

*Continúa en la página siguiente*

## 2 Funciones

---

2.89 GetUASUserName - Obtener el nombre de usuario del usuario

RobotWare - OS

Continuación

---

### Información relacionada

| Para obtener más información sobre        | Consulte  |
|---|---|
| User Authorization System (UAS)           | <i>Manual del operador - RobotStudio</i>                                    |
| Comprobar si el cliente de usuario existe | <a href="#">UIClientExist - Existe cliente de usuario en la página 1552</a> |

## 2.90 GInput - Lee el valor de una señal de entrada de grupo

### Utilización

GInput se utiliza para leer el valor actual de un grupo de señales digitales de entrada.



#### Nota

Recuerde que la función GInput es una función antigua que ya no es necesario utilizar. Consulte los ejemplos relativos a una forma alternativa y recomendada de programar.

### Ejemplos básicos

El ejemplo siguiente ilustra la función GInput.

#### Ejemplo 1

```
IF GInput(gi2) = 5 THEN ...
...
IF gi2 = 5 THEN ...
```

Si el valor actual de la señal gi2 es igual a 5, entonces ...

### Valor de retorno

Tipo de dato: num

El valor actual de la señal (un entero positivo).

Se leen los valores de las distintas señales del grupo, que se interpretan como números binarios sin signo. Estos números binarios se convierten a continuación a enteros.

El valor devuelto se encuentra dentro de un rango que depende del número de señales del grupo.

| Número de señales | Valor permitido |
|-------------------|-----------------|
| 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          |

*Continúa en la página siguiente*

## 2 Funciones

### 2.90 GInput - Lee el valor de una señal de entrada de grupo

RobotWare - OS

Continuación

| Número de señales | Valor permitido |
|-------------------|-----------------|
| 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       |
| 23                | 0-8388607       |

#### Argumentos

GInput (Signal)

Signal

Tipo de dato: signalgi

El nombre del grupo de señales a leer.

#### Sintaxis

```
GInput '('  
  [Signal ':=' ] <variable (VAR) of signalgi>')
```

Una función con un valor de retorno del tipo de dato num.

#### Información relacionada

| Para obtener más información sobre                               | Consulte   |
|--|--|
| Leer el valor de una señal de entrada de grupo de más de 23 bits | <a href="#">GInputDnum - Lee el valor de una señal de entrada de grupo en la página 1329</a>                     |
| Instrucciones de entrada/salida                                  | <i>Manual de referencia técnica - RAPID Overview, sección Resúmen sobre RAPID - Señales de entrada y salida</i>  |
| Funcionalidad de entrada/salida en general                       | <i>Manual de referencia técnica - RAPID Overview, sección Principios de movimiento y E/S - Principios de E/S</i> |
| Configuración de E/S   | <i>Manual de referencia técnica - Parámetros del sistema</i>   |

## 2.91 GInputDnum - Lee el valor de una señal de entrada de grupo

### Utilización

GInputDnum se utiliza para leer el valor actual de un grupo de señales digitales de entrada de más de 23 bits.

### Ejemplos básicos

Los siguientes ejemplos ilustran la función GInputDnum.

#### Ejemplo 1

```
IF GInputDnum(gi2) = 55 THEN ...
```

Si el valor actual de la señal gi2 es igual a 55, entonces ...

#### Ejemplo 2

```
IF GInputDnum(gi2) = 4294967295 THEN ...
```

Si el valor actual de la señal gi2 es igual a 4294967295, entonces ...

### Valor de retorno

Tipo de dato: dnum

El valor actual de la señal (un entero positivo).

Se leen los valores de las distintas señales del grupo, que se interpretan como números binarios sin signo. Estos números binarios se convierten a continuación a enteros.

El valor devuelto se encuentra dentro de un rango que depende del número de señales del grupo.

| Número de señales | Valor permitido |
|-------------------|-----------------|
| 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        |

*Continúa en la página siguiente*

## 2 Funciones

### 2.91 GInputDnum - Lee el valor de una señal de entrada de grupo

RobotWare - OS

Continuación

| Número de señales | Valor permitido |
|-------------------|-----------------|
| 18                | 0-262143        |
| 19                | 0-524287        |
| 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    |

#### Argumentos

GInputDnum (Signal)

Signal

Tipo de dato: signalgi

El nombre del grupo de señales a leer.

#### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre             | Causa del error   |
|--------------------|---|
| ERR_NO_ALIASIO_DEF | La variable de señal es una variable declarada en RAPID. No se ha conectado a una señal E/S definida en la configuración de E/S con la instrucción <code>AliasIO</code> .                             |
| ERR_NORUNUNIT      | Se ha perdido el contacto con el dispositivo de E/S.  |
| ERR_SIG_NOT_VALID  | La señal de E/S no está disponible. Los motivos pueden ser que el dispositivo de E/S no está en funcionamiento o que exista un error en la configuración (válido solamente para el bus de campo ICI). |

#### Sintaxis

```
GInputDnum '('  
  [ Signal ':' = ' ] < variable (VAR) of signalgi > ')'
```

Una función con un valor de retorno del tipo de dato `dnum`.

Continúa en la página siguiente

2.91 GInputDnum - Lee el valor de una señal de entrada de grupo  
RobotWare - OS  
Continuación

## Información relacionada

| Para obtener más información sobre             | Consulte   |
|--|--|
| Leer el valor de una señal de entrada de grupo | <a href="#">GInput - Lee el valor de una señal de entrada de grupo en la página 1327</a>                         |
| Instrucciones de entrada/salida                | <i>Manual de referencia técnica - RAPID Overview, sección Resumen sobre RAPID - Señales de entrada y salida</i>  |
| Funcionalidad de entrada/salida en general     | <i>Manual de referencia técnica - RAPID Overview, sección Principios de movimiento y E/S - Principios de E/S</i> |
| Configuración de E/S                           | <i>Manual de referencia técnica - Parámetros del sistema</i>   |

## 2 Funciones

2.92 GOutput - Lee el valor de un grupo de señales digitales de salida  
RobotWare - OS

### 2.92 GOutput - Lee el valor de un grupo de señales digitales de salida

#### Utilización

GOutput se utiliza para leer el valor actual de un grupo de señales digitales de salida.

#### Ejemplos básicos

El ejemplo siguiente ilustra la función GOutput.

#### Ejemplo 1

```
IF GOutput(go2) = 5 THEN ...
```

Si el valor actual de la señal go2 es 5, entonces ...

#### Valor de retorno

Tipo de dato: num

El valor actual de la señal (un entero positivo).

Se leen los valores de las distintas señales del grupo, que se interpretan como números binarios sin signo. Estos números binarios se convierten a continuación a enteros.

El valor devuelto se encuentra dentro de un rango que depende del número de señales del grupo.

| Número de señales | Valor permitido |
|-------------------|-----------------|
| 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       |

Continúa en la página siguiente



## 2.92 GOutput - Lee el valor de un grupo de señales digitales de salida

RobotWare - OS

Continuación

| Número de señales | Valor permitido |
|-------------------|-----------------|
| 21                | 0-2097151       |
| 22                | 0-4194303       |
| 23                | 0-8388607       |

## Argumentos

GOutput (Signal)

Signal

Tipo de dato: signalgo

El nombre del grupo de señales a leer.

## Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre             | Causa del error   |
|--------------------|---|
| ERR_NO_ALIASIO_DEF | La variable de señal es una variable declarada en RAPID. No se ha conectado a una señal E/S definida en la configuración de E/S con la instrucción <code>AliasIO</code> .                             |
| ERR_NORUNUNIT      | Se ha perdido el contacto con el dispositivo de E/S.  |
| ERR_SIG_NOT_VALID  | La señal de E/S no está disponible. Los motivos pueden ser que el dispositivo de E/S no está en funcionamiento o que exista un error en la configuración (válido solamente para el bus de campo ICI). |

## Sintaxis

```
GOutput '('
  [ Signal ::= ] < variable (VAR) of signalgo > ')'
```

Una función con un valor de retorno del tipo de dato `num`.

## Información relacionada

| Para obtener más información sobre               | Consulte   |
|--|--|
| Establecimiento de un grupo de señales de salida | <a href="#">SetGO - Cambia el valor de un grupo de señales digitales de salida en la página 707</a>              |
| Lee un grupo de señales de salida                | <a href="#">GOutputDnum - Lee el valor de una señal de salida de grupo en la página 1334</a>                     |
| Lee un grupo de señales de entrada               | <a href="#">GInputDnum - Lee el valor de una señal de entrada de grupo en la página 1329</a>                     |
| Instrucciones de entrada/salida                  | <i>Manual de referencia técnica - RAPID Overview, sección Resumen sobre RAPID - Señales de entrada y salida</i>  |
| Funcionalidad de entrada/salida en general       | <i>Manual de referencia técnica - RAPID Overview, sección Principios de movimiento y E/S - Principios de E/S</i> |
| Configuración de E/S                             | <i>Manual de referencia técnica - Parámetros del sistema</i>   |

## 2 Funciones

---

2.93 GOutputDnum - Lee el valor de una señal de salida de grupo  
RobotWare - OS

### 2.93 GOutputDnum - Lee el valor de una señal de salida de grupo

---

#### Utilización

GOutputDnum se utiliza para leer el valor actual de un grupo de señales digitales de salida de más de 23 bits.

---

#### Ejemplos básicos

Los siguientes ejemplos ilustran la función GOutputDnum.

##### Ejemplo 1

```
IF GOutputDnum(go2) = 55 THEN ...
```

Si el valor actual de la señal go2 es igual a 55, entonces ...

##### Ejemplo 2

```
IF GOutputDnum(go2) = 4294967295 THEN ...
```

Si el valor actual de la señal go2 es igual a 4294967295, entonces ...

---

#### Valor de retorno

Tipo de dato: dnum

El valor actual de la señal (un entero positivo).

Se leen los valores de las distintas señales del grupo, que se interpretan como números binarios sin signo. Estos números binarios se convierten a continuación a enteros.

El valor devuelto se encuentra dentro de un rango que depende del número de señales del grupo.

| Número de señales | Valor permitido |
|-------------------|-----------------|
| 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        |

Continúa en la página siguiente

2.93 GOutputDnum - Lee el valor de una señal de salida de grupo  
RobotWare - OS  
Continuación

| Número de señales | Valor permitido |
|-------------------|-----------------|
| 18                | 0-262143        |
| 19                | 0-524287        |
| 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    |

### Argumentos

GOutputDnum (Signal)

Signal

Tipo de dato: signalgo

El nombre del grupo de señales a leer.

### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre             | Causa del error   |
|--------------------|---|
| ERR_NO_ALIASIO_DEF | La variable de señal es una variable declarada en RAPID. No se ha conectado a una señal E/S definida en la configuración de E/S con la instrucción AliasIO.   |
| ERR_NORUNUNIT      | Se ha perdido el contacto con el dispositivo de E/S.  |
| ERR_SIG_NOT_VALID  | La señal de E/S no está disponible. Los motivos pueden ser que el dispositivo de E/S no está en funcionamiento o que exista un error en la configuración (válido solamente para el bus de campo ICI). |

### Sintaxis

```
GOutputDnum '('
  [ Signal ':=' ] < variable (VAR) of signalgo > ')'
```

Una función con un valor de retorno del tipo de dato dnum.

Continúa en la página siguiente

## 2 Funciones

---

2.93 GOutputDnum - Lee el valor de una señal de salida de grupo

RobotWare - OS

Continuación

---

### Información relacionada

| Para obtener más información sobre               | Consulte  |
|--|---|
| Establecimiento de un grupo de señales de salida | <a href="#">SetGO - Cambia el valor de un grupo de señales digitales de salida en la página 707</a>                           |
| Instrucciones de entrada/salida                  | <i>Manual de referencia técnica - Descripción general de RAPID, sección Resumen sobre RAPID - Señales de entrada y salida</i> |
| Funcionalidad de entrada/salida en general       | <i>Manual de referencia técnica - Descripción general de RAPID, sección Principios de movimiento y E/S</i>                    |
| Configuración de E/S                             | <i>Manual de referencia técnica - Parámetros del sistema</i>  |

## 2.94 HexToDec - Convierte de hexadecimal a decimal

### Utilización

`HexToDec` se usa para convertir un número especificado en una cadena que admita lectura de la base 16 a la base 10.

La cadena de entrada se construye con el conjunto de caracteres [0-9,A-F,a-f].

Esta rutina admite los números del 0 al 9223372036854775807 en decimal o 7FFFFFFFFFFFFFFF en hexadecimal.

### Ejemplos básicos

El ejemplo siguiente ilustra la función `HexToDec`.

#### Ejemplo 1

```
VAR string str;

str := HexToDec("5F5E0FF");
```

Se asigna a la variable `str` el valor "99999999".

### Valor de retorno

Tipo de dato: `string`

La cadena convertida a la representación decimal a partir del número indicado en la cadena del parámetro de entrada.

### Argumentos

```
HexToDec ( Str )
```

`Str`

*String*

Tipo de dato: `string`

La cadena a convertir.

### Sintaxis

```
HexToDec '('  
  [ Str ':=' ] <expression (IN) of string> ')'
```

Una función con un valor de retorno del tipo de dato `string`.

### Información relacionada

| Para obtener más información sobre   | Consulte   |
|--------------------------------------|--|
| Funciones para cadenas de caracteres | <i>Manual de referencia técnica - RAPID Overview</i> , sección <i>Resumen sobre RAPID - Funciones para cadenas de caracteres</i> |
| Definición de cadena de caracteres   | <a href="#">string - Cadenas en la página 1778</a>   |
| Valores de cadena de caracteres      | <i>Manual de referencia técnica - RAPID Overview</i> , sección <i>Características básicas - Elementos básicos</i>                |

## 2 Funciones

---

### 2.95 IndInpos - Estado de posición de un eje independiente *Independent Axis*

### 2.95 IndInpos - Estado de posición de un eje independiente

---

#### Utilización

IndInpos se utiliza para comprobar si un eje independiente ha alcanzado la posición seleccionada.

---

#### Ejemplos básicos

El ejemplo siguiente ilustra la función IndInpos.

#### Ejemplo 1

```
IndAMove Station_A,1\ToAbsNum:=90,20;  
WaitUntil IndInpos(Station_A,1) = TRUE;  
WaitTime 0.2;
```

Se espera hasta que el eje 1 de Station\_A se encuentre en la posición de 90 grados.

---

#### Valor de retorno

Tipo de dato: bool

La tabla describe los valores de retorno de IndInpos:

| Valor de retorno | Estado del eje   |
|------------------|--|
| TRUE             | En posición y con velocidad cero.                              |
| FALSE            | Aún no situado en la posición y/o no tiene una velocidad cero. |

---

#### Argumentos

```
IndInpos ( MecUnit Axis )
```

MecUnit

*Mechanical Unit*

Tipo de dato: mecunit

El nombre de la unidad mecánica.

Eje

Tipo de dato: num

El número del eje actual de la unidad mecánica (del 1 al 6).

---

#### Limitaciones

Un eje independiente ejecutado con la instrucción IndCMove devuelve siempre el valor FALSE, incluso si la velocidad tiene el valor cero.

Se debe añadir un periodo de espera de 0,2 segundos tras la instrucción, para garantizar que se ha alcanzado el estado correcto. Este periodo debe ser mayor en el caso de los ejes externos que presenten un rendimiento deficiente.

---

*Continúa en la página siguiente*

**Gestión de errores**

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre       | Causa del error                          |
|--------------|--|
| ERR_AXIS_ACT | El eje no está activado.                 |
| ERR_AXIS_IND | El eje no está en el modo independiente. |

**Sintaxis**

```
IndInpos '('
  [MecUnit ':='] <variable (VAR) of mecunit>', '
  [Axis ':='] <expression (IN) of num>')'
```

Una función con un valor de retorno del tipo de dato `bool`.

**Información relacionada**

| Para obtener más información sobre                              | Consulte   |
|---|--|
| Ejes independientes en general                                  | <i>Manual de referencia técnica - RAPID Overview, sección Principios de movimiento y E/S - Posicionamiento durante la ejecución del programa</i> |
| Otras instrucciones y funciones independientes                  | <i>Manual de referencia técnica - RAPID Overview, sección Resumen sobre RAPID - Movimiento</i>   |
| Comprobación del estado de velocidad de los ejes independientes | <a href="#">IndInpos - Estado de velocidad de un eje independiente en la página 1340</a>   |
| Activación de ejes independientes                               | <i>Manual de referencia técnica - Parámetros del sistema, tema Motion, tipo Arm</i>  |

## 2 Funciones

---

### 2.96 IndInpos - Estado de velocidad de un eje independiente *Independent Axis*

### 2.96 IndInpos - Estado de velocidad de un eje independiente

---

#### Utilización

IndSpeed se utiliza para comprobar si un eje independiente ha alcanzado la velocidad seleccionada.

---

#### Ejemplos básicos

El ejemplo siguiente ilustra la función IndSpeed.

#### Ejemplo 1

```
IndCMove Station_A, 2, 3.4;  
WaitUntil IndSpeed(Station_A,2 \InSpeed) = TRUE;  
WaitTime 0.2;
```

Se espera hasta que el eje 2 de Station\_A haya alcanzado la velocidad de 3,4 grados/s.

---

#### Valor de retorno

Tipo de dato: bool

La tabla describe los valores de retorno de IndSpeed \IndSpeed:

| Valor de retorno | Estado del eje                                |
|------------------|---|
| TRUE             | Se ha alcanzado la velocidad seleccionada.    |
| FALSE            | No se ha alcanzado la velocidad seleccionada. |

La tabla describe los valores de retorno de IndSpeed \ZeroSpeed:

| Valor de retorno | Estado del eje             |
|------------------|----------------------------|
| TRUE             | Velocidad cero.            |
| FALSE            | Velocidad distinta de cero |

---

#### Argumentos

```
IndSpeed ( MecUnit Axis [ \InSpeed ] | [ \ZeroSpeed ] )
```

MecUnit

**Mechanical Unit**

Tipo de dato: mecunit

El nombre de la unidad mecánica.

Axis

Tipo de dato: num

El número del eje actual de la unidad mecánica (del 1 al 6).

[ \InSpeed ]

Tipo de dato: switch

IndSpeed devuelve el valor TRUE si el eje ha alcanzado la velocidad seleccionada, o FALSE si no lo ha hecho.

---

*Continúa en la página siguiente*



[ \ZeroSpeed ]

**Tipo de dato:** switch

IndSpeed devuelve el valor TRUE si el eje tiene la velocidad cero, o FALSE si no la tiene.

Si se utiliza tanto el argumento \InSpeed como \ZeroSpeed, se muestra un mensaje de error.

### Limitación

La función IndSpeed\InSpeed siempre devuelve el valor FALSE en las situaciones siguientes:

- El robot se encuentra en el modo manual a velocidad reducida.
- La velocidad se reduce mediante la instrucción VelSet.
- La velocidad se reduce desde la ventana de producción.

Se debe añadir un periodo de espera de 0,2 segundos tras la instrucción, para garantizar que se ha obtenido el estado correcto. Este periodo debe ser mayor en el caso de los ejes externos que presenten un rendimiento deficiente.

### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema ERRNO cambiará a:

| Nombre       | Causa del error                          |
|--------------|--|
| ERR_AXIS_ACT | El eje no está activado.                 |
| ERR_AXIS_IND | El eje no está en el modo independiente. |

### Sintaxis

```
IndSpeed '('
  [MecUnit ':='] <variable (VAR) of mecunit>','
  [Axis ':='] <expression (IN) of num>
  ['\' InSpeed] | ['\' ZeroSpeed]')'
```

Una función con un valor de retorno del tipo de dato bool.

### Información relacionada

| Para obtener más información sobre                             | Consulte   |
|--|--|
| Ejes independientes en general                                 | <i>Manual de referencia técnica - RAPID Overview, sección Principios de movimiento y E/S - Posicionamiento durante la ejecución del programa</i> |
| Otras instrucciones y funciones independientes                 | <i>Manual de referencia técnica - RAPID Overview, sección Resumen sobre RAPID - Movimiento</i>   |
| Más ejemplos   | <a href="#">IndCMove - Movimiento independiente continuo en la página 268</a>  |
| Comprobación del estado de posición de los ejes independientes | <a href="#">IndInpos - Estado de posición de un eje independiente en la página 1338</a>  |
| Activación de ejes independientes                              | <i>Manual de referencia técnica - Parámetros del sistema, tema Motion, tipo Arm</i>  |

## 2 Funciones

---

### 2.97 IOUnitState - Obtener el estado actual de un dispositivo de E/S

RobotWare - OS

### 2.97 IOUnitState - Obtener el estado actual de un dispositivo de E/S

---

#### Utilización

IOUnitState se utiliza para determinar el estado actual de un dispositivo de E/S. Su estado físico y su estado lógico definen el estado de un dispositivo de E/S.

---

#### Ejemplos básicos

Los siguientes ejemplos ilustran la función IOUnitState.

##### Ejemplo 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
```

Se realiza una comprobación para ver si el dispositivo de E/S UNIT1 está funcionando correctamente.

##### Ejemplo 2

```
IF (IOUnitState("UNIT1" \Logic)=IOUNIT_LOG_STATE_DISABLED) THEN
  ! Unit is disabled by user from RAPID or FlexPendant
ELSE
  ! Unit is enabled.
ENDIF
```

Se realiza una comprobación para ver si el dispositivo de E/S UNIT1 está deshabilitado.

---

#### Valor de retorno

Tipo de dato: iounit\_state

El valor de retorno tiene valores diferentes en función de si se utilizan los argumentos opcionales \Logic o \Phys o de si no se utiliza ningún argumento opcional.

Los estados lógicos del dispositivo de E/S describen el estado cuya activación en el dispositivo de E/S puede solicitar el usuario. El estado del dispositivo de E/S se define en la tabla siguiente cuando se utiliza el argumento opcional \Logic.

| Valor de retorno | Constante simbólica       | Comentario   |
|------------------|---------------------------|--|
| 10               | IOUNIT_LOG_STATE_DISABLED | El dispositivo de E/S ha sido deshabilitado por el usuario desde RAPID, FlexPendant o parámetros del sistema.  |
| 11               | IOUNIT_LOG_STATE_ENABLED  | El dispositivo de E/S ha sido habilitado por el usuario desde RAPID, FlexPendant o parámetros del sistema. Es la opción predeterminada tras el inicio. |

Cuando el dispositivo de E/S es habilitado lógicamente por el usuario y el controlador del bus de campo intenta poner un dispositivo de E/S en el estado

*Continúa en la página siguiente*

físico IOUNIT\_PHYS\_STATE\_RUNNING, el dispositivo de E/S podría terminar en otros estados por distintos motivos (consulte la tabla siguiente).

El estado del dispositivo de E/S se define en la tabla siguiente cuando se utiliza el argumento opcional `\Phys`.

| Valor de retorno | Constante simbólica            | Comentario   |
|------------------|--------------------------------|--|
| 20               | IOUNIT_PHYS_STATE_DEACTIVATED  | Dispositivo de E/S sin funcionamiento, deshabilitado por el usuario  |
| 21               | IOUNIT_PHYS_STATE_RUNNING      | Dispositivo de E/S en funcionamiento   |
| 22               | IOUNIT_PHYS_STATE_ERROR        | El dispositivo de E/S no funciona a causa de algún error de tiempo de ejecución                            |
| 23               | IOUNIT_PHYS_STATE_UNCONNECTED  | El dispositivo de E/S está configurado pero no está conectado a la red de E/S o la red de E/S está parada. |
| 24               | IOUNIT_PHYS_STATE_UNCONFIGURED | El dispositivo de E/S no está configurado aunque está conectado a la red de E/S. <sup>1)</sup>             |
| 25               | IOUNIT_PHYS_STATE_STARTUP      | El dispositivo de E/S está en el modo de puesta en marcha. <sup>1)</sup>                                   |
| 26               | IOUNIT_PHYS_STATE_INIT         | El dispositivo de E/S ha sido creado. <sup>1)</sup>  |



#### Nota

El estado del dispositivo de E/S se define en la tabla siguiente cuando no se utiliza ninguno de los argumentos opcionales, `\Phys` ni `\Logic`.

| Valor de retorno | Constante simbólica | Comentario   |
|------------------|---------------------|--|
| 1                | IOUNIT_RUNNING      | El dispositivo de E/S funciona correctamente                                       |
| 2                | IOUNIT_RUNERROR     | El dispositivo de E/S no funciona a causa de algún error de tiempo de ejecución    |
| 3                | IOUNIT_DISABLE      | El dispositivo de E/S ha sido desactivado por el usuario desde RAPID o FlexPendant |
| 4                | IOUNIT_OTHERERR     | Otros errores de configuración o puesta en marcha                                  |

<sup>1)</sup> No es posible obtener este estado en el programa de RAPID con la versión actual de RobotWare - OS.

#### Argumentos

```
IOUnitState (UnitName [\Phys] | [\Logic])
```

Continúa en la página siguiente

## 2 Funciones

### 2.97 IOUnitState - Obtener el estado actual de un dispositivo de E/S

RobotWare - OS

Continuación

UnitName

Tipo de dato: string

El nombre del dispositivo de E/S que se desea comprobar (con el mismo nombre con el que se configuró).

[\Phys]

*Físico*

Tipo de dato: switch

Si se utiliza este parámetro, se lee el estado físico del dispositivo de E/S .

[\Logic]

*Lógico*

Tipo de dato: switch

Si se utiliza este parámetro, se lee el estado lógico del dispositivo de E/S.

---

#### Sintaxis

```
IOUnitState '('  
  [ UnitName ':=' ] < expression (IN) of string >  
  [ '\ Phys' | [ '\ Logic' ] )'
```

Una función con un valor de retorno del tipo de dato `iounit_state`.

---

#### Información relacionada

| Para obtener más información sobre         | Consulte   |
|--|--|
| Estado del dispositivo de E/S              | <a href="#">iounit_state - Estado del dispositivo de E/S en la página 1690</a>                                   |
| Habilitación de un dispositivo de E/S      | <a href="#">IOEnable - Activar un dispositivo de E/S en la página 296</a>  |
| Deshabilitación de un dispositivo de E/S   | <a href="#">IODisable - Desactivar un dispositivo de E/S en la página 293</a>                                    |
| Instrucciones de entrada/salida            | <i>Manual de referencia técnica - RAPID Overview, sección Resumen sobre RAPID - Señales de entrada y salida</i>  |
| Funcionalidad de entrada/salida en general | <i>Manual de referencia técnica - RAPID Overview, sección Principios de movimiento y E/S - Principios de E/S</i> |
| Configuración de E/S                       | <i>Manual de referencia técnica - Parámetros del sistema</i>   |

## 2.98 IsBrakeCheckActive: Comprobar si la prueba de frenos está en marcha

### Utilización

IsBrakeCheckActive se utiliza para comprobar si se está realizando alguna prueba de frenos, es decir, si los procedimientos CyclicBrakeCheck o BrakeCheck están activos (en ejecución o detenidos) en cualquier nivel de ejecución.

### Ejemplos básicos

El ejemplo siguiente ilustra la función IsBrakeCheckActive.

#### Ejemplo 1

```
WHILE IsBrakeCheckActive() = TRUE THEN
  WaitTime 1;
ENDWHILE
...
```

Compruebe si hay alguna rutina de prueba de frenos activa. Si está activa, espere a que termine.

### Valor de retorno

Tipo de dato: bool

La función devuelve TRUE si se está realizando una prueba de frenos.

### Sintaxis

```
IsBrakeCheckActive '(' '')
```

Una función con un valor de retorno del tipo de dato bool.

### Información relacionada

| Para obtener más información sobre | Consulte                                   |
|------------------------------------|--|
| BrakeCheck                         | Manual del operador - IRC5 con FlexPendant |

## 2 Funciones

---

2.99 IsCollFree: comprueba si la posición colisionaría  
RobotWare - OS

### 2.99 IsCollFree: comprueba si la posición colisionaría

---

#### Utilización

IsCollFree se utiliza para probar si el robot colisionaría si se situara en una posición de ejes, proporcionada como un argumento para la función. En sistemas *MultiMove*, pueden proporcionarse varias posiciones de ejes para diferentes robots.

---

#### Ejemplos básicos

El ejemplo siguiente ilustra la función IsCollFree.

##### Ejemplo 1

```
IF IsCollFree(testpos) THEN
  MoveAbsJ testpos, v50, fine, tool1;
ENDIF
```

Comprueba si la posición (jointtarget) testpos está libre de colisiones antes de realizar el movimiento hasta la posición concreta.

##### Ejemplo 2

```
IF IsCollFree(testpos1 \Rob2Pos:=testpos2 \Rob3Pos:=testpos3) THEN
  TPWRITE "No collisions"
ENDIF
```

Asumiéndose que una llamada de la tarea RAPID controla Rob1. Comprueba si las posiciones testpos1 para Rob1, testpos2 para Rob2 y testpos3 para Rob3 provocarán o no una colisión.

---

#### Valor de retorno

Tipo de dato: bool

La función devuelve TRUE si las posiciones de ejes especificadas no provocarán una colisión; en caso contrario FALSE.

---

#### Argumentos

```
IsCollFree ThisRobotPos [\Rob1Pos] [\Rob2Pos] [\Rob3Pos] [\Rob4Pos]
```

ThisRobotPos

Tipo de dato: jointtarget

La posición para la que se comprueba si está libre o no de colisiones, para el robot controlado por la llamada de la tarea RAPID.

Rob1Pos

Tipo de dato: jointtarget

Comprobación para verificar si ocurre una colisión cuando Rob1 también tiene la posición Rob1Pos en sistemas *MultiMove*. No se puede establecer si la llamada de la tarea RAPID controla Rob1, ya que ThisRobotPos especifica la posición de este robot.

Rob2Pos

Tipo de dato: jointtarget

*Continúa en la página siguiente*

Comprobación para verificar si ocurre una colisión cuando Rob2 también tiene la posición `Rob2Pos` en sistemas *MultiMove*. No se puede establecer si la llamada de la tarea RAPID controla Rob2, ya que `ThisRobotPos` especifica la posición de este robot.

Rob3Pos

Tipo de dato: `jointtarget`

Comprobación para verificar si ocurre una colisión cuando Rob3 también tiene la posición `Rob3Pos` en sistemas *MultiMove*. No se puede establecer si la llamada de la tarea RAPID controla Rob3, ya que `ThisRobotPos` especifica la posición de este robot.

Rob4Pos

Tipo de dato: `jointtarget`

Comprobación para verificar si ocurre una colisión cuando Rob4 también tiene la posición `Rob4Pos` en sistemas *MultiMove*. No se puede establecer si la llamada de la tarea RAPID controla Rob4, ya que `ThisRobotPos` especifica la posición de este robot.

### Limitaciones

- En el sistema *MultiMove*, si no se proporciona un argumento opcional, la función utilizará la posición medida del robot. Para obtener resultados fiables, los robots no indicados en `IsCollFree` deben estar en reposo.
- No comprueba si el movimiento hasta la posición provista provocará una colisión. Como en el Ejemplo 1 anterior, la posición `testpos` puede estar libre de colisiones, pero el movimiento hacia esta posición puede provocar una colisión.

### Sintaxis

```
IsCollFree '('
  [ ThisRobotPos ':' '=' ] < expression (IN) of jointtarget > ','
  [ '\ ' Rob1Pos ':' '=' < expression (IN) of jointtarget > ]
  [ '\ ' Rob2Pos ':' '=' < expression (IN) of jointtarget > ]
  [ '\ ' Rob3Pos ':' '=' < expression (IN) of jointtarget > ]
  [ '\ ' Rob4Pos ':' '=' < expression (IN) of jointtarget > ]
  ')'
```

Una función con un valor de retorno del tipo de dato `bool`.

### Información relacionada

| Para obtener más información sobre | Consulte   |
|------------------------------------|--|
| Predicción de colisiones           | <i>Application manual - Controller software IRC5</i> |

## 2 Funciones

---

### 2.100 IsCyclicBool - Comprueba si una variable persistente es un Cyclic bool RobotWare - OS

### 2.100 IsCyclicBool - Comprueba si una variable persistente es un Cyclic bool

---

#### Utilización

IsCyclicBool se utiliza para comprobar si un booleano persistente es un Cyclic bool, es decir, si se ha conectado una condición lógica a la variable booleana persistente con la instrucción SetupCyclicBool.

---

#### Ejemplos básicos

Los siguientes ejemplos ilustran la función IsCyclicBool.

##### Ejemplo 1

```
PERS bool cyclicflag1;

PROC main()
  TPWrite "cyclicflag1 is a cyclic bool:
        "\Bool:=IsCyclicBool(cyclicflag1);
  SetupCyclicBool cyclicflag1, dil=1 AND do2=1;
  TPWrite "cyclicflag1 is a cyclic bool:
        "\Bool:=IsCyclicBool(cyclicflag1);
  ...
```

El texto `cyclicflag1 is a cyclic bool: FALSE` se escribe primero en FlexPendant. Después de la ejecución de SetupCyclicBool la variable booleana persistente es un Cyclic bool y el segundo texto será `cyclicflag1 is a cyclic bool: TRUE`.

##### Ejemplo 2

```
TASK PERS bool cyclicflag1;

PROC main()
  SetupCyclicBool cyclicflag1, dil=1 AND do2=1;
  TPWrite "cyclicflag1 is a cyclic bool: "
        "\Bool:=IsCyclicBool("cyclicflag1");
  ...
```

Uso de una cadena de texto como entrada para especificar el nombre del Cyclic bool en la función IsCyclicBool. El texto escrito en FlexPendant será `cyclicflag1 is a cyclic bool: TRUE`.

##### Ejemplo 3

```
..
TPWrite "cyclicflag1 is a cyclic bool: "
        "\Bool:=IsCyclicBool("cyclicflag1", \TaskName:="T_ROB1");
..
```

Uso de una cadena de texto como entrada para especificar el nombre de Cyclic bool en la función IsCyclicBool. El texto escrito en FlexPendant será `cyclicflag1 is a cyclic bool: TRUE` si se ha conectado cyclicflag1 a una condición lógica con la instrucción SetupCyclicBool en la tarea de RAPID T\_ROB1; de lo contrario, el texto escrito en FlexPendant será `cyclicflag1 is a cyclic bool: FALSE`.

*Continúa en la página siguiente*



---

#### Valor de retorno

Tipo de dato: bool

La función devolverá **TRUE** si se ha conectado una condición lógica al booleano persistente con la instrucción `SetupCyclicBool`, de lo contrario **FALSE**.

---

#### Argumentos

`IsCyclicBool (Flag | Name [\TaskRef] | [\TaskName])`

Flag

Tipo de dato: bool

La variable booleana persistente que debe comprobarse.

Name

Tipo de dato: string

El nombre de la variable booleana persistente que debe comprobarse.

[\TaskRef]

#### *Task Reference*

Tipo de dato: taskid

La identidad de la tarea de programa donde se ha ejecutado la instrucción `SetupCyclicBool`. Este argumento solo debe utilizarse para un Cyclic bool que se declare como **TASK PERS** y al utilizar la función `IsCyclicBool` desde una tarea de **RAPID** que no ha conectado la condición lógica a la variable booleana persistente con la instrucción `SetupCyclicBool`.

Existen variables predefinidas con el tipo de dato `taskid` para todas las tareas de programa del sistema. La identificación de la variable será "nombre\_tarea"+"ID". Por ejemplo, para la tarea `T_ROB1` la identificación de la variable es `T_ROB1Id`.

[\TaskName]

Tipo de dato: string

El nombre de la tarea de programa donde se ha ejecutado la instrucción `SetupCyclicBool`. Este argumento solo debe utilizarse para un Cyclic bool que se declare como **TASK PERS** y al utilizar la función `IsCyclicBool` desde una tarea de **RAPID** que no ha conectado la condición lógica a la variable booleana persistente con la instrucción `SetupCyclicBool`.

Si no se especifica ninguno de los argumentos, ni `\TaskRef` ni `\TaskName`, se usa la tarea actual.

---

#### Ejecución de programas

Los nombres de los Cyclic bools se almacenan en el sistema como una cadena de caracteres. Para **PERS bool m1** el nombre almacenado es `m1`. Para **TASK PERS bool m2** el nombre será `"T_ROB1/m2"` si la configuración se realiza con la instrucción `SetupCyclicBool` en la tarea de **RAPID** `T_ROB1`.

Si la función se utiliza con el argumento `Flag` o `Name`, comprueba primero si existe el nombre persistente en la lista de Cyclic bools para ver si es una variable declarada **PERS** que se ha conectado a una condición con `SetupCyclicBool`.

Si no encontrara ningún Cyclic bool con ese nombre, también comprueba si es

*Continúa en la página siguiente*

## 2 Funciones

### 2.100 IsCyclicBool - Comprueba si una variable persistente es un Cyclic bool

RobotWare - OS

Continuación

una TASK PERS añadiendo la tarea que se está ejecutando actualmente antes del nombre del nombre persistente ("T\_ROB1/name").

#### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema ERRNO cambiará a:

| Nombre       | Causa del error  |
|--------------|--|
| ERR_TASKNAME | El nombre de tarea de programa en el argumento \TaskName no puede encontrarse en el sistema. |

#### Sintaxis

```
IsCyclicBool '('  
  [ [ Flag ':= ' ] <persistent (PERS) of bool>  
  | [ [ Name ':= ' ] <expression (IN) of string> ', '  
  | '\ ' TaskRef ':= ' <variable (VAR) of taskid>]  
  | [ '\ ' TaskName ':= ' <expression (IN) of string> ] )'
```

Una función con un valor de retorno del tipo de dato bool.

#### Información relacionada

| Para obtener más información sobre                      | Consulte   |
|---|--|
| Configurar una condición de Cyclic bool                 | <a href="#">SetupCyclicBool - Configurar una condición de Cyclic bool en la página 716</a>           |
| Eliminar una condición de Cyclic bool                   | <a href="#">RemoveCyclicBool - Eliminar una condición de Cyclic bool en la página 596</a>            |
| Eliminar todas las condiciones de Cyclic bool           | <a href="#">RemoveAllCyclicBool - Eliminar todas las condiciones de Cyclic bool en la página 594</a> |
| Condiciones lógicas evaluadas cíclicamente, Cyclic bool | <a href="#">Application manual - Controller software IRC5</a>  |
| Configuración de Cyclic bool                            | <a href="#">Manual de referencia técnica - Parámetros del sistema</a>                                |

---

## 2.101 IsFile - Comprobar el tipo de un archivo

---

### Utilización

La función `IsFile` obtiene información acerca del archivo o directorio indicado y comprueba si es del mismo tipo que el especificado. Si no se especifica ningún tipo, sólo se realiza una comprobación de existencia.

El nombre del archivo se especifica en un argumento de trayectoria. No se requieren permisos de lectura, escritura ni ejecución para el archivo indicado, pero debe ser posible leer el contenido de todos los directorios indicados en la ruta que conduce hasta el archivo.

---

### Ejemplos básicos

El ejemplo siguiente ilustra la función `IsFile`.

Consulte también [Más ejemplos en la página 1352](#).

#### Ejemplo 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
```

En este ejemplo se imprime en el FlexPendant el `filename` y el tipo del archivo especificado.

---

### Valor de retorno

Tipo de dato: `bool`

La función devuelve `TRUE` si el tipo especificado y el tipo real coinciden. De lo contrario, devuelve `FALSE`. Si no se especifica ningún tipo, devuelve `TRUE` si el archivo existe y `FALSE` si no existe.

*Continúa en la página siguiente*

## 2 Funciones

---

### 2.101 IsFile - Comprobar el tipo de un archivo

RobotWare - OS

Continuación

---

#### Argumentos

```
IsFile (Path [\Directory] [\Fifo] [\RegFile] [\BlockSpec]
        [\CharSpec])
```

Path

Tipo de dato: string

El archivo especificado con una ruta completa o relativa.

[ \Directory ]

Tipo de dato: switch

Indica si el archivo es un directorio.

[ \Fifo ]

Tipo de dato: switch

Indica si el archivo es un archivo fifo.

[ \RegFile ]

Tipo de dato: switch

Indica si el archivo es un archivo normal, es decir, un archivo binario normal, ISO 8859-1 (Latin-1) o UTF8.

[ \BlockSpec ]

Tipo de dato: switch

Indica si el archivo es un archivo con bloques especiales.

[ \CharSpec ]

Tipo de dato: switch

Indica si el archivo es un archivo con caracteres especiales.

---

#### Ejecución de programas

Esta función devuelve un valor `bool` que especifica si la comprobación es cierta o no.

---

#### Más ejemplos

A continuación aparecen más ejemplos de la función `IsFile`.

##### Ejemplo 1

Este ejemplo implementa una función de recorrido genérico de una estructura de directorios.

```
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
    ENDIF
  ENDIF
```

*Continúa en la página siguiente*

---

```

        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 of HOME:
    searchdir "HOME:", "listfile";
ENDPROC

```

Este programa recorre la estructura de directorios que existe dentro de "HOME:" y con cada archivo encontrado, ejecuta el procedimiento `listfile`. `searchdir` es la parte genérica, que no tiene ninguna información sobre el inicio de la búsqueda ni sobre a qué rutina se debe llamar con cada archivo. Utiliza `IsFile` para comprobar si se ha encontrado un subdirectorio o un archivo y utiliza el mecanismo de enlazamiento en tiempo de ejecución para llamar al procedimiento especificado en `actionproc` con todos los archivos encontrados. La rutina `actionproc` debe ser un procedimiento con un parámetro de tipo `string`.

### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                   | Causa del error                                  |
|--------------------------|--|
| <code>ERR_FILEACC</code> | El archivo no existe y hay un tipo especificado. |

### Limitaciones

No es posible usar esta función con canales serie ni buses de campo.

Si se usa con discos montados de tipo `FTP` o `NFS`, la existencia del archivo o la información del tipo no siempre se actualiza. Para obtener una información correcta, es posible que sea necesaria una orden explícita con la ruta de búsqueda (con la instrucción `Open`) antes de usar `IsFile`.

### Sintaxis

```

Isfile '('
    [ Path ':'= ] < expression (IN) of string>
    [ '\ ' Directory ]
    | [ '\ ' Fifo ]
    | [ '\ ' RegFile ]
    | [ '\ ' BlockSpec ]
    | [ '\ ' CharSpec ] ')'

```

*Continúa en la página siguiente*

## 2 Funciones

---

### 2.101 IsFile - Comprobar el tipo de un archivo

RobotWare - OS

Continuación

Una función con un valor de retorno del tipo de dato `bool`.

---

#### Información relacionada

| Para obtener más información sobre              | Consulte   |
|---|--|
| Directorio                                      | <a href="#">dir</a> - Estructura de directorio de archivos en la página 1659           |
| Apertura de un directorio                       | <a href="#">OpenDir</a> - Abre un directorio en la página 494                          |
| Cierre de un directorio                         | <a href="#">CloseDir</a> - Cierra un directorio en la página 153                       |
| Lectura de un directorio                        | <a href="#">ReadDir</a> - Lee la siguiente entrada de un directorio en la página 1437  |
| Creación de un directorio                       | <a href="#">MakeDir</a> - Crea un nuevo directorio en la página 358                    |
| Eliminación de un directorio                    | <a href="#">RemoveDir</a> - Elimina un directorio en la página 598                     |
| Cambio del nombre de un archivo                 | <a href="#">RenameFile</a> - Cambia el nombre de un archivo en la página 604           |
| Eliminación de un archivo                       | <a href="#">RemoveFile</a> - Elimina un archivo en la página 600                       |
| Copia de un archivo                             | <a href="#">CopyFile</a> - Copia un archivo en la página 169                           |
| Comprobación del tamaño del archivo             | <a href="#">FileSize</a> - Obtiene el tamaño de un archivo en la página 1285           |
| Comprobación del tamaño del sistema de archivos | <a href="#">FSSize</a> - Obtiene el tamaño de un sistema de archivos en la página 1291 |
| Gestión de archivos y dispositivos de E/S       | <a href="#">Application manual</a> - Controller software IRC5                          |

## 2.102 IsLeadThrough - Comprobar el estado del proceso de guiado

### Utilización

`IsLeadThrough` se utiliza para obtener información sobre el estado del proceso de guiado para un robot TCP.

### Ejemplos básicos

Los siguientes ejemplos ilustran la función `IsLeadThrough`.

#### Ejemplo 1

```
VAR bool leadthrough:=FALSE;
leadthrough:=IsLeadThrough();
```

Comprueba si el proceso de guiado está establecido para el robot TCP activo en esta tarea. Por ejemplo, si se ejecuta en la tarea de RAPID `T_ROB_L`, comprueba si el proceso de guiado está establecido para el robot TCP `ROB_L`.

#### Ejemplo 2

```
VAR bool leadthrough:=FALSE;
leadthrough:=IsLeadThrough(\MechUnit:=ROB_R);
```

Comprueba si el proceso de guiado está establecido para el robot TCP `ROB_R`.

#### Ejemplo 3

```
VAR bool leadthrough:=FALSE;
leadthrough:=IsLeadThrough(\MechUnit:=ROB_R \Active);
```

Comprueba si el proceso de guiado está activo para el robot TCP `ROB_R`.

#### Ejemplo 4

```
SetLeadThrough \On \NoStopMove;
TPWrite "Set: "+ValToStr(IsLeadThrough(\MechUnit:=ROB_R \Set));
TPWrite "Active: "+ValToStr(IsLeadThrough(\MechUnit:=ROB_R
\Active));
..
StopMove;
TPWrite "Set: "+ValToStr(IsLeadThrough(\MechUnit:=ROB_R \Set));
TPWrite "Active: "+ValToStr(IsLeadThrough(\MechUnit:=ROB_R
\Active));
```

Comprueba si el proceso de guiado está establecido y activo para el robot TCP `ROB_R`. El proceso de guiado no está activo hasta que se haya ejecutado una instrucción `StopMove` o se haya detenido la ejecución del programa.

La información mostrada será:

Set: TRUE

Active: FALSE

Set: TRUE

Active: TRUE

### Valor de retorno

Tipo de dato: `bool`

*Continúa en la página siguiente*

## 2 Funciones

---

### 2.102 IsLeadThrough - Comprobar el estado del proceso de guiado

YuMi

Continuación

---

#### Argumentos

IsLeadThrough (\MechUnit \Active | \Set)

\MechUnit

**Mechanical unit**

Tipo de dato: mecunit

El nombre del robot TCP.

El argumento \MechUnit es opcional. Si se omite, la comprobación se realizará para la unidad mecánica representada por la variable predefinida RAPID ROB\_ID, que es una referencia al robot con TCP de la tarea de programa actual.

Si se omite \MechUnit e IsLeadThrough se utiliza desde una tarea sin movimiento, la comprobación se realizará para el robot TCP en la tarea de movimiento conectada.

\Active

Tipo de dato: switch

TRUE si se ha activado el proceso de guiado.

FALSE si se ha desactivado el proceso de guiado con SetLeadThrough \Off.

FALSE también puede devolverse si se ha ejecutado un SetLeadThrough \On \NoStopMove. Es la orden StopMove o la parada de la ejecución del programa lo que activa el proceso de guiado.

\Set

Tipo de dato: switch

TRUE si se ha establecido el proceso de guiado.

FALSE si se ha restablecido el proceso de guiado.

Si no se utiliza ninguno de los modificadores, el comportamiento por defecto es \Set.

---

#### Limitaciones

La unidad mecánica debe ser un robot TCP.

La función IsLeadThrough solo puede usarse con los robots YuMi.

---

#### Sintaxis

```
IsLeadThrough '('  
  ['\MecUnit :=' < variable (VAR) of mecunit> ','  
  ['\Active] | ['\Set'])'
```

Una función con un valor de retorno del tipo de dato bool.

---

#### Información relacionada

| Para obtener más información sobre     | Consulte   |
|--|--|
| Activar y desactivar proceso de guiado | <a href="#">SetLeadThrough - Activar y desactivar proceso de guiado en la página 711</a> |



## 2.103 IsMechUnitActive - Indica si una unidad mecánica está activa

### Utilización

IsMechUnitActive (*Is Mechanical Unit Active*) se utiliza para comprobar si una unidad mecánica está activada.

### Ejemplos básicos

El ejemplo siguiente ilustra la función IsMechUnitActive.

#### Ejemplo 1

```
IF IsMechUnitActive(SpotWeldGun)
  CloseGun SpotWeldGun;
```

Si la unidad mecánica SpotWeldGun está activada, se llama a la rutina CloseGun, que se utiliza para cerrar la pistola.

### Valor de retorno

Tipo de dato: bool

La función devuelve lo siguiente:

- TRUE, si la unidad mecánica está activada
- FALSE, si la unidad mecánica está desactivada

### Argumentos

```
IsMechUnitActive ( MechUnit )
```

MechUnit

*Mechanical Unit*

Tipo de dato: mecunit

El nombre de la unidad mecánica.

### Sintaxis

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

Una función con un valor de retorno del tipo de dato bool.

### Información relacionada

| Para obtener más información sobre  | Consulte   |
|-------------------------------------|--|
| Activación de unidades mecánicas    | <a href="#">ActUnit - Activa una unidad mecánica en la página 30</a>       |
| Desactivación de unidades mecánicas | <a href="#">DeactUnit - Desactiva una unidad mecánica en la página 187</a> |
| Unidades mecánicas                  | <a href="#">mecunit - Unidad mecánica en la página 1703</a>                |

## 2 Funciones

---

### 2.104 IsPers - Determina si es una variable persistente

RobotWare - OS

### 2.104 IsPers - Determina si es una variable persistente

---

#### Utilización

IsPers se utiliza para comprobar si un objeto de datos es una variable persistente o no.

---

#### Ejemplos básicos

El ejemplo siguiente ilustra la función IsPers.

#### Ejemplo 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
```

El procedimiento procedure1 toma caminos diferentes en función de si el parámetro recibido parameter1 es una variable o una variable persistente.

---

#### Valor de retorno

Tipo de dato: bool

TRUE si el parámetro INOUT comprobado es una variable persistente. FALSE si el parámetro INOUT comprobado no es una variable persistente.

---

#### Argumentos

IsPers (DatObj)

DatObj

Tipo de dato: string

El número de serie del sistema de robot que indica la identidad del sistema.

*Data Object*

Tipo de dato: cualquier tipo

El nombre formal del parámetro INOUT.

---

#### Sintaxis

```
IsPers '('
  [ DatObj ':= ' ] < var or pers (INOUT) of any type > ')'
```

Una función con un valor de retorno del tipo de dato bool.

---

*Continúa en la página siguiente*

#### Información relacionada

| Para obtener más información sobre         | Consulte  |
|--|---|
| Comprobación de si un dato es una variable | <a href="#">IsVar - Determina si un dato es una variable en la página 1367</a>                          |
| Tipos de parámetros (modos de acceso)      | Manual de referencia técnica - <i>RAPID Overview</i> , sección <i>Características básicas - Rutinas</i> |

## 2 Funciones

---

### 2.105 IsStopMoveAct - Está activo el indicador de movimiento de paro *RobotWare - OS*

### 2.105 IsStopMoveAct - Está activo el indicador de movimiento de paro

---

#### Utilización

`IsStopMoveAct` se usa para obtener el estado de los indicadores de movimiento de paro de la tarea de movimiento actual o conectada.

---

#### Ejemplos básicos

Los siguientes ejemplos ilustran la función `IsStopMoveAct`.

##### Ejemplo 1

```
stopflag2:= IsStopMoveAct(\FromNonMoveTask);
```

`stopflag2` será **TRUE** si el indicador de movimiento de paro de las tareas sin movimiento está activado en la tarea de movimiento actual o conectada. De lo contrario, será **FALSE**.

##### Ejemplo 2

```
IF IsStopMoveAct(\FromMoveTask) THEN  
  StartMove;  
ENDIF
```

Si el indicador de movimiento de paro de la tarea de movimiento está activado en la tarea de movimiento actual, será restablecido por la instrucción `StartMove`.

---

#### Valor de retorno

Tipo de dato: `bool`

El valor de retorno será **TRUE** si el indicador de movimiento de paro seleccionado está activado. De lo contrario, el valor de retorno será **FALSE**.

---

#### Argumentos

```
IsStopMoveAct ( [\FromMoveTask] | [\FromNonMoveTask] )
```

`[\FromMoveTask]`

Tipo de dato: `switch`

`FromMoveTask` se usa para obtener el estado del indicador de movimiento del tipo de una tarea de movimiento privada.

Este tipo de indicador de movimiento de paro sólo puede ser activado por:

- La propia tarea de movimiento con la instrucción `StopMove`
- Tras la salida del nivel `RestoPath` en el programa.
- En la ejecución de un gestor de errores asíncrono para errores de proceso o movimiento antes de cualquier `StorePath` y tras cualquier `RestoPath`

`[\FromNonMoveTask]`

Tipo de dato: `switch`

`FromNonMoveTask` se usa para obtener el estado del indicador de movimiento del tipo de cualquier tarea sin movimiento. Este tipo de indicador de movimiento de paro sólo puede ser activado por cualquier tarea sin movimiento de las tareas de movimiento conectadas o de todas ellas, con la instrucción `StopMove`.

---

*Continúa en la página siguiente*

#### Sintaxis

```
IsStopMoveAct '('  
  ['\' FromMoveTask]  
  | ['\' FromNonMoveTask] ')'
```

Una función con un valor de retorno del tipo de dato `bool`.

#### Información relacionada

| Para obtener más información sobre   | Consulte   |
|--------------------------------------|--|
| Detención del movimiento del robot   | <a href="#">StopMove - Detiene el movimiento del robot en la página 828</a>  |
| Reanudación del movimiento del robot | <a href="#">StartMove - Reanuda el movimiento del robot en la página 796</a> |

## 2 Funciones

---

### 2.106 IsStopStateEvent - Comprueba si se ha movido el puntero de programa RobotWare - OS

### 2.106 IsStopStateEvent - Comprueba si se ha movido el puntero de programa

---

#### Utilización

IsStopStateEvent devuelve información acerca del movimiento del puntero de programa (PP) de la tarea de programa actual.

---

#### Ejemplos básicos

El ejemplo siguiente ilustra la función IsStopStateEvent.

#### Ejemplo 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
```

---

#### Valor de retorno

Tipo de dato: bool

El estado de si el PP se ha movido durante el último estado de paro de programa y cómo se ha movido.

TRUE Si el PP se ha movido durante el último paro.

FALSE Si el PP no se ha movido durante el último paro.

Si el PP se ha movido a la rutina Main, tanto \PPMoved como \PPToMain devuelven TRUE.

Si el PP se ha movido a otra rutina, tanto \PPMoved como \PPToMain devuelven TRUE.

Si el PP se ha movido dentro de una lista de una rutina, \PPMoved devuelve TRUE y \PPToMain devuelve FALSE.

Tras llamar a una rutina de servicio (manteniendo el contexto de ejecución en la secuencia de programa principal), \PPMove devuelve FALSE y \PPToMain devuelve FALSE.

---

#### Argumentos

```
IsStopStateEvent ([\PPMoved] | [\PPToMain])
```

[ \PPMoved ]

Tipo de dato: switch

Comprueba si el PP se ha movido.

[ \PPToMain ]

Tipo de dato: switch

Comprueba si el PP se ha movido a Main o a una rutina.

---

*Continúa en la página siguiente*

#### Limitaciones

En la mayoría de los casos, esta función no puede usarse con la ejecución hacia atrás o hacia delante, porque el sistema se encuentra en el estado de paro entre un paso y el siguiente.

#### Sintaxis

```
IsStopStateEvent '('  
    ['\' PPMoved] | ['\' PPToMain] ')'
```

Una función con un valor de retorno del tipo de dato `bool`.

#### Información relacionada

| Para obtener más información sobre | Consulte  |
|------------------------------------|---|
| Creación de instrucciones propias  | <i>Manual de referencia técnica - RAPID Overview</i>            |
| <i>Advanced RAPID</i>              | <i>Especificaciones del producto - Controller software IRC5</i> |

## 2 Funciones

---

2.107 IsSyncMoveOn - Comprueba si el modo de movimiento sincronizado está activado  
RobotWare - OS

### 2.107 IsSyncMoveOn - Comprueba si el modo de movimiento sincronizado está activado

---

#### Utilización

IsSyncMoveOn se utiliza para comprobar si la tarea de programa actual de tipo Motion Task tiene activado el modo de movimiento sincronizado.

Desde algunas Non Motion Task también es posible comprobar si la Motion Task conectada se encuentra en el modo de movimiento sincronizado. El parámetro del sistema *Controller/Tasks/Use Mechanical Unit Group* define la Motion Task conectada.

Cuando la Motion Task se está ejecutando en el nivel StorePathIsSyncMoveOn comprueba si la tarea está en el modo sincronizado en ese nivel, independientemente del modo sincronizado en el nivel original.

La instrucción IsSyncMoveOn suele utilizarse en un sistema *MultiMove* con la opción *Coordinated Robots*, pero puede usarse en cualquier sistema y con cualquier tarea de programa.

---

#### Ejemplos básicos

El ejemplo siguiente ilustra la función IsSyncMoveOn.

##### Ejemplo 1

##### Ejemplo de programa de la tarea 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;
MoveL * \ID:=10, v100, z10, tcp1 \WOBJ:= rob2_obj;
MoveL * \ID:=20, v100, fine, tcp1 \WOBJ:= rob2_obj;
SyncMoveOff sync3;
UNDO
SyncMoveUndo;
ENDPROC
```

##### Ejemplo de programa de la tarea BCK1

```
PROC main()
...
```

*Continúa en la página siguiente*



## 2.107 IsSyncMoveOn - Comprueba si el modo de movimiento sincronizado está activado

RobotWare - OS

Continuación

```

IF IsSyncMoveOn() THEN
    ! Connected Motion Task is in synchronized movement mode
ELSE
    ! Connected Motion Task is in independent mode
ENDIF
...
ENDPROC

```

En el momento de la ejecución de `IsSyncMoveOn` en la tarea de segundo plano `BCK1`, se comprueba si la tarea de movimiento conectada en ese momento se encuentra en el modo de movimiento sincronizado.

## Valor de retorno

Tipo de dato: `bool`

`TRUE` si la tarea de programa conectada se encuentra en el modo de movimiento sincronizado o, de lo contrario, `FALSE`.

## Ejecución de programas

Se comprueba si la tarea de programa actual o conectada se encuentra en el modo de movimiento sincronizado. Cuando la `MotionTask` se está ejecutando en el `StorePath level`, `SyncMoveOn` comprueba si la tarea está en el modo sincronizado en el `StorePath level`, no en el nivel original.

## Sintaxis

```
IsSyncMoveOn '(' ' ')
```

Una función con un valor de retorno del tipo de dato `bool`.

## Información relacionada

| Para obtener más información sobre                          | Consulte  |
|---|---|
| Especificación de tareas de programa cooperativas           | <a href="#">tasks - Tareas de programa RAPID en la página 1791</a>                                      |
| Identidad para punto de sincronización                      | <a href="#">syncident - Identidad de punto de sincronización en la página 1786</a>                      |
| Inicio de movimientos sincronizados coordinados             | <a href="#">SyncMoveOn - Inicia los movimientos sincronizados coordinados en la página 854</a>          |
| Fin de movimientos sincronizados coordinados                | <a href="#">SyncMoveOff - Finaliza los movimientos sincronizados coordinados en la página 848</a>       |
| Definición de movimientos independientes                    | <a href="#">SyncMoveUndo - Activa los movimientos independientes en la página 864</a>                   |
| Almacenamiento de trayectoria y ejecución en un nuevo nivel | <a href="#">StorePath - Almacena la trayectoria cuando se produce una interrupción en la página 835</a> |

## 2 Funciones

---

### 2.108 IsSysID - Comprobar la identidad del sistema RobotWare - OS

### 2.108 IsSysID - Comprobar la identidad del sistema

---

#### Utilización

IsSysId (*System Identity*) puede usarse para comprobar la identidad del sistema, mediante le número de serie del sistema.

---

#### Ejemplos básicos

El ejemplo siguiente ilustra la función IsSysId.

#### Ejemplo 1

```
IF NOT IsSysId("6400-1234") THEN
  ErrWrite "System identity fault", "Faulty system identity for
  this program";
  EXIT;
ENDIF
```

Este programa se ha diseñado para un sistema de robot especial con el número de serie 6400-1234 y no puede utilizarse con otro sistema de robot.

---

#### Valor de retorno

Tipo de dato: bool

TRUE = El número de serie del sistema de robot es el mismo que el especificado en la comprobación.

FALSE = El número de serie del sistema de robot no es el mismo que el especificado en la comprobación.

---

#### Argumentos

IsSysId ( SystemId)

SystemId

Tipo de dato: string

El número de serie del sistema de robot que indica la identidad del sistema.

---

#### Sintaxis

```
IsSysId '('
  [ SystemId ':= ' ] < expression (IN) of string> ')'
```

Una función con un valor de retorno del tipo de dato bool.

---

#### Información relacionada

| Para obtener más información sobre | Consulte  |
|------------------------------------|---|
| Lectura de información del sistema | <a href="#">GetSysInfo - Obtener información acerca del sistema en la página 1316</a> |

## 2.109 IsVar - Determina si un dato es una variable

### Utilización

IsVar se utiliza para comprobar si un objeto de datos es una variable.

### Ejemplos básicos

El ejemplo siguiente ilustra la función IsVar.

#### Ejemplo 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

```

El procedimiento `procedure1` toma caminos diferentes en función de si el parámetro recibido `parameter1` es una variable o una variable persistente.

### Valor de retorno

Tipo de dato: `bool`

TRUE si el parámetro INOUT comprobado es una variable. FALSE si el parámetro INOUT comprobado no es una variable.

### Argumentos

IsVar (DatObj)

DatObj

*Data Object*

Tipo de dato: cualquier tipo

El nombre formal del parámetro INOUT.

### Sintaxis

```

IsVar '('
  [ DatObj ':= ' ] < var or pers (INOUT) of any type > ')'

```

Una función con un valor de retorno del tipo de dato `bool`.

### Información relacionada

| Para obtener más información sobre                     | Consulte  |
|--|---|
| Comprobación de si un dato es una variable persistente | <a href="#">IsPers - Determina si es una variable persistente en la página 1358</a> |

*Continúa en la página siguiente*

## 2 Funciones

---

### 2.109 IsVar - Determina si un dato es una variable

RobotWare - OS

Continuación

| Para obtener más información sobre    | Consulte  |
|---------------------------------------|---|
| Tipos de parámetros (modos de acceso) | <i>Manual de referencia técnica - RAPID Overview, sección Características básicas - Rutinas</i> |

## 2.110 Max - Obtener el mayor de dos valores

### Utilización

Max devuelve el mayor de dos argumentos.

### Ejemplos básicos

El ejemplo siguiente ilustra la función Max.

#### Ejemplo 1

```
reg1 := Max(reg2, reg3)
```

A reg1 se asigna el valor mayor de reg2 y reg3.

### Valor de retorno

Tipo de dato: num

Devuelve el mayor de los dos argumentos.

### Argumentos

Max (A, B)

A

Tipo de dato: num

Primer valor numérico.

B

Tipo de dato: num

Segundo valor numérico.

### Sintaxis

```
Max '('  
  [A ':=' ] < expression (IN) of num > ','  
  [B ':=' ] < expression (IN) of num > ')'
```

Una función con un valor de retorno del tipo de dato num.

### Información relacionada

| Para obtener más información sobre    | Consulte  |
|---------------------------------------|---|
| Función Min                           | <a href="#">Min - Obtener el menor de dos valores en la página 1374</a> |
| Instrucciones y funciones matemáticas | <i>Manual de referencia técnica - RAPID Overview</i>                    |

## 2 Funciones

---

### 2.111 MaxExtLinearSpeed - Velocidad máxima de eje adicional

RobotWare - OS

### 2.111 MaxExtLinearSpeed - Velocidad máxima de eje adicional

---

#### Utilización

`MaxExtLinearSpeed` (Velocidad lineal máxima de eje adicional) devuelve la velocidad lineal máxima para los ejes adicionales en la tarea de movimiento actual.

---

#### Ejemplos básicos

El siguiente ejemplo muestra la función `MaxExtLinearSpeed`.

#### Ejemplo 1

```
TPWrite "Max. Linear speed in mm/s for my axis="\Num:=  
MaxExtLinearSpeed ();
```

El mensaje `Max. Linear speed in mm/s for my axis = 5000` se escribe en el FlexPendant (el valor depende de la configuración).

---

#### Valor de retorno

Tipo de dato: `num`

Devuelve la velocidad lineal máxima ( $v_{max}$ ) en mm/s para los ejes adicionales en esta tarea.

---

#### Sintaxis

```
MaxExtLinearSpeed '(' '')
```

Una función con un valor de retorno del tipo de dato `num`.

---

#### Información relacionada

| Para obtener más información sobre                         | Consulte   |
|--|--|
| Definición de velocidad $v_{max}$                          | <a href="#">speeddata - Datos de velocidad en la página 1767</a>   |
| Configuración de la velocidad lineal máx. de eje adicional | <i>Manual de referencia técnica - Parámetros del sistema, parámetro Ext. Axis Linear Max Speed (m/s)</i> |

## 2.112 MaxExtReorientSpeed - Velocidad de giro máxima de eje adicional

### Utilización

MaxExtReorientSpeed (Velocidad de giro máxima de eje adicional) devuelve la velocidad de giro máxima para los ejes adicionales en la tarea de movimiento actual.

### Ejemplos básicos

El ejemplo siguiente ilustra la función MaxExtReorientSpeed.

#### Ejemplo 1

```
TPWrite "Max. Rotational speed in deg/s for my axis="\Num:=
MaxExtReorientSpeed ();
```

El mensaje Max. Rotational speed in deg/s for my axis = 1000 se escribe en el FlexPendant (el valor depende de la configuración).

### Valor de retorno

Tipo de dato: num

Devuelve la velocidad de giro máxima ( $v_{max}$ ) en grados/s para los ejes adicionales en esta tarea.

### Sintaxis

```
MaxExtReorientSpeed '( ' )'
```

Una función con un valor de retorno del tipo de dato num.

### Información relacionada

| Para obtener más información sobre                          | Consulte  |
|---|---|
| Definición de velocidad $v_{max}$                           | <a href="#">speeddata - Datos de velocidad en la página 1767</a>  |
| Configuración de la velocidad de giro máx. de eje adicional | Manual de referencia técnica - Parámetros del sistema, parámetro Ext. Axis Rotational Max Speed (deg/s) |

## 2 Funciones

---

### 2.113 MaxRobReorientSpeed - Velocidad de reorientación máxima del robot

RobotWare - OS

### 2.113 MaxRobReorientSpeed - Velocidad de reorientación máxima del robot

---

#### Utilización

MaxRobReorientSpeed (Velocidad de reorientación máxima del robot) devuelve la velocidad de reorientación máxima de TCP para el robot.

---

#### Ejemplos básicos

El ejemplo siguiente ilustra la función MaxRobReorientSpeed.

#### Ejemplo 1

```
TPWrite "TCP Reorient Max Speed in deg/s for my robot="\Num:=  
MaxRobReorientSpeed ();
```

El mensaje TCP Reorient Max Speed in deg/s for my robot = 500 se escribe en el FlexPendant (el valor depende de la configuración).

---

#### Valor de retorno

Tipo de dato: num

Devuelve la velocidad de reorientación máxima del TCP ( $v_{max}$ ) en grados/s para el robot utilizado y valores normales prácticos para el TCP.

Si se utilizan valores de TCP extremadamente altos en la base de coordenadas de la herramienta, puede crear su propio speeddata con velocidad de reorientación de TCP menor que la devuelta por MaxRobReorientSpeed.

---

#### Sintaxis

```
MaxRobReorientSpeed '(' '')
```

Una función con un valor de retorno del tipo de dato num.

---

#### Información relacionada

| Para obtener más información sobre                         | Consulte  |
|--|---|
| Definición de velocidad $v_{max}$                          | <a href="#">speeddata - Datos de velocidad en la página 1767</a>                                |
| Configuración de la velocidad de reorientación máx. de TCP | Manual de referencia técnica - Parámetros del sistema, parámetro TCP Reorient Max Speed (deg/s) |



## 2.114 MaxRobSpeed - Velocidad máxima del robot

### Utilización

MaxRobSpeed (*Maximum Robot Speed*) devuelve la velocidad máxima de TCP para el robot.

### Ejemplos básicos

El ejemplo siguiente ilustra la función MaxRobSpeed.

#### Ejemplo 1

```
TPWrite "Max. TCP speed in mm/s for my robot="\Num:=MaxRobSpeed();
```

El mensaje Max. TCP speed in mm/s for my robot = 7000 se escribe en el FlexPendant (el valor depende de la configuración).

### Valor de retorno

Tipo de dato: num

Devuelve la velocidad máxima del TCP ( $v_{max}$ ) en mm/s para el robot utilizado y valores normales prácticos para el TCP.

Si se utilizan valores de TCP extremadamente elevados en la base de coordenadas de la herramienta, puede crear su propio speeddata con una velocidad mayor de TCP que la devuelta por MaxRobSpeed y usar VelSet para permitir una velocidad mayor.

### Sintaxis

```
MaxRobSpeed '(' ' ')
```

Una función con un valor de retorno del tipo de dato num.

### Información relacionada

| Para obtener más información sobre        | Consulte   |
|---|--|
| Definición de velocidad $v_{max}$         | <a href="#">speeddata - Datos de velocidad en la página 1767</a>                                   |
| Definición de la velocidad máxima         | <a href="#">VelSet - Cambia la velocidad programada en la página 1029</a>                          |
| Configuración de la velocidad máx. de TCP | <i>Manual de referencia técnica - Parámetros del sistema, parámetro TCP Linear Max Speed (m/s)</i> |

## 2 Funciones

---

### 2.115 Min - Obtener el menor de dos valores

RobotWare - OS

### 2.115 Min - Obtener el menor de dos valores

---

#### Utilización

Min devuelve el menor de dos argumentos.

---

#### Ejemplos básicos

El ejemplo siguiente ilustra la función Min.

##### Ejemplo 1

```
reg1 := Min(reg2, reg3)
```

A reg1 se asigna el valor menor de reg2 y reg3.

---

#### Valor de retorno

Tipo de dato: num

Devuelve el menor de los dos argumentos.

---

#### Argumentos

Min (A, B)

A

Tipo de dato: num

Primer valor numérico.

B

Tipo de dato: num

Segundo valor numérico.

---

#### Sintaxis

```
Min '('  
  [A ':=' ] < expression (IN) of num > ','  
  [B ':=' ] < expression (IN) of num > ')'
```

Una función con un valor de retorno del tipo de dato num.

---

#### Información relacionada

| Para obtener más información sobre    | Consulte  |
|---------------------------------------|---|
| Función Max                           | <a href="#">Max - Obtener el mayor de dos valores en la página 1369</a> |
| Instrucciones y funciones matemáticas | <i>Manual de referencia técnica - RAPID Overview</i>                    |

## 2.116 MirPos - Obtención de la posición espejo de una posición

### Utilización

`MirPos` (*Mirror Position*) se utiliza para obtener los valores espejo de las partes de traslación y rotación de una posición.

### Ejemplos básicos

El ejemplo siguiente ilustra la función `MirPos`.

#### Ejemplo 1

```
CONST robtarget p1:= [...];
VAR robtarget p2;
PERS wobjdata mirror:= [...];
...
p2 := MirPos(p1, mirror);
```

`p1` es un valor de tipo `robtarget` que contiene una posición del robot y una orientación de la herramienta. La información espejo se obtiene en el plano xy de la base de coordenadas definido por `mirror`, respecto del sistema de coordenadas mundo. El resultado es un nuevo dato de tipo `robtarget`, que se almacena en `p2`.

### Valor de retorno

Tipo de dato: `robtarget`

La nueva posición, que es la posición espejo de la posición de entrada.

### Argumentos

```
MirPos (Point MirPlane [\Wobj] [\MirY])
```

Point

Tipo de dato: `robtarget`

La posición de entrada del robot. La parte de orientación de esta posición define la orientación actual del sistema de coordenadas de la herramienta.

MirPlane

*Mirror Plane*

Tipo de dato: `wobjdata`

Los datos del objeto de trabajo que definen el plano espejo. El plano espejo es el plano xy de la base de coordenadas del objeto definida en `MirPlane`. La ubicación de la base de coordenadas del objeto se define respecto de la base de coordenadas del usuario (también definida en `MirPlane`) que a su vez se define respecto de la base de coordenadas mundo.

[\Wobj]

*Work Object*

Tipo de dato: `wobjdata`

Se definen los datos del objeto de trabajo que definen la base de coordenadas del objeto y la base de coordenadas del usuario respecto de la posición de entrada,

*Continúa en la página siguiente*

## 2 Funciones

### 2.116 MirPos - Obtención de la posición espejo de una posición

RobotWare - OS

Continuación

*Point*. Si no se utiliza el argumento, la posición se define respecto del sistema de coordenadas mundo.

#### ¡IMPORTANTE!

Si la posición se crea con un objeto de trabajo activo, es necesario hacer referencia al objeto en el argumento.

[ \MirY ]

#### Mirror Y

Tipo de dato: switch

Si se omite este modificador, que es el comportamiento predeterminado, se obtiene una imagen espejo de la base de coordenadas de la herramienta en cuanto al eje x y al eje z. Si se utiliza este modificador, se obtiene una imagen espejo de la base de coordenadas de la herramienta en cuanto al eje y, y al eje z.

#### Limitaciones

No se realiza ningún recálculo de la parte de configuración del robot que se incluye en los datos de entrada *robtarget*.

Si se utiliza una base de coordenadas, la unidad coordinada debe estar dentro de la misma tarea que el robot.

#### Sintaxis

```
MirPos '('  
  [ Point ':= ' ] < expression (IN) of robtarget> ','  
  [ MirPlane ':= ' ] <expression (IN) of wobjdata> ','  
  [ '\ ' WObj ':= ' <expression (IN) of wobjdata> ]  
  [ '\ ' MirY ] ')'
```

Una función con un valor de retorno del tipo de dato *robtarget*.

#### Información relacionada

| Para obtener más información sobre    | Consulte  |
|---------------------------------------|---|
| Instrucciones y funciones matemáticas | <i>Manual de referencia técnica - RAPID Overview</i> , sección <i>Resumen sobre RAPID - Matemáticas</i> |
| Datos de posición                     | <a href="#">robtarget - Datos de posición en la página 1750</a>   |
| Datos del objeto de trabajo           | <a href="#">wobjdata - Datos del objeto de trabajo en la página 1823</a>                                |

## 2.117 MOD - Evalúa un módulo de entero

### Utilización

MOD es una expresión condicional utilizada para evaluar el módulo, el resto, de una división de enteros.

### Ejemplos básicos

Los siguientes ejemplos ilustran la función MOD.

#### Ejemplo 1

```
reg1 := 14 MOD 4;
```

El valor de retorno es 2 porque 14 dividido entre 4 da como resultado el módulo 2.

#### Ejemplo 2

```
VAR dnum mydnum1 := 11;
VAR dnum mydnum2 := 5;
VAR dnum mydnum3;
...
mydnum3 := mydnum1 MOD mydnum2;
```

El valor de retorno es 1 porque 11 dividido entre 5 da como resultado el módulo 2.

### Valor de retorno

Tipo de dato: num, dnum

Devuelve el módulo, el resto, de una división de enteros.

### Sintaxis

```
<expression of num> MOD <expression of num>
```

Una función con un valor de retorno del tipo de dato num.

```
<expression of dnum> MOD <expression of dnum>
```

Una función con un valor de retorno del tipo de dato dnum.

### Información relacionada

| Para obtener más información sobre | Consulte   |
|------------------------------------|--|
| num - Valores numéricos            | <a href="#">num - Valores numéricos en la página 1712</a>          |
| dnum - Valores numéricos dobles    | <a href="#">dnum - Valores numéricos dobles en la página 1660</a>  |
| DIV                                | <a href="#">DIV - Evalúa una división entera en la página 1269</a> |
| Expresiones                        | <a href="#">Manual de referencia técnica - RAPID Overview</a>      |

## 2 Funciones

---

### 2.118 ModExist - Comprobar si un módulo de programa existe

RobotWare - OS

### 2.118 ModExist - Comprobar si un módulo de programa existe

---

#### Utilización

`ModExist` (*Module Exist*) se utiliza para comprobar si un módulo determinado existe o no en la tarea de programa.

La búsqueda se realiza en primer lugar en los módulos cargados y, a continuación, si no se encuentra ninguno, en los módulos instalados.

---

#### Ejemplos básicos

El ejemplo siguiente ilustra la función `ModExist`.

##### Ejemplo 1

```
VAR bool mod_exist;  
mod_exist:=ModExist ("MyModule");
```

Si el módulo `MyModule` existe en la tarea, la función devuelve `TRUE`. Si no es así, la función devuelve `FALSE`.

---

#### Valor de retorno

Tipo de dato: `bool`

`TRUE` si el módulo se encuentra, y `FALSE` si no se encuentra.

---

#### Argumentos

```
ModExist (ModuleName)
```

ModuleName

Tipo de dato: `string`

El nombre del módulo a buscar.

---

#### Sintaxis

```
ModExist '('  
  [ ModuleName ':=' ] < expression (IN) of string > ')'
```

Una función con un valor de retorno del tipo de dato `bool`.

---

#### Información relacionada

| Para obtener más información sobre                     | Consulte   |
|--|--|
| Búsqueda de la hora de modificación del módulo cargado | <a href="#">ModTimeDnum - Obtener la hora de modificación del módulo cargado en la página 1379</a> |

---

## 2.119 ModTimeDnum - Obtener la hora de modificación del módulo cargado

### Utilización

`ModTimeDnum` (*Modify Time*) se usa para obtener la hora más reciente de modificación del archivo del módulo cargado. El módulo se especifica con su nombre y debe encontrarse en la memoria de tareas. La hora se indica en segundos a partir de las 00:00:00 horas, hora de Greenwich, del 1 de enero de 1970. La hora se devuelve como un valor de tipo `dnum` y opcionalmente también en un valor de tipo `stringdig`.

### Ejemplos básicos

El ejemplo siguiente ilustra la función `ModTimeDnum`.  
 Consulte también [Más ejemplos en la página 1380](#).

#### Ejemplo 1

```
MODULE mymod
  VAR dnum mytime;
  PROC printMyTime()
    mytime := ModTimeDnum("mymod");
    TPWrite "My time is "+ValToStr(mytime);
  ENDPROC
ENDMODULE
```

### Valor de retorno

Tipo de dato: `dnum`

El tiempo, medido en segundos, desde las 00:00:00 horas, horario de Greenwich, del 1 de enero de 1970.

### Argumentos

```
ModTimeDnum ( Object [\StrDig] )
```

Object

Tipo de dato: `string`  
 El nombre del módulo.

[\StrDig]

*String Digit*

Tipo de dato: `stringdig`

Para obtener la hora de carga del módulo en una representación de tipo `stringdig`.

### Ejecución de programas

Esta función devuelve un valor numérico que especifica la hora de última modificación del archivo, antes de que fuera cargado como un módulo de programa en el sistema.

*Continúa en la página siguiente*

## 2 Funciones

### 2.119 ModTimeDnum - Obtener la hora de modificación del módulo cargado

#### Continuación

#### Más ejemplos

A continuación aparecen más ejemplos de la función `ModTimeDnum`.

#### Ejemplo 1

```
IF FileTimeDnum ("HOME:/mymod.mod" \ModifyTime) > ModTimeDnum
  ("mymod") THEN
  UnLoad "HOME:/mymod.mod";
  Load \Dynamic, "HOME:/mymod.mod";
ENDIF
```

Este programa recarga un módulo si el archivo de origen es más reciente. Utiliza `ModTimeDnum` para obtener la hora de la última modificación del módulo especificado y para compararlo con los valores de `FileTimeDnum` ("HOME:/mymod.mod" \ModifyTime) del origen. A continuación, si el origen es más reciente, el programa descarga y carga de nuevo el módulo.

#### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                          | Causa del error  |
|---------------------------------|--|
| <code>ERR_MOD_NOT_LOADED</code> | Ningún módulo con el nombre especificado se encuentra en la tarea de programa. |

#### Limitaciones

Esta función devuelve siempre 0 si se usa en un módulo que está codificado o instalado de forma compartida.

#### Sintaxis

```
ModTimeDnum '('
  [Object ':='] <expression (IN) of string>
  ['\' StrDig ':='] <variable (VAR) of stringdig>']'
```

Una función con un valor de retorno del tipo de dato `dnum`.

#### Información relacionada

| Para obtener más información sobre                  | Consulte  |
|---|---|
| Obtención de información de tiempo sobre un archivo | <a href="#">FileTimeDnum - Obtener información de tiempo sobre un archivo en la página 1288</a> |
| Cadena de caracteres con sólo dígitos               | <a href="#">stringdig - Cadena de caracteres con sólo dos dígitos en la página 1780</a>         |
| Comparar dos cadenas que sólo contienen dígitos     | <a href="#">StrDigCmp - Comparar dos cadenas que sólo contienen dígitos en la página 1494</a>   |



## 2.120 MotionPlannerNo - Obtiene el número de planificador de movimientos conectado

### Utilización

MotionPlannerNo devuelve el número de planificador de movimientos conectado. Si ejecuta MotionPlannerNo en una tarea de movimiento, éste devuelve su número de planificador. En caso contrario, si ejecuta MotionPlannerNo en una tarea sin movimiento se devuelve el número de planificador de movimientos conectado acorde con la configuración de los parámetros del sistema.

### Ejemplos básicos

El ejemplo siguiente ilustra la función MotionPlannerNo.

#### Ejemplo 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
```

*Continúa en la página siguiente*

## 2 Funciones

---

### 2.120 MotionPlannerNo - Obtiene el número de planificador de movimientos conectado

RobotWare - OS

Continuación

```
!Background task BCK2
PERS string buffer{6};
VAR num motion_planner;
VAR string status;

PROC main()
...
motion_planner := MotionPlannerNo();
status := buffer{motion_planner};
...
ENDPROC
```

Utilice la función `MotionPlannerNo` para determinar qué número de planificador de movimientos está conectado a la tarea. Puede implementarse exactamente el mismo código en todas las tareas de movimiento y las tareas en segundo plano. Después todas las tareas en segundo plano pueden comprobar el estado de su tarea de movimiento conectada.

---

#### Valor de retorno

Tipo de dato: num

El nombre del planificador de movimientos conectado. En las tareas sin movimiento, se devuelve el número de planificador de movimientos de la unidad mecánica asociada.

El rango de valor de retorno es de 1 a 6.

---

#### Sintaxis

```
MotionPlannerNo '(' '')
```

Una función con un valor de retorno del tipo de dato `num`.

---

#### Información relacionada

| Para obtener más información sobre                | Consulte  |
|---|---|
| Especificación de tareas de programa cooperativas | <i>Manual de referencia técnica - Parámetros del sistema, sección Controller - Task</i> |

## 2.121 NonMotionMode - Lee el modo de ejecución sin movimiento

### Utilización

`NonMotionMode` (*Non-Motion Execution Mode*) se usa para leer el modo de ejecución sin movimiento actual de la tarea del programa. El modo de ejecución sin movimiento se selecciona o deselecciona desde el FlexPendant, dentro de la opción de menú *ABB\Panel de control\Supervisión*.

### Ejemplos básicos

El ejemplo siguiente ilustra la función `NonMotionMode`.

#### Ejemplo 1

```
IF NonMotionMode() =TRUE THEN
  ...
ENDIF
```

La sección de programa sólo se ejecuta si el programa no se encuentra en el modo de ejecución sin movimiento.

### Valor de retorno

Tipo de dato: `bool`

El modo sin movimiento actual, con uno de los valores definidos en la tabla siguiente.

| Valor de retorno | Constante simbólica | Comentario                            |
|------------------|---------------------|---------------------------------------|
| 0                | FALSE               | No se usa la ejecución sin movimiento |
| 1                | TRUE                | Se usa la ejecución sin movimiento    |

### Argumentos

```
NonMotionMode ( [ \Main ] )
```

[ \Main ]

Tipo de dato: `switch`

Devuelve el modo de ejecución actual para la tarea de movimiento conectada. Se utiliza en un sistema multitarea para obtener el modo de ejecución actual para la tarea real, si se trata de una tarea de movimiento o una tarea de movimiento conectada, si la función `NonMotionMode` se ejecuta en una tarea sin movimiento.

Si se omite el argumento, el valor de retorno siempre refleja el modo de ejecución opuesto de la tarea de programa que ejecuta la función `NonMotionMode`.

Recuerde que el modo de ejecución está conectado al sistema y no a ninguna tarea. Esto quiere decir que todas las tareas de un sistema obtienen el mismo valor de retorno de `NonMotionMode`.

### Sintaxis

```
NonMotionMode '(' ['\ Main'] ')'
```

Una función con un valor de retorno del tipo de dato `bool`.

*Continúa en la página siguiente*

## 2 Funciones

---

2.121 NonMotionMode - Lee el modo de ejecución sin movimiento

RobotWare - OS

Continuación

---

### Información relacionada

| Para obtener más información sobre | Consulte   |
|------------------------------------|--|
| Lectura del modo de funcionamiento | <a href="#">OpMode - Lee el modo de funcionamiento en la página 1393</a> |

## 2.122 NOT - Invierte un valor lógico

### Utilización

NOT es una expresión condicional utilizada para invertir un valor lógico (verdadero/falso).

### Ejemplos básicos

Los ejemplos siguientes ilustran la expresión condicional NOT.

#### Ejemplo 1

```
VAR bool mybool;  
mybool := NOT mybool;
```

Si `mybool` tiene el valor `TRUE`, el valor de retorno es `FALSE`.

Si `mybool` tiene el valor `FALSE`, el valor de retorno es `TRUE`.

#### Ejemplo 2

```
VAR bool a;  
VAR bool b;  
VAR bool c;  
...  
c := a AND (NOT b);
```

El valor de retorno `c` es `TRUE` si `a` tiene el valor `TRUE` y `b` tiene el valor `FALSE`

### Valor de retorno

Tipo de dato: `bool`

Devuelve el valor invertido.

### Sintaxis

```
NOT <logical term>
```

### Información relacionada

| Para obtener más información sobre | Consulte   |
|------------------------------------|--|
| AND                                | <a href="#">AND - Evalúa un valor lógico en la página 1159</a> |
| OR                                 | <a href="#">OR - Evalúa un valor lógico en la página 1394</a>  |
| XOR                                | <a href="#">XOR - Evalúa un valor lógico en la página 1605</a> |
| Expresiones                        | <i>Manual de referencia técnica - RAPID Overview</i>           |

## 2 Funciones

---

### 2.123 NOrient - Normaliza la orientación

RobotWare - OS

### 2.123 NOrient - Normaliza la orientación

---

#### Utilización

NOrient (*Normalize Orientation*) se utiliza para normalizar una orientación no normalizada (cuaternio).

---

#### Descripción

La orientación debe estar normalizada, es decir, la suma de los cuadrados debe ser igual a 1:

$$q_1^2 + q_2^2 + q_3^2 + q_4^2 = 1$$

xx0500002452

Si la orientación está ligeramente desnormalizada, es posible normalizarla. El error de normalización es el valor absoluto de la suma de los cuadrados de los componentes de orientación. Se considera que la orientación está ligeramente desnormalizada si el error de normalización es superior a 0,00001 e inferior a 0,1. Si el error de normalización es superior a 0,1, no puede utilizarse la orientación.

$$\text{ABS}(\sqrt{q_1^2 + q_2^2 + q_3^2 + q_4^2} - 1) = \text{normerr}$$

xx0500002453

|                                      |                          |
|--------------------------------------|--------------------------|
| normerr > 0,1                        | No utilizable            |
| normerr > 0,00001 AND normerr <= 0,1 | Levemente no normalizada |
| normerr <= 0,00001                   | Normalizada              |

---

#### Ejemplos básicos

El ejemplo siguiente ilustra la función NOrient.

##### Ejemplo 1

Tenemos una posición ligeramente desnormalizada (0,707170, 0, 0, 0,707170).

$$\text{ABS}(\sqrt{0,707170^2 + 0^2 + 0^2 + 0,707170^2} - 1) = 0,0000894$$

0,0000894 > 0,00001 ⇒ unnormalized

xx0500002451

```
VAR orient unnormorient := [0.707170, 0, 0, 0.707170];
VAR orient normorient;
...
...
normorient := NOrient(unnormorient);
```

La normalización de la orientación (0,707170, 0, 0, 0,707170) se convierte en (0,707107, 0, 0, 0,707107).

---

#### Valor de retorno

Tipo de dato: orient

La orientación normalizada.

*Continúa en la página siguiente*

---

#### Argumentos

NOrient (Rotation)

#### Rotación

Tipo de dato: orient

La orientación que debe ser normalizada.

---

#### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema ERRNO cambiará a:

| Nombre           | Causa del error  |
|------------------|--|
| ERR_ORIENT_VALUE | Valor de orientación incorrecto en la función NOrient. |

---

#### Sintaxis

```
NOrient '('  
  [Rotation ':='] <expression (IN) of orient> ')'
```

Una función con un valor de retorno del tipo de dato orient.

---

#### Información relacionada

| Para obtener más información sobre    | Consulte  |
|---------------------------------------|---|
| Instrucciones y funciones matemáticas | <i>Manual de referencia técnica - RAPID Overview, sección Resúmen sobre RAPID - Matemáticas</i> |

## 2 Funciones

---

### 2.124 NumToDnum - Convierte num a dnum

RobotWare - OS

### 2.124 NumToDnum - Convierte num a dnum

---

#### Utilización

NumToDnum **convierte un valor num en un valor dnum.**

---

#### Ejemplos básicos

El ejemplo siguiente ilustra la función NumToDnum.

#### Ejemplo 1

```
VAR num mynum:=55;  
VAR dnum mydnum:=0;  
mydnum:=NumToDnum(mynum);
```

El valor num 55 es devuelto por la función como el valor dnum 55.

---

#### Valor de retorno

**Tipo de dato:** dnum

El valor de retorno del tipo dnum tendrá el mismo valor que el valor de entrada del tipo num.

---

#### Argumentos

NumToDnum (Value)

Value

**Tipo de dato:** num

El valor numérico a convertir.

---

#### Sintaxis

```
NumToDnum '( '  
[ Value ':=' ] < expression (IN) of num > ')'
```

Una función con un valor de retorno del tipo de dato dnum.

---

#### Información relacionada

| Para obtener más información sobre | Consulte  |
|------------------------------------|---|
| Num tipo de dato                   | <a href="#">num - Valores numéricos en la página 1712</a>         |
| Dnum tipo de dato                  | <a href="#">dnum - Valores numéricos dobles en la página 1660</a> |



---

## 2.125 NumToStr - Convierte un valor numérico en una cadena de caracteres

---

### Utilización

NumToStr (*Numeric To String*) se utiliza para convertir un valor numérico en una cadena.

---

### Ejemplos básicos

Los siguientes ejemplos ilustran la función NumToStr.

#### Ejemplo 1

```
VAR string str;
str := NumToStr(0.38521,3);
```

Se asigna a la variable `str` el valor "0.385".

#### Ejemplo 2

```
reg1 := 0.38521;
str := NumToStr(reg1, 2\Exp);
```

Se asigna a la variable `str` el valor "3.85E-01".

---

### Valor de retorno

Tipo de dato: `string`

El valor numérico, convertido en una cadena con el número especificado de decimales, en notación científica si así se solicita. Si es necesario, el valor numérico se redondea. Si no se incluye ningún decimal, se suprime el punto decimal.

---

### Argumentos

```
NumToStr (Val Dec [\Exp])
```

Val

*Value*

Tipo de dato: `num`

El valor numérico a convertir.

Dec

*Decimals*

Tipo de dato: `num`

Número de decimales. El número de decimales no debe ser negativo ni mayor que la precisión disponible para los valores numéricos.

[\Exp]

*Exponent*

Tipo de dato: `switch`

Para usar exponente en el valor de retorno.

---

### Sintaxis

```
NumToStr '('
  [ Val ':' ] <expression (IN) of num>
  [ Dec ':' ] <expression (IN) of num>
```

*Continúa en la página siguiente*

## 2 Funciones

---

### 2.125 NumToStr - Convierte un valor numérico en una cadena de caracteres

RobotWare - OS

Continuación

[ '\ ' Exp ' ) ' ]

Una función con un valor de retorno del tipo de dato `string`.

---

#### Información relacionada

| Para obtener más información sobre                           | Consulte   |
|--|--|
| Funciones para cadenas de caracteres                         | <i>Manual de referencia técnica - RAPID Overview, sección Resumen sobre RAPID - Funciones para cadenas de caracteres</i> |
| Definición de cadena de caracteres                           | <a href="#">string - Cadenas en la página 1778</a>   |
| Valores de cadena de caracteres                              | <i>Manual de referencia técnica - RAPID Overview, sección Características básicas - Elementos básicos</i>                |
| Convertir un valor numérico dnum en una cadena de caracteres | <a href="#">DnumToStr - Convierte un valor numérico en una cadena de caracteres en la página 1272</a>                    |

---

## 2.126 Offs - Desplaza una posición del robot

---

### Utilización

`Offs` se utiliza para añadir un offset en el sistema de coordenadas de objeto a una posición de robot.

---

### Ejemplos básicos

Los siguientes ejemplos ilustran la función `Offs`.

Consulte también [Más ejemplos en la página 1391](#).

#### Ejemplo 1

```
MoveL Offs(p2, 0, 0, 10), v1000, z50, tool1;
```

Se mueve el robot a un punto que se encuentra a 10 mm de la posición `p2` (en la dirección `z`).

#### Ejemplo 2

```
p1 := Offs (p1, 5, 10, 15);
```

Se desplaza la posición del robot `p1` 5 mm en la dirección `x`, 10 mm en la dirección `y`, y 15 mm en la dirección `z`.

---

### Valor de retorno

Tipo de dato: `robtarget`

Los datos de la posición desplazada.

---

### Argumentos

```
Offs (Point XOffset YOffset ZOffset)
```

Point

Tipo de dato: `robtarget`

Los datos de posición del desplazamiento.

XOffset

Tipo de dato: `num`

El desplazamiento en mm en la dirección `x` del sistema de coordenadas de objeto.

YOffset

Tipo de dato: `num`

El desplazamiento en la dirección `y` del sistema de coordenadas de objeto.

ZOffset

Tipo de dato: `num`

El desplazamiento en la dirección `z` del sistema de coordenadas de objeto.

---

### Más ejemplos

A continuación aparecen más ejemplos de la función `Offs`.

#### Ejemplo 1

```
PROC pallet (num row, num column, num distance, PERS tooldata tool,  
            PERS wobjdata wobj)
```

*Continúa en la página siguiente*

## 2 Funciones

### 2.126 Offs - Desplaza una posición del robot

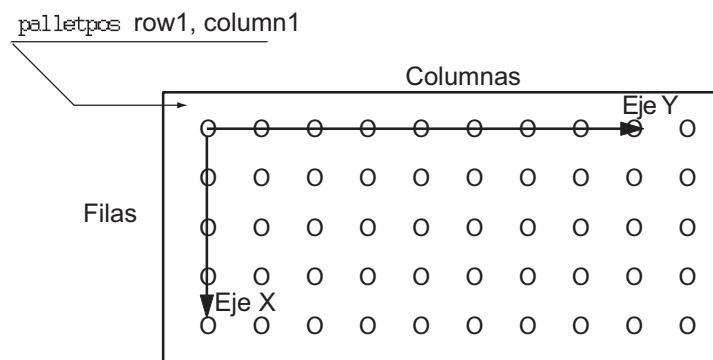
RobotWare - OS

Continuación

```
VAR robtarget palletpos:=[[0, 0, 0], [1, 0, 0, 0], [0, 0, 0, 0],
    [9E9, 9E9, 9E9, 9E9, 9E9, 9E9]];
palletpos := Offs (palletpos, (row-1)*distance, (column-1)*distance,
    0);
MoveL palletpos, v100, fine, tool\WObj:=wobj;
ENDPROC
```

Se crea una rutina para tomar piezas de un palé. Cada palé se define como un objeto de trabajo (consulte la figura siguiente). La pieza que debe tomarse (fila y columna) y la distancia existente entre las partes se indican como parámetros de entrada. El aumento del índice de fila y columna se realiza fuera de la rutina.

La figura muestra la posición y la orientación del palé que se especifican mediante la definición de un objeto de trabajo.



xx050002300\_

### Sintaxis

```
Offs '('
    [Point ':=' ] <expression (IN) of robtarget> ', '
    [XOffset ':=' ] <expression (IN) of num> ', '
    [YOffset ':=' ] <expression (IN) of num> ', '
    [ZOffset ':=' ] <expression (IN) of num> ')'
```

Una función con un valor de retorno del tipo de dato `robtarget`.

### Información relacionada

| Para obtener más información sobre    | Consulte  |
|---------------------------------------|---|
| Datos de posición                     | <a href="#">robtarget - Datos de posición en la página 1750</a>                                 |
| Instrucciones y funciones matemáticas | <i>Manual de referencia técnica - RAPID Overview, sección Resumen sobre RAPID - Matemáticas</i> |
| Instrucciones de posicionamiento      | <i>Manual de referencia técnica - RAPID Overview, sección Resumen sobre RAPID - Movimiento</i>  |

## 2.127 OpMode - Lee el modo de funcionamiento

### Utilización

OpMode(*Operating Mode*) se utiliza para leer el modo de funcionamiento actual del sistema.

### Ejemplos básicos

El ejemplo siguiente ilustra la función OpMode.

#### Ejemplo 1

```
TEST OpMode()
CASE OP_AUTO:
  ...
CASE OP_MAN_PROG:
  ...
CASE OP_MAN_TEST:
  ...
DEFAULT:
  ...
ENDTEST
```

Se ejecutan secciones distintas del programa en función del modo de funcionamiento actual.

### Valor de retorno

Tipo de dato: `symnum`

El modo de funcionamiento actual, con uno de los valores definidos en la tabla siguiente.

| Valor de retorno | Constante simbólica | Comentario  |
|------------------|---------------------|---|
| 0                | OP_UNDEF            | Modo de funcionamiento no definido                      |
| 1                | OP_AUTO             | Modo de funcionamiento automático                       |
| 2                | OP_MAN_PROG         | Modo de funcionamiento manual a 250 mm/seg como máximo  |
| 3                | OP_MAN_TEST         | Modo de funcionamiento manual a máxima velocidad, 100 % |

### Sintaxis

```
OpMode '( ' )'
```

Una función con un valor de retorno del tipo de dato `symnum`.

### Información relacionada

| Para obtener más información sobre | Consulte   |
|------------------------------------|--|
| Distintos modos de funcionamiento  | <i>Manual del operador - IRC5 con FlexPendant</i>                        |
| Lectura del modo de ejecución      | <a href="#">RunMode - Obtiene el modo de ejecución en la página 1465</a> |

## 2 Funciones

---

2.128 OR - Evalúa un valor lógico  
RobotWare - OS

### 2.128 OR - Evalúa un valor lógico

---

#### Utilización

OR es una expresión condicional utilizada para evaluar un valor lógico (verdadero/falso).

---

#### Ejemplos básicos

Los siguientes ejemplos ilustran la función OR.

##### Ejemplo 1

```
VAR num a;  
VAR num b;  
VAR bool c;  
...  
c := a>5 OR b=3;
```

El valor de retorno de `c` es `TRUE` si `a` es mayor de 5 o `b` es igual a 3. De lo contrario, el valor de retorno es `FALSE`.

##### Ejemplo 2

```
VAR num mynum;  
VAR string mystring;  
VAR bool mybool;  
VAR bool result;  
...  
result := mystring="Hello" OR mynum<15 AND mybool;
```

El valor de retorno `result` es `TRUE` si `mystring` tiene el valor "Hello". O bien, si `mynum` es menor de 15 y `mybool` tiene el valor `TRUE`. De lo contrario, el valor de retorno es `FALSE`.

En primer lugar se evalúa la sentencia `AND` y, a continuación, la sentencia `OR`. Esto se ilustra mediante los paréntesis de la fila que aparece a continuación.

```
result := mystring="Hello" OR (mynum<15 AND mybool);
```

---

#### Valor de retorno

Tipo de dato: `bool`

El valor de retorno es `TRUE` si una de las expresiones condicionales es correcta o ambas son correctas; de lo contrario, el valor de retorno es `FALSE`.

---

#### Sintaxis

```
<expression of bool> OR <expression of bool>
```

Una función con un valor de retorno del tipo de dato `bool`.

---

#### Información relacionada

| Para obtener más información sobre | Consulte   |
|------------------------------------|--|
| AND                                | <a href="#">AND - Evalúa un valor lógico en la página 1159</a>   |
| XOR                                | <a href="#">XOR - Evalúa un valor lógico en la página 1605</a>   |
| NOT                                | <a href="#">NOT - Invierte un valor lógico en la página 1385</a> |

Continúa en la página siguiente

|   |  |
|---|--|
| <b>Para obtener más información sobre</b> | <b>Consulte</b>                                      |
| Expresiones                               | <i>Manual de referencia técnica - RAPID Overview</i> |

## 2 Funciones

---

2.129 OrientZYX - Genera una orientación a partir de ángulos Euler  
RobotWare - OS

### 2.129 OrientZYX - Genera una orientación a partir de ángulos Euler

---

#### Utilización

`OrientZYX` (*Orient from Euler ZYX angles*) se utiliza para generar una variable de tipo `orient` a partir de ángulos Euler.

---

#### Ejemplos básicos

El ejemplo siguiente ilustra la función `OrientZYX`.

#### Ejemplo 1

```
VAR num anglex;  
VAR num angley;  
VAR num anglez;  
VAR pose object;  
...  
object.rot := OrientZYX(anglez, angley, anglex)
```

---

#### Valor de retorno

Tipo de dato: `orient`

La orientación obtenida a partir de ángulos Euler.

---

#### Argumentos

`OrientZYX` (ZAngle YAngle XAngle)



#### Nota

Las rotaciones se realizan en el orden siguiente:

- 1 giro alrededor del eje z
- 2 giro alrededor del nuevo eje y
- 3 giro alrededor del nuevo eje x

ZAngle

Tipo de dato: `num`

La rotación, en grados, alrededor del eje Z.

YAngle

Tipo de dato: `num`

La rotación, en grados, alrededor del eje Y.

XAngle

Tipo de dato: `num`

La rotación, en grados, alrededor del eje X.

---

#### Sintaxis

```
OrientZYX '('  
  [ZAngle ':='] <expression (IN) of num> ','  
  [YAngle ':='] <expression (IN) of num> ','  
  [XAngle ':='] <expression (IN) of num> ')'
```

*Continúa en la página siguiente*



Una función con un valor de retorno del tipo de dato `orient`.

---

#### Información relacionada

| Para obtener más información sobre    | Consulte  |
|---------------------------------------|---|
| Instrucciones y funciones matemáticas | <i>Manual de referencia técnica - RAPID Overview, sección Matemáticas</i> |

## 2 Funciones

---

### 2.130 ORobT - Elimina el desplazamiento de programa de una posición

RobotWare - OS

### 2.130 ORobT - Elimina el desplazamiento de programa de una posición

---

#### Utilización

ORobT (*Object Robot Target*) se utiliza para transformar una posición del sistema de coordenadas de desplazamiento de programa al sistema de coordenadas del objeto y/o eliminar un offset de los ejes externos.

---

#### Ejemplos básicos

El ejemplo siguiente ilustra la función ORobT.

Consulte también [Más ejemplos en la página 1399](#).

#### Ejemplo 1

```
VAR robtarget p10;  
VAR robtarget p11;  
VAR num wobj_diameter;  
  
p10 := CRobT(\Tool:=tool1 \WObj:=wobj_diameter);  
p11 := ORobT(p10);
```

Se almacenan en p10 y p11 las posiciones actuales de los ejes del robot y de los ejes externos. Los valores almacenados en p10 dependen del sistema de coordenadas ProgDisp/ExtOffs. Los valores almacenados en p11 dependen del sistema de coordenadas del objeto sin ningún desplazamiento de programa ni offset en los ejes externos.

---

#### Valor de retorno

Tipo de dato: robtarget

Los datos de posición transformados.

---

#### Argumentos

```
ORobT (OrgPoint [\InPDisp] | [\InEOffs])
```

OrgPoint

##### *Original Point*

Tipo de dato: robtarget

El punto original que debe transformarse.

[\InPDisp]

##### *In Program Displacement*

Tipo de dato: switch

Devuelve la posición del TCP en el sistema de coordenadas ProgDisp, es decir, sólo elimina el offset de los ejes externos.

[\InEOffs]

##### *In External Offset*

Tipo de dato: switch

Devuelve los ejes externos en el sistema de coordenadas del offset, es decir, sólo elimina el desplazamiento de programa del robot.

---

*Continúa en la página siguiente*

**Más ejemplos**

A continuación aparecen más ejemplos de cómo usar la función ORobT.

**Ejemplo 1**

```
p10 := ORobT(p10 \InEOffs );
```

La función ORobT eliminará cualquier desplazamiento de programa que esté activo, dejando la posición del TCP respecto del sistema de coordenadas del objeto. Los ejes externos permanecen en el sistema de coordenadas del offset.

**Ejemplo 2**

```
p10 := ORobT(p10 \InPDisp );
```

La función ORobT eliminará cualquier offset de los ejes externos. La posición del TCP permanece en el sistema de coordenadas ProgDisp.

**Sintaxis**

```
ORobT '('  
  [ OrgPoint ':= ' ] < expression (IN) of robtarget >  
  ['\' InPDisp] | ['\' InEOffs] ')'
```

Una función con un valor de retorno del tipo de dato robtarget.

**Información relacionada**

| Para obtener más información sobre                      | Consulte   |
|---|--|
| Definición de desplazamientos de programa para el robot | <a href="#">PDispOn</a> - Activa el desplazamiento de programa en la página 533<br><a href="#">PDispSet</a> - Activa un desplazamiento de programa a partir de una base de coordenadas conocida en la página 538 |
| Definición de un offset para los ejes externos          | <a href="#">EOffsOn</a> - Activa un offset de ejes adicionales en la página 195<br><a href="#">EOffsSet</a> - Activa un offset de ejes adicionales a partir de valores conocidos en la página 197                |
| Sistemas de coordenadas                                 | <i>Manual de referencia técnica - RAPID Overview, sección Principios de movimiento y E/S - Sistemas de coordenadas</i>   |

## 2 Funciones

---

### 2.131 ParIdPosValid - Posición de robot válida para la identificación de parámetros *RobotWare - OS*

### 2.131 ParIdPosValid - Posición de robot válida para la identificación de parámetros

---

#### Utilización

`ParIdPosValid` (*Parameter Identification Position Valid*) comprueba si la posición del robot es válida para la identificación actual de parámetros, por ejemplo la identificación de carga de la herramienta o de la carga útil.

Esta instrucción sólo puede usarse en la tarea `main` o, si se cuenta con un sistema *MultiMove*, en las tareas de movimiento.

---

#### Ejemplos básicos

El ejemplo siguiente ilustra la función `ParIdPosValid`.

#### Ejemplo 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
```

Se comprueba si la posición del robot es válida antes de realizar la identificación de carga de la herramienta.

---

#### Valor de retorno

Tipo de dato: `bool`

`TRUE` si la posición del robot es válida para la identificación de parámetros actual.

`FALSE` si la posición del robot no es válida para la identificación de parámetros actual.

---

#### Argumentos

```
ParIdPosValid (ParIdType Pos AxValid [\ConfAngle])
```

*Continúa en la página siguiente*

2.131 ParIdPosValid - Posición de robot válida para la identificación de parámetros  
*RobotWare - OS*  
*Continuación*

ParIdType

Tipo de dato: paridnum

Un tipo de identificación de parámetros de los definidos en la tabla siguiente

| Valor | Constante simbólica | Comentario   |
|-------|---------------------|--|
| 1     | TOOL_LOAD_ID        | Identificación de la carga de la herramienta                       |
| 2     | PAY_LOAD_ID         | Identificación de la carga útil (consulte la instrucción GripLoad) |
| 3     | IRBP_K              | Identificación de la carga del manipulador externo IRBP K          |
| 4     | IRBP_L              | Identificación de la carga del manipulador externo IRBP L          |
| 4     | IRBP_C              | Identificación de la carga del manipulador externo IRBP C          |
| 4     | IRBP_C_INDEX        | Identificación de la carga del manipulador externo IRBP C_INDEX    |
| 4     | IRBP_T              | Identificación de la carga del manipulador externo IRBP T          |
| 5     | IRBP_R              | Identificación de la carga del manipulador externo IRBP R          |
| 6     | IRBP_A              | Identificación de la carga del manipulador externo IRBP A          |
| 6     | IRBP_B              | Identificación de la carga del manipulador externo IRBP B          |
| 6     | IRBP_D              | Identificación de la carga del manipulador externo IRBP D          |

Pos

Tipo de dato: jointtarget

La variable especifica los ángulos actuales de todos los ejes del robot y de los ejes externos. La variable se actualiza mediante ParIdPosValid de acuerdo con la tabla siguiente.

| Valor de eje de entrada | Valor de eje de salida     |
|-------------------------|----------------------------|
| Válido                  | No cambia                  |
| No válido               | Cambia a un valor adecuado |

AxValid

Tipo de dato: bool

Una variable de matriz con 12 elementos que corresponden a 6 ejes del robot y 6 ejes externos. La variable se actualiza mediante ParIdPosValid de acuerdo con la tabla siguiente.

| Valor de eje de entrada de Pos | Estado de salida de AxValid |
|--------------------------------|-----------------------------|
| Válido                         | TRUE                        |
| No válido                      | FALSE                       |

*Continúa en la página siguiente*

## 2 Funciones

### 2.131 ParIdPosValid - Posición de robot válida para la identificación de parámetros

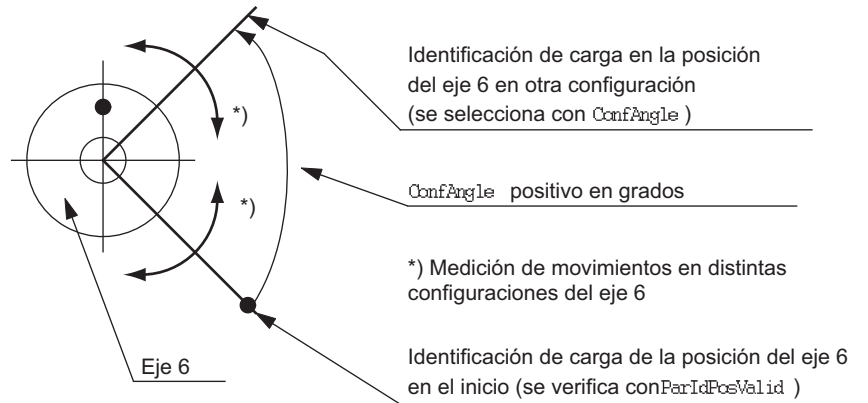
RobotWare - OS

Continuación

[ \ConfAngle ]

Tipo de dato: num

El argumento opcional para la especificación de un ángulo +/- grados de configuración determinado para su uso en la identificación de parámetros.



xx0500002493

Si no se especifica este argumento, el valor predeterminado es +90 grados.  
Mín. + ó -30 grados. Valor óptimo + o -90 grados.

### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                        | Causa del error           |
|-------------------------------|---------------------------|
| <code>ERR_PID_RAISE_PP</code> | Se ha producido un error. |

### Sintaxis

```
ParIdPosValid '('  
  [ ParIdType ':=' ] <expression (IN) of paridnum> ','  
  [ Pos ':=' ] <variable (VAR) of jointtarget> ','  
  [ AxValid ':=' ] <array variable {*} (VAR) of bool>  
  [ '\ ' ConfAngle ':=' <expression (IN) of num> ] ')'
```

Una función con un valor de retorno del tipo de dato `bool`.

### Información relacionada

| Para obtener más información sobre                        | Consulte  |
|---|---|
| Tipo de identificación de parámetro                       | <a href="#">paridnum - Tipo de identificación de parámetro en la página 1721</a>                            |
| Tipo de robot válido                                      | <a href="#">ParIdRobValid - Tipo de robot válido para la identificación de parámetros en la página 1403</a> |
| Identificación de carga de la herramienta o la carga útil | <a href="#">LoadId - Identificación de carga de la herramienta o la carga útil en la página 352</a>         |
| Identificación de carga de posicionadores (IRBP)          | <a href="#">ManLoadIdProc - Identificación de carga de los manipuladores IRBP en la página 360</a>          |

## 2.132 ParIdRobValid - Tipo de robot válido para la identificación de parámetros

### Utilización

ParIdRobValid (*Parameter Identification Robot Valid*) comprueba si el tipo de robot o manipulador es válido para la identificación actual de parámetros, por ejemplo la identificación de carga de la herramienta o de la carga útil.

Esta instrucción sólo puede usarse en la tarea principal T\_ROB1 o, si se cuenta con un sistema *MultiMove*, en las tareas de movimiento.

### Ejemplos básicos

El ejemplo siguiente ilustra la función ParIdRobValid.

#### Ejemplo 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
```

### Valor de retorno

Tipo de dato: paridvalidnum

Indica si la identificación del parámetro especificado puede realizarse con el tipo de robot o manipulador actual, con los valores definidos en la tabla siguiente.

| Valor | Constante simbólica | Comentario   |
|-------|---------------------|--|
| 10    | ROB_LOAD_VAL        | Tipo de robot o manipulador válido para la identificación actual de parámetros                             |
| 11    | ROB_NOT_LOAD_VAL    | No es ningún tipo válido para la identificación actual de parámetros                                       |
| 12    | ROB_LM1_LOAD_VAL    | Tipo de robot válido IRB 6400FHD para la identificación actual de parámetros, si la carga real es < 200 kg |

### Argumentos

```
ParIdRobValid(ParIdType [\MechUnit] [\AxisNo])
```

*Continúa en la página siguiente*

## 2 Funciones

### 2.132 ParIdRobValid - Tipo de robot válido para la identificación de parámetros

RobotWare - OS

Continuación

ParIdType

Tipo de dato: `paridnum`

Un tipo de identificación de parámetros de los definidos en la tabla siguiente.

| Valor | Constante simbólica | Comentario   |
|-------|---------------------|--|
| 1     | TOOL_LOAD_ID        | Identificación de la carga de la herramienta del robot                       |
| 2     | PAY_LOAD_ID         | Identificación de la carga útil del robot (consulte la instrucción GripLoad) |
| 3     | IRBP_K              | Identificación de la carga del manipulador externo IRBP K                    |
| 4     | IRBP_L              | Identificación de la carga del manipulador externo IRBP L                    |
| 4     | IRBP_C              | Identificación de la carga del manipulador externo IRBP C                    |
| 4     | IRBP_C_INDEX        | Identificación de la carga del manipulador externo IRBP C_INDEX              |
| 4     | IRBP_T              | Identificación de la carga del manipulador externo IRBP T                    |
| 5     | IRBP_R              | Identificación de la carga del manipulador externo IRBP R                    |
| 6     | IRBP_A              | Identificación de la carga del manipulador externo IRBP A                    |
| 6     | IRBP_B              | Identificación de la carga del manipulador externo IRBP B                    |
| 6     | IRBP_D              | Identificación de la carga del manipulador externo IRBP D                    |

[ `\MechUnit` ]

#### *Mechanical Unit*

Tipo de dato: `mecunit`

La unidad mecánica utilizada para la identificación de carga. Sólo debe especificarse en el caso de los manipuladores externos. Si se omite este argumento, se utiliza el robot de TCP de la tarea.

[ `\AxisNo` ]

#### *Axis number*

Tipo de dato: `num`

Dentro de la unidad mecánica, el número del eje que sostiene la carga que se desea identificar. Sólo debe especificarse en el caso de los manipuladores externos.

Si se utiliza el argumento `\MechUnit`, debe utilizarse `\AxisNo`. El argumento `\AxisNo` no puede usarse sin `\MechUnit`.

### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                        | Causa del error           |
|-------------------------------|---------------------------|
| <code>ERR_PID_RAISE_PP</code> | Se ha producido un error. |

Continúa en la página siguiente



**Sintaxis**

```

ParIdRobValid '('
  [ParIdType ':=' ] <expression (IN) of paridnum>
  ['\' MechUnit ':=' <variable (VAR) of mecunit>]
  ['\' AxisNo ':=' <expression (IN) of num> ] ')'

```

Una función con un valor de retorno del tipo de dato paridvalidnum.

**Información relacionada**

| Para obtener más información sobre                                  | Consulte  |
|---|---|
| Tipo de identificación de parámetro                                 | <a href="#">paridnum - Tipo de identificación de parámetro en la página 1721</a>                                |
| Unidad mecánica a identificar                                       | <a href="#">mecunit - Unidad mecánica en la página 1703</a>   |
| Resultado de esta función   | <a href="#">paridvalidnum - Resultado de ParIdRobValid en la página 1723</a>                                    |
| Posición de robot válida  | <a href="#">ParIdPosValid - Posición de robot válida para la identificación de parámetros en la página 1400</a> |
| Identificación de carga de la herramienta o la carga útil del robot | <a href="#">LoadId - Identificación de carga de la herramienta o la carga útil en la página 352</a>             |
| Identificación de carga de cargas de posicionador                   | <a href="#">ManLoadIdProc - Identificación de carga de los manipuladores IRBP en la página 360</a>              |

## 2 Funciones

---

2.133 PathLengthGet - Lee el valor de longitud de trayectoria actual del contador  
*RobotWare - OS*

### 2.133 PathLengthGet - Lee el valor de longitud de trayectoria actual del contador

---

#### Utilización

`PathLengthGet` se utiliza para leer el valor del contador que mide la longitud de trayectoria recorrida por el TCP del robot. El valor de retorno se indica en milímetros y se mide siempre en relación con el objeto de trabajo.

La llamada a esta función puede realizarse en cualquier momento, aunque se recomienda hacerlo cuando el robot esté en reposo para obtener un comportamiento predecible.

Se lee el valor de longitud de trayectoria para el robot TCP en la tarea de movimiento real o la conectada.

#### Ejemplos básicos

El ejemplo siguiente ilustra la función `PathLengthGet`.

##### Ejemplo 1

```
PathLengthStart;  
MoveJ p10, v1000, z50, L10tip;  
...  
MoveL p40, v1000, fine, L10tip;  
PathLengthStop;  
TPWrite "PathLengthGet: "+ValToStr(PathLengthGet());  
PathLengthReset;
```

Este ejemplo realiza la lectura del valor del contador que mide la longitud de trayectoria recorrida por el TCP del robot. El valor se escribe en el FlexPendant.

#### Ejecución de programas

La medición de longitud de trayectoria se aplica a la siguiente instrucción de movimiento de robot ejecutada, de cualquier tipo, y se mantiene vigente hasta la ejecución de una instrucción `PathLengthStop`.

La medición de longitud de trayectoria se desactiva, y el contador de medición de longitud de trayectoria se establece en cero, cuando se ejecuta una instrucción `PathLengthReset`. El valor predeterminado, medición de longitud de trayectoria desactivada, se establece automáticamente:

- Cuando se utiliza el modo de reinicio *Restablecer RAPID*.
- al cargar un nuevo programa o un nuevo módulo.
- Cuando se inicia la ejecución del programa desde el principio
- Cuando se mueve el puntero del programa a rutina principal.
- al mover el puntero del programa a una rutina.
- al mover el puntero de programa de una forma que se pierde el orden de la ejecución.

#### Limitaciones

Las mediciones de longitud de trayectoria solo son aplicables a robots TCP.

*Continúa en la página siguiente*

### 2.133 PathLengthGet - Lee el valor de longitud de trayectoria actual del contador

RobotWare - OS

Continuación

#### Sintaxis

```
PathLengthGet '(' '')
```

Una función con un valor de retorno del tipo de dato num.

#### Información relacionada

| Para obtener más información sobre | Consulte  |
|------------------------------------|---|
| PathLengthReset                    | <a href="#">PathLengthReset - Restablece el valor de longitud de trayectoria actual del contador en la página 508</a> |
| PathLengthStart                    | <a href="#">PathLengthStart - Activa el contador que monitoriza la longitud de trayectoria en la página 510</a>       |
| PathLengthStop                     | <a href="#">PathLengthStop - Detiene el contador que monitoriza la longitud de trayectoria en la página 512</a>       |

## 2 Funciones

---

### 2.134 PathLevel - Obtiene el nivel de trayectoria actual

RobotWare - OS

### 2.134 PathLevel - Obtiene el nivel de trayectoria actual

---

#### Utilización

`PathLevel` se utiliza para obtener el nivel de trayectoria actual. Esta función mostrará si la tarea se está ejecutando en el nivel original o si la trayectoria de movimiento original se ha almacenado y se está ejecutando un nuevo movimiento temporal. Para saber más sobre *Path Recovery* en *Application manual - Controller software IRC5*.

#### Ejemplos básicos

El ejemplo siguiente ilustra la función `PathLevel`.

Consulte también [Más ejemplos en la página 1408](#).

#### Ejemplo 1

```
VAR num level;  
level:= PathLevel();
```

La variable `level` es 1 si se ejecutó en una trayectoria de movimiento original o 2 si se ejecutó en una nueva trayectoria de movimiento temporal.

#### Valor de retorno

Tipo de dato: num

Hay dos valores de retorno posibles.

| Valor de retorno | Descripción   |
|------------------|---|
| 1                | Ejecución en una trayectoria de movimiento original.                                  |
| 2                | Ejecución en una trayectoria StorePath, una nueva trayectoria de movimiento temporal. |

#### Más ejemplos

A continuación aparecen más ejemplos de cómo usar la función `PathLevel`.

#### Ejemplo 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;  
...
```

La variable `level` es 2.

*Continúa en la página siguiente*

#### Limitaciones

La opción Path Recovery de RobotWare debe estar instalada para poder utilizar la función `PathLevel` en el nivel de trayectoria 2

#### Sintaxis

```
PathLevel '(' '')
```

Una función con un valor de retorno del tipo de dato `num.`

#### Información relacionada

| Para obtener más información sobre            | Consulte  |
|---|---|
| Path Recovery                                 | <i>Application manual - Controller software IRC5</i>  |
| Almacenamiento y restauración de trayectorias | <a href="#">StorePath - Almacena la trayectoria cuando se produce una interrupción en la página 835</a><br><a href="#">RestoPath - Restablece la trayectoria después de una interrupción en la página 615</a> |
| Detención e inicio del movimiento             | <a href="#">StartMove - Reanuda el movimiento del robot en la página 796</a><br><a href="#">StopMove - Detiene el movimiento del robot en la página 828</a>   |

## 2 Funciones

### 2.135 PathRecValidBwd - Comprueba si existe una trayectoria de retroceso válida guardada *Path Recovery*

### 2.135 PathRecValidBwd - Comprueba si existe una trayectoria de retroceso válida guardada

#### Utilización

PathRecValidBwd se utiliza para comprobar si la grabadora de trayectorias está activa y si está disponible una trayectoria hacia atrás grabada.

#### Ejemplos básicos

El ejemplo siguiente ilustra la función PathRecValidBwd.

Consulte también [Más ejemplos en la página 1411](#).

#### Ejemplo 1

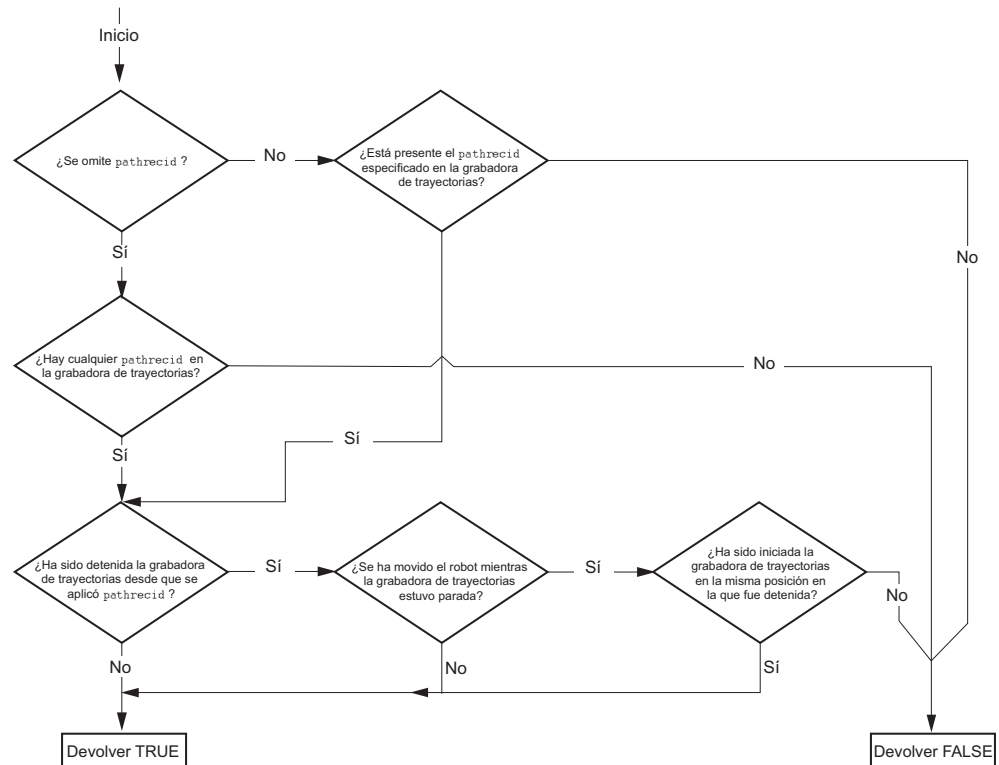
```
VAR bool bwd_path;  
VAR pathrecid fixture_id;  
bwd_path := PathRecValidBwd (\ID:=fixture_id);
```

La variable `bwd_path` tiene el valor TRUE si es posible retroceder hasta una posición con el identificador `fixture_id`. En caso negativo, `bwd_path` recibe el valor FALSE.

#### Valor de retorno

Tipo de dato: `bool`

El valor de retorno de la función puede determinarse a partir del diagrama de flujo siguiente:



xx0500002132

Continúa en la página siguiente

**Argumentos**

```
PathRecValidBwd ([\ID])
```

```
[\ID]
```

**Identifier**

Tipo de dato: pathrecid

Una variable que especifica el nombre de la posición de inicio de la grabación. El tipo de dato `pathrecid` es de un tipo sin valor y sólo se utiliza como un identificador para asignar un nombre a la posición de grabación.

**Ejecución de programas**

Antes de pedir con `PathRecMoveBwd` a la grabadora de trayectorias que retroceda por la trayectoria, es posible comprobar con `PathRecValidBwd` si existe una trayectoria grabada válida.

**Más ejemplos**

Los siguientes ejemplos ilustran la función `PathRecValidBwd`.

**Ejemplo 1**

```
PathRecStart id1;
MoveL p1, vmax, z50, tool1;
MoveL p2, vmax, z50, tool1;
bwd_path := PathRecValidBwd (\ID := id1);
```

Se inicia la grabadora de trayectorias y se ejecutan dos instrucciones de movimiento. `PathRecValidBwd` devuelve TRUE y la trayectoria de retroceso será: Desde `p2` a `p1` y de ahí a la posición de inicio.

**Ejemplo 2**

```
PathRecStart id1;
MoveL p1, vmax, z50, tool1;
MoveL p2, vmax, z50, tool1;
PathRecStop \Clear;
bwd_path:= PathRecValidBwd (\ID := id1);
```

Se inicia la grabadora de trayectorias y se ejecutan dos instrucciones de movimiento. A continuación, se detiene y vacía la grabadora de trayectorias. `PathRecValidBwd` devuelve FALSE.

**Ejemplo 3**

```
PathRecStart id1;
MoveL p1, vmax, z50, tool1;
PathRecStart id2;
MoveL p2, vmax, z50, tool1;
bwd_path := PathRecValidBwd ();
```

Se inicia la grabadora de trayectorias y se ejecuta una instrucción de movimiento. A continuación, se inicia un identificador de trayectoria adicional, seguido de una instrucción de movimiento. `PathRecValidBwd` devuelve TRUE y la trayectoria de retroceso será:

De `p2` a `p1`.

*Continúa en la página siguiente*

## 2 Funciones

### 2.135 PathRecValidBwd - Comprueba si existe una trayectoria de retroceso válida guardada

#### Path Recovery

#### Continuación

#### Ejemplo 4

```
PathRecStart id1;  
MoveL p1, vmax, z50, tool1;  
WaitSyncTask sync101, tasklist_rl01;  
MoveL p2, vmax, z50, tool1;  
bwd_path1 := PathRecValidBwd ();  
bwd_path2 := PathRecValidBwd (\ID := id1);
```

La ejecución del programa anterior dará lugar a que la variable booleana `bwd_path1` reciba el valor TRUE, dado que existe una trayectoria de retroceso válida hacia `WaitSyncTask`. La variable booleana `bwd_path2` recibe el valor FALSE dado que no es posible retroceder más arriba de una sentencia `WaitSyncTask`.

#### Sintaxis

```
PathRecValidBwd '('  
  ['\' ID :=' < variable (VAR) of pathrecid > ] ')'
```

Una función con un valor de retorno del tipo de dato `bool`.

#### Información relacionada

| Para obtener más información sobre                             | Consulte   |
|--|--|
| Identificadores de grabadora de trayectorias                   | <a href="#">pathrecid - Identificador de grabadora de trayectorias en la página 1725</a>   |
| Inicio y detención de la grabadora de trayectorias             | <a href="#">PathRecStart - Inicia la grabadora de trayectorias en la página 524</a><br><a href="#">PathRecStop - Detiene la grabadora de trayectorias en la página 527</a> |
| Reproducción de la grabación de trayectorias hacia atrás       | <a href="#">PathRecMoveBwd - Hace retroceder la grabadora de trayectorias en la página 514</a>   |
| Comprobación de si existe una trayectoria válida hacia delante | <a href="#">PathRecValidFwd - Comprueba si existe una trayectoria de avance válida guardada en la página 1413</a>  |
| Reproducción de la grabación de trayectorias hacia delante     | <a href="#">PathRecMoveFwd - Hace avanzar la grabadora de trayectorias en la página 521</a>  |
| Movimiento en general  | <a href="#">Manual de referencia técnica - RAPID Overview</a>  |



## 2.136 PathRecValidFwd - Comprueba si existe una trayectoria de avance válida guardada

### Utilización

PathRecValidFwd se utiliza para comprobar si la grabadora de trayectorias puede usarse para avanzar. La posibilidad de avanzar con la grabadora de trayectorias implica que se debe haber solicitado previamente el movimiento hacia atrás a la grabadora de trayectorias.

### Ejemplos básicos

El ejemplo siguiente ilustra la función PathRecValidFwd.

Consulte también [Más ejemplos en la página 1414](#).

#### Ejemplo 1

```
VAR bool fwd_path;
VAR pathrecid fixture_id;

fwd_path:= PathRecValidFwd (\ID:=fixture_id);
```

La variable fwd\_path tiene el valor TRUE si es posible avanzar hasta una posición con el identificador fixture\_id. En caso negativo, fwd\_path recibe el valor FALSE.

### Valor de retorno

Tipo de dato: bool

El valor de retorno de PathRecValidFwd sin el valor especificado en \ID es:

TRUE si:

- La grabadora de trayectorias ha movido el robot hacia atrás, usando PathRecMoveBwd.
- El robot no se ha alejado de la trayectoria ejecutada por PathRecMoveBwd.

FALSE si:

- No se cumplen las condiciones indicadas anteriormente.

El valor de retorno de PathRecValidFwd con el valor especificado en \ID es:

TRUE si:

- La grabadora de trayectorias ha movido el robot hacia atrás, usando PathRecMoveBwd.
- El robot no se ha alejado de la trayectoria ejecutada por PathRecMoveBwd.
- La \ID especificada fue entregada durante el movimiento de retroceso.

FALSE si:

- No se cumplen las condiciones indicadas anteriormente.

### Argumentos

```
PathRecValidFwd ([\ID])
```

[\ID]

*Identifier*

*Continúa en la página siguiente*

## 2 Funciones

---

### 2.136 PathRecValidFwd - Comprueba si existe una trayectoria de avance válida guardada

#### *Path Recovery*

#### *Continuación*

Tipo de dato: pathrecid

Una variable que especifica el nombre de la posición de inicio de la grabación. El tipo de dato `pathrecid` es de un tipo sin valor y sólo se utiliza como un identificador para asignar un nombre a la posición de grabación.

---

#### Ejecución de programas

Después de pedir con `PathRecMoveBwd` a la grabadora de trayectorias que retroceda, es posible comprobar si existe una trayectoria válida grabada sobre la que mover el robot. Si se omite el identificador `\ID`, `PathRecValidFwd` indica si es posible avanzar hasta la posición en la que se inició el movimiento de retroceso.

---

#### Más ejemplos

El ejemplo siguiente ilustra la función `PathRecValidFwd`.

#### Ejemplo 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 the 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
```

*Continúa en la página siguiente*

2.136 PathRecValidFwd - Comprueba si existe una trayectoria de avance válida guardada  
*Path Recovery*  
 Continuación

ENDIF  
 !See figure 7 in the following table.  
 PathRecMoveFwd;  
 RestoPath;  
 StartMove;  
 RETRY;

|          |   |
|----------|---|
| <p>1</p> | <p>Posición de inicio<br/> <i>id1</i></p> <p>xx0500002121</p> |
| <p>2</p> | <p>xx0500002124</p>   |
| <p>3</p> | <p>xx0500002126</p>   |
| <p>4</p> | <p>Posición de inicio<br/> <i>id1</i></p> <p>xx0500002127</p> |
| <p>5</p> | <p>Posición de inicio<br/> <i>id1</i></p> <p>xx0500002128</p> |
| <p>6</p> | <p>xx0500002129</p>   |
| <p>7</p> | <p>xx0500002130</p>   |
| <p>8</p> | <p>xx0500002131</p>   |

Continúa en la página siguiente

## 2 Funciones

### 2.136 PathRecValidFwd - Comprueba si existe una trayectoria de avance válida guardada

#### *Path Recovery*

#### *Continuación*

En el ejemplo anterior, se inicia la grabadora de trayectorias y se añaden identificadores en tres ubicaciones distintas a lo largo de la trayectoria ejecutada. La imagen anterior contiene referencias al código de ejemplo y describe cómo se moverá el robot si se produce un error durante la ejecución hacia delante hasta el punto p3. PathRecValidBwd y PathRecValidFwd se utilizan respectivamente, dado que no es posible avanzar para determinar en qué punto del programa se ha producido un error posible.

#### Sintaxis

```
PathRecValidFwd '('  
    ['\' ID ' := ' < variable (VAR) of pathrecid >'] ')'
```

Una función con un valor de retorno del tipo de dato bool.

#### Información relacionada

| Para obtener más información sobre                            | Consulte   |
|---|--|
| Identificadores de grabadora de trayectorias                  | <a href="#">pathrecid - Identificador de grabadora de trayectorias en la página 1725</a>   |
| Inicio y detención de la grabadora de trayectorias            | <a href="#">PathRecStart - Inicia la grabadora de trayectorias en la página 524</a><br><a href="#">PathRecStop - Detiene la grabadora de trayectorias en la página 527</a> |
| Comprobación de si existe una trayectoria de retroceso válida | <a href="#">PathRecValidBwd - Comprueba si existe una trayectoria de retroceso válida guardada en la página 1410</a>   |
| Reproducción de la grabación de trayectorias hacia atrás      | <a href="#">PathRecMoveBwd - Hace retroceder la grabadora de trayectorias en la página 514</a>   |
| Reproducción de la grabación de trayectorias hacia delante    | <a href="#">PathRecMoveFwd - Hace avanzar la grabadora de trayectorias en la página 521</a>  |
| Movimiento en general   | <a href="#">Manual de referencia técnica - RAPID Overview</a>  |

## 2.137 PFRestart - Comprueba si se ha interrumpido una trayectoria después de un fallo de alimentación

*RobotWare - OS*

### 2.137 PFRestart - Comprueba si se ha interrumpido una trayectoria después de un fallo de alimentación

#### Utilización

**PFRestart** (*Power Failure Restart*) se utiliza para comprobar si la trayectoria fue interrumpida como consecuencia de la caída de alimentación. Si es así, es posible que sea necesario realizar algunas acciones específicas. La función comprueba la trayectoria en el nivel actual, el nivel de base y el nivel de interrupciones.

#### Ejemplos básicos

El ejemplo siguiente ilustra la función **PFRestart**.

##### Ejemplo 1

```
IF PFRestart() = TRUE THEN
```

Se comprueba si existe una trayectoria interrumpida en el nivel actual. Si es así, la función devuelve **TRUE**.

#### Valor de retorno

Tipo de dato: **bool**

**TRUE** si existe una trayectoria interrumpida en el nivel de trayectoria especificado. De lo contrario, devuelve **FALSE**.

#### Argumentos

```
PFRestart([\Base] | [\Irpt])
```

[ \Base ]

##### *Base Level*

Tipo de dato: **switch**

Devuelve **TRUE** si existe una trayectoria interrumpida en el nivel de base.

[ \Irpt ]

##### *Interrupt Level*

Tipo de dato: **switch**

Devuelve **TRUE** si existe una trayectoria interrumpida en el nivel **StorePath**.

Si no se indica ningún argumento, la función devuelve **TRUE** si existe una trayectoria interrumpida en el nivel actual.

#### Sintaxis

```
PFRestart '('  
  ['\' Base] | ['\' Irpt] ')'
```

Una función con un valor de retorno del tipo de dato **bool**.

#### Información relacionada

| Para obtener más información sobre | Consulte  |
|------------------------------------|---|
| <i>Advanced RAPID</i>              | <i>Especificaciones del producto - Controller software IRC5</i> |

## 2 Funciones

2.138 PoseInv - Invierte los datos de pose  
RobotWare - OS

### 2.138 PoseInv - Invierte los datos de pose

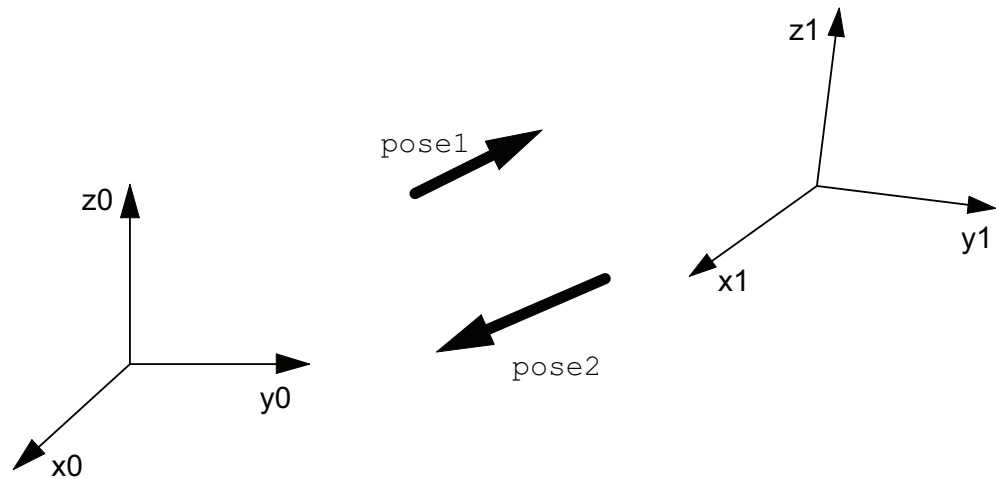
#### Utilización

PoseInv (*Pose Invert*) calcula la transformación inversa de una pose.

#### Ejemplos básicos

El ejemplo siguiente ilustra la función PoseInv.

#### Ejemplo 1



xx0500002443

pose1 representa el sistema de coordenadas 1 relacionado con el sistema de coordenadas 0. La transformación que indica el sistema de coordenadas 0 relacionado con el sistema de coordenadas 1 se obtiene mediante la transformación inversa, almacenada en pose2.

```
VAR pose pose1;  
VAR pose pose2;  
...  
pose2 := PoseInv(pose1);
```

#### Valor de retorno

Tipo de dato: pose  
El valor de la pose inversa.

#### Argumentos

PoseInv (Pose)

Pose

Tipo de dato: pose  
La pose a invertir.

#### Sintaxis

```
PoseInv ('  
  [Pose ::='] <expression (IN) of pose>  
'')
```

Continúa en la página siguiente

Una función con un valor de retorno del tipo de dato `pose`.

---

### Información relacionada

| Para obtener más información sobre    | Consulte   |
|---------------------------------------|--|
| Instrucciones y funciones matemáticas | <i>Manual de referencia técnica - RAPID Overview</i> |

## 2 Funciones

### 2.139 PoseMult - Multiplica datos de pose RobotWare - OS

### 2.139 PoseMult - Multiplica datos de pose

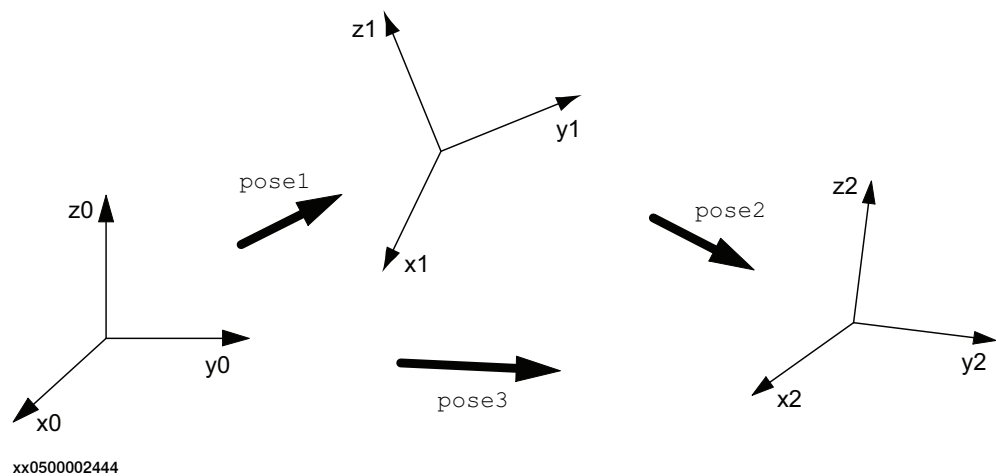
#### Utilización

`PoseMult` (*Pose Multiply*) se utiliza para calcular el producto de dos transformaciones de pose. Una aplicación típica es calcular una nueva pose como resultado de un desplazamiento que actúa sobre una pose original.

#### Ejemplos básicos

El ejemplo siguiente ilustra la función `PoseMult`.

#### Ejemplo 1



`pose1` representa el sistema de coordenadas 1 relacionado con el sistema de coordenadas 0. `pose2` representa el sistema de coordenadas 2 relacionado con el sistema de coordenadas 1. La transformación que indica `pose3`, el sistema de coordenadas 2 relacionado con el sistema de coordenadas 0, se obtiene mediante el producto de las dos transformaciones:

```
VAR pose pose1;  
VAR pose pose2;  
VAR pose pose3;  
...  
pose3 := PoseMult(pose1, pose2);
```

#### Valor de retorno

Tipo de dato: `pose`  
El valor del producto de dos poses.

#### Argumentos

```
PoseMult (Pose1 Pose2)
```

`Pose1`

Tipo de dato: `pose`  
La primera pose.

*Continúa en la página siguiente*



Pose2

**Tipo de dato:** pose

La segunda pose.

---

#### Sintaxis

```
PoseMult '('  
  [Pose1 ':=' ] <expression (IN) of pose> ','  
  [Pose2 ':=' ] <expression (IN) of pose> ')'
```

Una función con un valor de retorno del tipo de dato pose.

---

#### Información relacionada

| Para obtener más información sobre    | Consulte   |
|---------------------------------------|--|
| Instrucciones y funciones matemáticas | <i>Manual de referencia técnica - RAPID Overview</i> |

## 2 Funciones

### 2.140 PoseVect - Aplica una transformación a un vector RobotWare - OS

#### 2.140 PoseVect - Aplica una transformación a un vector

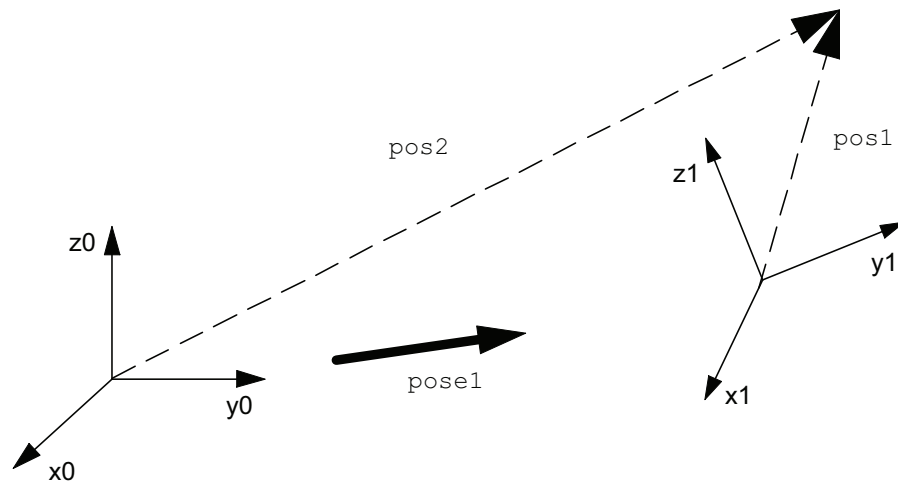
##### Utilización

PoseVect (*vector de pose*) se utiliza para calcular el producto de una pose y un vector. Se suele utilizar para calcular un vector como resultado del efecto de un desplazamiento o de un vector original.

##### Ejemplos básicos

El ejemplo siguiente ilustra la función PoseVect.

##### Ejemplo 1



xx0500002445

pose1 representa el sistema de coordenadas 1 relacionado con el sistema de coordenadas 0.

pos1 es un vector relacionado con sistema de coordenadas 1. El vector correspondiente relacionado con el sistema de coordenadas 0 se obtiene mediante el producto;

```
VAR pose pose1;  
VAR pos pos1;  
VAR pos pos2;  
...  
...  
pos2:= PoseVect(pose1, pos1);
```

##### Valor de retorno

Tipo de dato: pos

El valor del producto de la pose y el valor pos original.

##### Argumentos

PoseVect (Pose Pos)

Pose

Tipo de dato: pose

La transformación a aplicar.

Continúa en la página siguiente

Pos

Tipo de dato: pos

El valor pos a transformar.

#### Sintaxis

```
PoseVect '('  
  [Pose ':=' ] <expression (IN) of pose> ','  
  [Pos ':=' ] <expression (IN) of pos> ')'
```

Una función con un valor de retorno del tipo de dato pos.

#### Información relacionada

| Para obtener más información sobre    | Consulte   |
|---------------------------------------|--|
| Instrucciones y funciones matemáticas | <i>Manual de referencia técnica - RAPID Overview</i> |

## 2 Funciones

---

### 2.141 Pow - Calcula el resultado de elevar un valor a una potencia

RobotWare - OS

### 2.141 Pow - Calcula el resultado de elevar un valor a una potencia

---

#### Utilización

Pow (*Power*) se utiliza para calcular el valor exponencial en cualquier base.

---

#### Ejemplos básicos

El ejemplo siguiente ilustra la función Pow.

#### Ejemplo 1

```
VAR num x;  
VAR num y  
VAR num reg1;  
...  
reg1:= Pow(x, y);  
reg1 recibe el valor  $x^y$ .
```

---

#### Valor de retorno

Tipo de dato: num

El valor de la Base elevado a la potencia del exponente, es decir  $\text{base}^{\text{Exponente}}$ .

---

#### Argumentos

Pow (Base Exponent)

Base

Tipo de dato: num

El valor del argumento usado como base.

Exponent

Tipo de dato: num

El valor del argumento de exponente.

---

#### Limitaciones

La ejecución de la función  $x^y$  genera un error en los casos siguientes:

- Si  $x < 0$  e  $y$  no es un entero;
  - Si  $x = 0$  e  $y \leq 0$ .
- 

#### Sintaxis

```
Pow '('  
  [Base ':=' ] <expression (IN) of num> ','  
  [Exponent ':=' ] <expression (IN) of num> ')'
```

Una función con un valor de retorno del tipo de dato num.

---

#### Información relacionada

| Para obtener más información sobre    | Consulte                                      |
|---------------------------------------|---|
| Instrucciones y funciones matemáticas | Manual de referencia técnica - RAPID Overview |

---

## 2.142 PowDnum - Calcula el resultado de elevar un valor a una potencia

### Utilización

PowDnum (*Power Dnum*) se utiliza para calcular el valor exponencial en cualquier base.

### Ejemplos básicos

El ejemplo siguiente ilustra la función PowDnum.

#### Ejemplo 1

```
VAR dnum x;
VAR num y
VAR dnum value;
...
value:= PowDnum(x, y);
value recibe el valor  $x^y$ .
```

### Valor de retorno

Tipo de dato: dnum

El valor de la Base elevado a la potencia del exponente, es decir  $\text{base}^{\text{Exponente}}$ .

### Argumentos

PowDnum (Base Exponent)

Base

Tipo de dato: dnum

El valor del argumento usado como base.

Exponent

Tipo de dato: num

El valor del argumento de exponente.

### Limitaciones

La ejecución de la función  $x^y$  genera un error en los casos siguientes:

- Si  $x < 0$  e  $y$  no es un entero;
- Si  $x = 0$  e  $y \leq 0$ .

### Sintaxis

```
PowDnum '('
  [Base ':=' ] <expression (IN) of dnum> ','
  [Exponent ':=' ] <expression (IN) of num> ')'
```

Una función con un valor de retorno del tipo de dato dnum.

### Información relacionada

| Para obtener más información sobre    | Consulte   |
|---------------------------------------|--|
| Instrucciones y funciones matemáticas | <i>Manual de referencia técnica - RAPID Overview</i> |

## 2 Funciones

---

2.143 PPMovedInManMode - Comprobar si el puntero de programa se ha movido en el modo manual.  
*RobotWare - OS*

### 2.143 PPMovedInManMode - Comprobar si el puntero de programa se ha movido en el modo manual.

---

#### Utilización

PPMovedInManMode devuelve TRUE si el usuario ha movido el puntero de programa mientras el controlador se encuentra en el modo manual, es decir, si la llave de operador está en las posiciones de manual a velocidad reducida o manual a máxima velocidad. El estado del puntero de programa movido se pone a cero cuando se sitúa la llave de Auto a Man o cuando se usa la instrucción ResetPPMoved.

---

#### Ejemplos básicos

El ejemplo siguiente ilustra la función PPMovedInManMode.

##### Ejemplo 1

```
IF PPMovedInManMode() THEN
  WarnUserOfPPMovement;
  DoJob;
ELSE
  DoJob;
ENDIF
```

---

#### Valor de retorno

Tipo de dato: `bool`

TRUE si el puntero de programa ha sido movido por el usuario en el modo manual.

---

#### Ejecución de programas

Comprobar si el puntero de programa de la tarea de programa actual ha sido movido en el modo manual.

---

#### Sintaxis

```
PPMovedInManMode '( ' )'
```

Una función con un valor de retorno del tipo de dato `bool`.

---

#### Información relacionada

| Para obtener más información sobre                                     | Consulte  |
|--|---|
| Comprobar si el puntero de programa se ha movido                       | <a href="#">IsStopStateEvent - Comprueba si se ha movido el puntero de programa en la página 1362</a>                   |
| Restablecer el estado del puntero de programa movido en el modo manual | <a href="#">ResetPPMoved - Restablecer el estado del puntero de programa movido en el modo manual. en la página 612</a> |

---

## 2.144 Present - Comprueba si se está usando un parámetro opcional

---

### Utilización

`Present` se utiliza para comprobar si se ha utilizado un argumento opcional al llamar a una rutina.

Los parámetros opcionales no pueden usarse si no se especificaron al llamar a la rutina. Esta función puede usarse para comprobar si se ha especificado un parámetro, con la finalidad de evitar la aparición de errores.

---

### Ejemplos básicos

El ejemplo siguiente ilustra la función `Present`.

Consulte también [Más ejemplos en la página 1427](#).

#### Ejemplo 1

```
PROC feeder (\switch on | switch off)
  IF Present (on) Set d01;
  IF Present (off) Reset d01;
ENDPROC
```

La salida `d01`, que controla un transportador, se activa o desactiva en función del argumento utilizado al llamar a la rutina.

---

### Valor de retorno

Tipo de dato: `bool`

TRUE = Se ha definido el valor de parámetro o el modificador al llamar a la rutina.

FALSE = No se ha definido el valor del parámetro o el modificador.

---

### Argumentos

`Present (OptPar)`

OptPar

*Optional Parameter*

Tipo de dato: cualquier tipo

El nombre del parámetro opcional cuya presencia se desea comprobar.

---

### Más ejemplos

El ejemplo siguiente ilustra la función `Present`.

#### Ejemplo 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
```

*Continúa en la página siguiente*

## 2 Funciones

---

### 2.144 Present - Comprueba si se está usando un parámetro opcional

RobotWare - OS

Continuación

Se crea una rutina de aplicación de adhesivo Si se especifica el argumento `\on` al llamar a la rutina, se genera un pulso en la señal `glue_on`. A continuación, el robot activa la señal analógica de salida `gluesignal`, que controla la pistola de adhesivo y la mueve hasta la posición final. Dado que el parámetro `wobj` es opcional, se utilizan instrucciones `MoveL` diferentes en función de si se utiliza o no este argumento.

---

#### Sintaxis

```
Present '('  
  [OptPar ':='] <reference (REF) of any type> ')'
```

En este caso, el parámetro REF requiere el nombre del parámetro opcional.

Una función con un valor de retorno del tipo de dato `bool`.

---

#### Información relacionada

| Para obtener más información sobre | Consulte   |
|------------------------------------|--|
| Parámetros de rutinas              | <i>Manual de referencia técnica - RAPID Overview</i> |



## 2.145 ProgMemFree - Obtiene el tamaño de memoria libre del programa

### Utilización

`ProgMemFree` (*Program Memory Free*) se usa para obtener el tamaño libre de la memoria de programas.

### Ejemplos básicos

El ejemplo siguiente ilustra la función `ProgMemFree`.

#### Ejemplo 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` se usa en una función que devuelve el valor de cuánta memoria se reserva para un módulo en la memoria de programas.

### Valor de retorno

Tipo de dato: `num`

El tamaño del espacio libre de la memoria de programas, en bytes.

### Sintaxis

```
ProgMemFree '(' ' ')
```

Una función con un valor de retorno del tipo de dato `num`.

### Información relacionada

| Para obtener más información sobre | Consulte  |
|------------------------------------|---|
| Carga de un módulo de programa     | <a href="#">Load - Carga un módulo de programa durante la ejecución en la página 347</a>        |
| Descarga de un módulo de programa  | <a href="#">UnLoad - Descargar un módulo de programa durante la ejecución en la página 1021</a> |

## 2 Funciones

---

### 2.146 PrxGetMaxRecordpos - Obtención de la posición máxima de sensor

*Machine Synchronization*

### 2.146 PrxGetMaxRecordpos - Obtención de la posición máxima de sensor

---

#### Utilización

PrxGetMaxRecordpos se utiliza para devolver la posición máxima en mm del registro activo.

La posición máxima del sensor puede usarse para escalar o limitar el argumento `max_sync` de la instrucción `SyncToSensor`.

---

#### Ejemplo básico

```
maxpos:=PrxGetMaxRecordpos Ssync1;
```

Obtiene la posición máxima para el perfil activo de la unidad mecánica `Ssync1`.

---

#### Valor de retorno

Tipo de dato: `num`

La posición máxima (en mm) del perfil grabado del movimiento del sensor.

---

#### Argumentos

```
PrxGetMaxRecordpos MechUnit
```

MechUnit

Tipo de dato: `mechunit`

El objeto de unidad mecánica móvil al que está sincronizado el movimiento del robot.

---

#### Ejecución de programas

La grabación debe haber finalizado y el registro debe estar activo.

---

#### Sintaxis

```
PrxGetMaxRecordpos '('  
  [ MechUnit ':=' ] < expression (IN) of mechunit > ')'
```

Una función con un valor de retorno del tipo de dato `num`.

---

#### Información relacionada

| Para obtener más información sobre | Consulte   |
|------------------------------------|--|
| <i>Machine Synchronization</i>     | <i>Application manual - Controller software IRC5</i> |

---

## 2.147 Rand - Genera un número aleatorio

---

### Utilización

Se utiliza `Rand` para generar un número entero aleatorio entre 0 y `RAND_MAX`.

---

### Ejemplos básicos

El ejemplo siguiente ilustra la función `Rand`.

#### Ejemplo 1

```
VAR num myrandomnumber;  
..  
myrandomnumber:=Rand();
```

Genera un número aleatorio y lo asigna a la variable `myrandomnumber`.

---

### Valor de retorno

Tipo de dato: `num`

Un número aleatorio entre 0 y 65535 (0 y 32767 para un controlador virtual).

---

### Argumentos

```
Rand ([\Seed] [\SeedEachNTime])
```

`\Seed`

Tipo de dato: `switch`

La hora actual se utiliza para el seed del generador de números aleatorios si se utiliza este interruptor. Si no se utiliza el interruptor el seed se realiza a la primera llamada de la función `Rand`, y la secuencia generada de números será siempre la misma.

`\SeedEachNTime`

Tipo de dato: `num`

Establece un contador de seeds que se reduce cada vez que se ejecuta la función `Rand`. Cuando el valor es 0 se realiza un nuevo seed y el contador se configura en el valor utilizado en la llamada precedente con el argumento opcional `SeedEachNTime`. El contador se configura con el nuevo valor cada vez que se utiliza `SeedEachNTime` en la función `Rand`.

---

### Sintaxis

```
Rand '('  
  ['\ Seed]  
  ['\ SeedEachNTime]')'
```

Una función con un valor de retorno del tipo de dato `num`.

---

## 2 Funciones

---

### 2.148 RawBytesLen - Obtiene la longitud de un dato de tipo rawbytes

RobotWare - OS

### 2.148 RawBytesLen - Obtiene la longitud de un dato de tipo rawbytes

---

#### Utilización

`RawBytesLen` se utiliza para obtener la longitud actual de los bytes válidos de una variable de tipo `rawbytes`.

---

#### Ejemplos básicos

El ejemplo siguiente ilustra la función `RawBytesLen`.

#### Ejemplo 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;
```

En este ejemplo, la variable `from_raw_data` del tipo `rawbytes` es borrada en primer lugar. Es decir, todos sus bytes cambian a 0 (el valor predeterminado tras la declaración). A continuación, el valor del entero se almacena en los primeros 2 bytes y, con ayuda de la función `RawBytesLen`, el valor de `float` se almacena en los 4 bytes siguientes (comenzando por el número de índice 3).

Después de guardar un dato en `from_raw_data`, el contenido (6 bytes) se copia a `to_raw_data`, empezando por la posición 3.

---

#### Valor de retorno

Tipo de dato: `num`

La longitud actual de los bytes válidos de una variable de tipo `rawbytes`; en el rango de 0 ... 1024.

En general, la longitud actual de los bytes válidos de una variable `rawbytes` es actualizada por el sistema al último byte escrito en la estructura de la variable `rawbytes`.

Para obtener más detalles, consulte el tipo de dato `rawbytes`, la instrucción `ClearRawBytes`, `CopyRawBytes`, `PackDNHeader`, `PackRawBytes` y `ReadRawBytes`.

---

#### Argumentos

`RawBytesLen (RawData)`

`RawData`

Tipo de dato: `rawbytes`

`RawData` es el contenedor de datos cuya longitud actual de bytes válidos se desea obtener.

---

*Continúa en la página siguiente*

## 2.148 RawBytesLen - Obtiene la longitud de un dato de tipo rawbytes

RobotWare - OS

Continuación

## Ejecución de programas

Durante la ejecución del programa, se devuelve la longitud actual de bytes válidos.

## Sintaxis

```
RawBytesLen '('
  [RawData ':=' ] < variable (VAR) of rawbytes> ')'
```

Una función con un valor de retorno del tipo de dato num.

## Información relacionada

| Para obtener más información sobre                              | Consulte   |
|---|--|
| rawbytes datos  | <a href="#">rawbytes - Datos sin formato en la página 1736</a>   |
| Borrado del contenido de un dato de tipo rawbytes               | <a href="#">ClearRawBytes - Borra el contenido de un dato de tipo rawbytes en la página 146</a>        |
| Copiado del contenido de un dato de tipo rawbytes               | <a href="#">CopyRawBytes - Copia el contenido de un dato de tipo rawbytes en la página 171</a>         |
| Empaquetamiento de un encabezado de DeviceNet en datos rawbytes | <a href="#">PackDNHeader - Empaqueta un encabezado de DeviceNet en datos rawbytes en la página 496</a> |
| Empaquetamiento de datos en datos rawbytes                      | <a href="#">PackRawBytes - Empaqueta datos en un dato de tipo rawbytes en la página 499</a>            |
| Lectura de un dato rawbytes                                     | <a href="#">ReadRawBytes - Lee datos de tipo rawbytes en la página 589</a>                             |
| Desempaquetamiento de datos de un dato rawbytes                 | <a href="#">UnpackRawBytes - Desempaqueta datos de un dato de tipo rawbytes en la página 1024</a>      |
| Escritura de un dato rawbytes                                   | <a href="#">WriteRawBytes - Escribe un dato de tipo rawbytes en la página 1111</a>                     |
| Gestión de archivos y dispositivos de E/S                       | <a href="#">Application manual - Controller software IRC5</a>  |

## 2 Funciones

---

2.149 ReadBin - Lee un byte de un archivo o dispositivo de E/S  
*RobotWare - OS*

### 2.149 ReadBin - Lee un byte de un archivo o dispositivo de E/S

---

#### Utilización

ReadBin (*Read Binary*) se utiliza para leer un byte (8 bits) de un archivo o dispositivo de E/S.

Esta función es compatible con archivos o dispositivos de E/S de tipo binario y alfanumérico.

---

#### Ejemplos básicos

El ejemplo siguiente ilustra la función ReadBin.

Consulte también [Más ejemplos en la página 1435](#).

#### Ejemplo 1

```
VAR num character;  
VAR iodev file1;  
...  
Open "HOME:", \File:= "FILE1.DOC", file1\Bin;  
character := ReadBin(file1);
```

Se lee un byte del archivo binario file1.

---

#### Valor de retorno

Tipo de dato: num

Un byte (8 bits) se lee de un archivo o un dispositivo de E/S especificado. Este byte se convierte en el valor numérico positivo correspondiente y se devuelve con el tipo de dato num. Si un archivo está vacío (se ha alcanzado el fin del archivo), se devuelve EOF\_BIN (el número -1).

---

#### Argumentos

```
ReadBin (IODevice [\Time])
```

IODevice

Tipo de dato: iodev

El nombre (referencia) del archivo o dispositivo de E/S que debe leerse.

[\Time]

Tipo de dato: num

El tiempo máximo para la operación de lectura (tiempo límite) en segundos. Si no se especifica este argumento, el tiempo máximo es de 60 segundos. Para esperar ininterrumpidamente, utilice la constante predefinida WAIT\_MAX.

Si se agota este tiempo antes de que se complete la operación de lectura, se llama al gestor de errores con el código de error ERR\_DEV\_MAXTIME. Si no hay ningún gestor de errores, se detiene la ejecución.

La función de tiempo límite se utiliza también durante un paro de programa y se observará en el programa de RAPID al poner en marcha el programa.

---

*Continúa en la página siguiente*

**Ejecución de programas**

La ejecución del programa espera hasta que se pueda leer un byte (8 bits) del archivo o del dispositivo de E/S.

En caso de un reinicio tras una caída de alimentación, todos los archivos o dispositivos de E/S abiertos del sistema se cierran y el descriptor de E/S de la variable del tipo `iodev` se restablece.

**Más ejemplos**

El ejemplo siguiente ilustra la función `ReadBin`.

**Ejemplo 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
```

Se lee el contenido del archivo binario `myfile.bin` desde el principio hasta el fin y se muestran los datos binarios recibidos (un carácter cada vez) en el FlexPendant, tras convertirlos en caracteres.

**Limitaciones**

Esta función sólo puede usarse con archivos y dispositivos de E/S que hayan sido abiertos con un acceso de lectura (`\Read` en el caso de los archivos alfanuméricos, `\Bin` o `\Append \Bin` en el caso de los archivos binarios).

**Gestión de errores**

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                       | Causa del error   |
|------------------------------|---|
| <code>ERR_FILEACC</code>     | Se produjo un error durante la lectura.                     |
| <code>ERR_DEV_MAXTIME</code> | Tiempo límite antes de que termine la operación de lectura. |

**Datos predefinidos**

La constante `EOF_BIN` puede usarse para detener la lectura al final del archivo.

```
CONST num EOF_BIN := -1;
```

**Sintaxis**

```
ReadBin '('
  [IODevice ']:='] <variable (VAR) of iodev>
  ['\' Time ']:='] <expression (IN) of num> ] ''
```

Una función con un valor de retorno del tipo de dato `num`.

*Continúa en la página siguiente*

## 2 Funciones

---

2.149 ReadBin - Lee un byte de un archivo o dispositivo de E/S

RobotWare - OS

Continuación

---

### Información relacionada

| Para obtener más información sobre                | Consulte  |
|---|---|
| Apertura, etc. de archivos o dispositivos de E/S  | <i>Manual de referencia técnica - RAPID Overview</i>  |
| Conversión de un byte en una cadena de caracteres | <a href="#"><i>ByteToStr - Convierte un byte en un dato de cadena de caracteres en la página 1202</i></a> |
| Gestión de archivos y dispositivos de E/S         | <i>Application manual - Controller software IRC5</i>  |



---

**2.150 ReadDir - Lee la siguiente entrada de un directorio**

---

**Utilización**

`ReadDir` se utiliza para obtener el nombre del siguiente archivo o subdirectorío existente dentro de un directorio abierto previamente con la instrucción `OpenDir`. Siempre y cuando la función devuelva `TRUE`, pueden existir más archivos o subdirectorios a obtener.

---

**Ejemplos básicos**

El ejemplo siguiente ilustra la función `ReadDir`.

Consulte también [Más ejemplos en la página 1438](#).

**Ejemplo 1**

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

Este ejemplo imprime los nombres de todos los archivos o subdirectorios que se encuentran dentro del directorio especificado.

---

**Valor de retorno**

Tipo de dato: `bool`

La función devuelve `TRUE` si ha obtenido un nombre. De lo contrario, devuelve `FALSE`.

---

**Argumentos**

`ReadDir (Dev FileName)`

Dev

Tipo de dato: `dir`

Una variable que hace referencia a un directorio y capturada con la instrucción `OpenDir`.

FileName

Tipo de dato: `string`

El nombre del archivo o subdirectorío obtenido.

---

**Ejecución de programas**

La función devuelve un valor booleano que especifica si la obtención de un nombre fue o no correcta.

*Continúa en la página siguiente*

## 2 Funciones

---

### 2.150 ReadDir - Lee la siguiente entrada de un directorio

RobotWare - OS

Continuación

---

#### Más ejemplos

A continuación aparecen más ejemplos de la función `ReadDir`

#### Ejemplo 1

Este ejemplo implementa una función de recorrido genérico de una estructura de directorios.

```
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

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
```

Este programa recorre la estructura de directorios que existe dentro de "HOME:" y con cada archivo encontrado, ejecuta el procedimiento `listfile`. `searchdir` es la parte genérica, que no tiene ninguna información sobre el inicio de la búsqueda ni sobre a qué rutina se debe llamar con cada archivo. Utiliza `IsFile` para comprobar si se ha encontrado un subdirectorio o un archivo y utiliza el mecanismo de enlazamiento en tiempo de ejecución para llamar al procedimiento especificado en `actionproc` con todos los archivos encontrados. La rutina `actionproc` debe ser un procedimiento con un parámetro de tipo `string`.

---

#### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                   | Causa del error   |
|--------------------------|---|
| <code>ERR_FILEACC</code> | El directorio no está abierto (consulte <code>OpenDir</code> ). |

Continúa en la página siguiente

**Sintaxis**

```
ReadDir '('  
  [ Dev ':= ' ] < variable (VAR) of dir> ','  
  [ FileName ':= ' ] < var or pers (INOUT) of string> ')'
```

Una función con un valor de retorno del tipo de dato `bool`.

**Información relacionada**

| Para obtener más información sobre        | Consulte   |
|---|--|
| Directorio                                | <a href="#">dir - Estructura de directorio de archivos en la página 1659</a> |
| Creación de un directorio                 | <a href="#">MakeDir - Crea un nuevo directorio en la página 358</a>          |
| Apertura de un directorio                 | <a href="#">OpenDir - Abre un directorio en la página 494</a>                |
| Cierre de un directorio                   | <a href="#">CloseDir - Cierra un directorio en la página 153</a>             |
| Eliminación de un directorio              | <a href="#">RemoveDir - Elimina un directorio en la página 598</a>           |
| Eliminación de un archivo                 | <a href="#">RemoveFile - Elimina un archivo en la página 600</a>             |
| Cambio del nombre de un archivo           | <a href="#">RenameFile - Cambia el nombre de un archivo en la página 604</a> |
| Gestión de archivos y dispositivos de E/S | <a href="#">Application manual - Controller software IRC5</a>                |

## 2 Funciones

---

### 2.151 ReadMotor - Lee los ángulos actuales de los motores

RobotWare - OS

### 2.151 ReadMotor - Lee los ángulos actuales de los motores

---

#### Utilización

`ReadMotor` se utiliza para leer los ángulos actuales de los distintos motores de los ejes del robot y de los ejes externos. La aplicación principal de esta función es la realización de procedimientos de calibración del robot.

---

#### Ejemplos básicos

El ejemplo siguiente ilustra la función `ReadMotor`.

Consulte también [Más ejemplos en la página 1441](#).

#### Ejemplo 1

```
VAR num motor_angle2;  
motor_angle2 := ReadMotor(2);
```

El ángulo actual del motor del segundo eje del robot se almacena en `motor_angle2`.

---

#### Valor de retorno

Tipo de dato: `num`

El ángulo actual del motor del eje indicado en radianes, ya sea un eje del robot o un eje externo.

---

#### Argumentos

```
ReadMotor [ \MecUnit ] Axis
```

MecUnit

**Mechanical Unit**

Tipo de dato: `mecunit`

El nombre de la unidad mecánica cuyos valores de eje se desea comprobar. Si se omite este argumento, se obtiene el valor de un eje del robot conectado.

Axis

Tipo de dato: `num`

El número del eje cuyo valor se desea obtener (de 1 a 6).

---

#### Ejecución de programas

El ángulo de motor devuelto representa la posición actual del motor en radianes sin ningún offset de calibración. El valor no depende de ninguna posición fija del robot, sólo de la posición cero interna del resolver, es decir, normalmente la posición cero del resolver más cercana a la posición de calibración (la diferencia existente entre la posición cero del resolver y la posición de calibración es el valor del offset de calibración). Este valor representa el movimiento completo de cada eje, si bien puede ser de varios giros.

*Continúa en la página siguiente*

**Limitaciones**

Solo es posible leer los ángulos de motor actuales de las unidades mecánicas controladas desde la tarea de programa actual. En el caso de las tareas sin movimiento, es posible leer los ángulos de las unidades mecánicas controladas por la tarea de movimiento conectada.

**Más ejemplos**

El ejemplo siguiente ilustra la función `ReadMotor`.

**Ejemplo 1**

```
VAR num motor_angle;
motor_angle := ReadMotor(\MecUnit:=STN1, 1);
```

El ángulo actual del motor del primer eje del robot de `STN1` se almacena en `motor_angle`.

**Gestión de errores**

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                    | Causa del error                            |
|---------------------------|--|
| <code>ERR_AXIS_PAR</code> | Parámetro de eje incorrecto en la función. |

**Sintaxis**

```
ReadMotor '('
  ['\' MecUnit ':= ' < variable (VAR) of mecunit> ',']
  [Axis ':= ' ] < expression (IN) of num> ')'
```

Una función con un valor de retorno del tipo de dato `num`.

**Información relacionada**

| Para obtener más información sobre | Consulte   |
|------------------------------------|--|
| Lectura del ángulo actual del eje  | <a href="#">CJointT - Lee los ángulos actuales de los ejes en la página 1231</a> |

## 2 Funciones

---

2.152 ReadNum - Lee un número de un archivo o dispositivo de E/S  
RobotWare - OS

### 2.152 ReadNum - Lee un número de un archivo o dispositivo de E/S

---

#### Utilización

ReadNum (*Read Numeric*) se utiliza para leer un número de un archivo o un dispositivo de E/S alfanumérico.

#### Ejemplos básicos

El ejemplo siguiente ilustra la función ReadNum.  
Consulte también [Más ejemplos en la página 1443](#).

#### Ejemplo 1

```
VAR iodev infile;
...
Open "HOME:/file.doc", infile\Read;
reg1 := ReadNum(infile);
```

Se asigna a reg1 un número leído del archivo file.doc.

#### Valor de retorno

Tipo de dato: num

El valor numérico leído de un archivo o un dispositivo de E/S especificado. Si el archivo está vacío (se ha alcanzado el fin del archivo), se devuelve un número mayor de EOF\_NUM (9.998E36).

#### Argumentos

```
ReadNum (IODevice [\Delim] [\Time])
```

IODevice

Tipo de dato: iodev

El nombre (referencia) del archivo o dispositivo de E/S que debe leerse.

[\Delim]

#### Delimiters

Tipo de dato: string

Una cadena que contiene los delimitadores que deben utilizarse para interpretar las líneas del archivo o del dispositivo de E/S. De forma predeterminada (sin \Delim), el archivo se lee una línea cada vez y el único delimitador que se tiene en cuenta es el carácter de salto de línea (\0A). Cuando se utiliza el argumento \Delim, cualquier carácter del argumento de cadena especificado se utilizará para determinar qué parte de la línea es significativa.

Cuando se utiliza el argumento \Delim, el sistema de control añade siempre los caracteres de retorno de carro (\0D) y salto de línea (\0A) a los delimitadores especificados por el usuario.

Para especificar caracteres no alfanuméricos, utilice \xx, donde xx es la representación hexadecimal del código ASCII del carácter (por ejemplo: TAB se especifica mediante \09).

*Continúa en la página siguiente*

[\Time]

Tipo de dato: num

El tiempo máximo para la operación de lectura (tiempo límite) en segundos. Si no se especifica este argumento, el tiempo máximo es de 60 segundos. Para esperar ininterrumpidamente, utilice la constante predefinida `WAIT_MAX`.

Si se agota este tiempo antes de que se complete la operación de lectura, se llama al gestor de errores con el código de error `ERR_DEV_MAXTIME`. Si no hay ningún gestor de errores, se detiene la ejecución.

La función de tiempo límite se utiliza también durante un paro de programa y se notificará en el programa de RAPID al poner en marcha el programa.

#### Ejecución de programas

A partir de la posición actual del archivo, la función lee y desecha cualquier delimitador de encabezado. Un delimitador de encabezado que se usa sin el argumento `\Delim` es el carácter de salto de línea. Los delimitadores de encabezado con el argumento `\Delim` son cualquier carácter del argumento `\Delim` más los caracteres de retorno de carro y salto de línea. Por tanto, se lee todo lo que se encuentre hasta el siguiente carácter delimitador e incluyéndolo (el delimitador se desecha posteriormente), pero no más de 80 caracteres. Si la parte significativa tiene más de 80 caracteres, el resto de los caracteres se leen en la siguiente operación de lectura.

A continuación, la cadena leída se convierte en un valor numérico. Por ejemplo, "234,4" se convierte en el valor numérico 234,4.

En caso de un reinicio tras una caída de alimentación, todos los archivos o dispositivos de E/S abiertos del sistema se cierran y el descriptor de E/S de la variable del tipo `iodev` se restablece.

#### Más ejemplos

El ejemplo siguiente ilustra la función `ReadNum`.

##### Ejemplo 1

```
reg1 := ReadNum(infile\Delim:="\09");
IF reg1 > EOF_NUM THEN
  TPWrite "The file is empty";
  ...
```

Lee un número en una línea en la que los números están separados por caracteres TAB ("`\09`") o SPACE (" "). Antes de usar el número leído del archivo, se realiza una comprobación para asegurarse de que el archivo no esté vacío.



#### Nota

Utilice `< o >` (menor que o mayor que) al comprobar si el archivo está vacío. No utilice `=` (igual a).

#### Limitaciones

La función sólo puede usarse con archivos alfanuméricos que hayan sido abiertos para lectura.

*Continúa en la página siguiente*

## 2 Funciones

---

### 2.152 ReadNum - Lee un número de un archivo o dispositivo de E/S

RobotWare - OS

Continuación

---

#### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                       | Causa del error   |
|------------------------------|---|
| <code>ERR_FILEACC</code>     | Se produjo un error de acceso durante la lectura.           |
| <code>ERR_RCVDATA</code>     | Hay un intento de leer datos no numéricos.                  |
| <code>ERR_DEV_MAXTIME</code> | Tiempo límite antes de que termine la operación de lectura. |

---

#### Datos predefinidos

La constante `EOF_NUM` puede usarse para detener la lectura al final del archivo.

```
CONST num EOF_NUM := 9.998E36;
```

---

#### Sintaxis

```
ReadNum '('  
  [IODevice ':='] <variable (VAR) of iodev>  
  ['\ ' Delim ':=' <expression (IN) of string>  
  ['\ ' Time ':=' <expression (IN) of num>] ')'
```

Una función con un valor de retorno del tipo de dato `num`.

---

#### Información relacionada

| Para obtener más información sobre               | Consulte   |
|--|--|
| Apertura, etc. de archivos o dispositivos de E/S | <i>Manual de referencia técnica - RAPID Overview</i> |
| Gestión de archivos y dispositivos de E/S        | <i>Application manual - Controller software IRC5</i> |

---



---

## 2.153 ReadStr - Lee una cadena de un archivo o dispositivo de E/S

---

### Utilización

`ReadStr` (*Read String*) se utiliza para leer una cadena de un archivo o un dispositivo de E/S alfanumérico.

---

### Ejemplos básicos

El ejemplo siguiente ilustra la función `ReadStr`.

Consulte también [Más ejemplos en la página 1447](#).

#### Ejemplo 1

```
VAR string text;
VAR iodev infile;
...
Open "HOME:/file.doc", infile\Read;
text := ReadStr(infile);
```

Se asigna a `text` una cadena leída del archivo `file.doc`.

---

### Valor de retorno

Tipo de dato: `string`

La cadena de caracteres leída del archivo o dispositivo de E/S especificado. Si el archivo está vacío (se ha alcanzado el fin del archivo), se devuelve la cadena de caracteres "EOF".

---

### Argumentos

```
ReadStr (IODevice [\Delim] [\RemoveCR] [\DiscardHeaders] [\Time]
[\Line])
```

`IODevice`

Tipo de dato: `iodev`

El nombre (referencia) del archivo o dispositivo de E/S que debe leerse.

`[\Delim]`

#### *Delimiters*

Tipo de dato: `string`

Una cadena que contiene los delimitadores que deben utilizarse para interpretar las líneas del archivo o del dispositivo de E/S. De forma predeterminada, el archivo se lee una línea cada vez y el único delimitador que se tiene en cuenta es el carácter de salto de línea (`\0A`). Cuando se utiliza el argumento `\Delim`, cualquier carácter del argumento de cadena especificado, además del carácter predeterminado de salto de línea, se utilizará para determinar qué parte de la línea es significativa.

Para especificar caracteres no alfanuméricos, utilice `\xx`, donde `xx` es la representación hexadecimal del código ASCII del carácter (por ejemplo: TAB se especifica con `\09`).

`[\RemoveCR]`

Tipo de dato: `switch`

---

*Continúa en la página siguiente*

## 2 Funciones

---

### 2.153 ReadStr - Lee una cadena de un archivo o dispositivo de E/S

RobotWare - OS

Continuación

Un modificador utilizado para eliminar el retorno de carro final al leer archivos de PC. En los archivos de PC, los cambios de línea se especifican mediante un retorno de carro y un salto de línea (CRLF). Al leer una línea en este tipo de archivos, el carácter de retorno de carro se lee de forma predeterminada en la cadena de retorno. Cuando se utiliza este argumento, el carácter de retorno de carro se lee del archivo pero no se incluye en la cadena de caracteres devuelta.

[ \DiscardHeaders ]

Tipo de dato: switch

Este argumento especifica si los delimitadores de encabezado (especificados en \Delim más el salto de línea predeterminado) deben ser omitidos o no antes de transferir los datos a la cadena de caracteres de retorno. De forma predeterminada, si el primer carácter de la posición actual del archivo es un delimitador, se lee pero no se transfiere a la cadena de caracteres de retorno, se detiene la interpretación de la línea y el valor devuelto será una cadena de caracteres vacía. Si se utiliza este argumento, todos los delimitadores incluidos en la línea se leerán del archivo pero se desechan y no se realizará el retorno hasta que la cadena de caracteres devuelta contenga los datos a partir del primer carácter de la línea que no sea un delimitador.

[ \Time ]

Tipo de dato: num

El tiempo máximo para la operación de lectura (tiempo límite) en segundos. Si no se especifica este argumento, el tiempo máximo es de 60 segundos. Para esperar ininterrumpidamente, utilice la constante predefinida WAIT\_MAX.

Si se agota este tiempo antes de que se complete la operación de lectura, se llama al gestor de errores con el código de error ERR\_DEV\_MAXTIME. Si no hay ningún gestor de errores, se detiene la ejecución.

La función de tiempo límite se utiliza también durante un paro de programa y se observará en el programa de RAPID al poner en marcha el programa.

[ \Line ]

Tipo de dato: num

Especifica qué línea del archivo se debe leer.

Si la línea no existe, devuelve la cadena "EOF".

---

### Ejecución de programas

A partir de la posición actual del archivo, si se utiliza el argumento \DiscardHeaders, la función lee y desecha cualquier delimitador de encabezado (caracteres de salto de línea y cualquier carácter especificado en el argumento \Delim). En todos los casos, lee todo hasta el siguiente carácter delimitador, pero no más de 80 caracteres. Si la parte significativa tiene más de 80 caracteres, el resto de los caracteres se leen en la siguiente operación de lectura. El delimitador que causa la detención de la interpretación se lee del archivo pero no se transfiere a la cadena devuelta. Si el último carácter de la cadena es un carácter de retorno de carro y se utiliza el argumento \RemoveCR, este carácter se elimina de la cadena.

*Continúa en la página siguiente*

En caso de un reinicio tras una caída de alimentación, todos los archivos o dispositivos de E/S abiertos del sistema se cierran y el descriptor de E/S de la variable del tipo `iodev` se restablece.

### Más ejemplos

Los siguientes ejemplos ilustran la función `ReadStr`.

#### Ejemplo 1

```
text := ReadStr(infile);
IF text = EOF THEN
  TPWrite "The file is empty";
...

```

Antes de usar la cadena de caracteres leída del archivo, se realiza una comprobación para asegurarse de que el archivo no esté vacío.

#### Ejemplo 2

Por ejemplo, veamos un archivo que contiene:

```
<LF><SPACE><TAB>Hello<SPACE><SPACE>World<CR><LF>
text := ReadStr(infile);

```

`text` será una cadena vacía: el primer carácter del archivo es el delimitador predeterminado `<LF>`.

```
text := ReadStr(infile\DiscardHeaders);

```

`text` contendrá `<SPACE><TAB>Hello<SPACE><SPACE>World<CR>`: el primer carácter del archivo, el delimitador predeterminado `<LF>`, se desecha.

```
text := ReadStr(infile\RemoveCR\DiscardHeaders);

```

`text` contendrá `<SPACE><TAB>Hello<SPACE><SPACE>World:`. El primer carácter del archivo, el delimitador predeterminado `<LF>`, se desecha. El carácter de retorno de carro final se elimina

```
text := ReadStr(infile\Delim:=" \09"\RemoveCR\DiscardHeaders);

```

`text` contendrá `"Hello"`: los primeros caracteres del archivo que coincidan con el delimitador predeterminado `<LF>` o con el conjunto de caracteres definido por `\Delim` (espacio y tabulador) se desechan. Se transfieren los datos existentes hasta el primer delimitador leído del archivo, aunque éste último no se transfiere a la cadena. Una nueva ejecución de la misma sentencia devuelve `"World"`.

#### Ejemplo 3

Por ejemplo, veamos un archivo que contiene:

```
<CR><LF>Hello<CR><LF>
text := ReadStr(infile);

```

`text` contendrá el carácter `<CR>` (`\0d`): los caracteres `<CR>` y `<LF>` se leen del archivo, pero sólo se transfiere `<CR>` a la cadena. Una nueva ejecución de la misma sentencia devuelve `"Hello\0d"`.

```
text := ReadStr(infile\RemoveCR);

```

`text` será una cadena vacía: los caracteres `<CR>` y `<LF>` se leen del archivo; `<CR>` se transfiere pero se elimina de la cadena. Una nueva ejecución de la misma sentencia devuelve `"Hello"`.

```
text := ReadStr(infile\Delim:="\0d");

```

*Continúa en la página siguiente*

## 2 Funciones

### 2.153 ReadStr - Lee una cadena de un archivo o dispositivo de E/S

RobotWare - OS

Continuación

text será una cadena vacía: <CR> se lee del archivo pero no se transfiere a la cadena devuelta. Una nueva ejecución de la misma instrucción devuelve de nuevo una cadena vacía: <LF> se lee del archivo pero no se transfiere a la cadena devuelta.

```
text := ReadStr(infile\Delim:="\0d"\DiscardHeaders);
```

text contendrá "Hello". una nueva ejecución de la misma instrucción devuelve "EOF" (final del archivo).

#### Limitaciones

La función sólo puede usarse con archivos o dispositivos de E/S que se han abierto para lectura en modo alfanumérico.

#### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                       | Causa del error   |
|------------------------------|---|
| <code>ERR_FILEACC</code>     | Se produjo un error durante la lectura.                     |
| <code>ERR_DEV_MAXTIME</code> | Tiempo límite antes de que termine la operación de lectura. |

#### Datos predefinidos

La constante `EOF` puede usarse para comprobar si el archivo estaba vacío al intentar leer del archivo o para detener la lectura al final del archivo.

```
CONST string EOF := "EOF";
```

#### Sintaxis

```
ReadStr '('  
  [IODevice ':='] <variable (VAR) of iodev>  
  ['\' Delim ':='] <expression (IN) of string>  
  ['\' RemoveCR]  
  ['\' DiscardHeaders]  
  ['\' Time ':='] <expression (IN) of num>  
  ['\' Line ':='] <expression (IN) of num> ] )'
```

Una función con un valor de retorno del tipo de dato `string`.

#### Información relacionada

| Para obtener más información sobre               | Consulte   |
|--|--|
| Apertura, etc. de archivos o dispositivos de E/S | <i>Manual de referencia técnica - RAPID Overview</i> |
| Gestión de archivos y dispositivos de E/S        | <i>Application manual - Controller software IRC5</i> |

---

## 2.154 ReadStrBin - Lee una cadena de un dispositivo de E/S o archivo binario

---

### Utilización

`ReadStrBin` (*Read String Binary*) se utiliza para leer una cadena de un dispositivo de E/S o archivo binario.

---

### Ejemplos básicos

El ejemplo siguiente ilustra la función `ReadStrBin`.

#### Ejemplo 1

```
VAR iodev file1;
VAR string text;
...
Open "HOME:", \File:= "FILE1.DOC", file1\Bin;
text := ReadStrBin (file1, 10);
IF text = EOF THEN
```

Se asigna la variable `text` a una cadena de texto de 10 caracteres leída del archivo `file1`.

Antes de usar la cadena de caracteres leída del archivo, se realiza una comprobación para asegurarse de que el archivo no esté vacío.

---

### Valor de retorno

Tipo de dato: `string`

La cadena de texto leída del dispositivo de E/S o archivo especificado. Si el archivo está vacío (se ha alcanzado el fin del archivo), se devuelve la cadena de caracteres "EOF".

---

### Argumentos

```
ReadStrBin (IODevice NoOfChars [\Time])
```

`IODevice`

Tipo de dato: `iodev`

El nombre (la referencia) del dispositivo de E/S o el archivo binario del que se desea leer.

`NoOfChars`

*Number of Characters*

Tipo de dato: `num`

El número de caracteres a leer del dispositivo de E/S o del archivo binario.

`[\Time]`

Tipo de dato: `num`

El tiempo máximo para la operación de lectura (tiempo límite) en segundos. Si no se especifica este argumento, el tiempo máximo es de 60 segundos. Para esperar ininterrumpidamente, utilice la constante predefinida `WAIT_MAX`.

---

*Continúa en la página siguiente*

## 2 Funciones

### 2.154 ReadStrBin - Lee una cadena de un dispositivo de E/S o archivo binario

RobotWare - OS

Continuación

Si se agota este tiempo antes de que se complete la operación de lectura, se llama al gestor de errores con el código de error `ERR_DEV_MAXTIME`. Si no hay ningún gestor de errores, se detiene la ejecución.

La función de tiempo límite se utiliza también durante un paro de programa y se observará en el programa de RAPID al poner en marcha el programa.

#### Ejecución de programas

La función lee el número especificado de caracteres del dispositivo de E/S o del archivo binario.

En caso de un reinicio tras una caída de alimentación, todos los archivos o dispositivos de E/S abiertos del sistema se cierran y el descriptor de E/S de la variable del tipo `iodev` se restablece.

#### Limitaciones

La función sólo puede usarse con dispositivos de E/S o archivos que hayan sido abiertos para lectura en modo binario.

#### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                       | Causa del error   |
|------------------------------|---|
| <code>ERR_FILEACC</code>     | Se produjo un error durante la lectura.                     |
| <code>ERR_DEV_MAXTIME</code> | Tiempo límite antes de que termine la operación de lectura. |

#### Datos predefinidos

La constante `EOF` puede usarse para comprobar si el archivo estaba vacío al intentar leer del archivo o para detener la lectura al final del archivo.

```
CONST string EOF := "EOF";
```

#### Sintaxis

```
ReadStrBin '('  
  [IODevice ':='] <variable (VAR) of iodev> ','  
  [NoOfChars ':='] <expression (IN) of num>  
  ['\'' Time ':='] <expression (IN) of num> ] )'
```

Una función con un valor de retorno del tipo de dato `string`.

#### Información relacionada

| Para obtener más información sobre               | Consulte  |
|--|---|
| Apertura, etc. de dispositivos de E/S o archivos | <a href="#">Manual de referencia técnica - RAPID Overview</a>                                       |
| Escritura de cadenas binarias                    | <a href="#">WriteStrBin - Escribe una cadena en un dispositivo de E/S binario en la página 1114</a> |
| Gestión de archivos y dispositivos de E/S        | <a href="#">Application manual - Controller software IRC5</a>                                       |

## 2.155 ReadVar - Lee una variable de un dispositivo

### Utilización

ReadVar se utiliza para leer una variable de un dispositivo que está conectado a la interfaz de sensores.

### Ejemplo de configuración

Éste es un ejemplo de configuración de un canal de sensor.

Estos parámetros corresponden al tipo *Transmission Protocol* del tema *Communication*.

| Name  | Type    | Remote Address  | Remote Port |
|-------|---------|-----------------|-------------|
| sen1: | SOCKDEV | 192.168.125.101 | 6344        |

### Ejemplos básicos

El ejemplo siguiente ilustra la función ReadVar.

#### Ejemplo 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);

```

### Argumentos

```
ReadVar (device, VarNo, [\TaskName])
```

device

**Tipo de dato:** string

El nombre del dispositivo de E/S configurado en sio.cfg para el sensor utilizado.

VarNo

**Tipo de dato:** num

El argumento VarNo se utiliza para seleccionar la variable que se desea leer.

[ \TaskName ]

**Tipo de dato:** string

El argumento TaskName hace posible el acceso a dispositivos de otras tareas de RAPID.

*Continúa en la página siguiente*

## 2 Funciones

### 2.155 ReadVar - Lee una variable de un dispositivo

#### Sensor Interface

#### Continuación

#### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre       | Causa del error                        |
|--------------|--|
| SEN_NO_MEAS  | Fallo de medición                      |
| SEN_NOREADY  | Sensor incapaz de gestionar el comando |
| SEN_GENERRO  | Error general del sensor               |
| SEN_BUSY     | Sensor ocupado                         |
| SEN_UNKNOWN  | Sensor desconocido                     |
| SEN_EXALARM  | Error de sensor externo                |
| SEN_CAALARM  | Error de sensor interno                |
| SEN_TEMP     | Error de temperatura del sensor        |
| SEN_VALUE    | Valor de comunicación no válido        |
| SEN_CAMCHECK | Fallo de comprobación de sensor        |
| SEN_TIMEOUT  | Error de comunicación                  |

#### Sintaxis

```
ReadVar '('  
    [device ':='] <expression(IN) of string> ','  
    [VarNo ':='] <expression (IN) of num> ','  
    ['\ ' TaskName ':=' <expression (IN) of string> ] ')'
```

Una función con un valor de retorno del tipo de dato `num`.

#### Información relacionada

| Para obtener más información sobre                         | Consulte   |
|--|--|
| Establecimiento de una conexión a un dispositivo de sensor | <a href="#">SenDevice - Establece una conexión a un dispositivo de sensor en la página 687</a> |
| Escritura de una variable de sensor                        | <a href="#">WriteVar - Escribir una variable en la página 1116</a>                             |
| Escritura de un bloque de datos de sensor                  | <a href="#">WriteBlock - Escribir un bloque de datos en un dispositivo en la página 1105</a>   |
| Lectura de un bloque de datos de sensor                    | <a href="#">ReadBlock - Lee un bloque de datos de un dispositivo en la página 580</a>          |
| Configuración de la comunicación del sensor                | <a href="#">Manual de referencia técnica - RAPID Overview</a>                                  |



## 2.156 RelTool - Hace un desplazamiento respecto de la herramienta

### Utilización

RelTool (*Relative Tool*) se utiliza para añadir un desplazamiento y/o una rotación, expresada en el sistema de coordenadas de la herramienta activa, a una posición del robot.

### Ejemplos básicos

Los siguientes ejemplos ilustran la función RelTool.

#### Ejemplo 1

```
MoveL RelTool (p1, 0, 0, 100), v100, fine, tool1;
```

Se mueve el robot a una posición que se encuentra a 100 mm de p1 en la dirección de la herramienta.

#### Ejemplo 2

```
MoveL RelTool (p1, 0, 0, 0 \Rz:= 25), v100, fine, tool1;
```

Se gira la herramienta 25° alrededor de su eje z.

### Valor de retorno

Tipo de dato: robtarget

La nueva posición, con la adición de un desplazamiento y/o una rotación, si la hay, respecto de la herramienta activa.

### Argumentos

```
RelTool (Point Dx Dy Dz [\Rx] [\Ry] [\Rz])
```



#### Nota

Los giros se realizarán en el siguiente orden si se especifican dos o tres giros al mismo tiempo:

- 1 giro alrededor del eje x
- 2 giro alrededor del nuevo eje y
- 3 giro alrededor del nuevo eje z

Point

Tipo de dato: robtarget

La posición de entrada del robot. La parte de orientación de esta posición define la orientación actual del sistema de coordenadas de la herramienta.

Dx

Tipo de dato: num

El desplazamiento en mm en la dirección x del sistema de coordenadas de la herramienta.

Dy

Tipo de dato: num

*Continúa en la página siguiente*

## 2 Funciones

### 2.156 RelTool - Hace un desplazamiento respecto de la herramienta

RobotWare - OS

Continuación

El desplazamiento en mm en la dirección y del sistema de coordenadas de la herramienta.

Dz

Tipo de dato: num

El desplazamiento en mm en la dirección z del sistema de coordenadas de la herramienta.

[\Rx]

Tipo de dato: num

La rotación en grados alrededor del eje x del sistema de coordenadas de la herramienta.

[\Ry]

Tipo de dato: num

La rotación en grados alrededor del eje y del sistema de coordenadas de la herramienta.

[\Rz]

Tipo de dato: num

La rotación en grados alrededor del eje z del sistema de coordenadas de la herramienta.

### Sintaxis

```
RelTool '('  
  [ Point ':= ' ] < expression (IN) of rotarget> ', '  
  [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> ] ')'
```

Una función con un valor de retorno del tipo de dato rotarget.

### Información relacionada

| Para obtener más información sobre    | Consulte   |
|---------------------------------------|--|
| Datos de posición                     | <a href="#">rotarget - Datos de posición en la página 1750</a> |
| Instrucciones y funciones matemáticas | <i>Manual de referencia técnica - RAPID Overview</i>           |
| Instrucciones de posicionamiento      | <i>Manual de referencia técnica - RAPID Overview</i>           |

## 2.157 RemainingRetries - Reintentos restantes aún pendientes

### Utilización

`RemainingRetries` se usa para determinar cuántos `RETRY` quedan por hacer en el gestor de errores del programa. El número máximo de reintentos se define en la configuración.

### Ejemplos básicos

El ejemplo siguiente ilustra la función `RemainingRetries`.

#### Ejemplo 1

```

...
ERROR
  IF RemainingRetries() > 0 THEN
    RETRY;
  ELSE
    TRYNEXT;
  ENDF
...

```

Este programa reintentará la instrucción, a pesar de error, hasta alcanzar el número máximo de reintentos y a continuación intentará la instrucción siguiente.

### Valor de retorno

Tipo de dato: `num`

El valor de retorno indica cuántos reintentos quedan por hacer del número total de reintentos.

### Sintaxis

```
RemainingRetries '(' '')
```

Una función con un valor de retorno del tipo de dato `num`.

### Información relacionada

| Para obtener más información sobre              | Consulte   |
|---|--|
| Gestores de errores                             | <i>Manual de referencia técnica - RAPID Overview</i>                                   |
| Reanudación de la ejecución después de un error | <a href="#">RETRY - Reanudar la ejecución después de un error en la página 617</a>     |
| Configuración del número máximo de reintentos   | <i>Manual de referencia técnica - Parámetros del sistema, sección General RAPID</i>    |
| Restablece el número de reintentos contado      | <a href="#">ResetRetryCount - Restablecer el número de reintentos en la página 613</a> |

## 2 Funciones

---

### 2.158 RMQGetSlotName - Obtener el nombre de un cliente de RMQ *FlexPendant Interface, PC Interface, or Multitasking*

### 2.158 RMQGetSlotName - Obtener el nombre de un cliente de RMQ

---

#### Utilización

`RMQGetSlotName` (*RAPID Mesasage Queue Get Slot Name*) se utiliza para obtener el nombre de ranura de un RMQ cliente de Robot Application Builder a partir de una identidad de ranura determinada, es decir, desde un `rmqslot` determinado.

---

#### Ejemplos básicos

El ejemplo siguiente ilustra la función `RMQGetSlotName`.

#### Ejemplo 1

```
VAR rmqslot slot;
VAR string client_name;
RMQFindSlot slot, "RMQ_T_ROB1";
...
client_name := RMQGetSlotName(slot);
TPWrite "Name of the client: " + client_name;
```

Este ejemplo ilustra cómo obtener el nombre de un cliente utilizando la identidad del cliente.

---

#### Valor de retorno

Tipo de dato: `string`

Se devuelve el nombre del cliente. Puede tratarse de un nombre de RMQ o del nombre de un cliente de Robot Application Builder que utiliza la funcionalidad de RMQ.

---

#### Argumentos

`RMQGetSlotName (Slot)`

Slot

Tipo de dato: `rmqslot`

El número de ranura de identidad del cliente cuyo nombre se desea encontrar.

---

#### Ejecución de programas

La instrucción `RMQGetSlotName` se usa para encontrar el nombre del cliente que tiene el número de identidad especificado en el argumento `Slot`. El cliente puede ser otro RMQ o un cliente de SDK.

---

#### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                       | Causa del error  |
|------------------------------|--|
| <code>ERR_RMQ_INVALID</code> | La ranura de destino no ha sido conectada o la ranura de destino ya no está disponible. Si no hay conexión, debe realizarse una llamada a <code>RMQFindSlot</code> . Si no está disponible, el motivo es que un cliente remoto se ha desconectado del controlador. |

---

*Continúa en la página siguiente*

## 2.158 RMQGetSlotName - Obtener el nombre de un cliente de RMQ FlexPendant Interface, PC Interface, or Multitasking Continuación

### Sintaxis

```
RMQGetSlotName '('
  [ Slot ':=' ] < variable (VAR) of rmqslot > ')'
```

Una función con un valor de retorno del tipo de dato `string`.

### Información relacionada

| Para obtener más información sobre   | Consulte  |
|--|---|
| Descripción de la funcionalidad de RAPID Message Queue   | <i>Application manual - Controller software IRC5, sección RAPID Message Queue.</i>                        |
| Buscar el número de identidad de una tarea de RAPID Message Queue o un cliente de SDK                | <a href="#">RMQFindSlot - Buscar una identidad de ranura para el nombre de ranura en la página 624</a>    |
| Enviar datos a la cola de una tarea de RAPID o de un cliente de SDK                                  | <a href="#">RMQSendMessage - Enviar un mensaje de datos de RMQ en la página 638</a>                       |
| Obtener el primer mensaje de una cola de RAPID Message Queue   | <a href="#">RMQGetMessage - Obtener un mensaje de RMQ en la página 626</a>                                |
| Enviar datos a la cola de una tarea de RAPID o un cliente de SDK y esperar una respuesta del cliente | <a href="#">RMQSendWait - Enviar un mensaje de datos de RMQ y esperar una respuesta en la página 642</a>  |
| Extraer los datos de encabezado de un <code>rmqmessage</code>  | <a href="#">RMQGetMsgHeader - Obtener información de encabezado de un mensaje de RMQ en la página 632</a> |
| Extraer los datos de un <code>rmqmessage</code>  | <a href="#">RMQGetMsgData - Obtener la parte de datos de un mensaje de RMQ en la página 629</a>           |
| Ordenar y habilitar interrupciones para un tipo de dato en concreto                                  | <a href="#">IRMQMessage - Ordenar interrupciones de RMQ para un tipo de dato en la página 305</a>         |
| RMQ Slot   | <a href="#">rmqslot - Número de identidad de un cliente de RMQ en la página 1748</a>                      |

## 2 Funciones

---

### 2.159 RobName - Obtiene el nombre del robot del TCP

RobotWare - OS

### 2.159 RobName - Obtiene el nombre del robot del TCP

---

#### Utilización

RobName (*Robot Name*) se utiliza para obtener el nombre del robot de TCP desde una tarea de programa. Si la tarea no controla ningún robot de TCP, esta función devuelve una cadena vacía.

---

#### Ejemplos básicos

El ejemplo siguiente ilustra la función RobName.

Consulte también [Más ejemplos en la página 1458](#).

#### Ejemplo 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
```

Se escribe en el FlexPendant el nombre del robot de TCP controlado desde esta tarea de programa. Si no se controla ningún robot de TCP, se indica que la tarea no controla ningún robot.

---

#### Valor de retorno

Tipo de dato: string

El nombre de unidad mecánica del robot de TCP controlado desde esta tarea de programa. Devuelve una cadena vacía si no se está controlando ningún robot de TCP.

---

#### Más ejemplos

A continuación aparecen más ejemplos de cómo usar la instrucción RobName.

#### Ejemplo 1

```
VAR string my_robot;
...
IF TaskRunRob() THEN
    my_robot := RobName();
    TPWrite "This task controls robot with name "+ my_robot;
ENDIF
```

Si esta tarea de programa controla algún robot de TCP, se escribe en el FlexPendant el nombre del robot de TCP.

---

#### Sintaxis

```
RobName '(' ' ' )'
```

Una función con un valor de retorno del tipo de dato string.

---

*Continúa en la página siguiente*

#### Información relacionada

| Para obtener más información sobre                          | Consulte  |
|---|---|
| Comprobación de si la tarea controla algún robot de TCP     | <a href="#">TaskRunRob - Comprueba si una tarea controla algún robot en la página 1516</a>                  |
| Comprobación de si la tarea controla alguna unidad mecánica | <a href="#">TaskRunMec - Comprueba si una tarea controla alguna unidad mecánica en la página 1515</a>       |
| Obtención del nombre de las unidades mecánicas del sistema  | <a href="#">GetNextMechUnit - Obtener el nombre y los datos de las unidades mecánicas en la página 1306</a> |
| Funciones para cadenas de caracteres                        | <i>Manual de referencia técnica - Instrucciones, funciones y tipos de datos de RAPID</i>                    |
| Definición de cadena de caracteres                          | <a href="#">string - Cadenas en la página 1778</a>  |

## 2 Funciones

---

2.160 RobOS - Comprueba si el programa se está ejecutando en RC o VC  
*RobotWare - OS*

### 2.160 RobOS - Comprueba si el programa se está ejecutando en RC o VC

---

#### Utilización

RobOS (*Robot Operating System*) puede usarse para comprobar si la ejecución se realiza en Robot Controller RC o en Virtual Controller VC.

---

#### Ejemplos básicos

El ejemplo siguiente ilustra la función RobOS.

#### Ejemplo 1

```
IF RobOS() THEN
  ! Execution statements in RC
ELSE
  ! Execution statements in VC
ENDIF
```

---

#### Valor de retorno

Tipo de dato: `bool`

TRUE si la ejecución se está realizando en el controlador de robot RC. De lo contrario, FALSE.

---

#### Sintaxis

```
RobOS '(' ' ')
```

Una función con un valor de retorno del tipo de dato `bool`.



---

## 2.161 Round - Redondear un valor numérico

---

### Utilización

Round se utiliza para redondear un valor numérico con un número determinado de decimales o a un valor entero.

---

### Ejemplos básicos

Los siguientes ejemplos ilustran la función Round.

#### Ejemplo 1

```
VAR num val;  
val := Round(0.3852138\Dec:=3);
```

Se asigna a la variable val el valor 0.385.

#### Ejemplo 2

```
val := Round(0.3852138\Dec:=1);
```

Se asigna a la variable val el valor 0.4.

#### Ejemplo 3

```
val := Round(0.3852138);
```

Se asigna a la variable val el valor 0.

#### Ejemplo 4

```
val := Round(0.3852138\Dec:=6);
```

Se asigna a la variable val el valor 0.385214.

---

### Valor de retorno

Tipo de dato: num

El valor numérico redondeado con el número especificado de decimales.

---

### Argumentos

```
Round ( Val [\Dec])
```

Val

*Value*

Tipo de dato: num

El valor numérico a redondear.

[\Dec]

*Decimals*

Tipo de dato: num

Número de decimales.

Si el número de decimales especificado es 0 o se omite el argumento, el valor se redondea a un entero.

El número de decimales no debe ser negativo ni mayor que la precisión disponible para los valores numéricos.

El número máximo de decimales que pueden usarse es 6.

---

*Continúa en la página siguiente*

## 2 Funciones

---

### 2.161 Round - Redondear un valor numérico

RobotWare - OS

Continuación

---

#### Sintaxis

```
Round '('  
  [ Val ':=' ] <expression (IN) of num>  
  [ \Dec ':=' <expression (IN) of num> ] ')'
```

Una función con un valor de retorno del tipo de dato num.

---

#### Información relacionada

| Para obtener más información sobre    | Consulte   |
|---------------------------------------|--|
| Instrucciones y funciones matemáticas | <i>Manual de referencia técnica - RAPID Overview</i>               |
| Truncación de un valor                | <a href="#">Trunc - Trunca un valor numérico en la página 1539</a> |

---

## 2.162 RoundDnum - Redondear un valor numérico

---

### Utilización

RoundDnum se utiliza para redondear un valor numérico con un número determinado de decimales o a un valor entero.

---

### Ejemplos básicos

Los siguientes ejemplos ilustran la función RoundDnum.

#### Ejemplo 1

```
VAR dnum val;  
val := RoundDnum(0.3852138754655357\Dec:=3);
```

Se asigna a la variable `val` el valor 0.385.

#### Ejemplo 2

```
val := RoundDnum(0.3852138754655357\Dec:=1);
```

Se asigna a la variable `val` el valor 0.4.

#### Ejemplo 3

```
val := RoundDnum(0.3852138754655357);
```

Se asigna a la variable `val` el valor 0.

#### Ejemplo 4

```
val := RoundDnum(0.3852138754655357\Dec:=15);
```

Se asigna a la variable `val` el valor 0.385213875465536.

#### Ejemplo 5

```
val := RoundDnum(1000.3852138754655357\Dec:=15);
```

Se asigna a la variable `val` el valor 1000.38521387547.

---

### Valor de retorno

Tipo de dato: `dnum`

El valor numérico redondeado con el número especificado de decimales.

---

### Argumentos

```
RoundDnum ( Val [\Dec] )
```

`Val`

*Value*

Tipo de dato: `dnum`

El valor numérico a redondear.

`[\Dec]`

*Decimals*

Tipo de dato: `num`

Número de decimales.

Si el número de decimales especificado es 0 o se omite el argumento, el valor se redondea a un entero.

---

*Continúa en la página siguiente*

## 2 Funciones

---

### 2.162 RoundDnum - Redondear un valor numérico

RobotWare - OS

Continuación

El número de decimales no debe ser negativo ni mayor que la precisión disponible para los valores numéricos.

El número máximo de decimales que pueden usarse es 15.

---

#### Sintaxis

```
RoundDnum '('  
  [ Val ':=' ] <expression (IN) of dnum>  
  [ \Dec ':=' <expression (IN) of num> ] ')'
```

Una función con un valor de retorno del tipo de dato dnum.

---

#### Información relacionada

| Para obtener más información sobre    | Consulte   |
|---------------------------------------|--|
| Instrucciones y funciones matemáticas | <i>Manual de referencia técnica - RAPID Overview</i>                   |
| Redondeo de un valor                  | <a href="#">Round - Redondear un valor numérico en la página 1461</a>  |
| Truncación de un valor                | <a href="#">Trunc - Trunca un valor numérico en la página 1539</a>     |
| Truncación de un valor                | <a href="#">TruncDnum - Trunca un valor numérico en la página 1541</a> |

## 2.163 RunMode - Obtiene el modo de ejecución

### Utilización

RunMode(*Running Mode*) se usa para leer el modo de ejecución actual de la tarea del programa.

### Ejemplos básicos

El ejemplo siguiente ilustra la función RunMode.

#### Ejemplo 1

```
IF RunMode() = RUN_CONT_CYCLE THEN
...
ENDIF
```

La sección de programa sólo se ejecuta para un funcionamiento continuado o cíclico.

### Valor de retorno

Tipo de dato: symnum

El modo de ejecución actual, con uno de los valores definidos como se describe en la tabla siguiente.

| Valor de retorno | Constante simbólica | Comentario   |
|------------------|---------------------|--|
| 0                | RUN_UNDEF           | Modo de ejecución no definido  |
| 1                | RUN_CONT_CYCLE      | Modo de ejecución continuo o en modo ciclo   |
| 2                | RUN_INSTR_FWD       | Modo de ejecución de avance de instrucciones   |
| 3                | RUN_INSTR_BWD       | Modo de ejecución hacia atrás  |
| 4                | RUN_SIM             | Modo de ejecución simulado. Aún no publicado.  |
| 5                | RUN_STEP_MOVE       | Instrucciones de movimiento en ejecución hacia delante e instrucciones lógicas en modo de ejecución continuo |

### Argumentos

```
RunMode ( [ \Main ] )
```

[ \Main ]

Tipo de dato: switch

Devuelve el modo actual de la tarea si ésta es una tarea de movimiento. Si se usa en una tarea sin movimiento, devuelve el modo actual de la tarea de movimiento a la que está conectada la tarea sin movimiento.

Si se omite el argumento, el valor de retorno siempre refleja el modo de ejecución opuesto de la tarea de programa que ejecuta la función RunMode.

### Sintaxis

```
RunMode '('
[ '\ Main ] ')'
```

Una función con un valor de retorno del tipo de dato symnum.

*Continúa en la página siguiente*

## 2 Funciones

---

2.163 RunMode - Obtiene el modo de ejecución

RobotWare - OS

Continuación

---

### Información relacionada

| Para obtener más información sobre | Consulte   |
|------------------------------------|--|
| Lectura del modo de funcionamiento | <a href="#">OpMode - Lee el modo de funcionamiento en la página 1393</a> |

2.164 SafetyControllerGetChecksum - Obtener la suma de comprobación del archivo de configuración de usuarios

*SafeMove Basic, SafeMove Pro, PROFIsafe*

## 2.164 SafetyControllerGetChecksum - Obtener la suma de comprobación del archivo de configuración de usuarios

### Utilización

`SafetyControllerGetChecksum` se utiliza para obtener la suma de comprobación del Safety Controller para el archivo de configuración de usuarios.

### Ejemplos básicos

El ejemplo siguiente ilustra la función `SafetyControllerGetChecksum`.

#### Ejemplo 1

```
VAR string mystring;
...
mystring:=SafetyControllerGetChecksum();
```

Obtener la suma de comprobación para el archivo de configuración de usuarios y almacenarlo en la variable `mystring`.

### Valor de retorno

Tipo de dato: `string`

La suma de comprobación para la configuración de usuarios.

### Sintaxis

```
SafetyControllerGetChecksum '(' '')
```

Una función con un valor de retorno del tipo de dato `string`.

### Información relacionada

| Para obtener más información sobre            | Consulte   |
|---|--|
| <code>SafetyControllerGetOpModePinCode</code> | <a href="#">SafetyControllerGetOpModePinCode - Obtener el código pin del modo de funcionamiento en la página 1468</a>            |
| <code>SafetyControllerGetUserChecksum</code>  | <a href="#">SafetyControllerGetUserChecksum - Obtener la suma de comprobación de los parámetros protegidos en la página 1470</a> |
| <code>SafetyControllerGetSWVersion</code>     | <a href="#">SafetyControllerGetSWVersion - Obtener la versión del firmware del Safety Controller en la página 1469</a>           |
| <code>SafetyControllerSyncRequest</code>      | <a href="#">SafetyControllerSyncRequest - Inicio del procedimiento de sincronización del hardware en la página 647</a>           |

## 2 Funciones

---

2.165 SafetyControllerGetOpModePinCode - Obtener el código pin del modo de funcionamiento  
*SafeMove Basic, SafeMove Pro, PROFIsafe*

### 2.165 SafetyControllerGetOpModePinCode - Obtener el código pin del modo de funcionamiento

---

#### Utilización

`SafetyControllerGetOpModePinCode` es utilizado para obtener el código PIN del modo de operación para el selector de modo sin llave.

---

#### Ejemplos básicos

El ejemplo siguiente ilustra la función `SafetyControllerGetOpModePinCode`.

#### Ejemplo 1

```
VAR string mystring;  
...  
mystring:=SafetyControllerGetOpModePinCode();
```

Obtener el código pin del modo de funcionamiento para el selector de modo sin llave y guardarlo en la variable `mystring`.

---

#### Valor de retorno

Tipo de dato: `string`

El código pin para el selector de modo sin llave.

---

#### Sintaxis

```
SafetyControllerGetOpModePinCode '(' '')
```

Una función con un valor de retorno del tipo de dato `string`.

---

#### Información relacionada

| Para obtener más información sobre           | Consulte   |
|--|--|
| <code>SafetyControllerGetChecksum</code>     | <a href="#">SafetyControllerGetChecksum - Obtener la suma de comprobación del archivo de configuración de usuarios en la página 1467</a> |
| <code>SafetyControllerGetUserChecksum</code> | <a href="#">SafetyControllerGetUserChecksum - Obtener la suma de comprobación de los parámetros protegidos en la página 1470</a>         |
| <code>SafetyControllerGetSWVersion</code>    | <a href="#">SafetyControllerGetSWVersion - Obtener la versión del firmware del Safety Controller en la página 1469</a>                   |
| <code>SafetyControllerSyncRequest</code>     | <a href="#">SafetyControllerSyncRequest - Inicio del procedimiento de sincronización del hardware en la página 647</a>                   |



## 2.166 SafetyControllerGetSWVersion - Obtener la versión del firmware del Safety Controller *SafeMove Basic, SafeMove Pro, PROFIsafe*

### 2.166 SafetyControllerGetSWVersion - Obtener la versión del firmware del Safety Controller

#### Utilización

`SafetyControllerGetSWVersion` se utiliza para obtener la versión de firmware del Safety Controller.

#### Ejemplos básicos

El ejemplo siguiente ilustra la función `SafetyControllerGetSWVersion`.

#### Ejemplo 1

```
VAR string mystring;
...
mystring:=SafetyControllerGetSWVersion();
```

Obtener la versión de firmware del Safety Controller y almacenarlo en la variable `mystring`.

#### Valor de retorno

Tipo de dato: `string`

La versión de firmware del Safety Controller. Se devuelve una cadena con "VC" si esta función se utiliza en el controlador virtual.

#### Sintaxis

```
SafetyControllerGetSWVersion '(' '')
```

Una función con un valor de retorno del tipo de dato `string`.

#### Información relacionada

| Para obtener más información sobre            | Consulte   |
|---|--|
| <code>SafetyControllerGetOpModePinCode</code> | <a href="#">SafetyControllerGetOpModePinCode - Obtener el código pin del modo de funcionamiento en la página 1468</a>            |
| <code>SafetyControllerGetUserChecksum</code>  | <a href="#">SafetyControllerGetUserChecksum - Obtener la suma de comprobación de los parámetros protegidos en la página 1470</a> |
| <code>SafetyControllerGetSWVersion</code>     | <a href="#">SafetyControllerGetSWVersion - Obtener la versión del firmware del Safety Controller en la página 1469</a>           |
| <code>SafetyControllerSyncRequest</code>      | <a href="#">SafetyControllerSyncRequest - Inicio del procedimiento de sincronización del hardware en la página 647</a>           |

## 2 Funciones

2.167 `SafetyControllerGetUserChecksum` - Obtener la suma de comprobación de los parámetros protegidos

*SafeMove Basic, SafeMove Pro, PROFIsafe*

### 2.167 `SafetyControllerGetUserChecksum` - Obtener la suma de comprobación de los parámetros protegidos

#### Utilización

`SafetyControllerGetUserChecksum` se utiliza para obtener la suma de comprobación del Safety Controller para el área con parámetros protegidos en el archivo de configuración de usuarios.

#### Ejemplos básicos

El ejemplo siguiente ilustra la función `SafetyControllerGetUserChecksum`.

##### Ejemplo 1

```
VAR string mystring;  
...  
mystring:=SafetyControllerGetUserChecksum();
```

Obtener la suma de comprobación para el área con parámetros protegidos en el archivo de configuración de usuarios y almacenarlo en la variable `mystring`.

#### Valor de retorno

Tipo de dato: `string`

La suma de comprobación para el área con parámetros protegidos.

#### Sintaxis

```
SafetyControllerGetUserChecksum '(' '')
```

Una función con un valor de retorno del tipo de dato `string`.

#### Información relacionada

| Para obtener más información sobre            | Consulte   |
|---|--|
| <code>SafetyControllerGetOpModePinCode</code> | <a href="#">SafetyControllerGetOpModePinCode - Obtener el código pin del modo de funcionamiento en la página 1468</a>                    |
| <code>SafetyControllerGetChecksum</code>      | <a href="#">SafetyControllerGetChecksum - Obtener la suma de comprobación del archivo de configuración de usuarios en la página 1467</a> |
| <code>SafetyControllerGetSWVersion</code>     | <a href="#">SafetyControllerGetSWVersion - Obtener la versión del firmware del Safety Controller en la página 1469</a>                   |
| <code>SafetyControllerSyncRequest</code>      | <a href="#">SafetyControllerSyncRequest - Inicio del procedimiento de sincronización del hardware en la página 647</a>                   |

---

## 2.168 Sin - Calcula el valor del seno

---

### Utilización

`Sin(Sine)` se utiliza para calcular el valor de seno de un valor de ángulo.

---

### Ejemplos básicos

El ejemplo siguiente ilustra la función `Sin`.

#### Ejemplo 1

```
VAR num angle;  
VAR num value;  
...  
...  
value := Sin(angle);  
value obtiene el valor de seno de angle.
```

---

### Valor de retorno

**Tipo de dato:** num

El valor del seno en el rango [-1, 1].

---

### Argumentos

`Sin (Angle)`

Angle

**Tipo de dato:** num

El valor del ángulo, expresado en grados.

---

### Sintaxis

```
Sin '('  
  [Angle ':=' ] <expression (IN) of num> ')'  
Una función con un valor de retorno del tipo de dato num.
```

---

### Información relacionada

| Para obtener más información sobre    | Consulte   |
|---------------------------------------|--|
| Instrucciones y funciones matemáticas | <i>Manual de referencia técnica - RAPID Overview</i> |

## 2 Funciones

---

### 2.169 SinDnum - Calcula el valor del seno

RobotWare - OS

### 2.169 SinDnum - Calcula el valor del seno

---

#### Utilización

`SinDnum` (*Sine dnum*) se utiliza para calcular el valor de seno de un valor de ángulo en los tipos de datos `dnum`.

---

#### Ejemplos básicos

El ejemplo siguiente ilustra la función `SinDnum`.

#### Ejemplo 1

```
VAR dnum angle;  
VAR dnum value;  
...  
...  
value := SinDnum(angle);  
value obtiene el valor de seno de angle.
```

---

#### Valor de retorno

Tipo de dato: `dnum`

El valor del seno en el rango [-1, 1].

---

#### Argumentos

`SinDnum` (`Angle`)

`Angle`

Tipo de dato: `dnum`

El valor del ángulo, expresado en grados.

---

#### Sintaxis

```
SinDnum '('  
  [Angle ':=' ] <expression (IN) of dnum> ')'
```

Una función con un valor de retorno del tipo de dato `dnum`.

---

#### Información relacionada

| Para obtener más información sobre    | Consulte   |
|---------------------------------------|--|
| Instrucciones y funciones matemáticas | <i>Manual de referencia técnica - RAPID Overview</i> |

---

#### 2.170 SocketGetStatus - Obtiene el estado actual de un zócalo

---

##### Utilización

SocketGetStatus devuelve el estado actual de un zócalo.

---

##### Ejemplos básicos

El ejemplo siguiente ilustra la función SocketGetStatus.

Consulte también [Más ejemplos en la página 1473](#).

##### Ejemplo 1

```
VAR socketdev socket1;
VAR socketstatus state;
...
SocketCreate socket1;
state := SocketGetStatus( socket1 );
```

El estado de zócalo SOCKET\_CREATED se almacena en la variable state.

---

##### Valor de retorno

Tipo de dato: socketstatus

El estado actual del zócalo.

Sólo es posible usar constantes simbólicas predefinidas del tipo socketstatus para comprobar el estado.

---

##### Argumentos

```
SocketGetStatus( Socket )
```

Socket

Tipo de dato: socketdev

La variable de zócalo cuyo estado se desea averiguar.

---

##### Ejecución de programas

Esta función devuelve uno de los estados de zócalo predefinidos siguientes:

SOCKET\_CREATED, SOCKET\_CONNECTED, SOCKET\_BOUND, SOCKET\_LISTENING o  
SOCKET\_CLOSED.

---

##### Más ejemplos

El ejemplo siguiente ilustra la función SocketGetStatus.

##### Ejemplo 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
```

*Continúa en la página siguiente*

## 2 Funciones

### 2.170 SocketGetStatus - Obtiene el estado actual de un zócalo

#### Socket Messaging

#### Continuación

```
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
```

Se crea un zócalo cliente, que se conecta a un ordenador remoto. Antes de que el zócalo se utilice en una instrucción `SocketSend`, se comprueba el estado del zócalo para verificar que aún está conectado.

#### Limitaciones

El estado de un zócalo sólo puede ser cambiado mediante la ejecución de una instrucción de zócalo de RAPID. Por ejemplo, si se conecta el zócalo pero la conexión se interrumpe, este hecho no será indicado por la función `SocketGetStatus`. En su lugar, se devolverá un error cuando se utiliza el zócalo en una instrucción `SocketSend` o `SocketReceive`.

#### Sintaxis

```
SocketGetStatus '('
    [Socket ':='] <variable (VAR) of socketdev>')'
```

Una función con un valor de retorno del tipo de dato `socketstatus`.

#### Información relacionada

| Para obtener más información sobre                         | Consulte   |
|--|--|
| Comunicación con zócalos en general                        | <a href="#">Application manual - Controller software IRC5</a>                                  |
| Creación de un nuevo zócalo                                | <a href="#">SocketCreate - Crea un nuevo zócalo en la página 747</a>                           |
| Conexión a un ordenador remoto (sólo cliente)              | <a href="#">SocketConnect - Establece una conexión a un ordenador remoto en la página 744</a>  |
| Envío de datos a un ordenador remoto                       | <a href="#">SocketSend - Envía datos a un ordenador remoto en la página 761</a>                |
| Recepción de datos desde un ordenador remoto               | <a href="#">SocketReceive - Recibe datos de un ordenador remoto en la página 751</a>           |
| Cierre del zócalo  | <a href="#">SocketClose - Cerrar un zócalo en la página 742</a>                                |
| Enlazamiento de un zócalo (sólo servidor)                  | <a href="#">SocketBind - Enlazar un zócalo a mi dirección IP y puerto en la página 739</a>     |
| Cómo permanecer a la escucha de conexiones (sólo servidor) | <a href="#">SocketListen - Permanece a la escucha de conexiones entrantes en la página 749</a> |

Continúa en la página siguiente

### 2.170 SocketGetStatus - Obtiene el estado actual de un zócalo

*Socket Messaging*

*Continuación*

| Para obtener más información sobre       | Consulte  |
|--|---|
| Aceptación de conexiones (sólo servidor) | <a href="#">SocketAccept - Aceptar una conexión entrante en la página 736</a> |

## 2 Funciones

---

### 2.171 SocketPeek - Prueba para comprobar la presencia de datos en un zócalo

#### Socket Messaging

### 2.171 SocketPeek - Prueba para comprobar la presencia de datos en un zócalo

---

#### Utilización

SocketPeek se utiliza para comprobar la presencia de datos en un zócalo. Devuelve el número de bytes que pueden recibirse en el zócalo especificado.

---

#### Ejemplos básicos

El ejemplo siguiente ilustra la función SocketPeek.

#### Ejemplo 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
```

Primero se crea un zócalo de servidor, que se enlaza al puerto 1025 en la dirección de la red del controlador 192.168.0.1. A continuación, SocketPeek se utiliza para comprobar si hay 64 bytes de datos disponibles para recibirlos en el zócalo.

---

#### Valor de retorno

Tipo de dato: num

El número de bytes disponibles en un zócalo específico.

---

#### Argumentos

SocketPeek ( Socket )

Socket

Tipo de dato: socketdev

La variable de zócalo que se desea ver.

---

#### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema ERRNO cambiará a:

| Nombre             | Causa del error   |
|--------------------|---|
| ERR_SOCK_CLOSED    | El zócalo se cierra. Conexión interrumpida.   |
| ERR_SOCK_NOT_BOUND | El zócalo no ha sido vinculado a una dirección. Cuando se utiliza el tipo del protocolo de datagramas UDP/IP. |

Continúa en la página siguiente

---



## 2.171 SocketPeek - Prueba para comprobar la presencia de datos en un zócalo

Socket Messaging

Continuación

| Nombre              | Causa del error             |
|---------------------|-----------------------------|
| ERR_SOCKET_NOT_CONN | El zócalo no está conectado |

## Limitaciones

Todos los zócalos están cerrados tras el reinicio de la caída de alimentación. Este problema puede ser gestionado por la recuperación en caso de error.

El tamaño máximo de los datos que pueden recibirse en una llamada está limitada a 1024 bytes. Por lo tanto, el valor máximo que puede devolverse de `SocketPeek` es 1024.

## Sintaxis

```
SocketPeek '('
  [Socket ':='] <variable (VAR) of socketdev>')'
```

Una función con un valor de retorno del tipo de dato `num`.

## Información relacionada

| Para obtener más información sobre                         | Consulte  |
|--|---|
| Comunicación con zócalos en general                        | <i>Application manual - Controller software IRC5</i>  |
| Creación de un nuevo zócalo                                | <a href="#">SocketCreate - Crea un nuevo zócalo en la página 747</a>                              |
| Conexión a un ordenador remoto (sólo cliente)              | <a href="#">SocketConnect - Establece una conexión a un ordenador remoto en la página 744</a>     |
| Envío de datos a un ordenador remoto                       | <a href="#">SocketSend - Envía datos a un ordenador remoto en la página 761</a>                   |
| Envío de datos a un ordenador remoto                       | <a href="#">SocketSendTo - Envío de datos a un ordenador remoto en la página 765</a>              |
| Cierre del zócalo  | <a href="#">SocketClose - Cerrar un zócalo en la página 742</a>                                   |
| Enlazamiento de un zócalo (sólo servidor)                  | <a href="#">SocketBind - Enlazar un zócalo a mi dirección IP y puerto en la página 739</a>        |
| Cómo permanecer a la escucha de conexiones (sólo servidor) | <a href="#">SocketListen - Permanece a la escucha de conexiones entrantes en la página 749</a>    |
| Aceptación de conexiones (sólo servidor)                   | <a href="#">SocketAccept - Aceptar una conexión entrante en la página 736</a>                     |
| Obtención del estado actual del zócalo                     | <a href="#">SocketGetStatus - Obtiene el estado actual de un zócalo en la página 1473</a>         |
| Aplicación de ejemplo de zócalos de cliente                | <a href="#">SocketSend - Envía datos a un ordenador remoto en la página 761</a>                   |
| Recepción de datos desde un ordenador remoto               | <a href="#">SocketReceive - Recibe datos de un ordenador remoto en la página 751</a>              |
| Recepción de datos desde un ordenador remoto               | <a href="#">SocketReceiveFrom - Recepción de datos desde un ordenador remoto en la página 756</a> |

## 2 Funciones

---

### 2.172 Sqrt - Calcula el valor de la raíz cuadrada

RobotWare - OS

### 2.172 Sqrt - Calcula el valor de la raíz cuadrada

---

#### Utilización

`Sqrt` (*Square root*) se utiliza para calcular la raíz cuadrada.

---

#### Ejemplos básicos

El ejemplo siguiente ilustra la función `Sqrt`.

##### Ejemplo 1

```
VAR num x_value;  
VAR num y_value;  
...  
...  
y_value := Sqrt( x_value);
```

y-value **obtiene el valor de la raíz cuadrada de x\_value, es decir**  $\sqrt{x\_value}$ .

---

#### Valor de retorno

Tipo de dato: num  
El valor de la raíz cuadrada ( $\sqrt{\quad}$ ).

---

#### Argumentos

`Sqrt (Value)`

Value

Tipo de dato: num  
El valor del argumento de la raíz cuadrada, es decir,  $\sqrt{\text{value}}$ .  
Value **debe ser**  $\geq 0$ .

---

#### Limitaciones

La ejecución de la función `Sqrt(x)` genera un error si  $x < 0$ .

---

#### Sintaxis

```
Sqrt '('  
    [Value ':=' ] <expression (IN) of num> ')'
```

Una función con un valor de retorno del tipo de dato num.

---

#### Información relacionada

| Para obtener más información sobre                           | Consulte  |
|--|---|
| Calcular el valor de raíz cuadrada de un valor numérico dnum | <a href="#">SqrtDnum - Calcula el valor de la raíz cuadrada en la página 1479</a> |
| Instrucciones y funciones matemáticas                        | <i>Manual de referencia técnica - RAPID Overview</i>                              |

## 2.173 SqrtDnum - Calcula el valor de la raíz cuadrada

### Utilización

`SqrtDnum` (*Square root dnum*) se utiliza para calcular la raíz cuadrada.

### Ejemplos básicos

El ejemplo siguiente ilustra la función `SqrtDnum`.

#### Ejemplo 1

```

VAR dnum x_value;
VAR dnum y_value;
...
...
y_value := SqrtDnum(x_value);

```

`y_value` obtiene el valor de la raíz cuadrada de `x_value`, es decir  $\sqrt{x\_value}$ .

### Valor de retorno

Tipo de dato: `dnum`

El valor de la raíz cuadrada ( $\sqrt{\quad}$ ).

### Argumentos

`SqrtDnum (Value)`

#### Value

Tipo de dato: `dnum`

El valor del argumento de la raíz cuadrada, es decir,  $\sqrt{\text{value}}$ .

`Value` debe ser  $\geq 0$ .

### Limitaciones

La ejecución de la función `Sqrt(x)` genera un error si  $x < 0$ .

### Sintaxis

```

SqrtDnum '('
  [ Value ':' '=' ] < expression (IN) of dnum > ')'

```

Una función con un valor de retorno del tipo de dato `dnum`.

### Información relacionada

| Para obtener más información sobre                                       | Consulte  |
|--|---|
| Calcular el valor de raíz cuadrada de un valor numérico <code>num</code> | <a href="#">Sqrt - Calcula el valor de la raíz cuadrada en la página 1478</a> |
| Instrucciones y funciones matemáticas                                    | <i>Manual de referencia técnica - RAPID Overview</i>                          |

## 2 Funciones

---

### 2.174 STCalcForce - Calcula fuerza de la punta de una herramienta servo

*Servo tool control*

### 2.174 STCalcForce - Calcula fuerza de la punta de una herramienta servo

---

#### Utilización

`STCalcForce` se utiliza para calcular la fuerza de la punta de una herramienta servo. Por ejemplo, esta función se utiliza para determinar la fuerza de punta máxima permitida con una herramienta servo.

---

#### Ejemplos básicos

El ejemplo siguiente ilustra la función `STCalcForce`.

#### Ejemplo 1

```
VAR num tip_force;  
tip_force := STCalcForce(gun1, 7);
```

Se calcula la fuerza de la punta con un par motor deseado de 7 Nm.

---

#### Valor de retorno

Tipo de dato: num

La fuerza de punta calculada [N].

---

#### Argumentos

```
STCalcForce ToolName MotorTorque
```

ToolName

Tipo de dato: string

El nombre de la unidad mecánica.

MotorTorque

Tipo de dato: num

El par motor deseado [Nm].

---

#### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre      | Causa del error   |
|-------------|---|
| ERR_NO_SGUN | El nombre especificado de la herramienta servo no es una herramienta servo configurada. |

---

#### Sintaxis

```
STCalcForce '('  
  [ ToolName ::= ' ] < expression (IN) of string > ','  
  [ MotorTorque ::= ' ] < expression (IN) of num > ';'
```

Una función con un valor de retorno del tipo de dato num.

---

*Continúa en la página siguiente*

2.174 STCalcForce - Calcula fuerza de la punta de una herramienta servo  
*Servo tool control*  
*Continuación*

### Información relacionada

| Para obtener más información sobre | Consulte   |
|------------------------------------|--|
| Apertura de una herramienta servo  | <a href="#">STOpen - Abre una herramienta servo en la página 825</a>                           |
| Cierre de una herramienta servo    | <a href="#">STClose - Cierra una herramienta servo en la página 807</a>                        |
| Cálculo del par motor              | <a href="#">STCalcTorque - Calcula el par motor de una herramienta servo en la página 1482</a> |

## 2 Funciones

### 2.175 STCalcTorque - Calcula el par motor de una herramienta servo

*Servo tool control*

### 2.175 STCalcTorque - Calcula el par motor de una herramienta servo

#### Utilización

STCalcTorque se utiliza para calcular el par motor de una herramienta servo. Por ejemplo, esta función se utiliza cuando se realiza una calibración de la fuerza.

#### Ejemplos básicos

El ejemplo siguiente ilustra la función STCalcTorque.

#### Ejemplo 1

```
VAR num curr_motortorque;  
curr_motortorque := STCalcTorque( gun1, 1000);
```

Se calcula el par motor con una fuerza de la punta deseada de 1000 Nm.

#### Valor de retorno

Tipo de dato: num  
El par motor calculado [Nm].

#### Argumentos

```
STCalcTorque ToolName TipForce
```

ToolName

Tipo de dato: string  
El nombre de la unidad mecánica.

TipForce

Tipo de dato: num  
La fuerza de punta deseada [N].

#### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema ERRNO cambiará a:

| Nombre      | Causa del error   |
|-------------|---|
| ERR_NO_SGUN | El nombre especificado de la herramienta servo no es una herramienta servo configurada. |

#### Sintaxis

```
STCalcTorque '('  
  [ ToolName ':= ' ] < expression (IN) of string > ','  
  [ TipForce ':= ' ] < expression (IN) of num > ';' )
```

Una función con un valor de retorno del tipo de dato num.

#### Información relacionada

| Para obtener más información sobre | Consulte   |
|------------------------------------|--|
| Apertura de una herramienta servo  | <a href="#">STOpen - Abre una herramienta servo en la página 825</a> |

*Continúa en la página siguiente*

### 2.175 STCalcTorque - Calcula el par motor de una herramienta servo

*Servo tool control*

*Continuación*

| Para obtener más información sobre | Consulte   |
|------------------------------------|--|
| Cierre de una herramienta servo    | <a href="#"><i>STClose - Cierra una herramienta servo en la página 807</i></a>                             |
| Cálculo de la fuerza de la punta   | <a href="#"><i>STCalcForce - Calcula fuerza de la punta de una herramienta servo en la página 1480</i></a> |

## 2 Funciones

---

### 2.176 STIsCalib - Compruebe si una herramienta servo está calibrada *Servo Tool Control*

### 2.176 STIsCalib - Compruebe si una herramienta servo está calibrada

---

#### Utilización

STIsCalib se utiliza para comprobar si una herramienta servo está calibrada, es decir, comprobar si las puntas de la pistola están calibradas o sincronizadas.

---

#### Ejemplos básicos

Los siguientes ejemplos ilustran la función STIsCalib.

##### Ejemplo 1

```
IF STIsCalib(gun1\sguninit) THEN
  ...
ELSE
  !Start the gun calibration
  STCalib gun1\TipChg;
ENDIF
```

##### Ejemplo 2

```
IF STIsCalib(gun1\sgunsynch) THEN
  ...
ELSE
  !Start the gun calibration to synchronize the gun position with
  the revolution counter
  STCalib gun1\ToolChg;
ENDIF
```

#### Valor de retorno

**Tipo de dato:** bool

TRUE si la herramienta comprobada está calibrada, es decir, que la distancia entre las puntas de la herramienta está calibrada, o si la herramienta comprobada está sincronizada, es decir, que la posición de las puntas de la herramienta está sincronizada con el cuentarrevoluciones de la herramienta.

FALSE si la herramienta comprobada no está calibrada ni sincronizada.

---

#### Argumentos

```
STIsCalib ToolName [\sguninit] | [\sgunsynch]
```

ToolName

**Tipo de dato:** string

El nombre de la unidad mecánica.

[ \sguninit ]

**Tipo de dato:** switch

Este argumento se utiliza para comprobar si la posición de la pistola está inicializada y calibrada.

[ \sgunsynch ]

**Tipo de dato:** switch

*Continúa en la página siguiente*

---



### 2.176 STIsCalib - Compruebe si una herramienta servo está calibrada

*Servo Tool Control*

*Continuación*

Este argumento se utiliza para comprobar si la posición está sincronizada con el cuentarrevoluciones.

#### Sintaxis

```
STIsCalib '('  
  [ ToolName ':=' ] < expression (IN) of string >  
  [ '\ ' sguninit ] | [ '\ 'sgunsynch ] ')'
```

Una función con un valor de retorno del tipo de dato `bool`.

#### Información relacionada

| Para obtener más información sobre   | Consulte   |
|--------------------------------------|--|
| Calibración de una herramienta servo | <a href="#">STCalib - Calibra una herramienta servo en la página 802</a> |

## 2 Funciones

---

### 2.177 STIsClosed - Comprueba si una herramienta servo está cerrada *Servo Tool Control*

### 2.177 STIsClosed - Comprueba si una herramienta servo está cerrada

---

#### Utilización

STIsClosed se utiliza para comprobar si una herramienta servo está cerrada.

---

#### Ejemplos básicos

Los siguientes ejemplos ilustran la función STIsClosed.

##### Ejemplo 1

```
IF STIsClosed(gun1) THEN
  !Start the weld process
  Set weld_start;
ELSE
  ...
ENDIF
```

Comprobar si la pistola está cerrada.

##### Ejemplo 2

```
STClose "sgun", 1000, 3 \Conc;
WHILE NOT(STIsClosed("sgun"\RetThickness:=thickness)) DO
  WaitTime 0.1;

ENDWHILE
IF thickness > max_thickness THEN...
```

Se empieza a cerrar la pistola con el nombre `sgun`. Se continúa inmediatamente con la siguiente instrucción, en la que el programa espera a que se cierre la pistola. Se lee el valor de grosor conseguido cuando la instrucción `STIsClosed` haya devuelto `TRUE`.

##### Ejemplo 3

#### Ejemplos de combinaciones no válidas

```
STClose "sgun", 1000, 3 \RetThickness:=thickness \Conc;
WHILE NOT(STIsClosed("sgun"\RetThickness:=thickness_2)) DO;
  ...
```

Se cierra la pistola. El parámetro `thickness` no incluye ningún valor válido ya que se utiliza el modificador `\Conc`. Espere a que se cierre la pistola. Cuando la pistola se cierra y `STIsClosed` devuelve `TRUE`, el parámetro `thickness_2` incluirá un valor válido ya que el modificador `\Conc` se utilizó para `STClose`.

```
STClose "sgun", 1000, 3 \RetThickness:=thickness;
WHILE NOT(STIsClosed("sgun"\RetThickness:=thickness_2)) DO;
  ...
```

Se cierra la pistola. El parámetro `thickness` incluirá un valor válido cuando la pistola se haya cerrado, dado que el modificador `\Conc` no se utiliza. El parámetro `thickness_2` no incluirá ningún valor válido, dado que el modificador `\Conc` no se utilizó en la instrucción `STClose`.

---

#### Valor de retorno

Tipo de dato: `bool`

*Continúa en la página siguiente*

### 2.177 STIsClosed - Comprueba si una herramienta servo está cerrada

*Servo Tool Control*

*Continuación*

TRUE si la herramienta comprobada está cerrada, es decir, si se consigue la fuerza de punta deseada.

FALSE si la herramienta comprobada no está cerrada.

#### Argumentos

```
STIsClosed ToolName [\RetThickness]
```

ToolName

Tipo de dato: string

El nombre de la unidad mecánica.

[\RetThickness]

Tipo de dato: num

El grosor conseguido [mm].

**¡ATENCIÓN!** Sólo es válido si \Conc se ha utilizado en la instrucción STClose precedente.

#### Sintaxis

```
STIsClosed '('
  [ ToolName ':=' ] < expression (IN) of string > ')'
  ['\ RetThickness ':=' < variable or persistent (INOUT) of num
  > ] ')'
```

Una función con un valor de retorno del tipo de dato bool.

#### Información relacionada

| Para obtener más información sobre                    | Consulte   |
|---|--|
| Apertura de una herramienta servo                     | <a href="#">STOpen - Abre una herramienta servo en la página 825</a>                         |
| Cierre de una herramienta servo                       | <a href="#">STClose - Cierra una herramienta servo en la página 807</a>                      |
| Comprobación de si una herramienta servo está abierta | <a href="#">STIsOpen - Comprueba si una herramienta servo está abierta en la página 1489</a> |

## 2 Funciones

2.178 STIsIndGun - Comprueba si una herramienta servo se encuentra en el modo independiente  
*Servo Tool Control*

### 2.178 STIsIndGun - Comprueba si una herramienta servo se encuentra en el modo independiente

#### Utilización

STIsIndGun se utiliza para comprobar si una herramienta servo se encuentra en el modo independiente.

#### Ejemplos básicos

El ejemplo siguiente ilustra la función STIsIndGun.

#### Ejemplo 1

```
IF STIsIndGun(gun1) THEN
  ! Start the gun calibration
  STCalib gun1\TipChg;
ELSE
  ...
ENDIF
```

#### Valor de retorno

Tipo de dato: bool

TRUE si la herramienta comprobada se encuentra en el modo independiente, es decir, puede moverse de forma independiente de los movimientos del robot.

FALSE si la herramienta comprobada *no* está en el modo independiente.

#### Argumentos

STIsIndGun ToolName

ToolName

Tipo de dato: string

El nombre de la unidad mecánica.

#### Sintaxis

```
STIsIndGun '('
  [ ToolName ':=' ] < expression (IN) of string > ')'
```

Una función con un valor de retorno del tipo de dato bool.

#### Información relacionada

| Para obtener más información sobre              | Consulte   |
|---|--|
| Calibración de una herramienta servo            | <a href="#">STCalib - Calibra una herramienta servo en la página 802</a>                               |
| Cambiar la pistola al modo independiente        | <a href="#">STIndGun: establece la herramienta servo en el modo independiente en la página 813</a>     |
| Restablecer el modo independiente de la pistola | <a href="#">STIndGunReset: restablece la herramienta servo del modo independiente en la página 815</a> |

## 2.179 STIsOpen - Comprueba si una herramienta servo está abierta

### Utilización

STIsOpen se utiliza para comprobar si una herramienta servo está abierta.

### Ejemplos básicos

Los siguientes ejemplos ilustran la función STIsOpen.

#### Ejemplo 1

```
IF STIsOpen(gun1) THEN
  !Start the motion
  MoveL ...
ELSE
  ...
ENDIF
```

Comprobar si la pistola está abierta.

#### Ejemplo 2

```
STCalib "sgun" \TipWear \Conc;
WHILE NOT(STIsOpen("sgun") \RetTipWear:=tipwear \RetPosAdj:=posadj)
  DO;
  WaitTime 0.1;

ENDWHILE
```

IF tipwear > 20...

IF posadj > 25...

Realizar una calibración de desgaste de puntas. Espere hasta que la pistola `sgun` esté abierta. Lea los valores de desgaste de la punta y el ajuste posicional.

#### Ejemplo 3

##### Ejemplos de combinaciones no válidas

```
STCalib "sgun" \TipWear \RetTipWear:=tipwear_1 \Conc;
WHILE NOT(STIsOpen("sgun") \RetTipWear:=tipwear_2) DO;
  WaitTime 0.1;

ENDWHILE
```

Iniciar una calibración de desgaste de puntas. El parámetro `tipwear_1` no incluye ningún valor válido ya que se utiliza el modificador `\Conc`. Cuando la calibración está preparada y `STIsOpen` devuelve `TRUE`, el parámetro `tipwear_2` contendrá un valor válido.

```
STCalib "sgun" \TipWear \RetTipWear:=tipwear_1;

WHILE NOT(STIsOpen("sgun") \RetTipWear:=tipwear_2) DO;
  WaitTime 0.1;

ENDWHILE
```

Realizar una calibración de desgaste de puntas. El parámetro `tipwear_1` incluye un valor válido ya que no se utiliza el modificador `\Conc`. Cuando `STIsOpen`

*Continúa en la página siguiente*

## 2 Funciones

### 2.179 STIsOpen - Comprueba si una herramienta servo está abierta

#### Servo Tool Control

#### Continuación

devuelve **TRUE**, el parámetro `tipwear_2` no contiene ningún valor válido dado que el modificador `\Conc` no se utilizó en `STCalib`.

#### Valor de retorno

Tipo de dato: `bool`

**TRUE** si la herramienta comprobada está abierta, es decir, si el brazo de la herramienta se encuentra en la posición abierta programada.

**FALSE** si la herramienta comprobada no está abierta.

#### Argumentos

```
STIsOpen ToolName [\RetTipWear] [\RetPosAdj]
```

ToolName

Tipo de dato: `string`

El nombre de la unidad mecánica.

[\RetTipWear]

Tipo de dato: `num`

El desgaste de puntas detectado [en mm].

**¡ATENCIÓN!** Sólo es válido si `\Conc` se ha usado en una instrucción `STCalib` precedente y si `STIsOpen` devuelve **TRUE**.

[\RetPosAdj]

Tipo de dato: `num`

El ajuste de posición desde la última calibración [mm].

**¡ATENCIÓN!** Sólo es válido si se ha usado `\Conc` en una instrucción `STCalib` precedente y si `STIsOpen` devuelve **TRUE**.

#### Sintaxis

```
STIsOpen '('  
  [ ToolName ':=' ] < expression (IN) of string > ')'  
  [ '\ RetTipWear ':=' < variable or persistent(INOUT) of num >  
    ] ';'   
  [ '\ RetPosAdj ':=' < variable or persistent(INOUT) of num > ]
```

Una función con un valor de retorno del tipo de dato `bool`.

#### Información relacionada

| Para obtener más información sobre                    | Consulte   |
|---|--|
| Apertura de una herramienta servo                     | <a href="#">STOpen - Abre una herramienta servo en la página 825</a>                           |
| Cierre de una herramienta servo                       | <a href="#">STClose - Cierra una herramienta servo en la página 807</a>                        |
| Comprobación de si una herramienta servo está cerrada | <a href="#">STIsClosed - Comprueba si una herramienta servo está cerrada en la página 1486</a> |

## 2.180 StrDigCalc - Operaciones aritméticas con el tipo de dato stringdig

### Utilización

StrDigCalc se utiliza para realizar operaciones aritméticas (+, -, \*, /, %) con dos cadenas de dígitos positivos, de la misma forma en que se realizan las operaciones aritméticas numéricas con valores enteros positivos.

Esta función permite gestionar enteros positivos superiores a 8.388.608 con una representación exacta.

### Ejemplos básicos

El ejemplo siguiente ilustra la función StrDigCalc.

Consulte también [Más ejemplos en la página 1492](#).

#### Ejemplo 1

```
res := StrDigCalc(str1, OpAdd, str2);
```

Se asigna a `res` el resultado de la operación de suma de los valores representados por las cadenas digitales `str1` y `str2`.

### Valor de retorno

Tipo de dato: stringdig

stringdig se utiliza para representar enteros positivos grandes en una cadena, usando únicamente dígitos.

Este tipo de dato ha sido introducido dado que el tipo de dato num no es capaz de manejar enteros positivos superiores a 8.388.608 con una representación exacta.

### Argumentos

```
StrDigCalc (StrDig1 Operation StrDig2)
```

StrDig1

*String Digit 1*

Tipo de dato: stringdig

Una cadena que representa un valor entero positivo.

Operation

*Arithmetic operator*

Tipo de dato: opcalc

Define la operación aritmética a realizar con las dos cadenas de dígitos. Es posible usar las operaciones aritméticas siguientes del tipo de dato opcalc; OpAdd, OpSub, OpMult, OpDiv y OpMod.

StrDig2

*String Digit 2*

Tipo de dato: stringdig

Una cadena que representa un valor entero positivo.

*Continúa en la página siguiente*

## 2 Funciones

---

### 2.180 StrDigCalc - Operaciones aritméticas con el tipo de dato stringdig

RobotWare - OS

Continuación

---

#### Ejecución de programas

Esta función permite:

- Comprobar que sólo haya dígitos del 0 al 9 en StrDig1 y StrDig2
  - Convertir las dos cadenas de dígitos a long integers
  - Realizar una operación aritmética con los dos long integers
  - Convertir el resultado de long integer a stringdig
- 

#### Más ejemplos

Los siguientes ejemplos ilustran la función StrDigCalc.

##### Ejemplo 1

```
res := StrDigCalc(str1, OpSub, str2);
```

Se asigna a *res* el resultado de la operación de resta de los valores representados por las cadenas digitales *str1* y *str2*.

##### Ejemplo 2

```
res := StrDigCalc(str1, OpMult, str2);
```

Se asigna a *res* el resultado de la operación de multiplicación de los valores representados por las cadenas digitales *str1* y *str2*.

##### Ejemplo 3

```
res := StrDigCalc(str1, OpDiv, str2);
```

Se asigna a *res* el resultado de la operación de división de los valores representados por las cadenas digitales *str1* y *str2*.

##### Ejemplo 4

```
res := StrDigCalc(str1, OpMod, str2);
```

Se asigna a *res* el resultado de la operación de módulo de los valores representados por las cadenas digitales *str1* y *str2*.

---

#### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre            | Causa del error  |
|-------------------|--|
| ERR_INT_NOTVAL    | Valores de entrada no sólo con dígitos o módulo entre cero |
| ERR_INT_MAXVAL    | Valor de entrada superior a 4294967295                     |
| ERR_CALC_OVERFLOW | Resultado dentro del rango 0...4.294.967.295               |
| ERR_CALC_NEG      | Sustracción negativa, es decir, StrDig2 > StrDig1          |
| ERR_CALC_DIVZERO  | División entre cero  |

---

#### Limitaciones

StrDigCalc sólo acepta cadenas que contengan únicamente dígitos (caracteres del 0 al 9). La existencia de cualquier otro carácter en stringdig da lugar a un error.

Esta función sólo puede manejar enteros positivos hasta 4.294.967.295.

Continúa en la página siguiente



**Sintaxis**

```
StrDigCalc '('  
  [ StrDig1 ':= ' ] < expression (IN) of stringdig > ','  
  [ Operation ':= ' ] < expression (IN) of opcalc > ','  
  [ StrDig2 ':= ' ] < expression (IN) of stringdig > ')'
```

Una función con un valor de retorno del tipo de dato stringdig.

**Información relacionada**

| Para obtener más información sobre      | Consulte  |
|---|---|
| Cadenas de caracteres con sólo dígitos. | <a href="#">stringdig - Cadena de caracteres con sólo dos dígitos en la página 1780</a> |
| Operadores aritméticos.                 | <a href="#">opcalc - Operador aritmético en la página 1714</a>                          |

## 2 Funciones

---

### 2.181 StrDigCmp - Comparar dos cadenas que sólo contienen dígitos

RobotWare - OS

### 2.181 StrDigCmp - Comparar dos cadenas que sólo contienen dígitos

---

#### Utilización

StrDigCmp se utiliza para comparar dos cadenas de dígitos positivos, de la misma forma en que se comparan numéricamente dos enteros positivos.

Esta función permite gestionar enteros positivos superiores a 8.388.608 con una representación exacta.

---

#### Ejemplos básicos

Los siguientes ejemplos ilustran la función StrDigCmp.

##### Ejemplo 1

```
VAR stringdig digits1 := "1234";
VAR stringdig digits2 := "1256";
VAR bool is_equal;
is_equal := StrDigCmp(digits1, EQ, digits2);
```

La variable `is_equal` cambia a `FALSE`, porque el valor numérico 1234 no es igual a 1256.

---

#### Valor de retorno

Tipo de dato: `bool`

`TRUE` si se cumple la condición indicada. `FALSE` en caso contrario.

---

#### Argumentos

```
StrDigCmp (StrDig1 Relation StrDig2)
```

StrDig1

*String Digit 1*

Tipo de dato: `stringdig`

La primera cadena a comparar numéricamente, sólo con dígitos.

Relation

Tipo de dato: `opnum`

Define cómo comparar las dos cadenas de dígitos. Es posible usar las constantes predefinidas del tipo de dato `opnumLT`, `LTEQ`, `EQ`, `NOTEQ`, `GTEQ` o bien `GT`.

StrDig2

*String Digit 2*

Tipo de dato: `stringdig`

La segunda cadena a comparar numéricamente, sólo con dígitos.

---

#### Ejecución de programas

Esta función permite:

- Comprobar que sólo se usen dígitos del 0 al 9 en `StrDig1` y `StrDig2`
  - Convertir las dos cadenas de dígitos a `long integers`
  - Comparar numéricamente los dos `long integers`
- 

Continúa en la página siguiente

**Gestión de errores**

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                      | Causa del error                    |
|-----------------------------|------------------------------------|
| <code>ERR_INT_NOTVAL</code> | Valores de entrada no sólo dígitos |
| <code>ERR_INT_MAXVAL</code> | Valor superior a 4294967295        |

**Limitaciones**

`StrDigCmp` sólo acepta cadenas que contengan únicamente dígitos (caracteres del 0 al 9). La existencia de cualquier otro carácter en `stringdig` da lugar a un error.

Esta función sólo puede manejar enteros positivos hasta 4.294.967.295.

**Sintaxis**

```
StrDigCmp '('
  [ StrDig1 ':= ' ] < expression (IN) of stringdig > ','
  [ Relation ':= ' ] < expression (IN) of opnum > ','
  [ StrDig2 ':= ' ] < expression (IN) of stringdig > ')'
```

Una función con un valor de retorno del tipo de dato `bool`.

**Información relacionada**

| Para obtener más información sobre                 | Consulte   |
|--|--|
| Cadena de caracteres con sólo dígitos              | <a href="#">stringdig - Cadena de caracteres con sólo dos dígitos en la página 1780</a>            |
| Operadores de comparación                          | <a href="#">opnum - Operador de comparación en la página 1715</a>                                  |
| Información de hora de archivo                     | <a href="#">FileTimeDnum - Obtener información de tiempo sobre un archivo en la página 1288</a>    |
| Hora de modificación de archivo del módulo cargado | <a href="#">ModTimeDnum - Obtener la hora de modificación del módulo cargado en la página 1379</a> |

## 2 Funciones

---

### 2.182 StrFind - Busca un carácter en una cadena de caracteres

RobotWare - OS

### 2.182 StrFind - Busca un carácter en una cadena de caracteres

---

#### Utilización

`StrFind` (*String Find*) se utiliza para buscar en una cadena, a partir de una posición especificada, un carácter que se encuentra dentro de un conjunto determinado de caracteres.

---

#### Ejemplos básicos

El ejemplo siguiente ilustra la función `StrFind`.

#### Ejemplo 1

```
VAR num found;  
found := StrFind("Robotics",1,"aeiou");
```

Se asigna a la variable `found` el valor 2.

```
found := StrFind("Robotics",1,"aeiou"\NotInSet);
```

Se asigna a la variable `found` el valor 1.

```
found := StrFind("IRB 6700",1,STR_DIGIT);
```

Se asigna a la variable `found` el valor 5.

```
found := StrFind("IRB 6700",1,STR_WHITE);
```

Se asigna a la variable `found` el valor 4.

---

#### Valor de retorno

Tipo de dato: `num`

La posición de carácter del primer carácter, ya sea en o después de la posición especificada, que pertenece al conjunto especificado. Si no se encuentra ninguno de los caracteres especificados, se devuelve la longitud de la cadena de caracteres +1.

---

#### Argumentos

```
StrFind (Str ChPos Set [\NotInSet])
```

Str

*String*

Tipo de dato: `string`

La cadena en la que se desea buscar.

ChPos

*Character Position*

Tipo de dato: `num`

La posición del carácter inicial. Si la posición está fuera de la cadena de caracteres, se genera un error de tiempo de ejecución.

Set

Tipo de dato: `string`

El conjunto de caracteres que se desea comprobar. Consulte también [Datos predefinidos en la página 1497](#).

---

Continúa en la página siguiente

[\NotInSet]

Tipo de dato: switch

Buscar un carácter que no se encuentra en el conjunto de caracteres presente en Set.

### Sintaxis

```
StrFind '('
  [Str ':='] <expression (IN) of string>', '
  [ChPos ':='] <expression (IN) of num>', '
  [Set ':='] <expression (IN) of string>
  ['\' NotInSet]')'
```

Una función con un valor de retorno del tipo de dato num.

### Datos predefinidos

El sistema dispone de varias constantes de cadena de caracteres predefinidas, que pueden usarse junto con las funciones para cadenas de caracteres.

| Nombre    | Conjunto de caracteres  |
|-----------|---|
| STR_DIGIT | <digit> ::=<br>0   1   2   3   4   5   6   7   8   9  |
| STR_UPPER | <upper case letter> ::=<br>A   B   C   D   E   F   G   H   I   J<br>  K   L   M   N   O   P   Q   R   S   T<br>  U   V   W   X   Y   Z   À   Á   Â   Ã<br>  Ä   Å   Æ   Ç   È   É   Ê   Ë   Ì   Í<br>  Î   Ï   1)   Ñ   Ò   Ó   Ô   Õ   Ö   Ø<br>  Ù   Ú   Û   Ü   2)   3)          |
| STR_LOWER | <lower case letter> ::=<br>a   b   c   d   e   f   g   h   i   j<br>  k   l   m   n   o   p   q   r   s   t<br>  u   v   w   x   y   z   à   á   â   ã<br>  ä   å   æ   ç   è   é   ê   ë   ì   í<br>  î   ï   1)   ñ   ò   ó   ô   õ   ö   ø<br>  ù   ú   û   ü   2)   3)   ß   ÿ- |
| STR_WHITE | <blank character> ::=   |

### Información relacionada

| Para obtener más información sobre   | Consulte   |
|--------------------------------------|--|
| Funciones para cadenas de caracteres | <i>Manual de referencia técnica - RAPID Overview</i> |
| Definición de cadena de caracteres   | <a href="#">string - Cadenas en la página 1778</a>   |
| Valores de cadena de caracteres      | <i>Manual de referencia técnica - RAPID Overview</i> |

## 2 Funciones

---

### 2.183 StrLen - Obtiene la longitud de una cadena

RobotWare - OS

### 2.183 StrLen - Obtiene la longitud de una cadena

---

#### Utilización

StrLen (*String Length*) se utiliza para obtener la longitud actual de la cadena.

---

#### Ejemplos básicos

El ejemplo siguiente ilustra la función StrLen.

#### Ejemplo 1

```
VAR num len;  
len := StrLen("Robotics");
```

Se asigna a la variable len el valor 8.

---

#### Valor de retorno

Tipo de dato: num

El número de caracteres de la cadena (>=0).

---

#### Argumentos

StrLen (Str)

Str

*String*

Tipo de dato: string

La cadena de caracteres cuyo número de caracteres se desea contar.

---

#### Sintaxis

```
StrLen '('  
  [Str ':=' ] <expression (IN) of string> ')'
```

Una función con un valor de retorno del tipo de dato num.

---

#### Información relacionada

| Para obtener más información sobre   | Consulte   |
|--------------------------------------|--|
| Funciones para cadenas de caracteres | <i>Manual de referencia técnica - Instrucciones, funciones y tipos de datos de RAPID</i> |
| Definición de cadena de caracteres   | <a href="#">string - Cadenas en la página 1778</a>                                       |
| Valores de cadena de caracteres      | <i>Manual de referencia técnica - Instrucciones, funciones y tipos de datos de RAPID</i> |

## 2.184 StrMap - Mapea una cadena de caracteres

### Utilización

`StrMap` (*String Mapping*) se utiliza para crear una copia de una cadena en la que todos los caracteres se convierten acorde con una especificación de mapeo especificada.

### Ejemplos básicos

Los siguientes ejemplos ilustran la función `StrMap`.

#### Ejemplo 1

```
VAR string str;
str := StrMap("Robotics","aeiou","AEIOU");
```

Se asigna a la variable `str` el valor `RObOtIcs`.

#### Ejemplo 2

```
str := StrMap("Robotics",STR_LOWER, STR_UPPER);
```

Se asigna a la variable `str` el valor `ROBOTICS`.

### Valor de retorno

Tipo de dato: `string`

La cadena de caracteres creada mediante la conversión de los caracteres de la cadena especificada, de la forma indicada por las cadenas de origen y destino. Cada carácter de la cadena especificada que se encuentre en la cadena de origen se reemplaza con el carácter de la posición correspondiente de la cadena de destino. Los caracteres para los que no se haya especificado ningún mapeo se copian sin cambios a la cadena resultante.

### Argumentos

```
StrMap ( Str FromMap ToMap)
```

`Str`

*String*

Tipo de dato: `string`

La cadena a convertir.

`FromMap`

Tipo de dato: `string`

Parte de índice del mapeo. Consulte también [Datos predefinidos en la página 1500](#).

`ToMap`

Tipo de dato: `string`

Parte de valor del mapeo. Consulte también [Datos predefinidos en la página 1500](#).

### Sintaxis

```
StrMap '('
  [ Str ':' ] <expression (IN) of string> ','
  [ FromMap ':' ] <expression (IN) of string> ','
```

*Continúa en la página siguiente*

## 2 Funciones

### 2.184 StrMap - Mapea una cadena de caracteres

RobotWare - OS

Continuación

```
[ ToMap '[:=' ] <expression (IN) of string> ')]'
```

Una función con un valor de retorno del tipo de dato string.

#### Datos predefinidos

El sistema dispone de varias constantes de cadena de caracteres predefinidas, que pueden usarse junto con las funciones para cadenas de caracteres.

| Nombre    | Conjunto de caracteres  |
|-----------|---|
| STR_DIGIT | <digit> ::=<br>0   1   2   3   4   5   6   7   8   9  |
| STR_UPPER | <upper case letter> ::=<br>A   B   C   D   E   F   G   H   I   J<br>  K   L   M   N   O   P   Q   R   S   T<br>  U   V   W   X   Y   Z   À   Á   Â   Ã<br>  Ä   Å   Æ   Ç   È   É   Ê   Ë   Ì   Í<br>  Î   Ï   1)   Ñ   Ò   Ó   Ô   Õ   Ö   Ø<br>  Ù   Ú   Û   Ü   2)   3)          |
| STR_LOWER | <lower case letter> ::=<br>a   b   c   d   e   f   g   h   i   j<br>  k   l   m   n   o   p   q   r   s   t<br>  u   v   w   x   y   z   à   á   â   ã<br>  ä   å   æ   ç   è   é   ê   ë   ì   í<br>  î   ï   1)   ñ   ò   ó   ô   õ   ö   ø<br>  ù   ú   û   ü   2)   3)   ß   ÿ- |
| STR_WHITE | <blank character> ::=   |

#### Información relacionada

| Para obtener más información sobre   | Consulte   |
|--------------------------------------|--|
| Funciones para cadenas de caracteres | <i>Manual de referencia técnica - RAPID Overview</i> |
| Definición de cadena de caracteres   | <a href="#">string - Cadenas en la página 1778</a>   |
| Valores de cadena de caracteres      | <i>Manual de referencia técnica - RAPID Overview</i> |



## 2.185 StrMatch - Busca un patrón dentro de una cadena de caracteres

### Utilización

`StrMatch` (*String Match*) se utiliza para buscar un patrón determinado dentro de otra cadena, a partir de una posición especificada.

### Ejemplos básicos

El ejemplo siguiente ilustra la función `StrMatch`.

#### Ejemplo 1

```
VAR num found;

found := StrMatch("Robotics",1,"bo");
```

Se asigna a la variable `found` el valor 3.

### Valor de retorno

Tipo de dato: `num`

La posición de carácter de la primera subcadena, en o después de la posición especificada, que es igual a la cadena de patrón especificada. Si no se encuentra la subcadena, se devuelve la longitud de la cadena + 1.

### Argumentos

```
StrMatch (Str ChPos Pattern)
```

Str

*String*

Tipo de dato: `string`

La cadena en la que se desea buscar.

ChPos

*Character Position*

Tipo de dato: `num`

La posición del carácter inicial. Si la posición está fuera de la cadena de caracteres, se genera un error de tiempo de ejecución.

Pattern

Tipo de dato: `string`

La cadena de caracteres de patrón que se desea buscar.

### Sintaxis

```
StrMatch '('
  [ Str ':= ' ] <expression (IN) of string> ','
  [ ChPos ':= ' ] <expression (IN) of num> ','
  [ Pattern ':= ' ] <expression (IN) of string> ')'
```

Una función con un valor de retorno del tipo de dato `num`.

*Continúa en la página siguiente*

## 2 Funciones

---

2.185 StrMatch - Busca un patrón dentro de una cadena de caracteres

RobotWare - OS

Continuación

---

### Información relacionada

| Para obtener más información sobre   | Consulte  |
|--------------------------------------|---|
| Funciones para cadenas de caracteres | <i>Manual de referencia técnica - RAPID Overview</i>      |
| Definición de cadena de caracteres   | <a href="#"><i>string - Cadenas en la página 1778</i></a> |
| Valores de cadena de caracteres      | <i>Manual de referencia técnica - RAPID Overview</i>      |

---

## 2.186 StrMemb - Comprueba si un carácter pertenece a un conjunto

---

### Utilización

StrMemb (*String Member*) se utiliza para comprobar si un carácter concreto de una cadena pertenece a un conjunto determinado de caracteres.

---

### Ejemplos básicos

El ejemplo siguiente ilustra la función StrMemb.

#### Ejemplo 1

```
VAR bool memb;  
memb := StrMemb("Robotics",2,"aeiou");
```

Se asigna a la variable memb el valor TRUE, ya que o forma parte del conjunto "aeiou".

```
memb := StrMemb("Robotics",3,"aeiou");
```

Se asigna a la variable memb el valor FALSE, ya que b no forma parte del conjunto "aeiou".

```
memb := StrMemb("S-721 68 VÄSTERÅS",3,STR_DIGIT);
```

Se asigna a la variable memb el valor TRUE, ya que 7 forma parte del conjunto STR\_DIGIT.

---

### Valor de retorno

Tipo de dato: bool

TRUE si el carácter de la posición especificada de la cadena especificada se encuentra dentro del conjunto de caracteres especificado.

---

### Argumentos

```
StrMemb (Str ChPos Set)
```

Str

*String*

Tipo de dato: string

La cadena que se desea comprobar.

ChPos

*Character Position*

Tipo de dato: num

La posición del carácter que se desea comprobar. Si la posición está fuera de la cadena de caracteres, se genera un error de tiempo de ejecución.

Set

Tipo de dato: string

El conjunto de caracteres que se desea comprobar.

---

### Sintaxis

```
StrMemb '('  
[ Str ':=' ] <expression (IN) of string> ','
```

*Continúa en la página siguiente*

## 2 Funciones

### 2.186 StrMemb - Comprueba si un carácter pertenece a un conjunto

RobotWare - OS

Continuación

```
[ ChPos ':= ' ] <expression (IN) of num> ','  
[ Set ':= ' ] <expression (IN) of string> ')'
```

Una función con un valor de retorno del tipo de dato bool.

#### Datos predefinidos

El sistema dispone de varias constantes de cadena de caracteres predefinidas, que pueden usarse junto con las funciones para cadenas de caracteres.

| Nombre    | Conjunto de caracteres  |
|-----------|---|
| STR_DIGIT | <digit> ::=<br>0   1   2   3   4   5   6   7   8   9  |
| STR_UPPER | <upper case letter> ::=<br>A   B   C   D   E   F   G   H   I   J<br>  K   L   M   N   O   P   Q   R   S   T<br>  U   V   W   X   Y   Z   À   Á   Â   Ã<br>  Î   Ï   1)   Ñ   Ò   Ó   Ô   Õ   Ö   Ø<br>  Ù   Ú   Û   Ü   2)   3)   |
| STR_LOWER | <lower case letter> ::=<br>a   b   c   d   e   f   g   h   i   j<br>  k   l   m   n   o   p   q   r   s   t<br>  u   v   w   x   y   z   à   á   â   ã<br>  ä   å   æ   ç   è   é   ê   ë   ì   í<br>  î   ï   1)   ñ   ò   ó   ô   õ   ö   ø<br>  ù   ú   û   ü   2)   3)   ß   ÿ- |
| STR_WHITE | <blank character> ::=   |

#### Información relacionada

| Para obtener más información sobre   | Consulte   |
|--------------------------------------|--|
| Funciones para cadenas de caracteres | <i>Manual de referencia técnica - RAPID Overview</i> |
| Definición de cadena de caracteres   | <a href="#">string - Cadenas en la página 1778</a>   |
| Valores de cadena de caracteres      | <i>Manual de referencia técnica - RAPID Overview</i> |

---

## 2.187 StrOrder - Comprueba si dos cadenas de caracteres están ordenadas

---

### Utilización

`StrOrder` (*String Order*) compara dos cadenas (carácter a carácter) y devuelve un valor booleano para indicar si las dos cadenas están en el mismo orden de acuerdo con la secuencia de orden de caracteres especificada.

---

### Ejemplos básicos

Los siguientes ejemplos ilustran la función `StrOrder`.

#### Ejemplo 1

```
VAR bool le;
```

```
le := StrOrder("FIRST", "SECOND", STR_UPPER);
```

Se asigna a la variable `le` el valor `TRUE`, porque "F" va antes de "S" en la secuencia de clasificación de caracteres `STR_UPPER`.

#### Ejemplo 2

```
VAR bool le;
```

```
le := StrOrder("FIRST", "FIRSTB", STR_UPPER);
```

Se asigna a la variable `le` el valor `TRUE`, porque "FIRSTB" contiene un carácter adicional en la secuencia de orden de caracteres (ningún carácter comparado con "B").

#### Ejemplo 3

```
VAR bool le;
```

```
le := StrOrder("FIRSTB", "FIRST", STR_UPPER);
```

Se asigna a la variable `le` el valor `FALSE`, porque "FIRSTB" contiene un carácter adicional en la secuencia de orden de caracteres ("B" comparado con ningún carácter).

---

### Valor de retorno

Tipo de dato: `bool`

`TRUE` si la primera cadena va antes de la segunda cadena (`Str1 <= Str2`) cuando se utiliza la clasificación de caracteres especificada.

Se supone que los caracteres que no están incluidos en el orden definido siguen a los caracteres presentes.

---

### Argumentos

```
StrOrder ( Str1 Str2 Order )
```

`Str1`

*String 1*

Tipo de dato: `string`

El primer valor de cadena.

---

*Continúa en la página siguiente*

## 2 Funciones

### 2.187 StrOrder - Comprueba si dos cadenas de caracteres están ordenadas

RobotWare - OS

Continuación

Str2

*String 2*

Tipo de dato: string

El segundo valor de cadena.

Order

Tipo de dato: string

Una secuencia de caracteres que define el orden. Consulte también [Datos predefinidos en la página 1506](#).

#### Sintaxis

```
StrOrder '('  
  [ Str1 ':=' ] <expression (IN) of string> ','  
  [ Str2 ':=' ] <expression (IN) of string> ','  
  [ Order ':=' ] <expression (IN) of string> ')'
```

Una función con un valor de retorno del tipo de dato bool.

#### Datos predefinidos

El sistema dispone de varias constantes de cadena de caracteres predefinidas, que pueden usarse junto con las funciones para cadenas de caracteres.

| Nombre    | Conjunto de caracteres  |
|-----------|---|
| STR_DIGIT | <digit> ::=<br>0   1   2   3   4   5   6   7   8   9  |
| STR_UPPER | <upper case letter> ::=<br>A   B   C   D   E   F   G   H   I   J<br>  K   L   M   N   O   P   Q   R   S   T<br>  U   V   W   X   Y   Z   À   Á   Â   Ã<br>  Ä   Å   Æ   Ç   È   É   Ê   Ë   Ì   Í<br>  Î   Ï   1)   Ñ   Ò   Ó   Ô   Õ   Ö   Ø<br>  Ù   Ú   Û   Ü   2)   3)          |
| STR_LOWER | <lower case letter> ::=<br>a   b   c   d   e   f   g   h   i   j<br>  k   l   m   n   o   p   q   r   s   t<br>  u   v   w   x   y   z   à   á   â   ã<br>  ä   å   æ   ç   è   é   ê   ë   ì   í<br>  î   ï   1)   ñ   ò   ó   ô   õ   ö   ø<br>  ù   ú   û   ü   2)   3)   ß   ÿ- |
| STR_WHITE | <blank character> ::=   |

#### Información relacionada

| Para obtener más información sobre   | Consulte   |
|--------------------------------------|--|
| Funciones para cadenas de caracteres | <i>Manual de referencia técnica - RAPID Overview</i> |
| Definición de cadena de caracteres   | <a href="#">string - Cadenas en la página 1778</a>   |
| Valores de cadena de caracteres      | <i>Manual de referencia técnica - RAPID Overview</i> |

### 2.188 StrPart - Busca una parte de una cadena

---

#### Utilización

StrPart (*String Part*) se utiliza para encontrar una parte de una cadena y obtenerla como una cadena nueva.

---

#### Ejemplos básicos

El ejemplo siguiente ilustra la función StrPart.

##### Ejemplo 1

```
VAR string part;  
part := StrPart("Robotics",1,5);
```

Se asigna a la variable `part` el valor "Robot".

---

#### Valor de retorno

Tipo de dato: `string`

La subcadena de la cadena especificada, que tiene la longitud especificada y comienza en la posición de carácter especificada.

---

#### Argumentos

```
StrPart (Str ChPos Len)
```

Str

*String*

Tipo de dato: `string`

La cadena en la que se desea buscar una parte.

ChPos

*Character Position*

La posición del carácter inicial. Si la posición está fuera de la cadena de caracteres, se genera un error de tiempo de ejecución.

Len

*Longitud*

Tipo de dato: `num`

La longitud de la cadena parcial. Si la longitud es negativa o es superior a la longitud de la cadena, o si la subcadena está (parcialmente) fuera de la cadena, se genera un error de tiempo de ejecución.

---

#### Sintaxis

```
StrPart '('  
[ Str ':= ' ] <expression (IN) of string> ','  
[ ChPos ':= ' ] <expression (IN) of num> ','  
[ Len ':= ' ] <expression (IN) of num> ')'
```

Una función con un valor de retorno del tipo de dato `string`.

*Continúa en la página siguiente*

## 2 Funciones

---

2.188 StrPart - Busca una parte de una cadena

RobotWare - OS

Continuación

---

### Información relacionada

| Para obtener más información sobre   | Consulte  |
|--------------------------------------|---|
| Funciones para cadenas de caracteres | <i>Manual de referencia técnica - RAPID Overview</i>      |
| Definición de cadena de caracteres   | <a href="#"><i>string - Cadenas en la página 1778</i></a> |
| Valores de cadena de caracteres      | <i>Manual de referencia técnica - RAPID Overview</i>      |



## 2.189 StrToByte - Convierte una cadena en un byte

### Utilización

`StrToByte` (*String To Byte*) se utiliza para convertir un dato del tipo `string` con un formato de dato de `byte` en un dato del tipo `byte`.

### Ejemplos básicos

El ejemplo siguiente ilustra la función `StrToByte`.

#### Ejemplo 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});
```

El contenido del componente de matriz `data_buffer{1}` será 10 en representación decimal después de la función `StrToByte` ....

```
data_buffer{2} := StrToByte(con_data_buffer{2}\Hex);
```

El contenido del componente de matriz `data_buffer{2}` será 174 en representación decimal después de la función `StrToByte` ....

```
data_buffer{3} := StrToByte(con_data_buffer{3}\Okt);
```

El contenido del componente de matriz `data_buffer{3}` será 126 en representación decimal después de la función `StrToByte` ... .

```
data_buffer{4} := StrToByte(con_data_buffer{4}\Bin);
```

El contenido del componente de matriz `data_buffer{4}` será 10 en representación decimal después de la función `StrToByte` ... .

```
data_buffer{5} := StrToByte(con_data_buffer{5}\Char);
```

El contenido del componente de matriz `data_buffer{5}` será 65 en representación decimal después de la función `StrToByte` ....

### Valor de retorno

Tipo de dato: `byte`

El resultado de la operación de conversión en representación decimal.

### Argumentos

```
StrToByte (ConStr [\Hex] | [\Okt] | [\Bin] | [\Char])
```

ConStr

*Convert String*

Tipo de dato: `string`

Los datos de cadena a convertir.

Si se omite el argumento modificador opcional, la cadena de caracteres se convierten al formato decimal (Dec).

[\Hex]

*Hexadecimal*

Tipo de dato: `switch`

*Continúa en la página siguiente*

## 2 Funciones

### 2.189 StrToByte - Convierte una cadena en un byte

RobotWare - OS

Continuación

La cadena de caracteres se convierte al formato hexadecimal.

[ \Okt ]

*Octal*

Tipo de dato: switch

La cadena de caracteres se convierte al formato octal.

[ \Bin ]

*Binary*

Tipo de dato: switch

La cadena de caracteres se convierte al formato binary.

[ \Char ]

*Character*

Tipo de dato: switch

La cadena de caracteres se convierte al formato de carácter ASCII.

### Limitaciones

En función del formato de la cadena que se desea convertir, se aplica el dato de cadena siguiente:

| Formato                              | Longitud de cadena | Rango             |
|--------------------------------------|--------------------|-------------------|
| Dec .....: 襍'- 袂'                    | 3                  | "0" - "255"       |
| Hex .....: 襍'- 袂', 'a'-'f', 'A'- 'F' | 2                  | "0" - "FF"        |
| Oct .....: 襍'- 袂'                    | 3                  | "0" - "377"       |
| Bin .....: 襍'- 襍'                    | 8                  | "0" - "11111111"  |
| Char ....: Cualquier carácter ASCII  | 1                  | Un carácter ASCII |

Es posible utilizar códigos de carácter de RAPID (por ejemplo "\07" para el carácter de control BEL) como argumentos de ConStr.

### Sintaxis

```
StrToByte '('  
  [ConStr ':= ' ] <expression (IN) of string>  
  ['\ ' Hex ] | ['\ ' Okt] | ['\ ' Bin] | ['\ ' Char]  
' )'
```

Una función con un valor de retorno del tipo de dato byte.

### Información relacionada

| Para obtener más información sobre                | Consulte   |
|---|--|
| Conversión de un byte en una cadena de caracteres | <a href="#">ByteToStr - Convierte un byte en un dato de cadena de caracteres en la página 1202</a> |
| Otras funciones de bits (byte)                    | <a href="#">Manual de referencia técnica - RAPID Overview</a>                                      |
| Otras funciones de cadenas de caracteres          | <a href="#">Manual de referencia técnica - RAPID Overview</a>                                      |

---

**2.190 StrToVal - Convierte una cadena de caracteres en un valor**

---

**Utilización**

StrToVal (*String To Value*) se utiliza para convertir una cadena en un valor de cualquier tipo de dato.

---

**Ejemplos básicos**

El ejemplo siguiente ilustra la función StrToVal.

Consulte también [Más ejemplos en la página 1511](#).

**Ejemplo 1**

```
VAR bool ok;  
VAR num nval;  
ok := StrToVal("3.85",nval);
```

Se asigna a la variable `ok` el valor `TRUE` y a `nval` el valor `3.85`.

---

**Valor de retorno**

Tipo de dato: `bool`

`TRUE` si la conversión solicitada tiene éxito y `FALSE` si no es así.

---

**Argumentos**

```
StrToVal ( Str Val )
```

Str

*String*

Tipo de dato: `string`

Un valor de cadena que contiene datos literales con un formato que corresponde al tipo de dato utilizado en el argumento `Val`. El formato válido es el de los agregados literales de RAPID.

Val

*Value*

Tipo de dato: `ANYTYPE`

El nombre de la variable o de la variable persistente de cualquier tipo para el almacenamiento del resultado de la conversión.

Puede usar todos los tipos de datos de valor con estructura atómica o de registro, componente de registro, matriz o elemento de matriz. El dato permanece sin cambios si la conversión solicitada ha fallado debido a que el formato no se corresponde con el dato utilizado en el argumento `Str`.

---

**Más ejemplos**

A continuación aparecen más ejemplos de la función StrToVal.

**Ejemplo 1**

```
VAR string str15 := "[600, 500, 225.3]";  
VAR bool ok;  
VAR pos pos15;
```

*Continúa en la página siguiente*

## 2 Funciones

---

### 2.190 StrToVal - Convierte una cadena de caracteres en un valor

RobotWare - OS

Continuación

```
ok := StrToVal(str15,pos15);
```

Se asigna a la variable `ok` el valor `TRUE` y se asigna a la variable `pos15` el valor especificado en la cadena `str15`.

---

#### Sintaxis

```
StrToVal '('  
  [ Str ':' ] <expression (IN) of string> ','  
  [ Val ':' ] <var or pers (INOUT) of ANYTYPE>  
' )'
```

Una función con un valor de retorno del tipo de dato `bool`.

---

#### Información relacionada

| Para obtener más información sobre   | Consulte   |
|--------------------------------------|--|
| Funciones para cadenas de caracteres | <i>Manual de referencia técnica - RAPID Overview</i> |
| Definición de cadena de caracteres   | <a href="#">string - Cadenas en la página 1778</a>   |
| Valores de cadena de caracteres      | <i>Manual de referencia técnica - RAPID Overview</i> |

## 2.191 Tan - Calcula la tangente

### Utilización

`Tan` (*Tangent*) se utiliza para calcular el valor de tangente desde un valor de ángulo.

### Ejemplos básicos

El ejemplo siguiente ilustra la función `Tan`.

#### Ejemplo 1

```

VAR num angle;
VAR num value;
...
...
value := Tan(angle);
value obtiene el valor de tangente de angle.

```

### Valor de retorno

Tipo de dato: num  
El valor de la tangente.

### Argumentos

`Tan` (Angle)

Angle

Tipo de dato: num  
El valor del ángulo, expresado en grados.

### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                              | Causa del error   |
|-------------------------------------|---|
| <code>ERR_TAN_DEGREES_NOTVAL</code> | El ángulo de 90 y 270 grados no está definido para <code>tan()</code> . |

### Sintaxis

```

Tan '('
  [Angle ':='] <expression (IN) of num>
  ')'

```

Una función con un valor de retorno del tipo de dato `num`.

### Información relacionada

| Para obtener más información sobre                          | Consulte  |
|---|---|
| Instrucciones y funciones matemáticas                       | <i>Manual de referencia técnica - RAPID Overview</i>                          |
| Arco tangente con un valor devuelto en el rango [-180, 180] | <a href="#">ATan2 - Calcula el valor de arco tangente 2 en la página 1171</a> |

## 2 Funciones

---

### 2.192 TanDnum - Calcula el valor de la tangente

RobotWare - OS

### 2.192 TanDnum - Calcula el valor de la tangente

---

#### Utilización

TanDnum (*Tangent*) se utiliza para calcular el valor de tangente de un valor de ángulo de los tipos de datos dnum.

---

#### Ejemplos básicos

El ejemplo siguiente ilustra la función TanDnum.

##### Ejemplo 1

```
VAR dnum angle;  
VAR dnum value;  
...  
...  
value := TanDnum(angle);  
value obtiene el valor de tangente de angle.
```

---

#### Valor de retorno

Tipo de dato: dnum

El valor de la tangente.

---

#### Argumentos

TanDnum (Angle)

Angle

Tipo de dato: dnum

El valor del ángulo, expresado en grados.

---

#### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema ERRNO cambiará a:

| Nombre                 | Causa del error   |
|------------------------|---|
| ERR_TAN_DEGREES_NOTVAL | El ángulo de 90 y 270 grados no está definido para tan(). |

---

#### Sintaxis

```
TanDnum '('  
  [Angle ':=' ] <expression (IN) of dnum>  
' )'
```

Una función con un valor de retorno del tipo de dato dnum.

---

#### Información relacionada

| Para obtener más información sobre                          | Consulte  |
|---|---|
| Instrucciones y funciones matemáticas                       | <i>Manual de referencia técnica - RAPID Overview</i>                              |
| Arco tangente con un valor devuelto en el rango [-180, 180] | <a href="#">ATan2Dnum - Calcula el valor de arco tangente 2 en la página 1172</a> |

---

## 2.193 TaskRunMec - Comprueba si una tarea controla alguna unidad mecánica

### Utilización

TaskRunMec se utiliza para comprobar si la tarea de programa controla cualquier unidad mecánica (robot con TCP o manipulador sin TCP).

### Ejemplos básicos

El ejemplo siguiente ilustra la función TaskRunMec.

#### Ejemplo 1

```
VAR bool flag;
...
flag := TaskRunMec( );
```

Si la tarea actual controla cualquier unidad mecánica, flag tendrá el valor TRUE. De lo contrario, tendrá el valor FALSE.

### Valor de retorno

Tipo de dato: bool

Si la tarea actual controla cualquier unidad mecánica, el valor de retorno será TRUE. De lo contrario, será FALSE.

### Ejecución de programas

Se comprueba si la tarea de programa actual controla alguna unidad mecánica.

### Sintaxis

```
TaskRunMec '( ' )'
```

Una función con un valor de retorno del tipo de dato bool.

### Información relacionada

| Para obtener más información sobre               | Consulte   |
|--|--|
| Comprobación de si la tarea controla algún robot | <a href="#">TaskRunRob - Comprueba si una tarea controla algún robot en la página 1516</a>   |
| Activación y desactivación de unidades mecánicas | <a href="#">ActUnit - Activa una unidad mecánica en la página 30</a><br><a href="#">DeactUnit - Desactiva una unidad mecánica en la página 187</a> |
| Configuración de unidades mecánicas              | <a href="#">Manual de referencia técnica - Parámetros del sistema</a>  |

## 2 Funciones

---

### 2.194 TaskRunRob - Comprueba si una tarea controla algún robot

RobotWare - OS

### 2.194 TaskRunRob - Comprueba si una tarea controla algún robot

---

#### Utilización

TaskRunRob se utiliza para comprobar si la tarea de programa controla cualquier robot (una unidad mecánica con TCP).

---

#### Ejemplos básicos

El ejemplo siguiente ilustra la función TaskRunRob.

#### Ejemplo 1

```
VAR bool flag;  
...  
flag := TaskRunRob();
```

Si la tarea actual controla algún robot, flag tendrá el valor TRUE. De lo contrario, tendrá el valor FALSE.

---

#### Valor de retorno

Tipo de dato: bool

Si la tarea actual controla algún robot, el valor de retorno será TRUE. De lo contrario, será FALSE.

---

#### Ejecución de programas

Se comprueba si la tarea de programa actual controla algún robot.

---

#### Sintaxis

```
TaskRunRob '(' ' ')
```

Una función con un valor de retorno del tipo de dato bool.

---

#### Información relacionada

| Para obtener más información sobre                          | Consulte   |
|---|--|
| Comprobación de si la tarea controla alguna unidad mecánica | <a href="#">TaskRunMec - Comprueba si una tarea controla alguna unidad mecánica en la página 1515</a>  |
| Activación y desactivación de unidades mecánicas            | <a href="#">ActUnit - Activa una unidad mecánica en la página 30</a><br><a href="#">DeactUnit - Desactiva una unidad mecánica en la página 187</a> |
| Configuración de unidades mecánicas                         | <a href="#">Manual de referencia técnica - Parámetros del sistema</a>  |



---

## 2.195 TasksInSync - Devuelve el número de tareas sincronizadas

---

### Utilización

`TasksInSync` se utiliza para obtener el número de tareas sincronizadas.

---

### Ejemplos básicos

El ejemplo siguiente ilustra la función `TaskInSync`.

#### Ejemplo 1

```
VAR tasks tasksInSyncList{6};
...

PROC main ()
  VAR num noOfSynchTasks;
  ...
  noOfSynchTasks:= TasksInSync (tasksInSyncList);
  TPWrite "No of synchronized tasks = "\Num:=noOfSynchTasks;
ENDPROC
```

Se asigna a la variable `noOfSynchTasks` el número de tareas sincronizadas y `tasksInSyncList` contendrá los nombres de las tareas sincronizadas. En este ejemplo, la lista de tareas es una variable pero también puede ser una variable persistente.

---

### Valor de retorno

Tipo de dato: `num`

El número de tareas sincronizadas.

---

### Argumentos

`TaskInSync (TaskList)`

`TaskList`

Tipo de dato: `tasks`

Un argumento inout que, en una lista de tareas (una matriz), presentará el nombre (`string`) de las tareas de programas que están sincronizadas. La lista de tareas puede ser del tipo `VAR` o `PERS`.

---

### Ejecución de programas

La función devuelve el número de tareas sincronizadas del sistema. Los nombres de las tareas sincronizadas se presentan en el argumento inout `TaskList`. En los casos en los que no hay ninguna tarea sincronizada, la lista sólo contendrá cadenas vacías.

---

### Limitaciones

En la actualidad sólo se admite un único grupo sincronizado, de forma que `TasksInSync` devuelve el número de tareas que están sincronizadas dentro de ese grupo.

---

*Continúa en la página siguiente*

## 2 Funciones

---

### 2.195 TasksInSync - Devuelve el número de tareas sincronizadas

RobotWare - OS

Continuación

---

#### Sintaxis

TasksInSync

```
[ TaskList ':=' ] < var or pers array {*} (INOUT) of tasks> ','
```

Una función con un valor de retorno del tipo de dato num.

---

#### Información relacionada

| Para obtener más información sobre                | Consulte   |
|---|--|
| Especificación de tareas de programa cooperativas | <a href="#">tasks - Tareas de programa RAPID en la página 1791</a>                             |
| Inicio de movimientos sincronizados coordinados   | <a href="#">SyncMoveOn - Inicia los movimientos sincronizados coordinados en la página 854</a> |

---

## 2.196 TaskIsActive: Comprobar si una tarea normal está activa

---

### Utilización

`TaskIsActive` se utiliza para comprobar si hay una tarea de programa normal activa en el Panel de selección de tareas de FlexPendant.

---

### Ejemplos básicos

El ejemplo siguiente ilustra la función `TaskIsActive`.

#### Ejemplo 1

```
IF TaskIsActive("T_ROB1") = TSP_STATUS_ACT THEN
  TPWrite "T_ROB1 is active in the Task Selection Panel";
ENDIF
```

Compruebe si la tarea `T_ROB1` del programa está activa en el Panel de selección de tareas de FlexPendant.

---

### Valor de retorno

Tipo de dato: `tsp_status`

El estado del panel de selección de tareas actuales.

---

### Datos predefinidos

Pueden utilizarse las siguientes constantes simbólicas predefinidas del tipo `tsp_status` para comprobar el valor de retorno:

```
CONST tsp_status TSP_STATUS_NOT_NORMAL_TASK := 0;
CONST tsp_status TSP_STATUS_DEACT := 1;
CONST tsp_status TSP_STATUS_DEACT_SERV_ROUT := 2;
CONST tsp_status TSP_STATUS_ACT := 3;
```

---

### Argumentos

`TaskIsActive ( TaskRef | TaskName )`

#### TaskRef

Tipo de dato: `taskid`

La identidad de la tarea de programa que debe comprobarse.

Las variables predefinidas del tipo de datos `taskid` está disponible para todas las tareas de programa del sistema.

La identidad variable es "*taskname*"+"Id", por ejemplo, la identidad variable de la tarea `T_ROB1` es `T_ROB1Id`.

#### TaskName

Tipo de dato: `string`

El nombre de la tarea de programa que debe comprobarse.

---

*Continúa en la página siguiente*

## 2 Funciones

### 2.196 TaskIsActive: Comprobar si una tarea normal está activa

RobotWare - OS

Continuación

#### Gestión de errores

Se generan los siguientes errores recuperables que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` se establecerá en:

|                           |   |
|---------------------------|---|
| <code>ERR_TASKNAME</code> | El nombre de tarea de programa en el argumento <code>\TaskName</code> no puede encontrarse en el sistema. |
|---------------------------|---|

#### Sintaxis

```
TaskIsActive '('  
  [ TaskRef ':= ' ] <variable (VAR) of taskid>  
  |[ TaskName ':= ' ] <expression (IN) of string> ')'
```

Una función con un valor de retorno del tipo de dato `tsp_status`.

#### Información relacionada

| Para obtener más información sobre                          | Consulte  |
|---|---|
| Estado de panel de selección de tareas                      | <a href="#">tsp_status: Estado de panel de selección de tareas en la página 1817</a>                        |
| Obtener el estado del panel de selección de tareas actuales | <a href="#">GetTSPStatus: Obtener el estado del panel de selección de tareas actuales en la página 1323</a> |
| Comprobar si la tarea se está ejecutando                    | <a href="#">TaskIsExecuting: Comprobar si la tarea se está ejecutando en la página 1521</a>                 |

## 2.197 TaskIsExecuting: Comprobar si la tarea se está ejecutando

### Utilización

`TaskIsExecuting` se utiliza para comprobar si una tarea de programa se está ejecutando.

### Ejemplos básicos

El ejemplo siguiente ilustra la función `TaskIsExecuting`.

#### Ejemplo 1

```
TPWrite "T_ROB1 is executing: " \Bool:=TaskIsExecuting("T_ROB1");
TPWrite "T_ROB2 is executing: " \Bool:=TaskIsExecuting("T_ROB2");
```

Comprueba si la tarea del programa `T_ROB1` y `T_ROB2` se está ejecutando y escribe el valor en FlexPendant.

### Valor de retorno

Tipo de dato: `bool`

Si la tarea del programa se está ejecutando, el valor de retorno es `TRUE`; de lo contrario, es `FALSE`.

### Argumentos

```
TaskIsExecuting ( TaskRef | TaskName )
```

#### TaskRef

Tipo de dato: `taskid`

La identidad de la tarea de programa que debe comprobarse.

Las variables predefinidas del tipo de datos `taskid` está disponible para todas las tareas de programa del sistema.

La identidad variable es "`taskname`"+"Id", por ejemplo, la identidad variable de la tarea `T_ROB1` es `T_ROB1Id`.

#### TaskName

Tipo de dato: `string`

El nombre de la tarea de programa que debe comprobarse.

### Gestión de errores

Se generan los siguientes errores recuperables que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` se establecerá en:

|                           |   |
|---------------------------|---|
| <code>ERR_TASKNAME</code> | El nombre de tarea de programa en el argumento <code>\TaskName</code> no puede encontrarse en el sistema. |
|---------------------------|---|

### Sintaxis

```
TaskIsExecuting '('
  [ TaskRef ':= ' ] <variable (VAR) of taskid>
  [ TaskName ':= ' ] <expression (IN) of string> ')'
```

Una función con un valor de retorno del tipo de dato `bool`.

*Continúa en la página siguiente*

## 2 Funciones

---

2.197 `TaskIsExecuting`: Comprobar si la tarea se está ejecutando

*RobotWare - OS*

*Continuación*

---

### Información relacionada

| Para obtener más información sobre        | Consulte  |
|---|---|
| Comprobar si una tarea normal está activa | <a href="#">TaskIsActive: Comprobar si una tarea normal está activa en la página 1519</a> |

## 2.198 TestAndSet - Comprueba una variable y la establece si no está establecida

### Utilización

`TestAndSet` puede usarse junto con un objeto de datos normal del tipo `bool`, como semáforo binario, para obtener el derecho exclusivo de acceso a áreas concretas del código de RAPID o a recursos del sistema. Esta función podría utilizarse tanto entre tareas de programa diferentes y niveles de ejecución diferentes (rutinas TRAP o de eventos) como dentro de una misma tarea de programa.

A continuación se enumeran algunos de los recursos que pueden necesitar protección de acceso al mismo tiempo:

- Uso de algunas rutinas de RAPID que presentan problemas de funcionamiento cuando se ejecutan en paralelo
- Uso del FlexPendant - Registro del operador

### Ejemplos básicos

El ejemplo siguiente ilustra la función `TestAndSet`.

Consulte también [Más ejemplos en la página 1524](#).

#### Ejemplo 1

**MAIN program task:**

```
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 program task:**

```
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;
```

Para evitar mezclar las líneas en el registro del operador, una de MAIN y una de BACK1, el uso de la función `TestAndSet` garantiza que las tres líneas de cada tarea no se separen.

Si la tarea de programa MAIN activa el semáforo `TestAndSet(tproutine_inuse)` en primer lugar, la tarea de programa BACK1 debe esperar hasta que la tarea de programa MAIN libere el semáforo.

### Valor de retorno

Tipo de dato: `bool`

TRUE si el semáforo ha sido activado por la tarea actual (la tarea que ejecuta la función `TestAndSet`). De lo contrario, FALSE.

*Continúa en la página siguiente*

## 2 Funciones

---

### 2.198 TestAndSet - Comprueba una variable y la establece si no está establecida

RobotWare - OS

Continuación

---

#### Argumentos

TestAndSet Object

Object

Tipo de dato: bool

Objeto de datos definido por el usuario para usarlo como semáforo.- El objeto de datos puede ser una variable VAR o una variable persistente PERS. Si se utilizan funciones TestAndSet entre tareas de programa diferentes, el objeto debe ser una variable persistente PERS o una variable instalada VAR (objetos compartidos entre tareas).

---

#### Ejecución de programas

Esta función comprueba en un paso indivisible si el usuario ha establecido el valor de la variable y, si no es así, lo establece y devuelve TRUE. De lo contrario, devuelve FALSE.

```
IF Object = FALSE THEN
  Object := TRUE;
  RETURN TRUE;
ELSE
  RETURN FALSE;
ENDIF
```

---

#### Más ejemplos

El ejemplo siguiente ilustra la función TestAndSet.

##### Ejemplo 1

```
LOCAL VAR bool doit_inuse := FALSE;
...
PROC doit(...)
  WaitUntil TestAndSet (doit_inuse);
  ...
  doit_inuse := FALSE;
ENDPROC
```

Si se instala, incorpora y comparte un módulo, es posible utilizar una variable de módulo local para la protección del acceso desde distintas tareas de programa al mismo tiempo.



#### Nota

En este caso con los módulos incorporados instalados y cuando se utiliza la variable persistente como objeto de semáforo: si la ejecución del programa se detiene en la rutina doit y el puntero de programa se traslada a main, la variable doit\_inuse no se restablecerá. Para evitarlo, devuelva la variable doit\_inuse a FALSE en la rutina de evento START.

---

#### Sintaxis

```
TestAndSet '('
  [ Object ':' '=' ] < variable or persistent (INOUT) of bool > ')'
```

Una función con un valor de retorno del tipo de dato bool.

Continúa en la página siguiente

---



### 2.198 TestAndSet - Comprueba una variable y la establece si no está establecida

*RobotWare - OS*

*Continuación*

#### Información relacionada

| Para obtener más información sobre   | Consulte   |
|--|--|
| Esperar hasta que la variable se desactiva y se activa de nuevo (tipo de espera con control de interrupción) | <a href="#">WaitTestAndSet - Esperar a que la variable cambie a FALSE y activarla a continuación en la página 1079</a> |

## 2 Funciones

---

2.199 TestDI - Se comprueba si una entrada digital está activada  
RobotWare - OS

### 2.199 TestDI - Se comprueba si una entrada digital está activada

---

#### Utilización

TestDI se usa para comprobar si una entrada digital está activada.

---

#### Ejemplos básicos

El ejemplo siguiente ilustra la función TestDI.

##### Ejemplo 1

```
IF TestDI (di2) THEN . . .
```

Si el valor actual de la señal di2 es 1,...

```
IF NOT TestDI (di2) THEN . . .
```

Si el valor actual de la señal di2 es 0,...

```
WaitUntil TestDI(di1) AND TestDI(di2);
```

La ejecución del programa continúa sólo cuando tanto la entrada di1 como la entrada di2 están activadas.

---

#### Valor de retorno

Tipo de dato: bool

TRUE = El valor actual de la señal es 1.

FALSE = El valor actual de la señal es 0.

---

#### Argumentos

```
TestDI (Signal)
```

Signal

Tipo de dato: signaldi

El nombre de la señal a comprobar.

---

#### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema ERRNO cambiará a:

| Nombre             | Causa del error   |
|--------------------|---|
| ERR_NO_ALIASIO_DEF | La variable de señal es una variable declarada en RAPID. No se ha conectado a una señal E/S definida en la configuración de E/S con la instrucción AliasIO.   |
| ERR_NORUNUNIT      | Se ha perdido el contacto con el dispositivo de E/S.  |
| ERR_SIG_NOT_VALID  | La señal de E/S no está disponible. Los motivos pueden ser que el dispositivo de E/S no está en funcionamiento o que exista un error en la configuración (válido solamente para el bus de campo ICI). |

---

#### Sintaxis

```
TestDI '('  
  [ Signal ':= ' ] < variable (VAR) of signaldi > ')'
```

Una función con un valor de retorno del tipo de dato bool.

---

Continúa en la página siguiente

2.199 TestDI - Se comprueba si una entrada digital está activada

RobotWare - OS

Continuación

### Información relacionada

| Para obtener más información sobre                | Consulte  |
|---|---|
| Lectura del valor de una señal digital de entrada | <a href="#">signalxx - Señales digitales y analógicas en la página 1763</a>             |
| Lectura del valor de una señal digital de salida  | <a href="#">DOutput - Lee el valor de una señal digital de salida en la página 1276</a> |
| Instrucciones de entrada/salida                   | <a href="#">Manual de referencia técnica - RAPID Overview</a>                           |

## 2 Funciones

---

2.200 TestSignRead - Obtiene el valor de una señal de test  
RobotWare - OS

### 2.200 TestSignRead - Obtiene el valor de una señal de test

---

#### Utilización

`TestSignRead` se utiliza para leer el valor actual de una señal de test. Esta función devuelve el valor momentáneo o el valor medio de los últimos muestreos, en función de la especificación de canal utilizada en la instrucción `TestSignDefine`.

---

#### Ejemplos básicos

El ejemplo siguiente ilustra la función `TestSignRead`. Consulte también [Más ejemplos en la página 1529](#).

#### Ejemplo 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;
```

Se asigna a `speed_value` el valor medio de los últimos 8 muestreos generados cada 0,5 mseg de la señal de test `speed` en el canal `speed_channel` definido como canal 1. El canal `speed_channel` mide la velocidad del eje `axis 1` en la unidad mecánica `orbit`.

---

#### Valor de retorno

Tipo de dato: `num`

El valor numérico en unidades SI del lado de motor del canal especificado, acorde con la definición de la instrucción `TestSignDefine`.

---

#### Argumentos

```
TestSignRead (Channel)
```

Channel

Tipo de dato: `num`

El número de canal, de 1 a 12, de la señal de test a leer. Este mismo número debe usarse en la instrucción de definición `TestSignDefine`.

---

#### Ejecución de programas

Devuelve el valor momentáneo o el valor medio de los últimos muestreos, en función de la especificación de canal utilizada en la instrucción `TestSignDefine`.

Para más información sobre las señales de test predefinidas con unidades SI válidas para los ejes de manipuladores externos, consulte el tipo de dato `testsignal`.

---

*Continúa en la página siguiente*

**Más ejemplos**

El ejemplo siguiente ilustra la función TestSignRead.

**Ejemplo 1**

```

CONST num torque_channel:=2;
VAR num torque_value;
VAR intnum timer_int;
CONST jointtarget psync := [...];
...
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

```

Quando la referencia de par del manipulador IRBP\_K y el eje 2 es por primera vez mayor que 6 Nm en el lado del motor durante el movimiento lento hacia la posición psync, la posición de ejes se muestra en el FlexPendant.

**Sintaxis**

```

TestSignRead '('
  [ Channel ':='] <expression (IN) of num> ')'

```

Una función con un valor de retorno del tipo de dato num.

**Información relacionada**

| Para obtener más información sobre | Consulte  |
|------------------------------------|---|
| Señal de prueba                    | <a href="#">testsignal - Señal de prueba en la página 1793</a>  |
| Definición de una señal de test    | <a href="#">TestSignDefine - Define una señal de prueba en la página 872</a>                            |
| Puesta a cero de señales de prueba | <a href="#">TestSignReset - Restablece todas las definiciones de señales de prueba en la página 874</a> |

## 2 Funciones

---

2.201 TextGet - Obtener un texto de las tablas de textos del sistema  
RobotWare - OS

### 2.201 TextGet - Obtener un texto de las tablas de textos del sistema

---

#### Utilización

TextGet se utiliza para obtener una cadena de texto de las tablas de textos del sistema.

---

#### Ejemplos básicos

Los siguientes ejemplos ilustran la función TextGet.

##### Ejemplo 1

```
VAR string text1;  
...  
text1 := TextGet(14, 5);
```

Se asigna a la variable `text1` el texto almacenado en el recurso de textos 14 y el número de índice 5.

##### Ejemplo 2

```
...  
TPWrite TextGet(14, 511 \Arg1:"robot" \Arg2:"fast");
```

Se lee el texto almacenado en el recurso de textos 14 y el número de índice 511.

El texto leído de las tablas de textos tiene el siguiente aspecto: `This is a {1} and this {1} is {2}`. Las cadenas utilizadas en los argumentos opcionales `Arg1` y `Arg2` sustituirán `{1}` y `{2}`. El resultado será:

```
This is a robot and this robot is fast
```

---

#### Valor de retorno

Tipo de dato: `string`

El texto especificado, tomado de las tablas de textos del sistema.

---

#### Argumentos

```
TextGet ( Table Index \Arg1 \Arg2 \Arg3 \Arg4 \Arg5 \Arg6 )
```

Table

Tipo de dato: `num`

El número de la tabla de textos (entero positivo).

Index

Tipo de dato: `num`

El número de índice (entero positivo) dentro de la tabla de textos.

[ \Arg1 ]

Tipo de dato: `string`

Si la cadena leída de la tabla de textos contiene `{1}`, el `{1}` se sustituye con la cadena utilizada en este argumento.

[ \Arg2 ]

Tipo de dato: `string`

---

*Continúa en la página siguiente*

Si la cadena leída de la tabla de textos contiene {2}, el {2} se sustituye con la cadena utilizada en este argumento.

[ \Arg3 ]

Tipo de dato: `string`

Si la cadena leída de la tabla de textos contiene {3}, el {3} se sustituye con la cadena utilizada en este argumento.

[ \Arg4 ]

Tipo de dato: `string`

Si la cadena leída de la tabla de textos contiene {4}, el {4} se sustituye con la cadena utilizada en este argumento.

[ \Arg5 ]

Tipo de dato: `string`

Si la cadena leída de la tabla de textos contiene {5}, el {5} se sustituye con la cadena utilizada en este argumento.

[ \Arg6 ]

Tipo de dato: `string`

Si la cadena leída de la tabla de textos contiene {6}, el {6} se sustituye con la cadena utilizada en este argumento.

#### Limitaciones

La longitud total de la cadena en RAPID es de 80 caracteres. Es la misma para todas las entradas como resultado de la función.

#### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                      | Causa del error   |
|-----------------------------|---|
| <code>ERR_TXTNOEXIST</code> | La tabla o índice no son válidos y no puede extraerse ninguna cadena de texto de las tablas de texto del sistema. |

#### Sintaxis

```
TextGet '('
  [Table ':='] <expression (IN) of num>','
  [Index ':='] <expression (IN) of num>
  ['\ ' Arg1 ':='] <expression (IN) of string>]
  ['\ ' Arg2 ':='] <expression (IN) of string>]
  ['\ ' Arg3 ':='] <expression (IN) of string>]
  ['\ ' Arg4 ':='] <expression (IN) of string>]
  ['\ ' Arg5 ':='] <expression (IN) of string>]
  ['\ ' Arg6 ':='] <expression (IN) of string>]')'
```

Una función con un valor de retorno del tipo de dato `string`.

*Continúa en la página siguiente*

## 2 Funciones

---

### 2.201 TextGet - Obtener un texto de las tablas de textos del sistema

RobotWare - OS

Continuación

---

#### Información relacionada

| Para obtener más información sobre        | Consulte  |
|---|---|
| Obtención de un número de tabla de textos | <a href="#">TextTabGet - Obtiene el número de una tabla de textos en la página 1535</a> |
| Instalación de una tabla de textos        | <a href="#">TextTabInstall - Instalación de una tabla de textos en la página 875</a>    |
| Formato de archivos de texto              | <i>Technical reference manual - RAPID kernel</i>  |
| Funciones para cadenas de caracteres      | <i>Manual de referencia técnica - RAPID Overview</i>                                    |
| Definición de cadena de caracteres        | <a href="#">string - Cadenas en la página 1778</a>                                      |
| Valores de cadena de caracteres           | <i>Manual de referencia técnica - RAPID Overview</i>                                    |
| <i>Advanced RAPID</i>                     | <i>Application manual - Controller software IRC5</i>                                    |



---

## 2.202 TextTabFreeToUse - Comprueba si una tabla de textos está libre para su uso

---

### Utilización

`TextTabFreeToUse` debe utilizarse para comprobar si un nombre de tabla de textos (una cadena de recursos de texto) puede utilizarse libremente (no está instalada en el sistema), es decir, si es posible instalar la tabla de textos en el sistema o no.

---

### Ejemplos básicos

El ejemplo siguiente ilustra la función `TextTabFreeToUse`.

#### Ejemplo 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.xml";
  ENDIF
ENDPROC
```

La primera vez que se ejecuta la rutina de evento `install_text`, la función `TextTabFreeToUse` devuelve `TRUE` y el archivo de texto `text_file.xml` se instala en el sistema. A continuación, es posible obtener con `RAPID` las cadenas de texto instaladas en el sistema, con ayuda de las funciones `TextTabGet` y `TextGet`.

La próxima vez que se ejecuta la rutina de evento `install_text`, la función `TextTabFreeToUse` devuelve `FALSE` y la instalación no se repite.

---

### Valor de retorno

Tipo de dato: `bool`

Esta función devuelve lo siguiente:

- `TRUE`, si la tabla de textos no está aún instalada en el sistema.
- `FALSE`, si la tabla de textos está ya instalada en el sistema.

---

### Argumentos

`TextTabFreeToUse ( TableName )`

`TableName`

Tipo de dato: `string`

El nombre de la tabla de textos (una cadena con 20 caracteres como máximo). Consulte `<text_resource>` en *Technical reference manual - RAPID kernel*, sección *Archivos de texto*. La cadena `text_resource` es el nombre de la tabla de textos.

*Continúa en la página siguiente*

## 2 Funciones

### 2.202 TextTabFreeToUse - Comprueba si una tabla de textos está libre para su uso

RobotWare - OS

Continuación

#### Limitaciones

Limitaciones de la instalación de tablas de textos (recursos de texto) en el sistema:

- No es posible instalar una misma tabla de textos más de una vez en el sistema.
- No es posible desinstalar (liberar) una sola tabla de textos del sistema. La única forma de desinstalar tablas de textos del sistema es reiniciar el controlador usando el modo de reinicio **Restablecer sistema**. De esta forma, se desinstalan todas las tablas de textos (tanto las del sistema como las definidas por el usuario).

#### Sintaxis

```
TextTabFreeToUse '('  
    [TableName ':='] <expression (IN) of string>')'
```

Una función con un valor de retorno del tipo de dato bool.

#### Información relacionada

| Para obtener más información sobre                      | Consulte   |
|---|--|
| Instalación de una tabla de textos                      | <a href="#">TextTabInstall - Instalación de una tabla de textos en la página 875</a>             |
| Formato de archivos de texto                            | <i>Technical reference manual - RAPID kernel</i>   |
| Obtención de un número de tabla de textos               | <a href="#">TextTabGet - Obtiene el número de una tabla de textos en la página 1535</a>          |
| Obtención de textos de las tablas de textos del sistema | <a href="#">TextGet - Obtener un texto de las tablas de textos del sistema en la página 1530</a> |
| Funciones para cadenas de caracteres                    | <i>Manual de referencia técnica - RAPID Overview</i>   |
| Definición de cadena de caracteres                      | <a href="#">string - Cadenas en la página 1778</a>   |
| <i>Advanced RAPID</i>                                   | <i>Especificaciones del producto - Controller software IRC5</i>                                  |

## 2.203 TextTabGet - Obtiene el número de una tabla de textos

### Utilización

`TextTabGet` se utiliza para obtener en tiempo de ejecución el número de tabla de textos de una tabla de textos definida por el usuario.

### Ejemplos básicos

Los siguientes ejemplos ilustran la función `TextTabGet`.

En ambos ejemplos se usa una nueva tabla de textos llamada `deburr_part1` para los textos definidos por el usuario. La nueva tabla de textos tiene el nombre de archivo `deburr.xml`. También se puede utilizar el formato `.eng` pero se recomienda usar el formato `.xml`.

### Código de ejemplo en formato .xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Resource Language="en" Name="deburr_part1">
  <Text Name="1">
    <Value>Part 1 is not in pos</Value>
    <Comment>Maximum length 80 chars</Comment>
  </Text>
  <Text Name="2">
    <Value>Identity of worked part: XYZ</Value>
    <Comment>Maximum length 80 chars</Comment>
  </Text>
  <Text Name="3">
    <Value>Part error in line 1</Value>
    <Comment>Maximum length 80 chars</Comment>
  </Text>
</Resource>
```

### Ejemplo 1

```
VAR num text_res_no;
...
text_res_no := TextTabGet("deburr_part1");
```

Se asigna a la variable `text_res_no` el número de tabla de texto de la tabla de texto definida `deburr_part1`.

### Ejemplo 2

```
ErrWrite TextGet(text_res_no, 1), TextGet(text_res_no, 2);
```

Se almacena un mensaje en el registro del robot. También se muestra en la pantalla del FlexPendant. Los mensajes se toman de la tabla de textos `deburr_part1`:

Part 1 is not in pos

Identity of worked part: XYZ

### Código de ejemplo en formato .eng

```
# deburr.eng - USERS deburr_part1 english text description file
#
# DESCRIPTION:
# Users text file for RAPID development
#
```

*Continúa en la página siguiente*

## 2 Funciones

### 2.203 TextTabGet - Obtiene el número de una tabla de textos

RobotWare - OS

Continuación

```
deburrr_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
```

---

#### Valor de retorno

Tipo de dato: num

El número de la tabla de textos definida.

---

#### Argumentos

```
TextTabGet ( TableName )
```

TableName

Tipo de dato: string

El nombre de la tabla de texto.

---

#### Sintaxis

```
TextTabGet '('
  [TableName '='] <expression (IN) of string>';)'
```

Una función con un valor de retorno del tipo de dato num.

---

#### Información relacionada

| Para obtener más información sobre                      | Consulte   |
|---|--|
| Obtención de textos de las tablas de textos del sistema | <a href="#">TextGet - Obtener un texto de las tablas de textos del sistema en la página 1530</a> |
| Instalación de una tabla de textos                      | <a href="#">TextTabInstall - Instalación de una tabla de textos en la página 875</a>             |
| Formato de archivos de texto                            | <i>Technical reference manual - RAPID kernel</i>   |
| Funciones para cadenas de caracteres                    | <i>Manual de referencia técnica - RAPID Overview</i>   |
| Definición de cadena de caracteres                      | <a href="#">string - Cadenas en la página 1778</a>   |
| Valores de cadena de caracteres                         | <i>Manual de referencia técnica - RAPID Overview</i>   |
| <i>Advanced RAPID</i>                                   | <i>Application manual - Controller software IRC5</i>   |

**2.204 TriggDataValid - Comprobar si el contenido de una variable de tipo triggdata es válido****Utilización**

La función `TriggDataValid` se utiliza para comprobar si una variable `triggdata` es válida. Una variable `triggdata` válida es una variable utilizada anteriormente en el programa en una de las instrucciones `TriggIO`, `TriggEquip`, `TriggInt`, `TriggSpeed`, `TriggCheckIO` o `TriggRampAO` para especificar condiciones de disparo y actividad de disparo.

**Ejemplos básicos**

El ejemplo siguiente ilustra la función `TriggDataValid`.

**Ejemplo 1**

```
VAR triggdata trigg_array{25};
...
PROC MyTriggProcL(robtargt 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;
  ...
```

El procedimiento `MyTriggProcL` que aparece arriba utiliza la instrucción `TriggDataValid` para comprobar si se utiliza cualquier dato válido en los argumentos opcionales `T1`, `T2` y `T3`.

**Valor de retorno**

Tipo de dato: `bool`

*Continúa en la página siguiente*

## 2 Funciones

### 2.204 TriggDataValid - Comprobar si el contenido de una variable de tipo triggdata es válido

RobotWare - OS

Continuación

Esta función devuelve lo siguiente:

- **TRUE**, si la variable es válida, es decir, se ha usado una de las instrucciones `TriggIO`, `TriggEquip`, `TriggInt`, `TriggSpeed`, `TriggCheckIO` o `TriggRampAO` para especificar condiciones de disparo y actividad de disparo.
- **FALSE**, si la variable no se ha usado al configurar ninguna condición de disparo ni actividad de disparo.

#### Argumentos

`TriggDataValid TriggData`

`TriggData`

Tipo de dato: `triggdata`

La variable `triggdata` cuya validez se desea comprobar.

#### Sintaxis

```
TriggDataValid  
[TriggData '[:=' ] < variable (VAR) of triggdata > '];'
```

Una función con un valor de retorno del tipo de dato `bool`.

#### Información relacionada

| Para obtener más información sobre   | Consulte   |
|--------------------------------------|--|
| Movimiento lineal con disparadores   | <a href="#">TriggL - Movimiento lineal del robot con eventos en la página 940</a>  |
| Movimiento de ejes con disparadores  | <a href="#">TriggJ - Movimientos de ejes del robot a partir de eventos en la página 932</a>  |
| Movimiento circular con disparadores | <a href="#">TriggC - Movimiento circular del robot con eventos en la página 895</a>  |
| Definición de disparadores           | <a href="#">TriggIO - Define un evento de E/S de posición o tiempo fijos cerca de un punto de paro en la página 926</a><br><a href="#">TriggEquip - Define un evento de E/S basado en la posición y el tiempo en la trayectoria en la página 914</a><br><a href="#">TriggInt - Define una interrupción dependiente de una posición en la página 921</a><br><a href="#">TriggCheckIO - Define una comprobación de E/S en una posición fija en la página 904</a><br><a href="#">TriggRampAO - Define un evento AO de rampa de posición fija en la trayectoria en la página 965</a><br><a href="#">TriggSpeed - Define la velocidad del TCP en proporción a una salida analógica con un evento de escala fija de posición-tiempo en la página 973</a> |
| Manejo de <code>triggdata</code>     | <a href="#">triggdata - Eventos de posicionamiento, trigg en la página 1805</a><br><a href="#">TriggDataReset - Restablecer el contenido en una variable de tipo triggdata en la página 912</a><br><a href="#">TriggDataCopy - Copiar el contenido de una variable de tipo triggdata en la página 910</a>  |

### 2.205 Trunc - Trunca un valor numérico

#### Utilización

`Trunc` (*Truncate*) se utiliza para trunca un valor numérico hasta un número especificado de decimales o a un valor entero.

#### Ejemplos básicos

Los siguientes ejemplos ilustran la función `Trunc`.

##### Ejemplo 1

```
VAR num val;  
val := Trunc(0.3852138\Dec:=3);
```

Se asigna a la variable `val` el valor 0.385.

##### Ejemplo 2

```
reg1 := 0.3852138;  
val := Trunc(reg1\Dec:=1);
```

Se asigna a la variable `val` el valor 0.3.

##### Ejemplo 3

```
val := Trunc(0.3852138);
```

Se asigna a la variable `val` el valor 0.

##### Ejemplo 4

```
val := Trunc(0.3852138\Dec:=6);
```

Se asigna a la variable `val` el valor 0.385213.

#### Valor de retorno

Tipo de dato: `num`

El valor numérico truncado con el número especificado de decimales.

#### Argumentos

```
Trunc ( Val [ \Dec ] )
```

`Val`

*Value*

Tipo de dato: `num`

El valor numérico a trunca.

`[ \Dec ]`

*Decimals*

Tipo de dato: `num`

Número de decimales.

Si el número de decimales especificado es 0 o se omite el argumento, el valor se trunca a un entero.

El número de decimales no debe ser negativo ni mayor que la precisión disponible para los valores numéricos.

El número máximo de decimales que pueden usarse es 6.

*Continúa en la página siguiente*

## 2 Funciones

---

### 2.205 Trunc - Trunca un valor numérico

RobotWare - OS

Continuación

---

#### Sintaxis

```
Trunc '('  
  [ Val ':=' ] <expression (IN) of num>  
  [ \Dec ':=' <expression (IN) of num> ] ')'
```

Una función con un valor de retorno del tipo de dato num.

---

#### Información relacionada

| Para obtener más información sobre    | Consulte  |
|---------------------------------------|---|
| Instrucciones y funciones matemáticas | <i>Manual de referencia técnica - RAPID Overview</i>                  |
| Redondeo de un valor                  | <a href="#">Round - Redondear un valor numérico en la página 1461</a> |



## 2.206 TruncDnum - Trunca un valor numérico

### Utilización

`TruncDnum` (*Truncate dnum*) se utiliza para trunca un valor numérico hasta un número especificado de decimales o a un valor entero.

### Ejemplos básicos

Los siguientes ejemplos ilustran la función `TruncDnum`.

#### Ejemplo 1

```
VAR dnum val;  
val := TruncDnum(0.3852138754655357\Dec:=3);
```

Se asigna a la variable `val` el valor 0.385.

#### Ejemplo 2

```
val := TruncDnum(0.3852138754655357\Dec:=1);
```

Se asigna a la variable `val` el valor 0.3.

#### Ejemplo 3

```
val := TruncDnum(0.3852138754655357);
```

Se asigna a la variable `val` el valor 0.

#### Ejemplo 4

```
val := TruncDnum(0.3852138754655357\Dec:=15);
```

Se asigna a la variable `val` el valor 0.385213875465535.

#### Ejemplo 5

```
val := TruncDnum(1000.3852138754655357\Dec:=15);
```

Se asigna a la variable `val` el valor 1000.38521387547.

### Valor de retorno

Tipo de dato: `dnum`

El valor numérico truncado con el número especificado de decimales.

### Argumentos

```
TruncDnum ( Val [\Dec] )
```

`Val`

*Value*

Tipo de dato: `dnum`

El valor numérico a trunca.

`[\Dec]`

*Decimals*

Tipo de dato: `num`

Número de decimales.

Si el número de decimales especificado es 0 o se omite el argumento, el valor se trunca a un entero.

*Continúa en la página siguiente*

## 2 Funciones

---

### 2.206 TruncDnum - Trunca un valor numérico

RobotWare - OS

Continuación

El número de decimales no debe ser negativo ni mayor que la precisión disponible para los valores numéricos.

El número máximo de decimales que pueden usarse es 15.

---

#### Sintaxis

```
TruncDnum '('  
  [ Val ':=' ] <expression (IN) of dnum>  
  [ \Dec ':=' <expression (IN) of num> ] ')'
```

Una función con un valor de retorno del tipo de dato dnum.

---

#### Información relacionada

| Para obtener más información sobre    | Consulte  |
|---------------------------------------|---|
| Instrucciones y funciones matemáticas | <i>Manual de referencia técnica - RAPID Overview</i>                      |
| Truncación de un valor                | <a href="#">Trunc - Trunca un valor numérico en la página 1539</a>        |
| Redondeo de un valor                  | <a href="#">Round - Redondear un valor numérico en la página 1461</a>     |
| Redondeo de un valor                  | <a href="#">RoundDnum - Redondear un valor numérico en la página 1463</a> |

---

## 2.207 Type - Obtiene el nombre del tipo de dato de una variable

---

### Utilización

`Type` se usa para obtener el nombre del tipo de dato de la variable especificada en el argumento `Data`.

---

### Ejemplos básicos

Los siguientes ejemplos ilustran la función `Type`.

#### Ejemplo 1

```
VAR string rettype;
VAR intnum intnumtype;
...
PROC main()
  rettype := Type(intnumtype);
  TPWrite "Data type name: " + rettype;
```

La información mostrada será: "Data type name: intnum"

#### Ejemplo 2

```
VAR string rettype;
VAR intnum intnumtype;
...
PROC main()
  rettype := Type(intnumtype \BaseName);
  TPWrite "Data type name: " + rettype;
```

La información mostrada será: "Data type name: num"

#### Ejemplo 3

```
VAR string rettype;
VAR num numtype;
...
PROC main()
  rettype := Type(numtype);
  TPWrite "Data type name: " + rettype;
```

La información mostrada será: "Data type name: num"

---

### Valor de retorno

Tipo de dato: `string`

Una cadena con el nombre del tipo de dato para la variable especificada en el argumento `Data`.

---

### Argumentos

```
Type (Data [\BaseName])
```

`Data`

*Data object name*

Tipo de dato: `anytype`

El nombre de la variable cuyo tipo de dato se desea obtener.

*Continúa en la página siguiente*

## 2 Funciones

---

### 2.207 Type - Obtiene el nombre del tipo de dato de una variable

RobotWare - OS

Continuación

[ \BaseName ]

**Base data type Name**

Tipo de dato: switch

Si se usa, la función devuelve el nombre del tipo de dato subyacente si el valor Data es un tipo de dato declarado como ALIAS.

---

### Sintaxis

```
Type '('  
  [ Data ' := ' ] < reference (REF) of anytype >  
  [ '\ ' BaseName ] ')'
```

Una función con un valor de retorno del tipo de dato string.

---

### Información relacionada

| Para obtener más información sobre | Consulte   |
|------------------------------------|--|
| Definición de los tipos Alias      | <i>Technical reference manual - RAPID kernel</i><br><a href="#">ALIAS - Asignación de un tipo de dato de alias en la página 1609</a> |

## 2.208 UIAlphaEntry - Introducción alfanumérica del usuario

### Utilización

UIAlphaEntry (*User Interaction Alpha Entry*) se usa para introducir una cadena desde un dispositivo de usuario disponible, como el FlexPendant. Se escribe un mensaje para el operador, que a su vez responde con una cadena de texto. A continuación, la cadena se transfiere al programa.

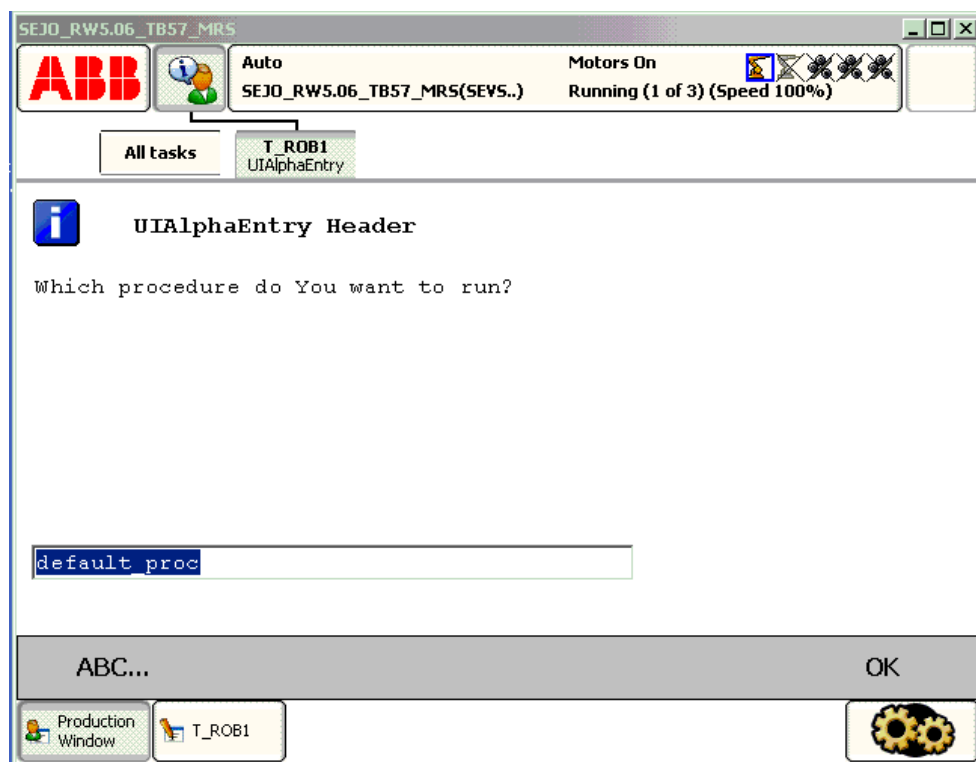
### Ejemplos básicos

El ejemplo siguiente ilustra la función UIAlphaEntry.

Consulte [Más ejemplos en la página 1549](#).

#### Ejemplo 1

```
VAR string answer;
...
answer := UIAlphaEntry(
  \Header:= "UIAlphaEntry Header",
  \Message:= "Which procedure do You want to run?"
  \Icon:=iconInfo
  \InitString:= "default_proc");
%answer%;
```



xx0500002437

Se muestra en el FlexPendant el cuadro de mensaje con icono, encabezado, mensaje y cadena inicial. El usuario puede editar la cadena o escribir una nueva cadena con el teclado alfanumérico admitido. La ejecución del programa espera hasta que se presione OK y, a continuación, la cadena escrita se devuelve en la

*Continúa en la página siguiente*

## 2 Funciones

---

### 2.208 UIAlphaEntry - Introducción alfanumérica del usuario

RobotWare-OS

Continuación

variable `answer`. A continuación, el programa llama al procedimiento especificado para su enlazamiento en tiempo de ejecución.

---

#### Valor de retorno

Tipo de dato: `string`

Esta función devuelve la cadena introducida.

Si la función es interrumpida por `\BreakFlag::`

- Si se especifica el parámetro `\InitString`, aparece esta cadena
- Si no se especifica el parámetro `\InitString`, aparece la cadena " " vacía.

Si la función se interrumpe a través del gestor de `ERROR`, no aparecerá ningún valor de retorno.

---

#### Argumentos

```
UIAlphaEntry ([\Header] [\Message][\MsgArray]
              [\Wrap][\Icon][\InitString] [\MaxTime] [\DIBreak] [\DIPassive]
              [\DOBreak] [\DOPassive] [\PersBoolBreak] [\PersBoolPassive]
              [\BreakFlag] [\UIActiveSignal])
```

`[\Header]`

Tipo de dato: `string`

El texto de título que debe escribirse en la parte superior del cuadro de mensaje. Máximo 40 caracteres.

`[\Message]`

Tipo de dato: `string`

Una línea de texto a escribir en la pantalla. Máximo 55 caracteres.

`[\MsgArray]`

*Message Array*

Tipo de dato: `string`

Several text lines from an array to be written on the display.

Sólo es posible usar uno de los parámetros, `\Message` o `\MsgArray` en cada momento.

El espacio máximo del diseño es de 9 líneas de 55 caracteres cada una.

`[\Wrap]`

Tipo de dato: `switch`

Si se selecciona, todas las cadenas especificadas en el argumento `\MsgArray` se concatenan para formar una cadena con un solo espacio entre las distintas cadenas individuales y distribuida en el número mínimo posible de líneas.

De forma predeterminada, cada una de las cadenas del argumento `\MsgArray` aparece en una línea separada en la pantalla.

`[\Icon]`

Tipo de dato: `icondata`

Define el icono a mostrar. Only one of the predefined icons of type `icondata` can be used. Consulte [Datos predefinidos en la página 1549](#).

Continúa en la página siguiente

---

De forma predeterminada, no se usa ningún icono.

[ \InitString ]

Tipo de dato: `string`

Una cadena inicial que debe mostrarse de forma predeterminada en el cuadro de introducción de texto.

[ \MaxTime ]

Tipo de dato: `num`

El periodo máximo, en segundos, que debe esperar el programa para continuar con la ejecución. Si no se presiona el botón OK en ese periodo, el programa sigue ejecutándose en el gestor de errores, a no ser que se utilice el indicador `BreakFlag` (que se documenta a continuación). La constante `ERR_TP_MAXTIME` puede usarse para comprobar si ha transcurrido ya el tiempo máximo establecido.

[ \DIBreak ]

#### *Digital Input Break*

Tipo de dato: `signal di`

La señal digital de entrada que puede interrumpir el diálogo con el operador. Si no se presiona el botón OK antes de que la señal cambie a 1 (o si ya tiene el valor 1), el programa sigue ejecutándose en el gestor de errores, a no ser que se utilice el indicador `BreakFlag` (que se documenta a continuación). La constante `ERR_TP_DIBREAK` puede usarse para comprobar si esto ha ocurrido.

[ \DIPassive ]

#### *Digital Input Passive*

Tipo de dato: `switch`

Este modificador redefine el comportamiento predeterminado con el argumento opcional `DIBreak`. En lugar de reaccionar cuando la señal cambia a 1 (o ya tiene el valor 1), la instrucción debe continuar en el gestor de errores (si no se utiliza `BreakFlag`) cuando la señal `DIBreak` cambia a 0 (o ya tiene el valor 0). La constante `ERR_TP_DIBREAK` puede usarse para comprobar si esto ha ocurrido.

[ \DOBreak ]

#### *Digital Output Break*

Tipo de dato: `signal do`

La señal digital de salida que puede interrumpir el diálogo con el operador. Si no se presiona el botón OK antes de que la señal cambie a 1 (o si ya tiene el valor 1), el programa sigue ejecutándose en el gestor de errores, a no ser que se utilice el indicador `BreakFlag` (que se documenta a continuación). La constante `ERR_TP_DOBREAK` puede usarse para comprobar si esto ha ocurrido.

[ \DOPassive ]

#### *Digital Output Passive*

Tipo de dato: `switch`

Este modificador redefine el comportamiento predeterminado con el argumento opcional `DOBreak`. En lugar de reaccionar cuando la señal cambia a 1 (o ya tiene

*Continúa en la página siguiente*

## 2 Funciones

---

### 2.208 UIAlphaEntry - Introducción alfanumérica del usuario

RobotWare-OS

Continuación

el valor 1), la instrucción debe continuar en el gestor de errores (si no se utiliza BreakFlag) cuando la señal DOBreak cambia a 0 (o ya tiene el valor 0). La constante ERR\_TP\_DOBREAK puede usarse para comprobar si esto ha ocurrido.

[\PersBoolBreak]

#### **Persistent Boolean Break**

Tipo de dato: bool

El booleano persistente que puede interrumpir el diálogo con el operador. Si no se selecciona ningún botón cuando el booleano persistente cambia a TRUE (o ya tiene el valor TRUE) entonces el programa continuará ejecutándose en el gestor de errores, a no ser que se utilice BreakFlag (que se documenta a continuación). La constante ERR\_TP\_PERSBOOLBREAK puede usarse para comprobar si esto ha ocurrido.

[\PersBoolPassive]

#### **Persistent Boolean Passive**

Tipo de dato: switch

Este interruptor redefine el comportamiento con el argumento opcional PersBoolBreak. En lugar de reaccionar cuando el booleano persistente cambia a TRUE (o ya tiene el valor TRUE), la instrucción debe continuar en el gestor de errores (si no se utiliza BreakFlag) cuando el booleano persistente PersBoolBreak cambia a FALSE (o ya tiene el valor FALSE). La constante ERR\_TP\_PERSBOOLBREAK puede usarse para comprobar si esto ha ocurrido.

[\BreakFlag]

Tipo de dato: errnum

Una variable que contiene el código de error si se utiliza MaxTime, DIBreak, DOBreak, o PersBoolBreak. Si se omite esta variable opcional, se ejecuta el gestor de errores. Las constantes ERR\_TP\_MAXTIME, ERR\_TP\_DIBREAK, ERR\_TP\_DOBREAK, y ERR\_TP\_PERSBOOLBREAK pueden usarse para seleccionar el motivo.

[\UIActiveSignal]

Tipo de dato: signaldo

La señal digital de salida utilizada en el argumento opcional UIActiveSignal se establece en 1 cuando se activa el cuadro de mensaje en FlexPendant. Cuando se ha realizado la selección de usuario y la ejecución continúa, la señal se vuelve a establecer en 0.

No existe ninguna supervisión de parada o reinicio. La señal se establece en 0 cuando la función está preparada o cuando se mueve el PP.

Continúa en la página siguiente



**Ejecución de programas**

El cuadro de mensaje alfanumérico, con su teclado alfanumérico, icono, título, líneas de mensaje y cadena inicial se muestra de acuerdo con los argumentos del programa. La ejecución del programa espera hasta que el usuario edite la cadena o cree una nueva y presione OK o hasta que el cuadro de mensaje sea interrumpido por un tiempo límite o una acción de señal. La cadena introducida y el motivo de la interrupción se devuelven al programa.

El nuevo cuadro de mensaje del nivel de rutina TRAP toma el foco del cuadro de mensaje del nivel básico.

**Datos predefinidos**

```
!Icons:
  CONST icondata iconNone := 0;
  CONST icondata iconInfo := 1;
  CONST icondata iconWarning := 2;
  CONST icondata iconError := 3;
```

**Más ejemplos**

El ejemplo siguiente ilustra la función UIAlphaEntry.

**Ejemplo 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
```

Aparece el cuadro de mensaje y el operador puede introducir una cadena y pulsar OK. El cuadro de mensaje también puede interrumpirse con un tiempo límite o mediante una señal de entrada digital. En el programa es posible encontrar la razón y realizar la acción adecuada.

*Continúa en la página siguiente*

## 2 Funciones

### 2.208 UIAlphaEntry - Introducción alfanumérica del usuario

RobotWare-OS

Continuación

#### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                          | Causa del error  |
|---------------------------------|--|
| <code>ERR_NO_ALIASIO_DEF</code> | La variable de señal es una variable declarada en <code>RAPID</code> y no se ha conectado a una señal de E/S definida en la configuración de E/S con la instrucción <code>AliasIO</code> . |
| <code>ERR_TP_NO_CLIENT</code>   | No hay ningún cliente, por ejemplo un <code>FlexPendant</code> , a cargo de la instrucción.  |

Si no se usa el parámetro `\BreakFlag`, estas situaciones pueden ser gestionadas en el gestor de errores:

Si se alcanza el tiempo límite (parámetro `\MaxTime`) antes de que responda el operador, la variable de sistema `ERRNO` cambia a `ERR_TP_MAXTIME` y la ejecución continúa en el gestor de errores.

Si se activa la entrada digital (parámetro `\DIBreak`) antes de que responda el operador, la variable de sistema `ERRNO` cambia a `ERR_TP_DIBREAK` y la ejecución continúa en el gestor de errores.

Si se activa la salida digital (parámetro `\DOBBreak`) antes de la acción del operador, la variable de sistema `ERRNO` cambia a `ERR_TP_DOBREAK` y la ejecución continúa en el gestor de errores.

Si se configura un booleano persistente (parámetro `\PersBoolBreak`) antes de la acción del operador, la variable de sistema `ERRNO` cambia a `ERR_TP_PERSBOOLBREAK` y la ejecución continúa en el gestor de errores.

#### Limitaciones

Evite usar un valor demasiado pequeño para el parámetro de tiempo límite `\MaxTime` si `UIAlphaEntry` se ejecuta frecuentemente, por ejemplo en un bucle. Si lo hace, puede dar lugar a un comportamiento impredecible del rendimiento del sistema, por ejemplo la ralentización de la respuesta del `FlexPendant`.

#### Sintaxis

```
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]  
  ['\' PersBoolBreak ::= ' <persistent (PERS) of bool>  
  ['\' PersBoolPassive]  
  ['\' BreakFlag ::= ' <var or pers (INOUT) of errnum>  
  ['\' UIActiveSignal ::= ' <variable (VAR) of signaldo> ] ')'
```

Continúa en la página siguiente

Una función con un valor de retorno del tipo de dato `string`.

#### Información relacionada

| Para obtener más información sobre                               | Consulte   |
|--|--|
| Datos de visualización de iconos                                 | <a href="#">icondata - Datos de visualización de iconos en la página 1683</a>                  |
| Cuadro de mensaje de interacción con el usuario de tipo básico   | <a href="#">UIMsgBox - Cuadro de mensaje de usuario de tipo básico en la página 1002</a>       |
| Cuadro de mensaje de interacción con el usuario de tipo avanzado | <a href="#">UIMessageBox - Cuadro de mensaje de usuario de tipo avanzado en la página 1577</a> |
| Introducción de número de interacción con el usuario             | <a href="#">UINumEntry - Introducción de número de usuario en la página 1585</a>               |
| Ajuste de número de interacción con el usuario                   | <a href="#">UINumTune - Ajuste de número de usuario en la página 1592</a>                      |
| Vista de lista de interacción con el usuario                     | <a href="#">UIListView - Vista de lista de usuario en la página 1568</a>                       |
| Sistema conectado al FlexPendant, etc.                           | <a href="#">UIClientExist - Existe cliente de usuario en la página 1552</a>                    |
| Llamada a procedimiento con enlazamiento en tiempo de ejecución  | <a href="#">Manual de referencia técnica - RAPID Overview</a>                                  |
| Borrado de la ventana de operador                                | <a href="#">TPErase - Borra el texto mostrado en el FlexPendant en la página 877</a>           |

## 2 Funciones

---

### 2.209 UIClientExist - Existe cliente de usuario

RobotWare - OS

### 2.209 UIClientExist - Existe cliente de usuario

---

#### Utilización

`UIClientExist` (*User Interaction Client Exist*) se usa para comprobar si hay algún dispositivo de usuario, por ejemplo el FlexPendant conectado al controlador.

---

#### Ejemplos básicos

El ejemplo siguiente ilustra la función `UIClientExist`.

#### Ejemplo 1

```
IF UIClientExist() THEN
    ! Possible to get answer from the operator
    ! The TReadFK and UIMsgBox ... can be used
ELSE
    ! Not possible to communicate with any operator
ENDIF
```

Se comprueba si es posible obtener alguna respuesta del operador del sistema.

---

#### Valor de retorno

Tipo de dato: `bool`

Devuelve `TRUE` si hay un FlexPendant conectado al sistema. De lo contrario, devuelve `FALSE`.

---

#### Limitaciones

`UIClientExist` devuelve `TRUE` durante un intervalo de hasta 16 segundos. A continuación, el FlexPendant es retirado. Tras ese intervalo, `UIClientExist` devuelve `FALSE` (es decir, a partir del momento en el que se detecta que se ha perdido la conexión con el FlexPendant). Se usa la misma limitación al volver a conectar el FlexPendant.

---

#### Sintaxis

```
UIClientExist '(' ')'
```

Una función con un valor de retorno del tipo de dato `bool`.

---

#### Información relacionada

| Para obtener más información sobre                               | Consulte   |
|--|--|
| Cuadro de mensaje de interacción con el usuario de tipo básico   | <a href="#">UIMsgBox - Cuadro de mensaje de usuario de tipo básico en la página 1002</a>       |
| Cuadro de mensaje de interacción con el usuario de tipo avanzado | <a href="#">UIMessageBox - Cuadro de mensaje de usuario de tipo avanzado en la página 1577</a> |
| Introducción de número de interacción con el usuario             | <a href="#">UINumEntry - Introducción de número de usuario en la página 1585</a>               |
| Ajuste de número de interacción con el usuario                   | <a href="#">UINumTune - Ajuste de número de usuario en la página 1592</a>                      |
| Introducción alfanumérica de interacción con el usuario          | <a href="#">UIAlphaEntry - Introducción alfanumérica del usuario en la página 1545</a>         |

*Continúa en la página siguiente*

| Para obtener más información sobre           | Consulte   |
|--|--|
| Vista de lista de interacción con el usuario | <a href="#">UIListView - Vista de lista de usuario en la página 1568</a>             |
| Borrado de la ventana de operador            | <a href="#">TPErase - Borra el texto mostrado en el FlexPendant en la página 877</a> |

## 2 Funciones

### 2.210 UIDnumEntry - Introducción de número de usuario RobotWare - OS

### 2.210 UIDnumEntry - Introducción de número de usuario

#### Utilización

UIDnumEntry (*User Interaction Number Entry*) se usa para introducir un valor numérico desde un dispositivo de usuario disponible, como el FlexPendant. Se escribe un mensaje para el operador, que a su vez responde con un valor numérico. El valor numérico es comprobado, autorizado y transferido de nuevo al programa.

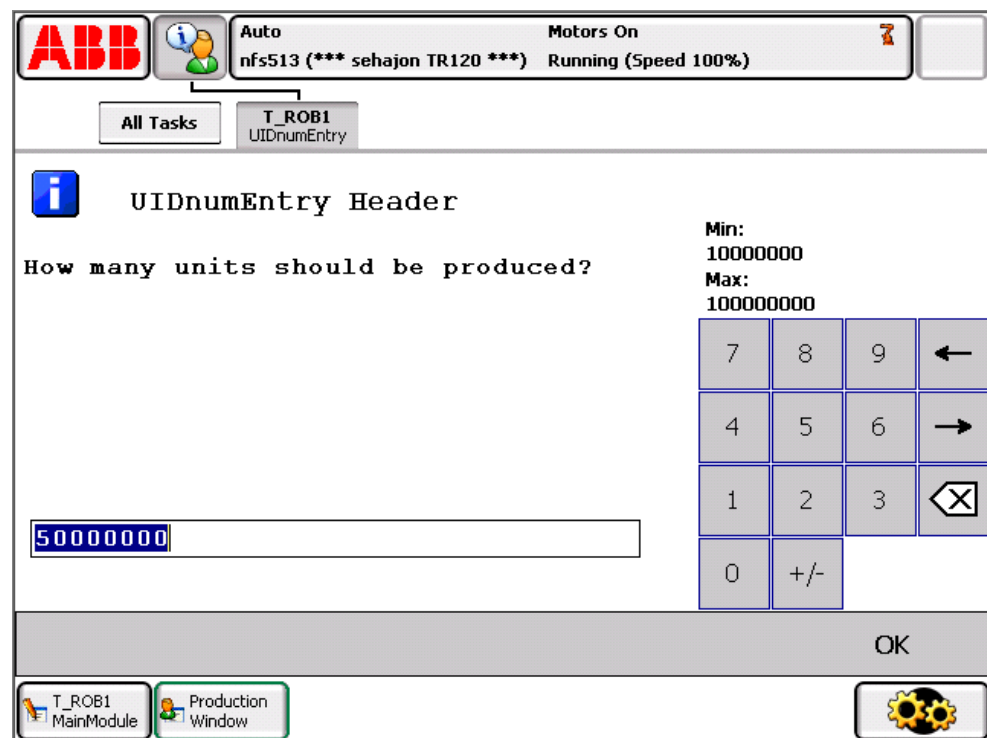
#### Ejemplos básicos

El ejemplo siguiente ilustra la función UIDnumEntry.

Consulte también [Más ejemplos en la página 1558](#).

#### Ejemplo 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

Se muestra en el FlexPendant el cuadro de mensaje numérico con icono, encabezado, mensaje y valores inicial, máximo y mínimo. El cuadro de mensaje comprueba que el operador seleccione un entero perteneciente al rango de valores.

Continúa en la página siguiente

La ejecución del programa espera hasta que se presione OK. A continuación, se devuelve el valor numérico seleccionado.

---

#### Valor de retorno

Tipo de dato: dnum

Esta función devuelve el valor numérico introducido.

Si la función es interrumpida por `\BreakFlag`:

- Si se especifica el parámetro `\InitValue`, se devuelve este valor.
- Si no se especifica el parámetro `\InitValue`, se devuelve el valor 0.

Si la función se interrumpe desde el gestor de `ERROR`, no tiene ningún valor de retorno en absoluto.

---

#### Argumentos

```
UIDnumEntry ( [\Header] [\Message] | [\MsgArray]
              [\Wrap][\Icon][\InitValue] [\MinValue] [\MaxValue]
              [\AsInteger][\MaxTime] [\DIBreak] [\DIPassive] [\DOBreak]
              [\DOPassive] [\PersBoolBreak] [\PersBoolPassive] [\BreakFlag]
              [\UIActiveSignal])
```

`[\Header]`

Tipo de dato: `string`

El texto de título que debe escribirse en la parte superior del cuadro de mensaje. Máximo 40 caracteres.

`[\Message]`

Tipo de dato: `string`

Una línea de texto a escribir en la pantalla. Máx. 40 caracteres.

`[\MsgArray]`

**Message Array**

Tipo de dato: `string`

Several text lines from an array to be written on the display.

Sólo es posible usar uno de los parámetros, `\Message` o `\MsgArray`, en cada momento.

El espacio máximo del diseño es de 9 líneas de 40 caracteres cada una.

`[\Wrap]`

Tipo de dato: `switch`

Si se selecciona, todas las cadenas especificadas en el argumento `\MsgArray` se concatenan para formar una cadena con un solo espacio entre las distintas cadenas individuales y distribuida en el número mínimo posible de líneas.

De forma predeterminada, cada una de las cadenas del argumento `\MsgArray` aparece en una línea separada en la pantalla.

`[\Icon]`

Tipo de dato: `icondata`

*Continúa en la página siguiente*

## 2 Funciones

---

### 2.210 UIDnumEntry - Introducción de número de usuario

RobotWare - OS

Continuación

Define el icono a mostrar. Sólo puede usarse uno de los iconos predefinidos de tipo `icondata`. Consulte [Datos predefinidos en la página 1558](#).

De forma predeterminada, no se usa ningún icono.

`[\InitValue]`

Tipo de dato: `dnum`

El valor inicial que se muestra en el cuadro de introducción.

`[\MinValue]`

Tipo de dato: `dnum`

El valor mínimo del valor de retorno.

`[\MaxValue]`

Tipo de dato: `dnum`

El valor máximo del valor de retorno.

`[\AsInteger]`

Tipo de dato: `switch`

Elimina el punto decimal del teclado numérico para garantizar que el valor devuelto sea un entero.

`[\MaxTime]`

Tipo de dato: `num`

El periodo máximo, en segundos, que debe esperar el programa para continuar con la ejecución. Si no se presiona el botón OK en ese periodo, el programa sigue ejecutándose en el gestor de errores, a no ser que se utilice el indicador `BreakFlag` (que se documenta a continuación). La constante `ERR_TP_MAXTIME` puede usarse para comprobar si ha transcurrido ya el tiempo máximo establecido.

`[\DIBreak]`

**Digital Input Break**

Tipo de dato: `signal`

La señal digital de entrada que puede interrumpir el diálogo con el operador. Si no se presiona el botón OK antes de que la señal cambie a 1 (o si ya tiene el valor 1), el programa sigue ejecutándose en el gestor de errores, a no ser que se utilice el indicador `BreakFlag` (que se documenta a continuación). La constante `ERR_TP_DIBREAK` puede usarse para comprobar si esto ha ocurrido.

`[\DIPassive]`

**Digital Input Passive**

Tipo de dato: `switch`

Este modificador redefine el comportamiento predeterminado con el argumento opcional `DIBreak`. En lugar de reaccionar cuando la señal cambia a 1 (o ya tiene el valor 1), la instrucción debe continuar en el gestor de errores (si no se utiliza `BreakFlag`) cuando la señal `DIBreak` cambia a 0 (o ya tiene el valor 0). La constante `ERR_TP_DIBREAK` puede usarse para comprobar si esto ha ocurrido.

Continúa en la página siguiente



[\`DOBreak`]

#### *Digital Output Break*

Tipo de dato: `signaldo`

La señal digital de salida que puede interrumpir el diálogo con el operador. Si no se presiona el botón OK antes de que la señal cambie a 1 (o si ya tiene el valor 1), el programa sigue ejecutándose en el gestor de errores, a no ser que se utilice el indicador `BreakFlag` (que se documenta a continuación). La constante `ERR_TP_DOBREAK` puede usarse para comprobar si esto ha ocurrido.

[\`DOPassive`]

#### *Digital Output Passive*

Tipo de dato: `switch`

Este modificador redefine el comportamiento predeterminado con el argumento opcional `DOBreak`. En lugar de reaccionar cuando la señal cambia a 1 (o ya tiene el valor 1), la instrucción debe continuar en el gestor de errores (si no se utiliza `BreakFlag`) cuando la señal `DOBreak` cambia a 0 (o ya tiene el valor 0). La constante `ERR_TP_DOBREAK` puede usarse para comprobar si esto ha ocurrido.

[\`PersBoolBreak`]

#### *Persistent Boolean Break*

Tipo de dato: `bool`

El booleano persistente que puede interrumpir el diálogo con el operador. Si no se selecciona ningún botón cuando el booleano persistente cambia a TRUE (o ya tiene el valor TRUE) entonces el programa continuará ejecutándose en el gestor de errores, a no ser que se utilice `BreakFlag` (que se documenta a continuación). La constante `ERR_TP_PERSBOOLBREAK` puede usarse para comprobar si esto ha ocurrido.

[\`PersBoolPassive`]

#### *Persistent Boolean Passive*

Tipo de dato: `switch`

Este interruptor redefine el comportamiento con el argumento opcional `PersBoolBreak`. En lugar de reaccionar cuando el booleano persistente cambia a TRUE (o ya tiene el valor TRUE), la instrucción debe continuar en el gestor de errores (si no se utiliza `BreakFlag`) cuando el booleano persistente `PersBoolBreak` cambia a FALSE (o ya tiene el valor FALSE). La constante `ERR_TP_PERSBOOLBREAK` puede usarse para comprobar si esto ha ocurrido.

[\`BreakFlag`]

Tipo de dato: `errnum`

Una variable que contiene el código de error si se utiliza `MaxTime`, `DIBreak`, `DOBreak`, o `PersBoolBreak`. Si se omite esta variable opcional, se ejecuta el gestor de errores. Las constantes `ERR_TP_MAXTIME`, `ERR_TP_DIBREAK`, `ERR_TP_DOBREAK`, y `ERR_TP_PERSBOOLBREAK` pueden usarse para seleccionar el motivo.

*Continúa en la página siguiente*

## 2 Funciones

---

### 2.210 UIDnumEntry - Introducción de número de usuario

RobotWare - OS

Continuación

[\UIActiveSignal]

Tipo de dato: signaldo

La señal digital de salida utilizada en el argumento opcional `UIActiveSignal` se establece en 1 cuando se activa el cuadro de mensaje en `FlexPendant`. Cuando se ha realizado la selección de usuario y la ejecución continúa, la señal se vuelve a establecer en 0.

No existe ninguna supervisión de parada o reinicio. La señal se establece en 0 cuando la función está preparada o cuando se mueve el PP.

---

### Ejecución de programas

Se muestra el cuadro de mensaje numérico, con teclado numérico, icono, encabezado, líneas de mensaje y valores inicial, máximo y mínimo, de acuerdo con los argumentos del programa. La ejecución del programa espera hasta que el usuario haya introducido un valor numérico autorizado y presione OK o hasta que el cuadro de mensaje sea interrumpido por un tiempo límite o una acción de señal. El valor numérico introducido y el motivo de la interrupción se devuelven al programa.

El nuevo cuadro de mensaje del nivel de rutina TRAP toma el foco del cuadro de mensaje del nivel básico.

---

### Datos predefinidos

```
!Icons:
  CONST icondata iconNone := 0;
  CONST icondata iconInfo := 1;
  CONST icondata iconWarning := 2;
  CONST icondata iconError := 3;
```

---

### Más ejemplos

El ejemplo siguiente ilustra la función `UIDnumEntry`.

#### Ejemplo 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
```

Continúa en la página siguiente

Se muestra el cuadro de mensaje y el operador puede introducir un valor numérico y presionar OK. El cuadro de mensaje también puede ser interrumpido con un tiempo límite o una interrupción con una señal digital de entrada. Es posible determinar el motivo desde el programa y tomar las acciones adecuadas.

#### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                          | Causa del error   |
|---------------------------------|---|
| <code>ERR_NO_ALIASIO_DEF</code> | La variable de señal es una variable declarada en RAPID y no se ha conectado a una señal de E/S definida en la configuración de E/S con la instrucción <code>AliasIO</code> .         |
| <code>ERR_TP_NO_CLIENT</code>   | No hay ningún cliente, por ejemplo un FlexPendant, a cargo de la instrucción.   |
| <code>ERR_UI_INITVALUE</code>   | El valor inicial (parámetro <code>\InitValue</code> ) no se ha especificado dentro del rango del valor mínimo y máximo (parámetros <code>\MinValue</code> y <code>\MaxValue</code> ). |
| <code>ERR_UI_MAXMIN</code>      | El valor mínimo (parámetro <code>\MinValue</code> ) es mayor que el valor máximo (parámetro <code>\MaxValue</code> ).   |
| <code>ERR_UI_NOTINT</code>      | El valor inicial (parámetro <code>\InitValue</code> ) no es un entero tal y como se especifica en el parámetro <code>\AsInteger</code> .  |

Si no se usa el parámetro `\BreakFlag`, estas situaciones pueden ser gestionadas en el gestor de errores:

- Si se alcanza el tiempo límite (parámetro `\MaxTime`) antes de que responda el operador, la variable de sistema `ERRNO` cambia a `ERR_TP_MAXTIME` y la ejecución continúa en el gestor de errores.
- Si se activa la entrada digital (parámetro `\DIBreak`) antes de que responda el operador, la variable de sistema `ERRNO` cambia a `ERR_TP_DIBREAK` y la ejecución continúa en el gestor de errores.
- Si se activa la salida digital (parámetro `\DOBBreak`) antes de la acción del operador, la variable de sistema `ERRNO` cambia a `ERR_TP_DOBREAK` y la ejecución continúa en el gestor de errores.
- Si se configura un booleano persistente (parámetro `\PersBoolBreak`) antes de la acción del operador, la variable de sistema `ERRNO` cambia a `ERR_TP_PERSBOOLBREAK` y la ejecución continúa en el gestor de errores.

Esta situación sólo puede ser gestionada desde el gestor de errores.

#### Limitaciones

Evite usar un valor demasiado pequeño para el parámetro de tiempo límite `\MaxTime` si `UIDnumEntry` se ejecuta frecuentemente, por ejemplo en un bucle. Si lo hace, puede dar lugar a un comportamiento impredecible del rendimiento del sistema, por ejemplo la ralentización de la respuesta del FlexPendant.

#### Sintaxis

```
UIDnumEntry '('
  ['\' Header ::= ' <expression (IN) of string>]
```

Continúa en la página siguiente

## 2 Funciones

### 2.210 UIDnumEntry - Introducción de número de usuario

RobotWare - OS

Continuación

```
[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]  
['\ ' PersBoolBreak ':=> <persistent (PERS) of bool>]  
['\ ' PersBoolPassive]  
['\ ' BreakFlag ':=> <var or pers (INOUT) of errnum>]  
['\ ' UIActiveSignal ':=> <variable (VAR) of signaldo>] ''
```

Una función con un valor de retorno del tipo de dato dnum.

#### Información relacionada

| Para obtener más información sobre                               | Consulte   |
|--|--|
| Datos de visualización de iconos                                 | <a href="#">icondata - Datos de visualización de iconos en la página 1683</a>                  |
| Cuadro de mensaje de interacción con el usuario de tipo básico   | <a href="#">UIMsgBox - Cuadro de mensaje de usuario de tipo básico en la página 1002</a>       |
| Cuadro de mensaje de interacción con el usuario de tipo avanzado | <a href="#">UIMessageBox - Cuadro de mensaje de usuario de tipo avanzado en la página 1577</a> |
| Introducción de número de interacción con el usuario             | <a href="#">UINumEntry - Introducción de número de usuario en la página 1585</a>               |
| Ajuste de número de interacción con el usuario                   | <a href="#">UIDnumTune - Ajuste de número de usuario en la página 1561</a>                     |
| Ajuste de número de interacción con el usuario                   | <a href="#">UINumTune - Ajuste de número de usuario en la página 1592</a>                      |
| Introducción alfanumérica de interacción con el usuario          | <a href="#">UIAlphaEntry - Introducción alfanumérica del usuario en la página 1545</a>         |
| Vista de lista de interacción con el usuario                     | <a href="#">UIListView - Vista de lista de usuario en la página 1568</a>                       |
| Sistema conectado al FlexPendant, etc.                           | <a href="#">UIClientExist - Existe cliente de usuario en la página 1552</a>                    |
| Borrado de la ventana de operador                                | <a href="#">TPErase - Borra el texto mostrado en el FlexPendant en la página 877</a>           |

## 2.211 UIDnumTune - Ajuste de número de usuario

### Utilización

UIDnumTune (*User Interaction Number Tune*) se usa para ajustar un valor numérico desde un dispositivo de usuario disponible, como el FlexPendant. Se escribe un mensaje para el operador, que a su vez ajusta un valor numérico. El valor numérico ajustado es comprobado, autorizado y transferido de nuevo al programa.

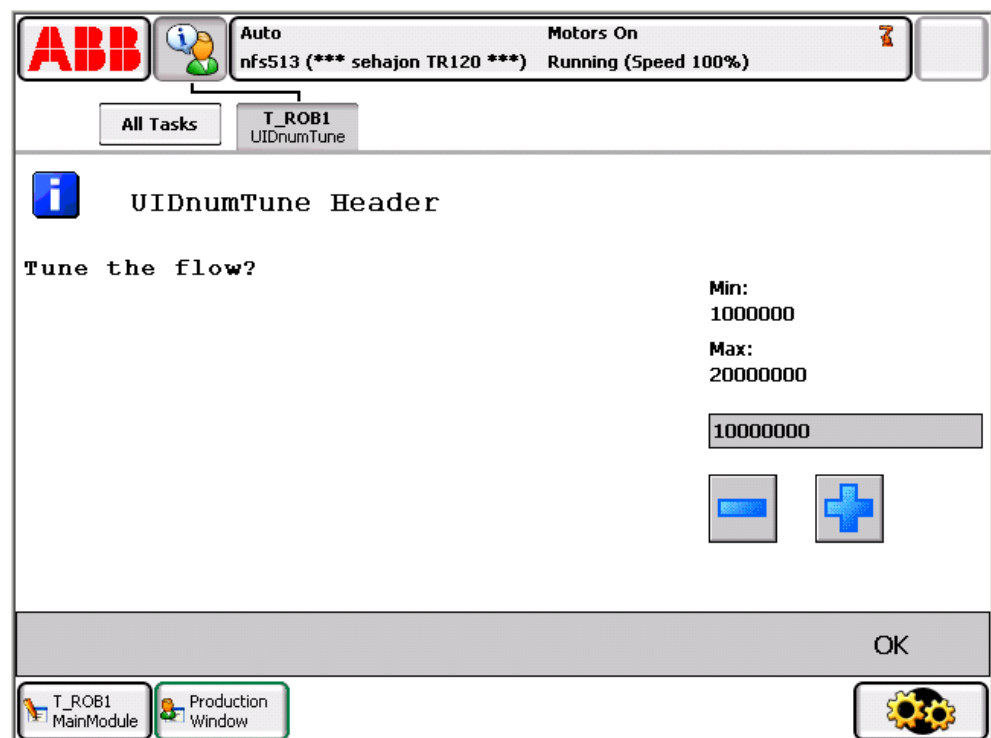
### Ejemplos básicos

El ejemplo siguiente ilustra la función UIDnumTune.

Consulte también [Más ejemplos en la página 1565](#).

#### Ejemplo 1

```
VAR dnum flow;
...
flow := UIDnumTune(
  \Header:="UIDnumTune Header"
  \Message:="Tune the flow?"
  \Icon:=iconInfo,
  10000000,
  1000000
  \MinValue:=1000000
  \MaxValue:=20000000);
```



xx0900001063

Se muestra en el FlexPendant el cuadro de mensaje de ajuste numérico anterior con icono, encabezado, mensaje y valores inicial, de incremento, máximo y mínimo. El cuadro de mensaje comprueba que el operador ajuste el valor de flow con

*Continúa en la página siguiente*

## 2 Funciones

---

### 2.211 UIDnumTune - Ajuste de número de usuario

RobotWare - OS

Continuación

pasos (incrementos) de 1000000, empezando con el valor inicial 10000000 y dentro del rango de valores de 1000000-20000000. La ejecución del programa espera hasta que se presione OK. A continuación, se devuelve el valor numérico seleccionado, que se almacena en la variable `flow`.

---

#### Valor de retorno

Tipo de dato: `dnum`

Esta función devuelve el valor numérico ajustado.

Si la función es interrumpida por `\BreakFlag`, se devuelve el valor `InitValue` especificado.

Si la función se interrumpe desde el gestor de `ERROR`, no se devuelve ningún valor de retorno en absoluto.

---

#### Argumentos

```
UIDnumTune ( [\Header] [\Message] | [\MsgArray] [\Wrap]
             [\Icon]InitValue Increment [\MinValue] [\MaxValue]
             [\MaxTime][\DIBreak] [\DIPassive] [\DOBreak] [\DOPassive]
             [\PersBoolBreak] [\PersBoolPassive] [\BreakFlag]
             [\UIActiveSignal])
```

`[\Header]`

Tipo de dato: `string`

El texto de título que debe escribirse en la parte superior del cuadro de mensaje. Máximo 40 caracteres.

`[\Message]`

Tipo de dato: `string`

Una línea de texto a escribir en la pantalla. Máx. 40 caracteres.

`[\MsgArray]`

**Message Array**

Tipo de dato: `string`

Several text lines from an array to be written on the display.

Sólo es posible usar uno de los parámetros, `\Message` o `\MsgArray`, en cada momento.

El espacio máximo del diseño es de 11 líneas de 40 caracteres cada una.

`[\Wrap]`

Tipo de dato: `switch`

Si se selecciona, todas las cadenas especificadas en el argumento `\MsgArray` se concatenan para formar una cadena con un solo espacio entre las distintas cadenas individuales y distribuida en el número mínimo posible de líneas.

De forma predeterminada, cada una de las cadenas del argumento `\MsgArray` aparece en una línea separada en la pantalla.

`[\Icon]`

Tipo de dato: `icondata`

*Continúa en la página siguiente*

Define el icono a mostrar. Only one of the predefined icons of type icondata can be used. Consulte [Datos predefinidos en la página 1565](#).

De forma predeterminada, no se usa ningún icono.

InitValue

#### *Initial Value*

Tipo de dato: dnum

El valor inicial que se muestra en el cuadro de introducción.

Increment

Tipo de dato: dnum

Este parámetro especifica en qué cantidad debe cambiar el valor al presionar los botones de más o menos.

[\MinValue]

Tipo de dato: dnum

El valor mínimo del valor de retorno.

[\MaxValue]

Tipo de dato: dnum

El valor máximo del valor de retorno.

[\MaxTime]

Tipo de dato: num

El periodo máximo, en segundos, que debe esperar el programa para continuar con la ejecución. Si no se presiona el botón OK en ese periodo, el programa sigue ejecutándose en el gestor de errores, a no ser que se utilice el indicador BreakFlag (que se documenta a continuación). La constante ERR\_TP\_MAXTIME puede usarse para comprobar si ha transcurrido ya el tiempo máximo establecido.

[\DIBreak]

#### *Digital Input Break*

Tipo de dato: signaldi

La señal digital de entrada que puede interrumpir el diálogo con el operador. Si no se presiona el botón OK antes de que la señal cambie a 1 (o si ya tiene el valor 1), el programa sigue ejecutándose en el gestor de errores, a no ser que se utilice el indicador BreakFlag (que se documenta a continuación). La constante ERR\_TP\_DIBREAK puede usarse para comprobar si esto ha ocurrido.

[\DIPassive]

#### *Digital Input Passive*

Tipo de dato: switch

Este modificador redefine el comportamiento predeterminado con el argumento opcional DIBreak. En lugar de reaccionar cuando la señal cambia a 1 (o ya tiene el valor 1), la instrucción debe continuar en el gestor de errores (si no se utiliza BreakFlag) cuando la señal DIBreak cambia a 0 (o ya tiene el valor 0). La constante ERR\_TP\_DIBREAK puede usarse para comprobar si esto ha ocurrido.

*Continúa en la página siguiente*

## 2 Funciones

---

### 2.211 UIDnumTune - Ajuste de número de usuario

RobotWare - OS

Continuación

[ \DOBreak ]

#### *Digital Output Break*

Tipo de dato: signaldo

La señal digital de salida que puede interrumpir el diálogo con el operador. Si no se presiona el botón OK antes de que la señal cambie a 1 (o si ya tiene el valor 1), el programa sigue ejecutándose en el gestor de errores, a no ser que se utilice el indicador `BreakFlag` (que se documenta a continuación). La constante `ERR_TP_DOBREAK` puede usarse para comprobar si esto ha ocurrido.

[ \DOPassive ]

#### *Digital Output Passive*

Tipo de dato: switch

Este modificador redefine el comportamiento predeterminado con el argumento opcional `DOBreak`. En lugar de reaccionar cuando la señal cambia a 1 (o ya tiene el valor 1), la instrucción debe continuar en el gestor de errores (si no se utiliza `BreakFlag`) cuando la señal `DOBreak` cambia a 0 (o ya tiene el valor 0). La constante `ERR_TP_DOBREAK` puede usarse para comprobar si esto ha ocurrido.

[ \PersBoolBreak ]

#### *Persistent Boolean Break*

Tipo de dato: bool

El booleano persistente que puede interrumpir el diálogo con el operador. Si no se selecciona ningún botón cuando el booleano persistente cambia a TRUE (o ya tiene el valor TRUE) entonces el programa continuará ejecutándose en el gestor de errores, a no ser que se utilice `BreakFlag` (que se documenta a continuación). La constante `ERR_TP_PERSBOOLBREAK` puede usarse para comprobar si esto ha ocurrido.

[ \PersBoolPassive ]

#### *Persistent Boolean Passive*

Tipo de dato: switch

Este interruptor redefine el comportamiento con el argumento opcional `PersBoolBreak`. En lugar de reaccionar cuando el booleano persistente cambia a TRUE (o ya tiene el valor TRUE), la instrucción debe continuar en el gestor de errores (si no se utiliza `BreakFlag`) cuando el booleano persistente `PersBoolBreak` cambia a FALSE (o ya tiene el valor FALSE). La constante `ERR_TP_PERSBOOLBREAK` puede usarse para comprobar si esto ha ocurrido.

[ \BreakFlag ]

Tipo de dato: errnum

Una variable que contiene el código de error si se utiliza `MaxTime`, `DIBreak`, `DOBreak`, o `PersBoolBreak`. Si se omite esta variable opcional, se ejecuta el gestor de errores. Las constantes `ERR_TP_MAXTIME`, `ERR_TP_DIBREAK`, `ERR_TP_DOBREAK`, y `ERR_TP_PERSBOOLBREAK` pueden usarse para seleccionar el motivo.

Continúa en la página siguiente



[\UIActiveSignal]

**Tipo de dato:** signaldo

La señal digital de salida utilizada en el argumento opcional `UIActiveSignal` se establece en 1 cuando se activa el cuadro de mensaje en `FlexPendant`. Cuando se ha realizado la selección de usuario y la ejecución continúa, la señal se vuelve a establecer en 0.

No existe ninguna supervisión de parada o reinicio. La señal se establece en 0 cuando la función está preparada o cuando se mueve el PP.

---

#### Ejecución de programas

Se muestra el cuadro de mensaje de ajuste numérico con botones de ajuste +/-, icono, encabezado, líneas de mensaje y valores inicial, de incremento, máximo y mínimo, de acuerdo con los argumentos del programa. La ejecución del programa espera hasta que el usuario haya ajustado el valor numérico y presione OK o hasta que el cuadro de mensaje sea interrumpido por un tiempo límite o una acción de señal. El valor numérico introducido y el motivo de la interrupción se devuelven al programa.

El nuevo cuadro de mensaje del nivel de rutina `TRAP` toma el foco del cuadro de mensaje del nivel básico.

---

#### Datos predefinidos

```
!Icons:
CONST icondata iconNone := 0;
CONST icondata iconInfo := 1;
CONST icondata iconWarning := 2;
CONST icondata iconError := 3;
```

---

#### Más ejemplos

El ejemplo siguiente ilustra la función `UIDnumTune`.

##### Ejemplo 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
```

*Continúa en la página siguiente*

## 2 Funciones

### 2.211 UIDnumTune - Ajuste de número de usuario

RobotWare - OS

Continuación

ENDTEST

Se muestra el cuadro de ajuste de mensaje y el operador puede ajustar el valor numérico y presionar OK. El cuadro de mensaje también puede ser interrumpido con un tiempo límite o una interrupción con una señal digital de entrada. Es posible determinar el motivo desde el programa y tomar las acciones adecuadas.

#### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                          | Causa del error   |
|---------------------------------|---|
| <code>ERR_NO_ALIASIO_DEF</code> | La variable de señal es una variable declarada en RAPID y no se ha conectado a una señal de E/S definida en la configuración de E/S con la instrucción <code>AliasIO</code> .         |
| <code>ERR_TP_NO_CLIENT</code>   | No hay ningún cliente, por ejemplo un FlexPendant, a cargo de la instrucción.   |
| <code>ERR_UI_INITVALUE</code>   | El valor inicial (parámetro <code>\InitValue</code> ) no se ha especificado dentro del rango del valor mínimo y máximo (parámetros <code>\MinValue</code> y <code>\MaxValue</code> ). |
| <code>ERR_UI_MAXMIN</code>      | El valor mínimo (parámetro <code>\MinValue</code> ) es mayor que el valor máximo (parámetro <code>\MaxValue</code> ).   |

Si no se usa el parámetro `\BreakFlag`, estas situaciones pueden ser gestionadas en el gestor de errores:

- Si se alcanza el tiempo límite (parámetro `\MaxTime`) antes de que responda el operador, la variable de sistema `ERRNO` cambia a `ERR_TP_MAXTIME` y la ejecución continúa en el gestor de errores.
- Si se activa la entrada digital (parámetro `\DIBreak`) antes de que responda el operador, la variable de sistema `ERRNO` cambia a `ERR_TP_DIBREAK` y la ejecución continúa en el gestor de errores.
- Si se activa la salida digital (parámetro `\DOBBreak`) antes de la acción del operador, la variable de sistema `ERRNO` cambia a `ERR_TP_DOBREAK` y la ejecución continúa en el gestor de errores.
- Si se configura un booleano persistente (parámetro `\PersBoolBreak`) antes de la acción del operador, la variable de sistema `ERRNO` cambia a `ERR_TP_PERSBOOLBREAK` y la ejecución continúa en el gestor de errores.

#### Limitaciones

Evite usar un valor demasiado pequeño para el parámetro de tiempo límite `\MaxTime` si `UIDnumTune` se ejecuta frecuentemente, por ejemplo en un bucle. Si lo hace, puede dar lugar a un comportamiento impredecible del rendimiento del sistema, por ejemplo la ralentización de la respuesta del FlexPendant.

#### Sintaxis

```
UIDnumTune '('  
  ['\ Header :=' <expression (IN) of string>]  
  ['\ Message :=' <expression (IN) of string> ]  
  | ['\ MsgArray :=' <array {*} (IN) of string>]  
  ['\ Wrap]
```

Continúa en la página siguiente

```
[ '\ ' 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 ]
[ '\ ' PersBoolBreak ':=' <persistent (PERS) of bool> ]
[ '\ ' PersBoolPassive ]
[ '\ ' BreakFlag ':=' <var or pers (INOUT) of errnum> ]
[ '\ ' UIActiveSignal ':=' <variable (VAR) of signaldo> ] '' ]
```

Una función con un valor de retorno del tipo de dato dnum.

### Información relacionada

| Para obtener más información sobre                               | Consulte   |
|--|--|
| Datos de visualización de iconos                                 | <a href="#">icondata - Datos de visualización de iconos en la página 1683</a>                  |
| Cuadro de mensaje de interacción con el usuario de tipo básico   | <a href="#">UIMsgBox - Cuadro de mensaje de usuario de tipo básico en la página 1002</a>       |
| Cuadro de mensaje de interacción con el usuario de tipo avanzado | <a href="#">UIMessageBox - Cuadro de mensaje de usuario de tipo avanzado en la página 1577</a> |
| Introducción de número de interacción con el usuario             | <a href="#">UIDnumEntry - Introducción de número de usuario en la página 1554</a>              |
| Introducción de número de interacción con el usuario             | <a href="#">UINumEntry - Introducción de número de usuario en la página 1585</a>               |
| Ajuste de número de interacción con el usuario                   | <a href="#">UINumTune - Ajuste de número de usuario en la página 1592</a>                      |
| Introducción alfanumérica de interacción con el usuario          | <a href="#">UIAlphaEntry - Introducción alfanumérica del usuario en la página 1545</a>         |
| Vista de lista de interacción con el usuario                     | <a href="#">UIListView - Vista de lista de usuario en la página 1568</a>                       |
| Sistema conectado al FlexPendant, etc.                           | <a href="#">UIClientExist - Existe cliente de usuario en la página 1552</a>                    |
| Borrado de la ventana de operador                                | <a href="#">TPErase - Borra el texto mostrado en el FlexPendant en la página 877</a>           |

## 2 Funciones

---

### 2.212 UICollection - Vista de lista de usuario

RobotWare - OS

### 2.212 UICollection - Vista de lista de usuario

---

#### Utilización

UICollection (*User Interaction List View*) se utiliza para definir listas de menús con textos e iconos opcionales en el dispositivo de usuario disponible, como el FlexPendant. El menú tiene dos estilos diferentes: uno con botones de validación y otro que reacciona instantáneamente a la selección del usuario.

---

#### Ejemplos básicos

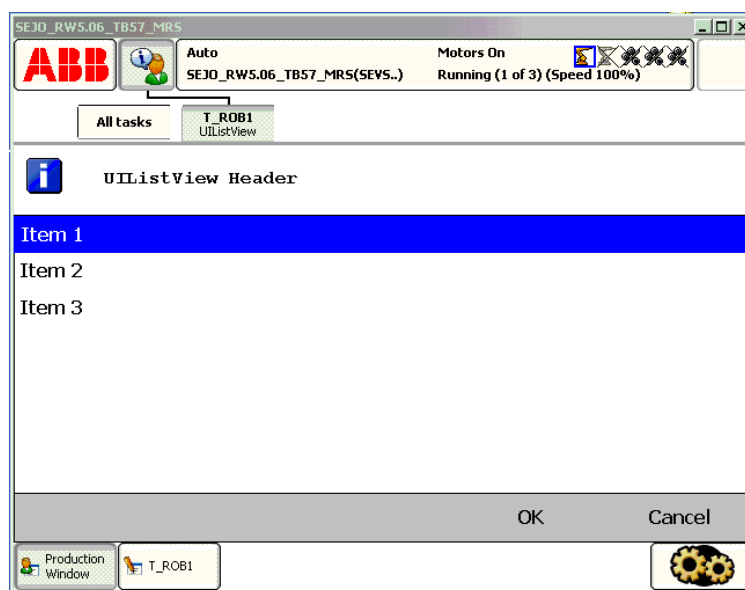
El ejemplo siguiente ilustra la función UICollection.

Consulte también [Más ejemplos en la página 1574](#).

#### Ejemplo 1

```
CONST listitem list{3} := [ ["", "Item 1"], ["", "Item 2"],
                           ["", "Item3"] ];
VAR num list_item;
VAR btnres button_answer;
...
list_item := UICollection (
  \Result:=button_answer
  \Header:="UICollection 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
```

*Continúa en la página siguiente*



xx0500002416

Se muestra en el FlexPendant la lista de menús con icono, encabezado, menú con los elementos de Item 1 ... Item 3 y los botones. La ejecución espera hasta que se presiona OK o Cancelar. Tanto la selección en la lista como el botón presionado se transfieren al programa.

### Valor de retorno

Tipo de dato: num

Esta función devuelve el elemento seleccionado por el usuario en el menú y correspondiente al índice de la matriz especificada en el parámetro `ListItems`.

Si la función es interrumpida por `\BreakFlag`:

- Si se especifica el parámetro `\DefaultIndex`, se devuelve este número de índice.
- Si no se especifica el parámetro `\DefaultIndex`, se devuelve 0.

Si la función se interrumpe desde el gestor de `ERROR`, no se devuelve ningún valor de retorno en absoluto.

### Argumentos

```
UListView ( [\Result] [\Header] ListItems [\Buttons] | [\BtnArray]
[\Icon] [\DefaultIndex ] [\MaxTime] [\DIBreak] [\DIPassive]
[\DOBreak] [\DOPassive] [\PersBoolBreak] [\PersBoolPassive]
[\BreakFlag] [\UIActiveSignal])
```

`[\Result]`

Tipo de dato: btnres

El valor numérico del botón seleccionado en el cuadro de menú de lista.

Si se usa el argumento `\Buttons`, se devuelven constantes simbólicas del tipo `btnres`. Si se usa el argumento `\BtnArray`, se devuelve el índice de matriz correspondiente.

*Continúa en la página siguiente*

## 2 Funciones

---

### 2.212 UICollection - Vista de lista de usuario

RobotWare - OS

Continuación

El argumento `\Result` con el valor da lugar a 0 si no se cumple ninguna de las condiciones siguientes:

- No se utiliza ninguno de los parámetros `\Buttons` o `\BtnArray`.
- Se utiliza el argumento `\Buttons:=btnNone`.
- si la función es interrumpida por `\BreakFlag` o el gestor de `ERROR`:

Consulte [Datos predefinidos en la página 1573](#).

`[\Header]`

Tipo de dato: `string`

El texto de título que debe escribirse en la parte superior del cuadro de menú de lista. Máximo 40 caracteres.

`ListItems`

Tipo de dato: `listitem`

Una matriz con uno o varios elementos de menú de lista para su visualización y compuestos de:

Componente `image` de tipo `string`:

El nombre de la imagen de icono que debe utilizarse. Para iniciar sus propias imágenes, las imágenes deben estar situadas en el directorio `HOME`: del sistema activo o directamente en el sistema activo.

La recomendación es situar los archivos en el directorio `HOME`: de forma que se incluyan en las operaciones de copia de seguridad y restauración.

Se requiere un Reinicio, tras lo cual el FlexPendant carga las imágenes.

Una exigencia del sistema es que se use la opción de RobotWare *FlexPendant Interface*.

La imagen a mostrar puede tener 28 píxeles de anchura y altura. Si la imagen es mayor, será redimensionada a únicamente 28 \* 28 píxeles.

No es posible especificar ningún valor exacto en cuanto al tamaño que una imagen puede tener o el número de imágenes que es posible cargar en el FlexPendant. Depende del tamaño de los demás archivos cargados en el FlexPendant. La ejecución de los programas continúa simplemente si se usa una imagen que no está cargada en el FlexPendant.

Utilice una cadena vacía "" o la constante `stEmpty` si no desea mostrar ningún icono.

Componente `text` de tipo `string`:

- El texto de la línea de menú a mostrar.
- Máximo de 75 caracteres por cada elemento del menú de lista.

`[\Buttons]`

Tipo de dato: `buttondata`

Define los pulsadores que se desea mostrar. Sólo puede mostrarse una de las combinaciones de botones predefinidas del tipo `buttondata`. Consulte [Datos predefinidos en la página 1573](#).

*Continúa en la página siguiente*

[\BtnArray]

#### **Button Array**

Tipo de dato: `string`

Definición propia de pulsadores almacenada en una matriz de cadenas. Esta función devuelve el índice de matriz cuando se selecciona la cadena correspondiente.

Sólo es posible usar uno de los parámetros, `\Buttons` o `\BtnArray`, en cada momento. Si no se usa ninguno de los parámetros `\Buttons` o `\BtnArray` o el argumento `\Buttons:=btnNone`, la lista de menú reacciona instantáneamente ante la selección por parte del usuario.

Como máximo, es posible utilizar 5 botones de 42 caracteres cada uno.

[\Icon]

Tipo de dato: `icondata`

Define el icono a mostrar. Sólo puede usarse uno de los iconos predefinidos de tipo `icondata`.

De forma predeterminada, no se usa ningún icono. Consulte [Datos predefinidos en la página 1573](#).

[\DefaultIndex]

Tipo de dato: `num`

La selección predeterminada del usuario en el menú de lista y correspondiente al índice de la matriz especificada en el parámetro `ListItems`.

[\MaxTime]

Tipo de dato: `num`

El periodo máximo, en segundos, que debe esperar el programa para continuar con la ejecución. Si no se presiona este botón o no selecciona ninguna opción en ese periodo, el programa sigue ejecutándose en el gestor de errores, a no ser que se utilice el indicador `BreakFlag` (que se documenta a continuación). La constante `ERR_TP_MAXTIME` puede usarse para comprobar si ha transcurrido ya el tiempo máximo establecido.

[\DIBreak]

#### **Digital Input Break**

Tipo de dato: `signal`

La señal digital de entrada que puede interrumpir el diálogo con el operador. Si no se presiona ningún botón o no se selecciona ninguna opción antes de que la señal cambie a 1 (o si ya tiene el valor 1), el programa sigue ejecutándose en el gestor de errores, a no ser que se utilice el indicador `BreakFlag` (que se documenta a continuación). La constante `ERR_TP_DIBREAK` puede usarse para comprobar si esto ha ocurrido.

[\DIPassive]

#### **Digital Input Passive**

Tipo de dato: `switch`

*Continúa en la página siguiente*

## 2 Funciones

---

### 2.212 UICollection - Vista de lista de usuario

RobotWare - OS

Continuación

Este modificador redefine el comportamiento predeterminado con el argumento opcional `DIBreak`. En lugar de reaccionar cuando la señal cambia a 1 (o ya tiene el valor 1), la instrucción debe continuar en el gestor de errores (si no se utiliza `BreakFlag`) cuando la señal `DIBreak` cambia a 0 (o ya tiene el valor 0). La constante `ERR_TP_DIBREAK` puede usarse para comprobar si esto ha ocurrido.

`[\DOBreak]()`

#### *Digital Output Break*

Tipo de dato: `signaldo`

La señal digital de salida que puede interrumpir el diálogo con el operador. Si no se presiona ningún botón o no se selecciona ninguna opción antes de que la señal cambie a 1 (o si ya tiene el valor 1), el programa sigue ejecutándose en el gestor de errores, a no ser que se utilice el indicador `BreakFlag` (que se documenta a continuación). La constante `ERR_TP_DOBREAK` puede usarse para comprobar si esto ha ocurrido.

`[\DOPassive]`

#### *Digital Output Passive*

Tipo de dato: `switch`

Este modificador redefine el comportamiento predeterminado con el argumento opcional `DOBreak`. En lugar de reaccionar cuando la señal cambia a 1 (o ya tiene el valor 1), la instrucción debe continuar en el gestor de errores (si no se utiliza `BreakFlag`) cuando la señal `DOBreak` cambia a 0 (o ya tiene el valor 0). La constante `ERR_TP_DOBREAK` puede usarse para comprobar si esto ha ocurrido.

`[\PersBoolBreak]`

#### *Persistent Boolean Break*

Tipo de dato: `bool`

El booleano persistente que puede interrumpir el diálogo con el operador. Si no se selecciona ningún botón cuando el booleano persistente cambia a `TRUE` (o ya tiene el valor `TRUE`) entonces el programa continuará ejecutándose en el gestor de errores, a no ser que se utilice `BreakFlag` (que se documenta a continuación). La constante `ERR_TP_PERSBOOLBREAK` puede usarse para comprobar si esto ha ocurrido.

`[\PersBoolPassive]`

#### *Persistent Boolean Passive*

Tipo de dato: `switch`

Este interruptor redefine el comportamiento con el argumento opcional `PersBoolBreak`. En lugar de reaccionar cuando el booleano persistente cambia a `TRUE` (o ya tiene el valor `TRUE`), la instrucción debe continuar en el gestor de errores (si no se utiliza `BreakFlag`) cuando el booleano persistente `PersBoolBreak` cambia a `FALSE` (o ya tiene el valor `FALSE`). La constante `ERR_TP_PERSBOOLBREAK` puede usarse para comprobar si esto ha ocurrido.

`[\BreakFlag]`

Tipo de dato: `errnum`

*Continúa en la página siguiente*



Una variable que contiene el código de error si se utiliza MaxTime, DIBreak, DOBreak, o PersBoolBreak. Si se omite esta variable opcional, se ejecuta el gestor de errores. Las constantes ERR\_TP\_MAXTIME, ERR\_TP\_DIBREAK, ERR\_TP\_DOBREAK, y ERR\_TP\_PERSBOOLBREAK pueden usarse para seleccionar el motivo.

[\UIActiveSignal]

Tipo de dato: signaldo

La señal digital de salida utilizada en el argumento opcional UIActiveSignal se establece en 1 cuando se activa el cuadro de mensaje en FlexPendant. Cuando se ha realizado la selección de usuario y la ejecución continúa, la señal se vuelve a establecer en 0.

No existe ninguna supervisión de parada o reinicio. La señal se establece en 0 cuando la función está preparada o cuando se mueve el PP.

---

### Ejecución de programas

Se muestra una lista de menú con icono, título, elementos de lista y un elemento predeterminado, de acuerdo con los argumentos del programa. La ejecución del programa espera hasta que el operador haya seleccionado una opción o hasta que la lista de menú sea interrumpida por un tiempo límite o una acción de señal. El elemento de lista seleccionado y el motivo de la interrupción se devuelven al programa.

La nueva lista de menú del nivel de rutina TRAP toma el foco de la lista de menú del nivel básico.

---

### Datos predefinidos

```
!Icons:
CONST icondata iconNone := 0;
CONST icondata iconInfo := 1;
CONST icondata iconWarning := 2;
CONST icondata iconError := 3;
!Buttons:
CONST buttodata btnNone := -1;
CONST buttodata btnOK := 0;
CONST buttodata btnAbrtRtryIgn := 1;
CONST buttodata btnOKCancel := 2;
CONST buttodata btnRetryCancel := 3;
CONST buttodata btnYesNo := 4;
CONST buttodata 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;
```

*Continúa en la página siguiente*

## 2 Funciones

### 2.212 UICollection - Vista de lista de usuario

RobotWare - OS

Continuación

#### Más ejemplos

El ejemplo siguiente ilustra la función `UICollection`.

#### Ejemplo 1

```
CONST listitem list{2} := [ ["", "Calibrate tool1"], ["", "Calibrate
    tool2"] ];
VAR num list_item;
VAR errnum err_var;
...
list_item := UICollection
    ( \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
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
```

Se muestra el cuadro de mensaje y el operador puede seleccionar un elemento de la lista. El cuadro de mensaje también puede ser interrumpido con un tiempo límite o una interrupción con una señal digital de entrada. Es posible determinar el motivo desde el programa y tomar las acciones adecuadas.

#### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                          | Causa del error  |
|---------------------------------|--|
| <code>ERR_NO_ALIASIO_DEF</code> | La variable de señal es una variable declarada en <code>RAPID</code> y no se ha conectado a una señal de E/S definida en la configuración de E/S con la instrucción <code>AliasIO</code> . |
| <code>ERR_TP_NO_CLIENT</code>   | No hay ningún cliente, por ejemplo un <code>FlexPendant</code> , a cargo de la instrucción.  |

Si no se usa el parámetro `\BreakFlag`, estas situaciones pueden ser gestionadas en el gestor de errores:

- Si se alcanza el tiempo límite (parámetro `\MaxTime`) antes de que responda el operador, la variable de sistema `ERRNO` cambia a `ERR_TP_MAXTIME` y la ejecución continúa en el gestor de errores.

Continúa en la página siguiente

- Si se activa la entrada digital (parámetro `\DIBreak`) antes de que responda el operador, la variable de sistema `ERRNO` cambia a `ERR_TP_DIBREAK` y la ejecución continúa en el gestor de errores.
- Si se activa la salida digital (parámetro `\DOBreak`) antes de la acción del operador, la variable de sistema `ERRNO` cambia a `ERR_TP_DOBREAK` y la ejecución continúa en el gestor de errores.
- Si se configura un booleano persistente (parámetro `\PersBoolBreak`) antes de la acción del operador, la variable de sistema `ERRNO` cambia a `ERR_TP_PERSBOOLBREAK` y la ejecución continúa en el gestor de errores.

#### Limitaciones

Evite usar un valor demasiado pequeño para el parámetro de tiempo límite `\MaxTime` si `UICollection` se ejecuta frecuentemente, por ejemplo en un bucle. Si lo hace, puede dar lugar a un comportamiento impredecible del rendimiento del sistema, por ejemplo la ralentización de la respuesta del FlexPendant.

#### Sintaxis

```
UICollection '('
  [['\ 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]
  [['\ PersBoolBreak ']:= <persistent (PERS) of bool>]
  [['\ PersBoolPassive]
  [['\ BreakFlag ']:= <var or pers (INOUT) of errnum>]
  [['\ UIActiveSignal ']:= <variable (VAR) of signaldo>] ')'

```

Una función con un valor de retorno del tipo de dato `num`.

#### Información relacionada

| Para obtener más información sobre                             | Consulte   |
|--|--|
| Datos de visualización de iconos                               | <a href="#">icondata - Datos de visualización de iconos en la página 1683</a>            |
| Datos de pulsador  | <a href="#">buttondata - Datos de botón en la página 1614</a>                            |
| Datos de resultado de pulsador                                 | <a href="#">btnres - Datos de resultado de pulsador en la página 1611</a>                |
| Estructura de datos de elementos de lista                      | <a href="#">listitem - Estructura de datos de elementos de lista en la página 1693</a>   |
| Cuadro de mensaje de interacción con el usuario de tipo básico | <a href="#">UIMsgBox - Cuadro de mensaje de usuario de tipo básico en la página 1002</a> |

Continúa en la página siguiente

## 2 Funciones

---

### 2.212 UIListView - Vista de lista de usuario

RobotWare - OS

Continuación

| Para obtener más información sobre                               | Consulte   |
|--|--|
| Cuadro de mensaje de interacción con el usuario de tipo avanzado | <a href="#">UIMessageBox - Cuadro de mensaje de usuario de tipo avanzado en la página 1577</a> |
| Introducción de número de interacción con el usuario             | <a href="#">UINumEntry - Introducción de número de usuario en la página 1585</a>               |
| Ajuste de número de interacción con el usuario                   | <a href="#">UINumTune - Ajuste de número de usuario en la página 1592</a>                      |
| Introducción alfanumérica de interacción con el usuario          | <a href="#">UIAlphaEntry - Introducción alfanumérica del usuario en la página 1545</a>         |
| Sistema conectado al FlexPendant, etc.                           | <a href="#">UIClientExist - Existe cliente de usuario en la página 1552</a>                    |
| Borrado de la ventana de operador                                | <a href="#">TPErase - Borra el texto mostrado en el FlexPendant en la página 877</a>           |

## 2.213 UIMessageBox - Cuadro de mensaje de usuario de tipo avanzado

### Utilización

UIMessageBox (*User Interaction Message Box*) se usa para comunicarse con el usuario del sistema de robot a través de un dispositivo de usuario disponible, como el FlexPendant. Se escribe un mensaje para el operador, que a su vez responde con la selección de un botón. A continuación, la selección de usuario se transfiere al programa.

### Ejemplos básicos

El ejemplo siguiente ilustra la función UIMessageBox.

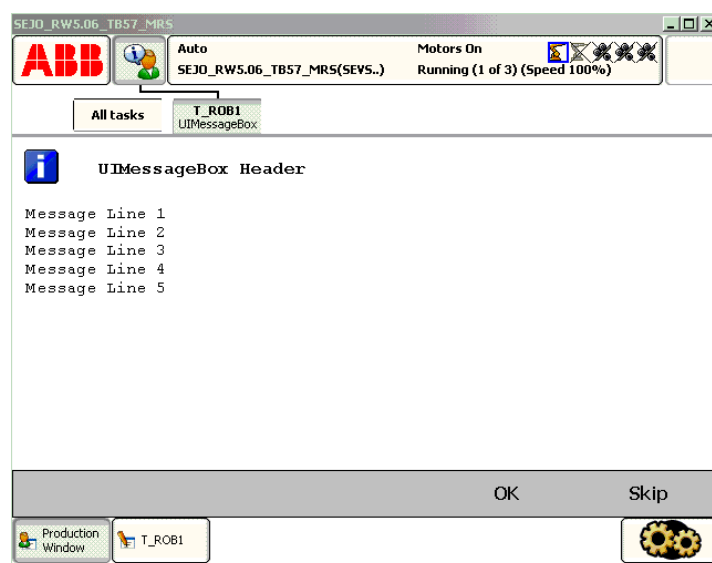
Consulte también [Más ejemplos en la página 1582](#).

#### Ejemplo 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

*Continúa en la página siguiente*

## 2 Funciones

---

### 2.213 UIMessageBox - Cuadro de mensaje de usuario de tipo avanzado

RobotWare - OS

Continuación

Se muestra en el FlexPendant el cuadro de mensaje con icono, encabezado, mensaje y pulsadores definidos por el usuario. La ejecución espera hasta que se presiona OK u Omitir. En otras palabras, se asigna a `answer` el valor 1 (OK) ó 2 (Omitir) en función de cuál de los botones se presione (índice de matriz correspondiente).



#### Nota

De Message Line 1 a Message Line 5 se muestran en las líneas separadas de la 1 a la 5 (el modificador `\Wrap` no se utiliza).

---

#### Valor de retorno

Tipo de dato: `btnres`

El valor numérico del botón seleccionado en el cuadro de mensaje.

Si se usa el argumento `\Buttons`, se devuelven constantes simbólicas del tipo `btnres`.

Si se usa el argumento `\BtnArray`, se devuelve el índice de matriz correspondiente.

Si la función es interrumpida por `\BreakFlag` o `\Buttons:=btnNone`:

- Si se especifica el parámetro `\DefaultBtn`, se devuelve este número de índice.
- Si no se especifica el parámetro `\DefaultBtn`, se devuelve `resUnkwn` igual a 0.

Si la función se interrumpe desde el gestor de `ERROR`, no tiene ningún valor de retorno en absoluto.

---

#### Argumentos

```
UIMessageBox ( [\Header] [\Message] | [\MsgArray] [\Wrap][\Buttons]  
| [\BtnArray] [\DefaultBtn] [\Icon][\Image] [\MaxTime]  
[\DIBreak] [\DIPassive] [\DOBreak] [\DOPassive]  
[\PersBoolBreak] [\PersBoolPassive] [\BreakFlag]  
[\UIActiveSignal])
```

`[\Header]`

Tipo de dato: `string`

El texto de título que debe escribirse en la parte superior del cuadro de mensaje. Máximo 40 caracteres.

`[\Message]`

Tipo de dato: `string`

Una línea de texto a escribir en la pantalla. Máximo 55 caracteres.

`[\MsgArray]`

*Message Array*

Tipo de dato: `string`

Several text lines from an array to be written on the display.

Continúa en la página siguiente

Sólo es posible usar uno de los parámetros, `\Message` o `\MsgArray`, en cada momento.

El espacio máximo del diseño es de 11 líneas de 55 caracteres cada una.

[`\Wrap`]

Tipo de dato: `switch`

Si se selecciona, todas las cadenas especificadas en el argumento `\MsgArray` se concatenan para formar una cadena con un solo espacio entre las distintas cadenas individuales y distribuida en el número mínimo posible de líneas.

De forma predeterminada, cada una de las cadenas del argumento `\MsgArray` aparece en una línea separada en la pantalla.

[`\Buttons`]

Tipo de dato: `buttondata`

Defines the push buttons to be displayed. Only one of the predefined buttons combination of type `buttondata` can be used. Consulte [Datos predefinidos en la página 1582](#).

De forma predeterminada, el sistema muestra el botón OK.

[`\BtnArray`]

**Button Array**

Tipo de dato: `string`

Definición propia de pulsadores almacenada en una matriz de cadenas. Esta función devuelve el índice de matriz cuando se selecciona la cadena correspondiente.

Sólo es posible usar uno de los parámetros, `\Buttons` o `\BtnArray`, en cada momento.

Como máximo, es posible utilizar 5 botones de 42 caracteres cada uno.

[`\DefaultBtn`]

**Default Button**

Tipo de dato: `btnres`

Permite especificar un valor que debe ser devuelto si el cuadro de mensaje es interrumpido por `\MaxTime`, `\DIBreak` o `\DOBreak`. Es posible especificar la constante simbólica predefinida del tipo `btnres` o cualquier valor definido por el usuario. Consulte [Datos predefinidos en la página 1582](#).

[`\Icon`]

Tipo de dato: `icondata`

Define el icono a mostrar. Sólo puede usarse uno de los iconos predefinidos de tipo `icondata`. Consulte [Datos predefinidos en la página 1582](#).

De forma predeterminada, no se usa ningún icono.

[`\Image`]

Tipo de dato: `string`

*Continúa en la página siguiente*

## 2 Funciones

---

### 2.213 UIMessageBox - Cuadro de mensaje de usuario de tipo avanzado

RobotWare - OS

Continuación

El nombre de la imagen que debe utilizarse. Para iniciar sus propias imágenes, las imágenes deben estar situadas en el directorio `HOME :` del sistema activo o directamente en el sistema activo.

La recomendación es situar los archivos en el directorio `HOME :` de forma que se incluyan en las operaciones de copia de seguridad y restauración.

Se requiere un **Reinicio**, tras lo cual el FlexPendant carga las imágenes.

Una exigencia del sistema es que se use la opción de RobotWare *FlexPendant Interface*.

La imagen a mostrar puede tener 185 píxeles de anchura y 300 píxeles de altura. Si la imagen tiene un tamaño mayor, sólo se muestran 185 \* 300 píxeles de la imagen a partir de su parte superior izquierda.

No es posible especificar ningún valor exacto en cuanto al tamaño que una imagen puede tener o el número de imágenes que es posible cargar en el FlexPendant. Depende del tamaño de los demás archivos cargados en el FlexPendant. La ejecución de los programas continúa simplemente si se usa una imagen que no está cargada en el FlexPendant.

[ \MaxTime ]

Tipo de dato: num

El periodo máximo, en segundos, que debe esperar el programa para continuar con la ejecución. Si no se selecciona ningún botón en ese periodo, el programa sigue ejecutándose en el gestor de errores, a no ser que se utilice el indicador `BreakFlag` (que se documenta a continuación). La constante `ERR_TP_MAXTIME` puede usarse para comprobar si ha transcurrido ya el tiempo máximo establecido.

[ \DIBreak ]

**Digital Input Break**

Tipo de dato: signaldi

La señal digital de entrada que puede interrumpir el diálogo con el operador. Si no se selecciona ningún botón cuando la señal cambia a 1 (o si ya tiene el valor 1), el programa sigue ejecutándose en el gestor de errores, a no ser que se utilice el indicador `BreakFlag` (que se documenta a continuación). La constante `ERR_TP_DIBREAK` puede usarse para comprobar si esto ha ocurrido.

[ \DIPassive ]

**Digital Input Passive**

Tipo de dato: switch

Este modificador redefine el comportamiento predeterminado con el argumento opcional `DIBreak`. En lugar de reaccionar cuando la señal cambia a 1 (o ya tiene el valor 1), la instrucción debe continuar en el gestor de errores (si no se utiliza `BreakFlag`) cuando la señal `DIBreak` cambia a 0 (o ya tiene el valor 0). La constante `ERR_TP_DIBREAK` puede usarse para comprobar si esto ha ocurrido.

[ \DOBreak ]

**Digital Output Break**

Tipo de dato: signaldo

Continúa en la página siguiente



La señal digital de salida que puede interrumpir el diálogo con el operador. Si no se selecciona ningún botón cuando la señal cambia a 1 (o si ya tiene el valor 1), el programa sigue ejecutándose en el gestor de errores, a no ser que se utilice el indicador `BreakFlag` (que se documenta a continuación). La constante `ERR_TP_DOBREAK` puede usarse para comprobar si esto ha ocurrido.

[`\DOPassive`]

#### *Digital Output Passive*

Tipo de dato: `switch`

Este modificador redefine el comportamiento predeterminado con el argumento opcional `DOBreak`. En lugar de reaccionar cuando la señal cambia a 1 (o ya tiene el valor 1), la instrucción debe continuar en el gestor de errores (si no se utiliza `BreakFlag`) cuando la señal `DOBreak` cambia a 0 (o ya tiene el valor 0). La constante `ERR_TP_DOBREAK` puede usarse para comprobar si esto ha ocurrido.

[`\PersBoolBreak`]

#### *Persistent Boolean Break*

Tipo de dato: `bool`

El booleano persistente que puede interrumpir el diálogo con el operador. Si no se selecciona ningún botón cuando el booleano persistente cambia a `TRUE` (o ya tiene el valor `TRUE`) entonces el programa continuará ejecutándose en el gestor de errores, a no ser que se utilice `BreakFlag` (que se documenta a continuación). La constante `ERR_TP_PERSBOOLBREAK` puede usarse para comprobar si esto ha ocurrido.

[`\PersBoolPassive`]

#### *Persistent Boolean Passive*

Tipo de dato: `switch`

Este interruptor redefine el comportamiento con el argumento opcional `PersBoolBreak`. En lugar de reaccionar cuando el booleano persistente cambia a `TRUE` (o ya tiene el valor `TRUE`), la instrucción debe continuar en el gestor de errores (si no se utiliza `BreakFlag`) cuando el booleano persistente `PersBoolBreak` cambia a `FALSE` (o ya tiene el valor `FALSE`). La constante `ERR_TP_PERSBOOLBREAK` puede usarse para comprobar si esto ha ocurrido.

[`\BreakFlag`]

Tipo de dato: `errnum`

Una variable que contiene el código de error si se utiliza `MaxTime`, `DIBreak`, `DOBreak`, o `PersBoolBreak`. Si se omite esta variable opcional, se ejecuta el gestor de errores. Las constantes `ERR_TP_MAXTIME`, `ERR_TP_DIBREAK`, `ERR_TP_DOBREAK`, y `ERR_TP_PERSBOOLBREAK` pueden usarse para seleccionar el motivo.

[`\UIActiveSignal`]

Tipo de dato: `signaldo`

La señal digital de salida utilizada en el argumento opcional `UIActiveSignal` se establece en 1 cuando se activa el cuadro de mensaje en `FlexPendant`. Cuando

*Continúa en la página siguiente*

## 2 Funciones

---

### 2.213 UIMessageBox - Cuadro de mensaje de usuario de tipo avanzado

RobotWare - OS

Continuación

se ha realizado la selección de usuario y la ejecución continúa, la señal se vuelve a establecer en 0.

No existe ninguna supervisión de parada o reinicio. La señal se establece en 0 cuando la función está preparada o cuando se mueve el PP.

---

#### Ejecución de programas

Se muestra un cuadro de mensaje con icono, título, líneas de mensaje, imágenes y botones, de acuerdo con los argumentos del programa. La ejecución del programa espera hasta que el usuario seleccione un botón o que el cuadro de mensaje sea interrumpido por un tiempo límite o una acción de señal. La opción seleccionada por el usuario y el motivo de la interrupción se devuelven al programa.

El nuevo cuadro de mensaje del nivel de rutina TRAP toma el foco del cuadro de mensaje del nivel básico.

---

#### Datos predefinidos

```
!Icons:
  CONST icondata iconNone := 0;
  CONST icondata iconInfo := 1;
  CONST icondata iconWarning := 2;
  CONST icondata iconError := 3;
!Buttons:
  CONST buttodata btnNone := -1;
  CONST buttodata btnOK := 0;
  CONST buttodata btnAbtrRtryIgn := 1;
  CONST buttodata btnOKCancel := 2;
  CONST buttodata btnRetryCancel := 3;
  CONST buttodata btnYesNo := 4;
  CONST buttodata 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;
```

---

#### Más ejemplos

El ejemplo siguiente ilustra la función UIMessageBox.

##### Ejemplo 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);
```

*Continúa en la página siguiente*

```

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

```

Se muestra el cuadro de mensaje y el operador puede responder OK o Cancelar. El cuadro de mensaje también puede ser interrumpido con un tiempo límite o una interrupción con una señal digital de entrada. Es posible determinar el motivo desde el programa.

### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                          | Causa del error   |
|---------------------------------|---|
| <code>ERR_NO_ALIASIO_DEF</code> | La variable de señal es una variable declarada en RAPID y no se ha conectado a una señal de E/S definida en la configuración de E/S con la instrucción <code>AliasIO</code> . |
| <code>ERR_TP_NO_CLIEN</code>    | No hay ningún cliente, por ejemplo un FlexPendant, a cargo de la instrucción.   |

Si no se usa el parámetro `\BreakFlag`, estas situaciones pueden ser gestionadas en el gestor de errores:

- Si se alcanza el tiempo límite (parámetro `\MaxTime`) antes de que responda el operador, la variable de sistema `ERRNO` cambia a `ERR_TP_MAXTIME` y la ejecución continúa en el gestor de errores.
- Si se activa la entrada digital (parámetro `\DIBreak`) antes de que responda el operador, la variable de sistema `ERRNO` cambia a `ERR_TP_DIBREAK` y la ejecución continúa en el gestor de errores.
- Si se activa la salida digital (parámetro `\DOBreak`) antes de la acción del operador, la variable de sistema `ERRNO` cambia a `ERR_TP_DOBREAK` y la ejecución continúa en el gestor de errores.
- Si se configura un booleano persistente (parámetro `\PersBoolBreak`) antes de la acción del operador, la variable de sistema `ERRNO` cambia a `ERR_TP_PERSBOOLBREAK` y la ejecución continúa en el gestor de errores.

### Limitaciones

Evite usar un valor demasiado pequeño para el parámetro de tiempo límite `\MaxTime` si `UIMessageBox` se ejecuta frecuentemente, por ejemplo en un bucle. Si lo hace, puede dar lugar a un comportamiento impredecible del rendimiento del sistema, por ejemplo la ralentización de la respuesta del FlexPendant.

*Continúa en la página siguiente*

## 2 Funciones

### 2.213 UIAlertView - Cuadro de mensaje de usuario de tipo avanzado

RobotWare - OS

Continuación

#### Sintaxis

```
UIAlertView '('  
  ['\' 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]  
  ['\' PersBoolBreak :=' <persistent (PERS) of bool>]  
  ['\' PersBoolPassive]  
  ['\' BreakFlag :=' <var or pers (INOUT) of errnum>]  
  ['\' UIActiveSignal :=' <variable (VAR) of signaldo>] ')'
```

Una función con un valor de retorno del tipo de dato btnres.

#### Información relacionada

| Para obtener más información sobre                             | Consulte  |
|--|---|
| Datos de visualización de iconos                               | <a href="#">icondata - Datos de visualización de iconos en la página 1683</a>               |
| Datos de pulsador  | <a href="#">buttondata - Datos de botón en la página 1614</a>                               |
| Datos de resultado de pulsador                                 | <a href="#">btnres - Datos de resultado de pulsador en la página 1611</a>                   |
| Cuadro de mensaje de interacción con el usuario de tipo básico | <a href="#">UIAlertView - Cuadro de mensaje de usuario de tipo básico en la página 1002</a> |
| Introducción de número de interacción con el usuario           | <a href="#">UINumEntry - Introducción de número de usuario en la página 1585</a>            |
| Ajuste de número de interacción con el usuario                 | <a href="#">UINumTune - Ajuste de número de usuario en la página 1592</a>                   |
| Introducción alfanumérica de interacción con el usuario        | <a href="#">UIAlphaEntry - Introducción alfanumérica del usuario en la página 1545</a>      |
| Vista de lista de interacción con el usuario                   | <a href="#">UIListView - Vista de lista de usuario en la página 1568</a>                    |
| Sistema conectado al FlexPendant, etc.                         | <a href="#">UIClientExist - Existe cliente de usuario en la página 1552</a>                 |
| <i>FlexPendant interface</i>                                   | <a href="#">Especificaciones del producto - Controller software IRC5</a>                    |
| Borrado de la ventana de operador                              | <a href="#">TPERase - Borra el texto mostrado en el FlexPendant en la página 877</a>        |

## 2.214 UINumEntry - Introducción de número de usuario

## Utilización

UINumEntry (*User Interaction Number Entry*) se usa para introducir un valor numérico desde un dispositivo de usuario disponible, como el FlexPendant. Se escribe un mensaje para el operador, que a su vez responde con un valor numérico. El valor numérico es comprobado, autorizado y transferido de nuevo al programa.

## Ejemplos básicos

El ejemplo siguiente ilustra la función UINumEntry.

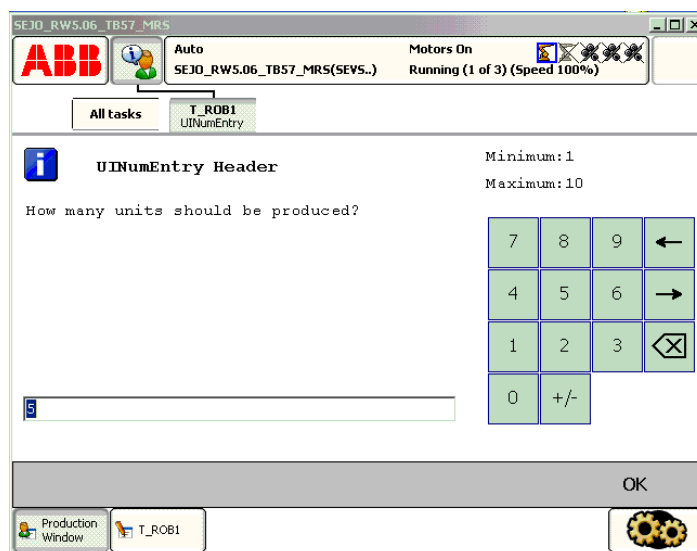
Consulte también [Más ejemplos en la página 1589](#).

## Ejemplo 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

Se muestra en el FlexPendant el cuadro de mensaje numérico con icono, encabezado, mensaje y valores inicial, máximo y mínimo. El cuadro de mensaje comprueba que el operador seleccione un entero perteneciente al rango de valores. La ejecución del programa espera hasta que se presione OK. A continuación, se

*Continúa en la página siguiente*

## 2 Funciones

---

### 2.214 UINumEntry - Introducción de número de usuario

RobotWare - OS

Continuación

devuelve el valor numérico seleccionado. A continuación, la rutina `produce_part` se repite el número de veces especificado a través del FlexPendant.

---

#### Valor de retorno

Tipo de dato: `num`

Esta función devuelve el valor numérico introducido.

Si la función es interrumpida por `\BreakFlag`:

- Si se especifica el parámetro `\InitValue`, se devuelve este valor.
- Si no se especifica el parámetro `\InitValue`, se devuelve el valor 0.

Si la función se interrumpe desde el gestor de `ERROR`, no tiene ningún valor de retorno en absoluto.

---

#### Argumentos

```
UINumEntry ( [\Header] [\Message] | [\MsgArray]
             [\Wrap][\Icon][\InitValue] [\MinValue] [\MaxValue]
             [\AsInteger][\MaxTime] [\DIBreak] [\DIPassive] [\DOBreak]
             [\DOPassive] [\PersBoolBreak] [\PersBoolPassive] [\BreakFlag]
             [\UIActiveSignal])
```

`[\Header]`

Tipo de dato: `string`

El texto de título que debe escribirse en la parte superior del cuadro de mensaje. Máximo 40 caracteres.

`[\Message]`

Tipo de dato: `string`

Una línea de texto a escribir en la pantalla. Máximo 40 caracteres.

`[\MsgArray]`

**Message Array**

Tipo de dato: `string`

Several text lines from an array to be written on the display.

Sólo es posible usar uno de los parámetros, `\Message` o `\MsgArray`, en cada momento.

El espacio máximo del diseño es de 9 líneas de 40 caracteres cada una.

`[\Wrap]`

Tipo de dato: `switch`

Si se selecciona, todas las cadenas especificadas en el argumento `\MsgArray` se concatenan para formar una cadena con un solo espacio entre las distintas cadenas individuales y distribuida en el número mínimo posible de líneas.

De forma predeterminada, cada una de las cadenas del argumento `\MsgArray` aparece en una línea separada en la pantalla.

`[\Icon]`

Tipo de dato: `icondata`

*Continúa en la página siguiente*

---

Define el icono a mostrar. Sólo puede usarse uno de los iconos predefinidos de tipo `icondata`. Consulte [Datos predefinidos en la página 1589](#).

De forma predeterminada, no se usa ningún icono.

`[\InitValue]`

Tipo de dato: `num`

El valor inicial que se muestra en el cuadro de introducción.

`[\MinValue]`

Tipo de dato: `num`

El valor mínimo del valor de retorno.

`[\MaxValue]`

Tipo de dato: `num`

El valor máximo del valor de retorno.

`[\AsInteger]`

Tipo de dato: `switch`

Elimina el punto decimal del teclado numérico para garantizar que el valor devuelto sea un entero.

`[\MaxTime]`

Tipo de dato: `num`

El periodo máximo, en segundos, que debe esperar el programa para continuar con la ejecución. Si no se presiona el botón OK en ese periodo, el programa sigue ejecutándose en el gestor de errores, a no ser que se utilice el indicador `BreakFlag` (que se documenta a continuación). La constante `ERR_TP_MAXTIME` puede usarse para comprobar si ha transcurrido ya el tiempo máximo establecido.

`[\DIBreak]`

**Digital Input Break**

Tipo de dato: `signal`

La señal digital de entrada que puede interrumpir el diálogo con el operador. Si no se presiona el botón OK antes de que la señal cambie a 1 (o si ya tiene el valor 1), el programa sigue ejecutándose en el gestor de errores, a no ser que se utilice el indicador `BreakFlag` (que se documenta a continuación). La constante `ERR_TP_DIBREAK` puede usarse para comprobar si esto ha ocurrido.

`[\DIPassive]`

**Digital Input Passive**

Tipo de dato: `switch`

Este modificador redefine el comportamiento predeterminado con el argumento opcional `DIBreak`. En lugar de reaccionar cuando la señal cambia a 1 (o ya tiene el valor 1), la instrucción debe continuar en el gestor de errores (si no se utiliza `BreakFlag`) cuando la señal `DIBreak` cambia a 0 (o ya tiene el valor 0). La constante `ERR_TP_DIBREAK` puede usarse para comprobar si esto ha ocurrido.

*Continúa en la página siguiente*

## 2 Funciones

---

### 2.214 UINumEntry - Introducción de número de usuario

RobotWare - OS

Continuación

[ \DOBreak ]

#### *Digital Output Break*

Tipo de dato: `signaldo`

La señal digital de salida que puede interrumpir el diálogo con el operador. Si no se presiona el botón OK antes de que la señal cambie a 1 (o si ya tiene el valor 1), el programa sigue ejecutándose en el gestor de errores, a no ser que se utilice el indicador `BreakFlag` (que se documenta a continuación). La constante `ERR_TP_DOBREAK` puede usarse para comprobar si esto ha ocurrido.

[ \DOPassive ]

#### *Digital Output Passive*

Tipo de dato: `switch`

Este modificador redefine el comportamiento predeterminado con el argumento opcional `DOBreak`. En lugar de reaccionar cuando la señal cambia a 1 (o ya tiene el valor 1), la instrucción debe continuar en el gestor de errores (si no se utiliza `BreakFlag`) cuando la señal `DOBreak` cambia a 0 (o ya tiene el valor 0). La constante `ERR_TP_DOBREAK` puede usarse para comprobar si esto ha ocurrido.

[ \PersBoolBreak ]

#### *Persistent Boolean Break*

Tipo de dato: `bool`

El booleano persistente que puede interrumpir el diálogo con el operador. Si no se selecciona ningún botón cuando el booleano persistente cambia a TRUE (o ya tiene el valor TRUE) entonces el programa continuará ejecutándose en el gestor de errores, a no ser que se utilice `BreakFlag` (que se documenta a continuación). La constante `ERR_TP_PERSBOOLBREAK` puede usarse para comprobar si esto ha ocurrido.

[ \PersBoolPassive ]

#### *Persistent Boolean Passive*

Tipo de dato: `switch`

Este interruptor redefine el comportamiento con el argumento opcional `PersBoolBreak`. En lugar de reaccionar cuando el booleano persistente cambia a TRUE (o ya tiene el valor TRUE), la instrucción debe continuar en el gestor de errores (si no se utiliza `BreakFlag`) cuando el booleano persistente `PersBoolBreak` cambia a FALSE (o ya tiene el valor FALSE). La constante `ERR_TP_PERSBOOLBREAK` puede usarse para comprobar si esto ha ocurrido.

[ \BreakFlag ]

Tipo de dato: `errnum`

Una variable que contiene el código de error si se utiliza `MaxTime`, `DIBreak`, `DOBreak`, o `PersBoolBreak`. Si se omite esta variable opcional, se ejecuta el gestor de errores. Las constantes `ERR_TP_MAXTIME`, `ERR_TP_DIBREAK`, `ERR_TP_DOBREAK`, y `ERR_TP_PERSBOOLBREAK` pueden usarse para seleccionar el motivo.

Continúa en la página siguiente



[\UIActiveSignal]

**Tipo de dato:** signaldo

La señal digital de salida utilizada en el argumento opcional `UIActiveSignal` se establece en 1 cuando se activa el cuadro de mensaje en `FlexPendant`. Cuando se ha realizado la selección de usuario y la ejecución continúa, la señal se vuelve a establecer en 0.

No existe ninguna supervisión de parada o reinicio. La señal se establece en 0 cuando la función está preparada o cuando se mueve el PP.

---

#### Ejecución de programas

Se muestra el cuadro de mensaje numérico, con teclado numérico, icono, encabezado, líneas de mensaje y valores inicial, máximo y mínimo, de acuerdo con los argumentos del programa. La ejecución del programa espera hasta que el usuario haya introducido un valor numérico autorizado y presione OK o hasta que el cuadro de mensaje sea interrumpido por un tiempo límite o una acción de señal. El valor numérico introducido y el motivo de la interrupción se devuelven al programa.

El nuevo cuadro de mensaje del nivel de rutina TRAP toma el foco del cuadro de mensaje del nivel básico.

---

#### Datos predefinidos

```
!Icons:
  CONST icondata iconNone := 0;
  CONST icondata iconInfo := 1;
  CONST icondata iconWarning := 2;
  CONST icondata iconError := 3;
```

---

#### Más ejemplos

El ejemplo siguiente ilustra la función `UINumEntry`.

##### Ejemplo 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
```

*Continúa en la página siguiente*

## 2 Funciones

### 2.214 UINumEntry - Introducción de número de usuario

RobotWare - OS

Continuación

Se muestra el cuadro de mensaje y el operador puede introducir un valor numérico y presionar OK. El cuadro de mensaje también puede ser interrumpido con un tiempo límite o una interrupción con una señal digital de entrada. Es posible determinar el motivo desde el programa y tomar las acciones adecuadas.

#### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                          | Causa del error   |
|---------------------------------|---|
| <code>ERR_NO_ALIASIO_DEF</code> | La variable de señal es una variable declarada en RAPID y no se ha conectado a una señal de E/S definida en la configuración de E/S con la instrucción <code>AliasIO</code> .         |
| <code>ERR_TP_NO_CLIENT</code>   | No hay ningún cliente, por ejemplo un <code>FlexPendant</code> , a cargo de la instrucción.   |
| <code>ERR_UI_INITVALUE</code>   | El valor inicial (parámetro <code>\InitValue</code> ) no se ha especificado dentro del rango del valor mínimo y máximo (parámetros <code>\MinValue</code> y <code>\MaxValue</code> ). |
| <code>ERR_UI_MAXMIN</code>      | El valor mínimo (parámetro <code>\MinValue</code> ) es mayor que el valor máximo (parámetro <code>\MaxValue</code> ).   |
| <code>ERR_UI_NOTINT</code>      | El valor inicial (parámetro <code>\InitValue</code> ) no es un entero tal y como se especifica en el parámetro <code>\AsInteger</code> .  |

Si no se usa el parámetro `\BreakFlag`, estas situaciones pueden ser gestionadas en el gestor de errores:

- Si se alcanza el tiempo límite (parámetro `\MaxTime`) antes de que responda el operador, la variable de sistema `ERRNO` cambia a `ERR_TP_MAXTIME` y la ejecución continúa en el gestor de errores.
- Si se activa la entrada digital (parámetro `\DIBreak`) antes de que responda el operador, la variable de sistema `ERRNO` cambia a `ERR_TP_DIBREAK` y la ejecución continúa en el gestor de errores.
- Si se activa la salida digital (parámetro `\DOBreak`) antes de la acción del operador, la variable de sistema `ERRNO` cambia a `ERR_TP_DOBREAK` y la ejecución continúa en el gestor de errores.
- Si se configura un booleano persistente (parámetro `\PersBoolBreak`) antes de la acción del operador, la variable de sistema `ERRNO` cambia a `ERR_TP_PERSBOOLBREAK` y la ejecución continúa en el gestor de errores.

#### Limitaciones

Evite usar un valor demasiado pequeño para el parámetro de tiempo límite `\MaxTime` si `UINumEntry` se ejecuta frecuentemente, por ejemplo en un bucle. Si lo hace, puede dar lugar a un comportamiento impredecible del rendimiento del sistema, por ejemplo la ralentización de la respuesta del `FlexPendant`.

#### Sintaxis

```
UINumEntry '('  
  ['\ Header ::= ' <expression (IN) of string>]  
  [Message ::= ' <expression (IN) of string> ]  
  | ['\ MsgArray ::= ' <array {*} (IN) of string>]
```

Continúa en la página siguiente

```

['\ ' 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]
['\ ' PersBoolBreak ':=' <persistent (PERS) of bool>]
['\ ' PersBoolPassive]
['\ ' BreakFlag ':=' <var or pers (INOUT) of errnum>]
['\ ' UIActiveSignal ':=' <variable (VAR) of signaldo>] ')'

```

Una función con un valor de retorno del tipo de dato num.

### Información relacionada

| Para obtener más información sobre                               | Consulte   |
|--|--|
| Datos de visualización de iconos                                 | <a href="#">icondata - Datos de visualización de iconos en la página 1683</a>                  |
| Cuadro de mensaje de interacción con el usuario de tipo básico   | <a href="#">UIMsgBox - Cuadro de mensaje de usuario de tipo básico en la página 1002</a>       |
| Cuadro de mensaje de interacción con el usuario de tipo avanzado | <a href="#">UIMessageBox - Cuadro de mensaje de usuario de tipo avanzado en la página 1577</a> |
| Ajuste de número de interacción con el usuario                   | <a href="#">UINumTune - Ajuste de número de usuario en la página 1592</a>                      |
| Introducción alfanumérica de interacción con el usuario          | <a href="#">UIAlphaEntry - Introducción alfanumérica del usuario en la página 1545</a>         |
| Vista de lista de interacción con el usuario                     | <a href="#">UIListView - Vista de lista de usuario en la página 1568</a>                       |
| Sistema conectado al FlexPendant, etc.                           | <a href="#">UIClientExist - Existe cliente de usuario en la página 1552</a>                    |
| Borrado de la ventana de operador                                | <a href="#">TPERase - Borra el texto mostrado en el FlexPendant en la página 877</a>           |

## 2 Funciones

### 2.215 UINumTune - Ajuste de número de usuario RobotWare - OS

### 2.215 UINumTune - Ajuste de número de usuario

#### Utilización

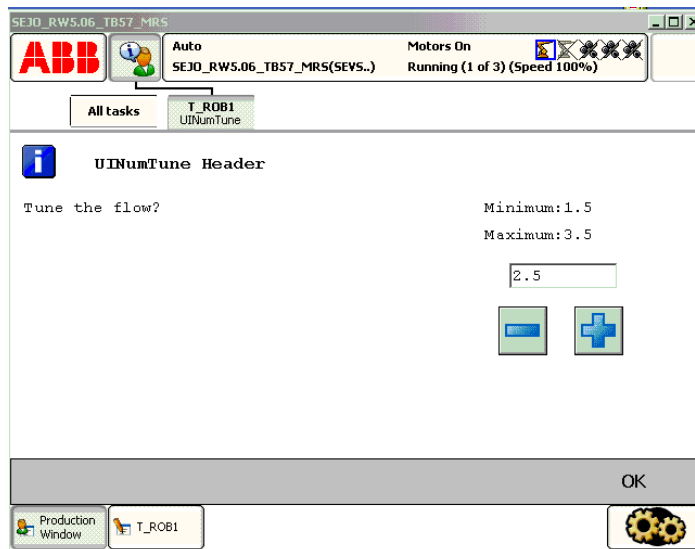
UINumTune (*User Interaction Number Tune*) se usa para ajustar un valor numérico desde un dispositivo de usuario disponible, como el FlexPendant. Se escribe un mensaje para el operador, que a su vez ajusta un valor numérico. El valor numérico ajustado es comprobado, autorizado y transferido de nuevo al programa.

#### Ejemplos básicos

El ejemplo siguiente ilustra la función UINumTune.  
Consulte también [Más ejemplos en la página 1596](#).

#### Ejemplo 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

Se muestra en el FlexPendant el cuadro de mensaje de ajuste numérico anterior con icono, encabezado, mensaje y valores inicial, de incremento, máximo y mínimo. El cuadro de mensaje comprueba que el operador ajuste el valor de flujo con pasos de 0.1, empezando con el valor inicial 2.5 y dentro del rango de valores de 1.5-3.5. La ejecución del programa espera hasta que se presione OK. A continuación, se devuelve el valor numérico seleccionado, que se almacena en la variable flow.

*Continúa en la página siguiente*

---

#### Valor de retorno

Tipo de dato: num

Esta función devuelve el valor numérico ajustado.

Si la función es interrumpida por `\BreakFlag`, se devuelve el valor `\InitValueInitValue` especificado.

Si la función se interrumpe desde el gestor de `ERROR`, no se devuelve ningún valor de retorno en absoluto.

---

#### Argumentos

```
UINumTune ( [\Header] [\Message] | [\MsgArray] [\Wrap] [\Icon]  
           InitValue Increment [\MinValue] [\MaxValue] [\MaxTime]  
           [\DIBreak] [\DIPassive] [\DOBreak] [\DOPassive]  
           [\PersBoolBreak] [\PersBoolPassive] [\BreakFlag]  
           [\UIActiveSignal])
```

`[\Header]`

Tipo de dato: string

El texto de título que debe escribirse en la parte superior del cuadro de mensaje. Máximo 40 caracteres.

`[\Message]`

Tipo de dato: string

Una línea de texto a escribir en la pantalla. Máximo 40 caracteres.

`[\MsgArray]`

*Message Array*

Tipo de dato: string

Varias líneas de texto de una matriz a escribir en la pantalla.

Sólo es posible usar uno de los parámetros, `\Message` o `\MsgArray`, en cada momento.

El espacio máximo del diseño es de 11 líneas de 40 caracteres cada una.

`[\Wrap]`

Tipo de dato: switch

Si se selecciona, todas las cadenas especificadas en el argumento `\MsgArray` se concatenan para formar una cadena con un solo espacio entre las distintas cadenas individuales y distribuida en el número mínimo posible de líneas.

De forma predeterminada, cada una de las cadenas del argumento `\MsgArray` aparece en una línea separada en la pantalla.

`[\Icon]`

Tipo de dato: icondata

Define el icono a mostrar. Sólo puede usarse uno de los iconos predefinidos de tipo `icondata`. Consulte [Datos predefinidos en la página 1596](#).

De forma predeterminada, no se usa ningún icono.

*Continúa en la página siguiente*

## 2 Funciones

---

### 2.215 UINumTune - Ajuste de número de usuario

RobotWare - OS

Continuación

InitValue

Tipo de dato: num

El valor inicial que se muestra en el cuadro de introducción.

Increment

Tipo de dato: num

Este parámetro especifica en qué cantidad debe cambiar el valor al presionar los botones de más o menos.

[\MinValue]

Tipo de dato: num

El valor mínimo del valor de retorno.

[\MaxValue]

Tipo de dato: num

El valor máximo del valor de retorno.

[\MaxTime]

Tipo de dato: num

El periodo máximo, en segundos, que debe esperar el programa para continuar con la ejecución. Si no se presiona el botón OK en ese periodo, el programa sigue ejecutándose en el gestor de errores, a no ser que se utilice el indicador `BreakFlag` (que se documenta a continuación). La constante `ERR_TP_MAXTIME` puede usarse para comprobar si ha transcurrido ya el tiempo máximo establecido.

[\DIBreak]

**Digital Input Break**

Tipo de dato: `signal di`

La señal digital de entrada que puede interrumpir el diálogo con el operador. Si no se presiona el botón OK antes de que la señal cambie a 1 (o si ya tiene el valor 1), el programa sigue ejecutándose en el gestor de errores, a no ser que se utilice el indicador `BreakFlag` (que se documenta a continuación). La constante `ERR_TP_DIBREAK` puede usarse para comprobar si esto ha ocurrido.

[\DIPassive]

**Digital Input Passive**

Tipo de dato: `switch`

Este modificador redefine el comportamiento predeterminado con el argumento opcional `DIBreak`. En lugar de reaccionar cuando la señal cambia a 1 (o ya tiene el valor 1), la instrucción debe continuar en el gestor de errores (si no se utiliza `BreakFlag`) cuando la señal `DIBreak` cambia a 0 (o ya tiene el valor 0). La constante `ERR_TP_DIBREAK` puede usarse para comprobar si esto ha ocurrido.

[\DOBBreak]

**Digital Output Break**

Tipo de dato: `signal do`

Continúa en la página siguiente

La señal digital de salida que puede interrumpir el diálogo con el operador. Si no se presiona el botón OK antes de que la señal cambie a 1 (o si ya tiene el valor 1), el programa sigue ejecutándose en el gestor de errores, a no ser que se utilice el indicador `BreakFlag` (que se documenta a continuación). La constante `ERR_TP_DOBREAK` puede usarse para comprobar si esto ha ocurrido.

[`\DOPassive`]

#### *Digital Output Passive*

Tipo de dato: `switch`

Este modificador redefine el comportamiento predeterminado con el argumento opcional `DOBreak`. En lugar de reaccionar cuando la señal cambia a 1 (o ya tiene el valor 1), la instrucción debe continuar en el gestor de errores (si no se utiliza `BreakFlag`) cuando la señal `DOBreak` cambia a 0 (o ya tiene el valor 0). La constante `ERR_TP_DOBREAK` puede usarse para comprobar si esto ha ocurrido.

[`\PersBoolBreak`]

#### *Persistent Boolean Break*

Tipo de dato: `bool`

El booleano persistente que puede interrumpir el diálogo con el operador. Si no se selecciona ningún botón cuando el booleano persistente cambia a `TRUE` (o ya tiene el valor `TRUE`) entonces el programa continuará ejecutándose en el gestor de errores, a no ser que se utilice `BreakFlag` (que se documenta a continuación). La constante `ERR_TP_PERSBOOLBREAK` puede usarse para comprobar si esto ha ocurrido.

[`\PersBoolPassive`]

#### *Persistent Boolean Passive*

Tipo de dato: `switch`

Este interruptor redefine el comportamiento con el argumento opcional `PersBoolBreak`. En lugar de reaccionar cuando el booleano persistente cambia a `TRUE` (o ya tiene el valor `TRUE`), la instrucción debe continuar en el gestor de errores (si no se utiliza `BreakFlag`) cuando el booleano persistente `PersBoolBreak` cambia a `FALSE` (o ya tiene el valor `FALSE`). La constante `ERR_TP_PERSBOOLBREAK` puede usarse para comprobar si esto ha ocurrido.

[`\BreakFlag`]

Tipo de dato: `errnum`

Una variable que contiene el código de error si se utiliza `MaxTime`, `DIBreak`, `DOBreak`, o `PersBoolBreak`. Si se omite esta variable opcional, se ejecuta el gestor de errores. Las constantes `ERR_TP_MAXTIME`, `ERR_TP_DIBREAK`, `ERR_TP_DOBREAK`, y `ERR_TP_PERSBOOLBREAK` pueden usarse para seleccionar el motivo.

[`\UIActiveSignal`]

Tipo de dato: `signaldo`

La señal digital de salida utilizada en el argumento opcional `UIActiveSignal` se establece en 1 cuando se activa el cuadro de mensaje en `FlexPendant`. Cuando

*Continúa en la página siguiente*

## 2 Funciones

---

### 2.215 UINumTune - Ajuste de número de usuario

RobotWare - OS

Continuación

se ha realizado la selección de usuario y la ejecución continúa, la señal se vuelve a establecer en 0.

No existe ninguna supervisión de parada o reinicio. La señal se establece en 0 cuando la función está preparada o cuando se mueve el PP.

---

#### Ejecución de programas

Se muestra el cuadro de mensaje de ajuste numérico con botones de ajuste +/-, icono, encabezado, líneas de mensaje y valores inicial, de incremento, máximo y mínimo, de acuerdo con los argumentos del programa. La ejecución del programa espera hasta que el usuario haya ajustado el valor numérico y presione OK o hasta que el cuadro de mensaje sea interrumpido por un tiempo límite o una acción de señal. El valor numérico introducido y el motivo de la interrupción se devuelven al programa.

El nuevo cuadro de mensaje del nivel de rutina TRAP toma el foco del cuadro de mensaje del nivel básico.

---

#### Datos predefinidos

```
!Icons:
  CONST icondata iconNone := 0;
  CONST icondata iconInfo := 1;
  CONST icondata iconWarning := 2;
  CONST icondata iconError := 3;
```

---

#### Más ejemplos

El ejemplo siguiente ilustra la función UINumTune.

##### Ejemplo 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
```

Se muestra el cuadro de ajuste de mensaje y el operador puede ajustar el valor numérico y presionar OK. El cuadro de mensaje también puede ser interrumpido

*Continúa en la página siguiente*



con un tiempo límite o una interrupción con una señal digital de entrada. Es posible determinar el motivo desde el programa y tomar las acciones adecuadas.

#### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                          | Causa del error   |
|---------------------------------|---|
| <code>ERR_NO_ALIASIO_DEF</code> | La variable de señal es una variable declarada en <code>RAPID</code> . No se ha conectado a una señal E/S definida en la configuración de E/S con la instrucción <code>AliasIO</code> . |
| <code>ERR_TP_NO_CLIENT</code>   | No hay ningún cliente, por ejemplo un <code>FlexPendant</code> , a cargo de la instrucción.   |
| <code>ERR_UI_INITVALUE</code>   | El valor inicial (parámetro <code>\InitValue</code> ) no se ha especificado dentro del rango del valor mínimo y máximo (parámetros <code>\MinValue</code> y <code>\MaxValue</code> ).   |
| <code>ERR_UI_MAXMIN</code>      | El valor mínimo (parámetro <code>\MinValue</code> ) es mayor que el valor máximo (parámetro <code>\MaxValue</code> ).   |

Si no se usa el parámetro `\BreakFlag`, estas situaciones pueden ser gestionadas en el gestor de errores:

- Si se alcanza el tiempo límite (parámetro `\MaxTime`) antes de que responda el operador, la variable de sistema `ERRNO` cambia a `ERR_TP_MAXTIME` y la ejecución continúa en el gestor de errores.
- Si se activa la entrada digital (parámetro `\DIBreak`) antes de que responda el operador, la variable de sistema `ERRNO` cambia a `ERR_TP_DIBREAK` y la ejecución continúa en el gestor de errores.
- Si se activa la salida digital (parámetro `\DOBreak`) antes de la acción del operador, la variable de sistema `ERRNO` cambia a `ERR_TP_DOBREAK` y la ejecución continúa en el gestor de errores.
- Si se configura un booleano persistente (parámetro `\PersBoolBreak`) antes de la acción del operador, la variable de sistema `ERRNO` cambia a `ERR_TP_PERSBOOLBREAK` y la ejecución continúa en el gestor de errores.

#### Limitaciones

Evite usar un valor demasiado pequeño para el parámetro de tiempo límite `\MaxTime` si `UINumTune` se ejecuta frecuentemente, por ejemplo en un bucle. Si lo hace, puede dar lugar a un comportamiento impredecible del rendimiento del sistema, por ejemplo la ralentización de la respuesta del `FlexPendant`.

#### Sintaxis

```
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>]
```

*Continúa en la página siguiente*

## 2 Funciones

### 2.215 UINumTune - Ajuste de número de usuario

RobotWare - OS

Continuación

```
['\' 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]  
['\' PersBoolBreak ':=' <persistent (PERS) of bool>]  
['\' PersBoolPassive]  
['\' BreakFlag ':=' <var or pers (INOUT) of errnum>]  
['\' UIActiveSignal ':=' <variable (VAR) of signaldo>] ')'
```

Una función con un valor de retorno del tipo de dato num.

#### Información relacionada

| Para obtener más información sobre                               | Consulte   |
|--|--|
| Datos de visualización de iconos                                 | <a href="#">icondata - Datos de visualización de iconos en la página 1683</a>                  |
| Cuadro de mensaje de interacción con el usuario de tipo básico   | <a href="#">UIMsgBox - Cuadro de mensaje de usuario de tipo básico en la página 1002</a>       |
| Cuadro de mensaje de interacción con el usuario de tipo avanzado | <a href="#">UIMessageBox - Cuadro de mensaje de usuario de tipo avanzado en la página 1577</a> |
| Introducción de número de interacción con el usuario             | <a href="#">UINumEntry - Introducción de número de usuario en la página 1585</a>               |
| Introducción alfanumérica de interacción con el usuario          | <a href="#">UIAlphaEntry - Introducción alfanumérica del usuario en la página 1545</a>         |
| Vista de lista de interacción con el usuario                     | <a href="#">UIListView - Vista de lista de usuario en la página 1568</a>                       |
| Sistema conectado al FlexPendant, etc.                           | <a href="#">UIClientExist - Existe cliente de usuario en la página 1552</a>                    |
| Borrado de la ventana de operador                                | <a href="#">TPErase - Borra el texto mostrado en el FlexPendant en la página 877</a>           |

## 2.216 ValidIO - Señal de E/S válida para su uso

### Utilización

ValidIO se utiliza para comprobar si la señal de E/S especificada puede utilizarse sin ningún error en este momento.

### Ejemplos básicos

El ejemplo siguiente ilustra la función ValidIO.

#### Ejemplo 1

```
IF ValidIO(mydosignal) SetDO mydosignal, 1;
```

Establezca la señal digital de salida `mydosignal` en 1 si su dispositivo de E/S está en funcionamiento.

### Valor de retorno

Tipo de dato: `bool`

Devuelve `TRUE` si la señal de E/S es válida y el dispositivo de E/S de la señal está en funcionamiento.

Devuelve `FALSE` si el dispositivo de E/S no está en funcionamiento o si no se ha ejecutado ninguna instrucción `AliasIO` para conectar una variable de señal declarada en el programa de RAPID a una señal definida en la configuración de E/S.

### Argumentos

ValidIO (Signal)

Signal

Tipo de dato: `signalxx`

El nombre de la señal de E/S. Debe ser el tipo de dato `signaldo`, `signaldi`, `signalgo`, `signalgi`, `signalao` o `signalai`.

### Ejecución de programas

Comportamiento de la ejecución:

- Comprobar si la señal de E/S es válida
- Compruebe si el dispositivo de E/S de la señal está en funcionamiento.

No se genera ningún mensaje de error.

### Sintaxis

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

Una función con un valor de retorno del tipo de dato `bool`.

### Información relacionada

| Para obtener más información sobre | Consulte   |
|------------------------------------|--|
| Instrucciones de entrada/salida    | <i>Manual de referencia técnica - RAPID Overview</i> |

*Continúa en la página siguiente*

## 2 Funciones

---

2.216 ValidIO - Señal de E/S válida para su uso

RobotWare - OS

Continuación

| Para obtener más información sobre                 | Consulte  |
|--|---|
| Funcionalidad de entrada/salida en general         | <i>Manual de referencia técnica - RAPID Overview</i>  |
| Configuración de E/S                               | <i>Manual de referencia técnica - Parámetros del sistema</i>                                      |
| Definir una señal de E/S con un nombre de alias    | <a href="#"><i>AliasIO - Define una señal de E/S con un nombre de alias en la página 37</i></a>   |
| Lectura de un atributo de un parámetro del sistema | <a href="#"><i>ReadCfgData - Lee un atributo de un parámetro del sistema en la página 582</i></a> |

---

## 2.217 ValToStr - Convierte un valor en una cadena

---

### Utilización

ValToStr (*Value To String*) se utiliza para convertir un valor de cualquier tipo de dato en una cadena.

---

### Ejemplos básicos

Los siguientes ejemplos ilustran la función ValToStr.

#### Ejemplo 1

```
VAR string str;
VAR pos p := [100,200,300];
str := ValToStr(p);
```

Se asigna a la variable `str` el valor "[100,200,300]".

#### Ejemplo 2

```
str := ValToStr(TRUE);
```

Se asigna a la variable `str` el valor `TRUE`.

#### Ejemplo 3

```
str := ValToStr(1.234567890123456789);
```

Se asigna a la variable `str` el valor "1.23456789012346".

#### Ejemplo 4

```
VAR num numtype:=1.234567890123456789;
```

```
str := ValToStr(numtype);
```

Se asigna a la variable `str` el valor "1.23457".

#### Ejemplo 5

```
VAR dnum dnumtype:=1.234567890123456789;
```

```
str := ValToStr(dnumtype);
```

Se asigna a la variable `str` el valor "1.23456789012346".

---

### Valor de retorno

Tipo de dato: `string`

El valor se convierte en una cadena con un formato estándar de `RAPID`. Esto significa en principio 6 dígitos significativos. Un valor literal interpretado como un valor `dnum` (consulte el ejemplo 3) y `dnum` variables (consulte el ejemplo 5) aunque tiene 15 dígitos significativos.

Si la cadena resultante es demasiado larga, se genera un error de tiempo de ejecución.

---

### Argumentos

```
ValToStr ( Val )
```

Val

*Value*

---

*Continúa en la página siguiente*

## 2 Funciones

---

### 2.217 ValToStr - Convierte un valor en una cadena

RobotWare - OS

Continuación

Tipo de dato: *anytype*

Un valor de cualquier tipo de dato. Puede usar todos los tipos de datos de valor con estructura atómica o de registro, componente de registro, matriz o elemento de matriz.

---

#### Sintaxis

```
ValToStr '('  
  [ Val ':= ' ] <expression (IN) of anytype> ')'
```

Una función con un valor de retorno del tipo de dato *string*.

---

#### Información relacionada

| Para obtener más información sobre   | Consulte  |
|--------------------------------------|---|
| Funciones para cadenas de caracteres | <i>Manual de referencia técnica - RAPID Overview</i>      |
| Definición de cadena de caracteres   | <a href="#"><i>string - Cadenas en la página 1778</i></a> |
| Valores de cadena de caracteres      | <i>Manual de referencia técnica - RAPID Overview</i>      |

## 2.218 VectMagn - Magnitud de un vector pos

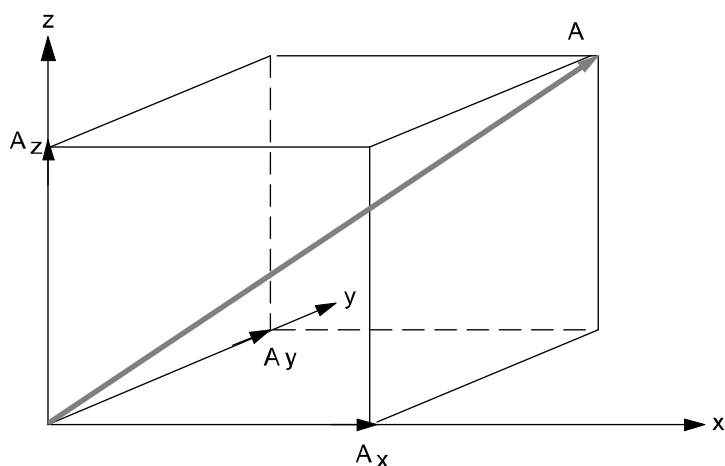
## Utilización

VectMagn (*Vector Magnitude*) se utiliza para calcular la magnitud de un vector pos .

## Ejemplos básicos

El ejemplo siguiente ilustra la función VectMagn.

## Ejemplo 1



xx0500002446

Un vector **A** puede escribirse como la suma de sus componentes en las tres direcciones ortogonales:

$$\mathbf{A} = A_x \mathbf{x} + A_y \mathbf{y} + A_z \mathbf{z}$$

La magnitud de **A** es:

$$|\mathbf{A}| = \sqrt{A_x^2 + A_y^2 + A_z^2}$$

El vector se describe mediante el tipo de dato pos y la magnitud mediante el tipo de dato num:

```
VAR num magnitud;
VAR pos vector;
...
vector := [1,1,1];
magnitud := VectMagn(vector);
```

## Valor de retorno

Tipo de dato: num

La magnitud del vector (tipo de dato pos).

## Argumentos

VectMagn (Vector)

*Continúa en la página siguiente*

## 2 Funciones

---

### 2.218 VectMagn - Magnitud de un vector pos

RobotWare - OS

Continuación

Vector

Tipo de dato: pos

El vector descrito mediante el tipo de dato pos.

---

#### Sintaxis

```
VectMagn '('  
    [Vector ':='] <expression (IN) of pos> ')'
```

Una función con un valor de retorno del tipo de dato num.

---

#### Información relacionada

| Para obtener más información sobre    | Consulte   |
|---------------------------------------|--|
| Instrucciones y funciones matemáticas | <i>Manual de referencia técnica - RAPID Overview</i> |



## 2.219 XOR - Evalúa un valor lógico

### Utilización

XOR (*Exclusive Or*) es una expresión condicional utilizada para evaluar un valor lógico (verdadero/falso).

### Ejemplos básicos

Los siguientes ejemplos ilustran la función XOR.

#### Ejemplo 1

```
VAR bool a;  
VAR bool b;  
VAR bool c;  
c := a XOR b;
```

El valor de retorno `c` es `TRUE` si sólo una de `a` o `b` es `TRUE`. De lo contrario, el valor de retorno es `FALSE`.

#### Ejemplo 2

```
VAR num a;  
VAR num b;  
VAR bool c;  
...  
c := a>5 XOR b=3;
```

El valor de retorno de `c` es `TRUE` si sólo una de las condiciones es `TRUE`. `a` es mayor que 5, o bien `b` es igual a 3. De lo contrario, el valor de retorno es `FALSE`.

### Valor de retorno

Tipo de dato: `bool`

El valor de retorno es `TRUE` si sólo una de las expresiones condicionales es correcta. De lo contrario, el valor de retorno es `FALSE`.

### Sintaxis

```
<expression of bool> XOR <expression of bool>
```

Una función con un valor de retorno del tipo de dato `bool`.

### Información relacionada

| Para obtener más información sobre | Consulte   |
|------------------------------------|--|
| AND                                | <a href="#">AND - Evalúa un valor lógico en la página 1159</a>   |
| OR                                 | <a href="#">OR - Evalúa un valor lógico en la página 1394</a>    |
| NOT                                | <a href="#">NOT - Invierte un valor lógico en la página 1385</a> |
| Expresiones                        | <a href="#">Manual de referencia técnica - RAPID Overview</a>    |

**Esta página se ha dejado vacía intencionadamente**

## 3 Tipos de datos

### 3.1 aiotrigg - Condición de disparo con E/S analógica

#### Utilización

`aiotrigg` (*Analog I/O Trigger*) se utiliza para definir la condición que genera una interrupción para una señal de entrada o salida analógica.

#### Descripción

Los datos de tipo `aiotrigg` definen la forma en que se usarán un umbral máximo y un umbral mínimo para determinar si el valor lógico de una señal analógica cumple las condiciones para generar una interrupción.

#### Ejemplos básicos

El siguiente ejemplo ilustra el tipo de dato `aiotrigg`:

##### Ejemplo 1

```
VAR intnum siglint;
PROC main()
  CONNECT siglint WITH iroutine1;
  ISignalAI \Single, ail, AIO_BETWEEN, 1.5, 0.5, 0, siglint;
```

Solicita una interrupción que debe producirse la primera vez que el valor lógico de la señal analógica de entrada `ail` se encuentre entre 0.5 y 1.5. En este caso, se realiza una llamada a la rutina TRAP `iroutine1`.

#### Datos predefinidos

Se han predefinido las constantes simbólicas siguientes del tipo de dato `aiotrigg`. Puede usarlas para especificar unas condiciones para las instrucciones `ISignalAI` y `ISignalAO`.

| Valor | Constante simbólica | Comentario   |
|-------|---------------------|--|
| 1     | AIO_ABOVE_HIGH      | La señal genera interrupciones si se encuentra por encima del valor máximo especificado  |
| 2     | AIO_BELOW_HIGH      | La señal genera interrupciones si se encuentra por debajo del valor máximo especificado  |
| 3     | AIO_ABOVE_LOW       | La señal genera interrupciones si se encuentra por encima del valor mínimo especificado  |
| 4     | AIO_BELOW_LOW       | La señal genera interrupciones si se encuentra por debajo del valor mínimo especificado  |
| 5     | AIO_BETWEEN         | La señal genera interrupciones si se encuentra entre los valores mínimo y máximo especificados                                     |
| 6     | AIO_OUTSIDE         | La señal genera interrupciones si se encuentra por debajo del valor mínimo especificado o por encima del valor máximo especificado |
| 7     | AIO_ALWAYS          | La señal siempre genera interrupciones   |

#### Características

`aiotrigg` es un tipo de dato de alias de `num` y por tanto hereda sus características.

*Continúa en la página siguiente*

### 3 Tipos de datos

---

#### 3.1 aiotrigg - Condición de disparo con E/S analógica

RobotWare - OS

Continuación

---

#### Información relacionada

| Para obtener más información sobre                      | Consulte   |
|---|--|
| Interrupción a partir de una señal analógica de entrada | <a href="#">ISignalAI - Interrupciones a partir de una señal analógica de entrada en la página 309</a> |
| Interrupción a partir de una señal analógica de salida  | <a href="#">ISignalAO - Interrupciones a partir de una señal analógica de salida en la página 319</a>  |
| Tipos de datos en general, tipos de datos de alias      | <i>Manual de referencia técnica - RAPID Overview, sección Características básicas - Tipos de datos</i> |

## 3.2 ALIAS - Asignación de un tipo de dato de alias

### Utilización

ALIAS se utiliza para definir un tipo de dato como equivalente a otro tipo de dato. Los tipos de alias proporcionan una forma de clasificar los objetos. El sistema puede utilizar la clasificación de alias para consultar y presentar los objetos relacionados con cada tipo. Un tipo de alias se introduce mediante una definición de alias.

Los tipos de alias incorporados son `errnum` y `intnum`, ambos alias de `num`.

### Tipo `errnum`

El tipo `errnum` es un alias para `num` y se utiliza para la representación de números de error.

### Tipo `intnum`

El tipo `intnum` es un alias para `num` y se utiliza para la representación de números de interrupción.

### Ejemplos básicos

El siguiente ejemplo ilustra la definición de ALIAS.

#### Ejemplo 1

```
ALIAS num level;
CONST level low := 2.5;
CONST level high := 4.0;
```

Se define un tipo de alias `level` (alias para `num`).

### Limitaciones

Para que RAPID lo reconozca, todas las definiciones de alias deben declararse en la parte más alta del programa o módulo de sistema, antes del resto de declaraciones de datos. El único tipo de datos que se permite declarar antes del alias es `RECORD`.

No es posible definir un tipo de alias sobre otro tipo de alias.

### Sintaxis

```
ALIAS <type name> <identifier> ';
```

Definición de alias.

### Información relacionada

| Para obtener más información sobre              | Consulte   |
|---|--|
| <code>errnum</code> - Número de error           | <a href="#">errnum - Número de error en la página 1664</a>           |
| <code>intnum</code> - Identidad de interrupción | <a href="#">intnum - Identidad de interrupción en la página 1687</a> |
| Elementos léxicos                               | <i>Technical reference manual - RAPID kernel</i>                     |

## 3 Tipos de datos

---

### 3.3 bool - Valores lógicos

RobotWare - OS

### 3.3 bool - Valores lógicos

---

#### Utilización

`bool` se utiliza con valores lógicos (verdadero/falso).

---

#### Descripción

El valor de un dato de tipo `bool` puede ser `TRUE` (verdadero) o `FALSE` (falso).

---

#### Ejemplos básicos

Los siguientes ejemplos ilustran el tipo de dato `bool` :

##### Ejemplo 1

```
flag1 := TRUE;
```

Se asigna al flag el valor `TRUE` (verdadero).

##### Ejemplo 2

```
VAR bool highvalue;  
VAR num reg1;  
...  
highvalue := reg1 > 100;
```

`highvalue` recibe el valor `TRUE` si `reg1` es mayor que 100; de lo contrario, se asigna `FALSE`.

##### Ejemplo 3

```
IF highvalue Set do1;
```

La señal `do1` se activa si `highvalue` es `TRUE`.

##### Ejemplo 4

```
highvalue := reg1 > 100;  
mediumvalue := reg1 > 20 AND NOT highvalue;  
mediumvalue recibe el valor TRUE si reg1 está entre 20 y 100.
```

---

#### Información relacionada

| Para obtener más información sobre | Consulte  |
|------------------------------------|---|
| Expresiones lógicas                | <i>Manual de referencia técnica - RAPID Overview, sección Características básicas - Expresiones</i> |
| Operaciones con valores lógicos    | <i>Manual de referencia técnica - RAPID Overview, sección Características básicas - Expresiones</i> |

## 3.4 btnres - Datos de resultado de pulsador

### Utilización

`btnres` (*button result*) se usa para representar la selección de usuario del pulsador en un dispositivo de usuario, por ejemplo el FlexPendant.

### Descripción

Las constantes de `btnres` se han diseñado para su uso al comprobar el valor de resultado de la instrucción `UIMsgBox` y el valor de retorno desde las funciones `UIMessageBox` y `UIListView`.

### Ejemplos básicos

El siguiente ejemplo ilustra el tipo de dato `btnres`:

#### Ejemplo 1

```
VAR btnres answer;

UIMsgBox "More ?" \Buttons:=btnYesNo \Result:= answer;
IF answer= resYes THEN
...
ELSEIF answer =ResNo THEN
...
ENDIF
```

La enumeración de botón estándar `btnYesNo` generará un pulsador Sí y un pulsador No en la interfaz de usuario. La selección del usuario se almacenará en la variable `answer`.

### Datos predefinidos

Se han predefinido en el sistema las constantes siguientes del tipo de dato `btnres`.

| Valor | Constantes             | Respuesta de botón    |
|-------|------------------------|-----------------------|
| 0     | <code>resUnkwn</code>  | Resultado desconocido |
| 1     | <code>resOK</code>     | Correcto              |
| 2     | <code>resAbort</code>  | Anular                |
| 3     | <code>resRetry</code>  | Reintentar            |
| 4     | <code>resIgnore</code> | Omitir                |
| 5     | <code>resCancel</code> | Cancelar              |
| 6     | <code>resYes</code>    | Sí                    |
| 7     | <code>resNo</code>     | No                    |

Es posible trabajar con botones definidos por el usuario que responden a las funciones `UIMessageBox` y `UIListView`.

### Características

`btnres` es un tipo de dato de alias de `num` y por tanto hereda sus características.

*Continúa en la página siguiente*

### 3 Tipos de datos

---

#### 3.4 btnres - Datos de resultado de pulsador

RobotWare - OS

Continuación

---

#### Información relacionada

| Para obtener más información sobre              | Consulte   |
|---|--|
| Cuadro de mensaje de interacción con el usuario | <a href="#">UIMsgBox - Cuadro de mensaje de usuario de tipo básico en la página 1002</a>       |
| Cuadro de mensaje de interacción con el usuario | <a href="#">UIMessageBox - Cuadro de mensaje de usuario de tipo avanzado en la página 1577</a> |
| Vista de lista de interacción con el usuario    | <a href="#">UICollection - Vista de lista de usuario en la página 1568</a>                     |
| Datos de botón de tipo de dato de alias         | <a href="#">buttondata - Datos de botón en la página 1614</a>                                  |



### 3.5 busstate - Estado de la red de E/S

#### Utilización

`busstate` se utiliza para representar el estado momentáneo de una red de E/S.

#### Descripción

Las constantes de `busstate` se han diseñado para usarlas al comprobar el valor de retorno de la instrucción `IOBusState`.

#### Ejemplos básicos

El siguiente ejemplo ilustra el tipo de dato `busstate`:

##### Ejemplo 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
```

#### Datos predefinidos

Las constantes simbólicas predefinidas del tipo de dato `busstate` pueden verse en la instrucción `IOBusState`.

#### Características

`busstate` es un tipo de dato de alias de `num` y por tanto hereda sus características.

#### Información relacionada

| Para obtener más información sobre            | Consulte   |
|---|--|
| Obtención del estado actual de una red de E/S | <a href="#">IOBusState - Obtener el estado actual de una red de E/S en la página 290</a>                         |
| Instrucciones de entrada/salida               | <i>Manual de referencia técnica - RAPID Overview, sección Resumen sobre RAPID - Señales de entrada y salida</i>  |
| Funcionalidad de entrada/salida en general    | <i>Manual de referencia técnica - RAPID Overview, sección Principios de movimiento y E/S - Principios de E/S</i> |
| Configuración de E/S                          | <i>Manual de referencia técnica - Parámetros del sistema</i>   |

## 3 Tipos de datos

---

### 3.6 buttndata - Datos de botón

RobotWare - OS

### 3.6 buttndata - Datos de botón

---

#### Utilización

`buttndata` se usa para representar una combinación de pulsadores estándar para su visualización en un dispositivo de usuario, por ejemplo en el FlexPendant.

---

#### Descripción

Las constantes `buttndata` se utilizan para representar los pulsadores de respuesta en la instrucción `UIMsgBox` y las funciones `UIMessageBox` y `UIListView`.

---

#### Ejemplos básicos

El siguiente ejemplo ilustra el tipo de dato `buttndata`:

##### Ejemplo 1

```
VAR btnres answer;  
UIMsgBox "More ?" \Buttons:=btnYesNo \Result:= answer;  
IF answer= resYes THEN  
...  
ELSE  
...  
ENDIF
```

La enumeración de botón estándar `btnYesNo` generará un pulsador Sí y un pulsador No.

---

#### Datos predefinidos

Se han predefinido en el sistema las constantes siguientes del tipo de dato `buttndata`.

| Valor | Constantes                  | Botón mostrado              |
|-------|-----------------------------|-----------------------------|
| - 1   | <code>btnNone</code>        | Ningún botón                |
| 0     | <code>btnOK</code>          | Correcto                    |
| 1     | <code>btnAbtrRtryIgn</code> | Anular, Reintentar y Omitir |
| 2     | <code>btnOKCancel</code>    | Aceptar y Cancelar          |
| 3     | <code>btnRetryCancel</code> | Reintentar y Cancelar       |
| 4     | <code>btnYesNo</code>       | Sí y No                     |
| 5     | <code>btnYesNoCancel</code> | Sí, No y Cancelar           |

Es posible mostrar botones definidos por el usuario con las funciones `UIMessageBox` y `UIListView`.

---

#### Características

`buttndata` es un tipo de dato de alias de `num` y por tanto hereda sus características.

---

Continúa en la página siguiente

#### Información relacionada

| Para obtener más información sobre                 | Consulte  |
|--|---|
| Cuadro de mensaje de interacción con el usuario    | <a href="#">UIAlert - Cuadro de mensaje de usuario de tipo básico en la página 1002</a>                         |
| Cuadro de mensaje de interacción con el usuario    | <a href="#">UIAlert - Cuadro de mensaje de usuario de tipo avanzado en la página 1577</a>                       |
| Vista de lista de interacción con el usuario       | <a href="#">UITableView - Vista de lista de usuario en la página 1568</a>                                       |
| Resultado de botón de tipo de dato de alias        | <a href="#">UIButton - Datos de resultado de pulsador en la página 1611</a>                                     |
| Tipos de datos en general, tipos de datos de alias | <a href="#">Manual de referencia técnica - RAPID Overview, sección Características básicas - Tipos de datos</a> |

## 3 Tipos de datos

---

### 3.7 byte - Valores enteros 0-255

RobotWare - OS

### 3.7 byte - Valores enteros 0-255

---

#### Utilización

`byte` se utiliza con valores decimales (de 0 a 255), acorde con el rango que permite un `byte`.

Este tipo de dato se utiliza junto con instrucciones y funciones dedicadas a la manipulación de bits y la conversión.

#### Descripción

Los datos de tipo `byte` representan valores de `byte` enteros.

#### Ejemplos básicos

Los siguientes ejemplos ilustran el tipo de dato `byte`:

##### Ejemplo 1

```
VAR byte data1 := 130;
```

Definición de una variable `data1` con el valor decimal 130.

##### Ejemplo 2

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

El bit número 8 (`parity_bit`) de la variable `data1` cambia a 0, con lo que el contenido de la variable `data1` cambia de 130 a 2 (en su valor decimal).

#### Gestión de errores

Si un argumento de tipo `byte` tiene un valor que queda fuera del rango de 0 a 255, se genera un error al ejecutar el programa.

#### Características

`byte` es un tipo de dato de alias de `num` y por tanto hereda sus características.

#### Información relacionada

| Para obtener más información sobre | Consulte  |
|------------------------------------|---|
| Tipos de datos de alias            | <i>Manual de referencia técnica - RAPID Overview</i>            |
| Funciones de bits                  | <i>Manual de referencia técnica - RAPID Overview</i>            |
| <i>Advanced RAPID</i>              | <i>Especificaciones del producto - Controller software IRC5</i> |

## 3.8 cameradev - dispositivo de cámara

### Utilización

`cameradev` (*dispositivo de cámara*) se utiliza para definir los dispositivos de cámara que pueden controlarse y utilizarse desde el programa de RAPID. El tipo de dato `cameradev` se utiliza para las instrucciones y funciones que se comunican con una cámara. Los nombres de las cámaras se definen en los parámetros del sistema y, por tanto, no debe definirlos en el programa.

### Descripción

Los datos del tipo `cameradev` sólo contienen una referencia al dispositivo de cámara.

### Limitaciones

No debe definir los datos de tipo `cameradev` en el programa. Sin embargo, si lo hace, aparecerá un mensaje de error tan pronto como se ejecute la instrucción o función que hace referencia a este dato `cameradev`. Sin embargo, sí es posible utilizarlos como parámetros al declarar una rutina.

### Datos predefinidos

Todas las cámaras definidas en los parámetros del sistema están predefinidas en cada tarea de programa.

### Ejemplos básicos

El siguiente ejemplo ilustra el tipo de dato `cameradev`.

#### Ejemplo 1

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

### Características

`cameradev` es un tipo de dato sin valor. Esto significa que los datos de este tipo no son compatibles con operaciones basadas en valores.

### Información relacionada

| Para obtener más información sobre | Consulte  |
|------------------------------------|---|
| Integrated Vision                  | <i>Manual de aplicaciones - Integrated Vision</i> |

## 3 Tipos de datos

### 3.9 camerastatus: estado de comunicación de la cámara

*Integrated Vision*

### 3.9 camerastatus: estado de comunicación de la cámara

#### Utilización

`camerastatus` se utiliza para la representación del estado de la comunicación con la cámara.

#### Descripción

#### Ejemplos básicos

El siguiente ejemplo ilustra el tipo de dato `camerastatus`:

#### Ejemplo 1

```
VAR camerastatus curr_camerastatus;  
...  
curr_camerastatus:=CamGetMode(mycamera);  
IF curr_camerastatus = CAMERA_DISCON THEN  
    TPWrite "Current mode of camera " + CamGetName(mycamera) + " is  
        CAMERA_DISCON";  
ELSEIF curr_camerastatus = CAMERA_STANDBY THEN  
    TPWrite "Current mode of camera "+CamGetName(mycamera) + " is  
        CAMERA_STANDBY";  
ELSEIF curr_camerastatus = CAMERA_RUNNING THEN  
    TPWrite "Current mode of camera " + CamGetName(mycamera) + " is  
        CAMERA_RUNNING";  
ENDIF
```

Obtenga el modo actual de la cámara y escriba el estado en el FlexPendant.

#### Datos predefinidos

Las siguientes constantes de tipo `camerastatus` están predefinidas:

| Constante de RAPID | Valor |                                      |
|--------------------|-------|--------------------------------------|
| CAMERA_DISCON      | 1     | La cámara está desconectada.         |
| CAMERA_STANDBY     | 2     | La cámara está en el modo de espera. |
| CAMERA_RUNNING     | 3     | La cámara está en funcionamiento.    |

#### Características

`camerastatus` es un tipo de dato de alias de `num` y por tanto hereda sus características.

#### Información relacionada

| Para obtener más información sobre                 | Consulte   |
|--|--|
| Integrated Vision                                  | <i>Manual de aplicaciones - Integrated Vision</i>  |
| Obtener el modo de la cámara                       | <a href="#">CamGetMode: obtener el modo actual de la cámara en la página 1224</a>                      |
| Tipos de datos en general, tipos de datos de alias | <i>Manual de referencia técnica - RAPID Overview, sección Características básicas - Tipos de datos</i> |

#### 3.10 cameratarget - datos de cámara

---

##### Utilización

`cameratarget` se utiliza para intercambiar datos de visión entre la imagen de cámara y el programa de RAPID.

---

##### Descripción

Los datos del tipo `cameratarget` son colecciones de datos definidas por el usuario y pueden configurarse para intercambiar datos de visión entre la imagen de cámara y el programa de RAPID.

Los datos cuentan con toda una variedad de componentes que pueden configurarse en función de las necesidades específicas de la aplicación de visión actual. El componente `cframe` está destinado a la transmisión de información acerca de la ubicación de un objeto, mientras que los valores numéricos y las cadenas están destinados a almacenar datos de inspección.

---

##### Componentes

El tipo de dato contiene los componentes siguientes:

`name`

Tipo de dato: `string`

El identificador de nombre del `cameratarget`.

`cframe`

*current frame*

Tipo de dato: `pose`

Para almacenar datos de posición utilizados normalmente para guiar al robot, a través de la modificación del objeto de trabajo.

`val1`

*value 1*

Tipo de dato: `num`

Para almacenar salidas numéricas, por ejemplo mediciones.

...

`val5`

*value 5*

Tipo de dato: `num`

Para almacenar salidas numéricas, por ejemplo mediciones.

`string1`

Tipo de dato: `string`

Para almacenar salidas numéricas de visión, por ejemplo una salida de inspección o identificación.

`string2`

Tipo de dato: `string`

*Continúa en la página siguiente*

## 3 Tipos de datos

---

### 3.10 cameratarget - datos de cámara

#### *Integrated Vision*

#### *Continuación*

Para almacenar salidas numéricas de visión, por ejemplo una salida de inspección o identificación.

type

Tipo de dato: num

Un identificador numérico del objetivo de la cámara. Tiene una finalidad similar a la del componente name.

cameraname

Tipo de dato: string

El nombre de la cámara.

sceneid

*scene identification*

Tipo de dato: num

El identificador único de la imagen utilizada para generar el cameratarget.

---

### Ejemplos básicos

El siguiente ejemplo ilustra el tipo de dato cameratarget.

#### Ejemplo 1

```
VAR cameratarget target1;
...
wobjmycamera.oframe := target1.cframe;
MoveL pickpart, v100, fine, mygripper \WObj:= wobjmycamera;
```

La transformación de coordenadas cframe se asigna a la base de coordenadas de objeto del objeto de trabajo. El robtarget pickpart ha sido ajustado anteriormente a una posición de recogida correcta dentro de la base de coordenadas de objeto del objeto de trabajo.

---

### Estructura

```
< 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 >
```

*Continúa en la página siguiente*



**Información relacionada**

| Para obtener más información sobre | Consulte  |
|------------------------------------|---|
| Integrated Vision                  | <i>Manual de aplicaciones - Integrated Vision</i> |

## 3 Tipos de datos

---

### 3.11 capaptrreferencedata - Datos de configuración variable para seguimiento del punto *Continuous Application Platform (CAP)*

### 3.11 capaptrreferencedata - Datos de configuración variable para seguimiento del punto

---

#### Utilización

capaptrreferencedata se usa para configurar la información necesaria para la configuración del proceso de corrección de seguimiento del punto mediante las instrucciones CapAPTrSetupAO, CapAPTrSetupAI y CapAPTrSetupPERS.

---

#### Componentes

reference\_y

Tipo de dato: num

Define la referencia para la posición Y.

reference\_z

Tipo de dato: num

Define la referencia para la posición Z.

threshold\_y

Tipo de dato: num

La diferencia entre la señal de entrada y el valor reference\_y debe ser mayor que el valor threshold\_y para que el regulador reaccione al cambio.

threshold\_z

Tipo de dato: num

La diferencia entre la señal de entrada y el valor reference\_z debe ser mayor que el valor threshold\_z para que el regulador reaccione al cambio.

gain\_y

Tipo de dato: num

La diferencia entre el valor reference\_y y el valor de la señal de entrada se escala con el valor gain\_y.

gain\_z

Tipo de dato: num

La diferencia entre el valor reference\_z y el valor de la señal de entrada se escala con el valor gain\_z.

---

#### Estructura

```
< data object of capaptrreferencedata >
  < reference_y of num >
  < reference_z of num >
  < threshold_y of num >
  < threshold_z of num >
  < gain_y of num >
  < gain_z of num >
```

*Continúa en la página siguiente*

3.11 capptrreferencedata - Datos de configuración variable para seguimiento del punto  
*Continuous Application Platform (CAP)*  
*Continuación*

**Información relacionada**

| Para obtener más información sobre     | Consulte  |
|--|---|
| Instrucción CapAPTrSetupAI             | <a href="#">CapAPTrSetupAI - Configurar un seguimiento del punto controlado por señales de entrada analógicas en la página 86</a> |
| Instrucción CapAPTrSetupAO             | <a href="#">CapAPTrSetupAO - Configurar un seguimiento del punto controlado por señales de salida analógicas en la página 89</a>  |
| Instrucción CapAPTrSetupPERS           | <a href="#">CapAPTrSetupPERS - Configurar un seguimiento del punto controlado por variables persistentes en la página 92</a>      |
| <i>Continuous Application Platform</i> | <i>Application manual - Continuous Application Platform</i>   |
| <i>Sensor Interface</i>                | <i>Application manual - Controller software IRC5</i>  |

## 3 Tipos de datos

### 3.12 capdata - Datos CAP

Continuous Application Platform (CAP)

### 3.12 capdata - Datos CAP

#### Utilización

capdata contiene todos los datos necesarios para definir el comportamiento del proceso CAP.

#### Componentes

##### start\_fly

*Inicio sobre la marcha*

Tipo de dato: bool

Define si se utiliza o no el inicio sobre la marcha:

| Valor | Consecuencia                            |
|-------|---|
| TRUE  | Se utiliza el inicio sobre la marcha    |
| FALSE | NO se utiliza el inicio sobre la marcha |

El inicio sobre la marcha significa que el movimiento del robot se inicia antes de que comience el proceso. El proceso se inicia en la ejecución (consulte [flypointdata - Datos para inicio/fin sobre la marcha en la página 1679](#)).

##### end\_fly

*Fin sobre la marcha*

Tipo de dato: bool

Define si se utiliza o no el fin sobre la marcha:

| Valor | Consecuencia                         |
|-------|--------------------------------------|
| TRUE  | Se utiliza el fin sobre la marcha    |
| FALSE | NO se utiliza el fin sobre la marcha |

Fin sobre la marcha quiere decir que el proceso CAP puede terminarse antes de que el robot alcance el punto final, permitiendo por tanto al robot abandonar la trayectoria del proceso en la ejecución, es decir, utilizando un punto de zona (consulte [flypointdata - Datos para inicio/fin sobre la marcha en la página 1679](#)).

##### first\_instr

*Primera instrucción*

Tipo de dato: bool

Define si la instrucción CapL/CapC es o no la primera instrucción de una secuencia de instrucciones CapL/CapC:

| Valor | Consecuencia   |
|-------|--|
| TRUE  | es la primera instrucción de una secuencia de instrucciones CapL/CapC    |
| FALSE | no es la primera instrucción de una secuencia de instrucciones CapL/CapC |

##### last\_instr

*Última instrucción*

Continúa en la página siguiente

Tipo de dato: bool

Define si la instrucción `CapL/CapC` es o no la última instrucción de una secuencia de instrucciones `CapL/CapC`:

| Valor | Consecuencia   |
|-------|--|
| TRUE  | es la última instrucción de una secuencia de instrucciones <code>CapL/CapC</code>    |
| FALSE | no es la última instrucción de una secuencia de instrucciones <code>CapL/CapC</code> |

restart\_dist

*Distancia de reinicio, unidad: mm*

Tipo de dato: num

Define la distancia que el robot tiene que retroceder en la trayectoria, cuando se reinicia después de experimentar una parada cuando un proceso CAP estaba activo.

En los sistemas MultiMove, todos los robots sincronizados deben utilizar la misma distancia de reinicio.

speed\_data

*Datos de velocidad para CAP*

Tipo de dato: capspeeddata

Define todos los datos CAP relativos a la velocidad (consulte [capspeeddata - Datos de velocidad para CAP en la página 1632](#)).

start\_fly\_point

Tipo de dato: flypointdata

Estos datos solo se tienen en cuenta cuando `start_fly` es TRUE.

Define la información de inicio sobre la marcha para el proceso CAP (consulte [flypointdata - Datos para inicio/fin sobre la marcha en la página 1679](#).)

end\_fly\_point

Tipo de dato: flypointdata

Estos datos solo se tienen en cuenta cuando `end_fly` es TRUE.

Define la información de fin sobre la marcha para el proceso CAP (consulte [flypointdata - Datos para inicio/fin sobre la marcha en la página 1679](#).)

sup\_timeouts

Tipo de dato: supervtimeouts

Define los tiempos límite utilizados para todas las fases de la supervisión del intercambio (consulte [supervtimeouts - Tiempos límite de supervisión de intercambio en la página 1781](#) y la sección *Supervisión en Application manual - Continuous Application Platform*).

proc\_times

Tipo de dato: processtimes

Define los tiempos límite utilizados para todas las fases PRE, POST1 y POST2 de la supervisión de estado (consulte [processtimes - Tiempos de proceso en la](#)

*Continúa en la página siguiente*

## 3 Tipos de datos

---

### 3.12 capdata - Datos CAP

#### Continuous Application Platform (CAP)

##### Continuación

[página 1732](#) y la sección *Supervisión y fases del proceso en Application manual - Continuous Application Platform*).

#### block\_at\_restart

Tipo de dato: restartblkdata

Define el comportamiento del proceso CAP durante un reinicio (consulte [restartblkdata - bloque de datos para el reinicio en la página 1738](#)).

---

#### Estructura

```
< 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 >
    < 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 >
```

*Continúa en la página siguiente*

#### Información relacionada

|  | Descrito en:  |
|--|---|
| <i>Continuous Application Platform</i>     | <i>Application manual - Continuous Application Platform</i>                                     |
| Tipo de dato <code>capspeeddata</code>     | <a href="#">capspeeddata - Datos de velocidad para CAP en la página 1632</a>                    |
| Tipo de dato <code>flypointdata</code>     | <a href="#">flypointdata - Datos para inicio/fin sobre la marcha en la página 1679</a>          |
| Tipo de dato <code>supervtimeouts</code>   | <a href="#">supervtimeouts - Tiempos límite de supervisión de intercambio en la página 1781</a> |
| Tipo de dato <code>processtimes</code>     | <a href="#">processtimes - Tiempos de proceso en la página 1732</a>                             |
| Tipo de dato <code>block_at_restart</code> | <a href="#">restartblkdata - bloque de datos para el reinicio en la página 1738</a>             |
| Instrucción <code>CapL</code>              | <a href="#">CapL - Instrucción de movimiento CAP lineal en la página 109</a>                    |
| Instrucción <code>CapC</code>              | <a href="#">CapC - Instrucción de movimiento CAP circular en la página 95</a>                   |

### 3 Tipos de datos

---

#### 3.13 capltrackdata - Datos de seguimiento de sensor de seguimiento anticipatorio de CAP *Continuous Application Platform (CAP)*

#### 3.13 capltrackdata - Datos de seguimiento de sensor de seguimiento anticipatorio de CAP

---

##### Utilización

capltrackdata contiene datos con los que el usuario puede influir en cómo las instrucciones CapL/CapC incorporan los datos de corrección de trayectoria generados por el sensor de seguimiento anticipatorio (por ejemplo, seguimiento láser). capltrackdata es parte de captrackdata.

---

##### Componentes

###### joint\_no

Tipo de dato: num

Define el tipo de eje, expresado en forma numérica, que el equipo de sensor usará durante el seguimiento.

###### filter

Tipo de dato: num

Define la constante de tiempo para un filtro contra bajas frecuencias aplicado a las correcciones de trayectoria. El componente puede tener valores entre 1 y 10, donde 1 da la respuesta más rápida (sin filtrado) a los errores de trayectoria detectados por el sensor.

###### calibframe\_no

Tipo de dato: num

Define qué base de coordenadas de calibración se utiliza de las tres bases de coordenadas definidas en CapLTrSetup.

| Valor | Base de coordenadas de calibración | Descripción                |
|-------|------------------------------------|----------------------------|
| 1     | calibframe                         | Obligatorio en CapLTrSetup |
| 2     | calibframe2                        | Opcional en CapLTrSetup    |
| 3     | calibframe3                        | Opcional en CapLTrSetup)   |

###### seamoffs\_y, seamoffs\_z

Tipo de dato: num

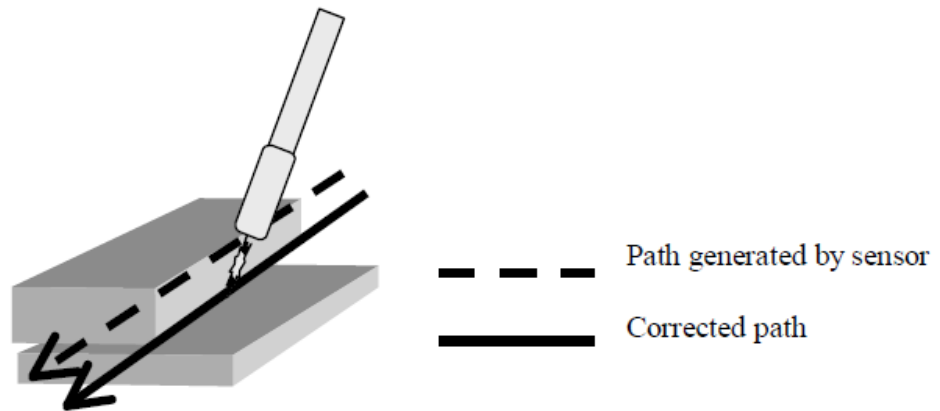
Los componentes de offset de cordón se utilizan para añadir offsets constantes a la trayectoria generada por el sensor (en mm). Si, por ejemplo, el sensor considera que el borde superior de una soldadura a solape está en la posición de cordón

*Continúa en la página siguiente*



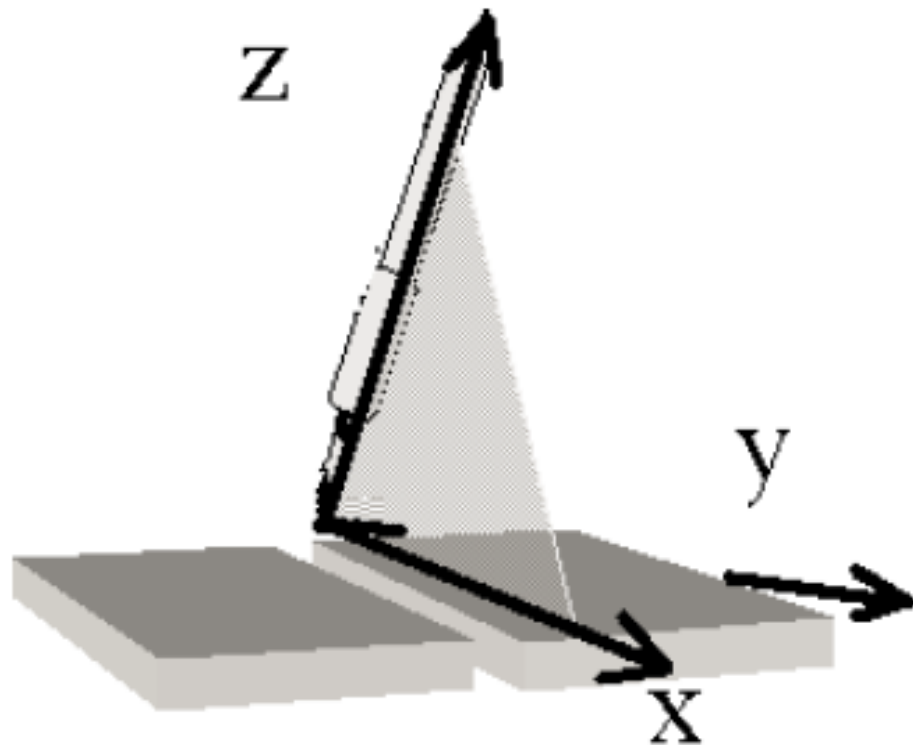
3.13 caplatrackdata - Datos de seguimiento de sensor de seguimiento anticipatorio de CAP  
*Continuous Application Platform (CAP)*  
 Continuación

correcta, como se indica en la figura siguiente, pueden utilizarse los offsets del cordón para corregir la trayectoria.



xx120000199

Las correcciones se definen en el sistema de coordenadas de la trayectoria, que es a derechas.



xx120000200

- La dirección X es paralela a la tangente de la trayectoria.
- La dirección Z es el vector Z de la herramienta.
- La dirección Y es perpendicular a un plano a través de las direcciones X y Z.

seamadapt\_y, seamadapt\_z

Tipo de dato: num

*Continúa en la página siguiente*

### 3 Tipos de datos

#### 3.13 caplatrackdata - Datos de seguimiento de sensor de seguimiento anticipatorio de CAP

##### Continuous Application Platform (CAP)

##### Continuación

Los componentes de adaptación de cordón son similares a los componentes de offset de cordón. Sin embargo, las magnitudes de los offsets no se indican como valores fijos. Los offsets se calculan como la separación medida del cordón, multiplicada por los valores de adaptación del cordón.

Los componentes se utilizan para adaptar el offset de la herramienta respecto al cordón, para optimizar el proceso para diferentes tamaños de separación.

track\_mode

Tipo de dato: num

Con el componente `track_mode` es posible influir de forma selectiva en el comportamiento de seguimiento de un seguimiento láser.

| Valor | Track Mode   |
|-------|--|
| 0     | Seguimiento normal. Se tienen en cuenta tanto las correcciones de Y como de Z  |
| 1     | Seguimiento como si las correcciones de Y enviadas por el seguimiento láser fueran cero. Se tienen en cuenta las correcciones de Z. <sup>i</sup>   |
| 2     | Seguimiento como si las correcciones de Z enviadas por el seguimiento láser fueran cero. Las correcciones de Y no se tienen en cuenta. <sup>i</sup>  |
| 3     | Seguimiento como si las correcciones de Y y Z enviadas por el seguimiento de láser fueran cero. <sup>i</sup>   |
| 4     | Corrección de Y totalmente desactivada, es decir la corrección del componente Y cambia a cero antes de que se envíe al robot. Se tiene en cuenta la corrección de Z. <sup>ii</sup>   |
| 5     | Corrección de Z totalmente desactivada, es decir la corrección del componente Z cambia a cero antes de que se envíe al robot. Se tiene en cuenta la corrección de Y. <sup>ii</sup>   |
| 6     | Correcciones de Y y Z totalmente desactivadas, es decir, la corrección del componente Y y Z cambia a cero antes de que se envíe al robot. <sup>ii</sup>  |
| 7     | La corrección de Y se atenúa; es decir, el TCP vuelve en rampa al componente Y programado de la trayectoria. La corrección de Z está activa.   |
| 8     | La corrección de Z se atenúa; es decir, el TCP vuelve en rampa al componente Z programado de la trayectoria. La corrección de Y está activa.   |
| 9     | Las correcciones de Y y Z se atenúan; es decir, el TCP vuelve en rampa a la trayectoria programada.  |
| 10    | La corrección de Y se intensifica; es decir, el TCP vuelve en rampa al componente Y programado de la trayectoria. La corrección de Z está activa.  |
| 11    | La corrección de Z se intensifica; es decir, el TCP vuelve en rampa al componente Z programado de la trayectoria. La corrección de Y está activa.  |
| 12    | Las correcciones de Y y de Z se intensifican; es decir, el TCP vuelve en rampa a la trayectoria programada.  |
| 13    | Seguimiento como si las correcciones de y enviadas por el seguimiento láser fueran cero. Se tienen en cuenta las correcciones de z. La diferencia con <code>track_mode 1</code> es que el modo se inicia en la posición del TCP del robot y no en la posición del TCP del sensor. <sup>i</sup> |
| 14    | Seguimiento como si las correcciones de z enviadas por el seguimiento láser fueran cero. Se tienen en cuenta las correcciones de y. La diferencia con <code>track_mode 2</code> es que el modo se inicia en la posición del TCP del robot y no en la posición del TCP del sensor. <sup>i</sup> |

Continúa en la página siguiente

## 3.13 capltrackdata - Datos de seguimiento de sensor de seguimiento anticipatorio de CAP

*Continuous Application Platform (CAP)**Continuación*

| Valor | Track Mode  |
|-------|---|
| 15    | Seguimiento como si las correcciones de y y z enviadas por el seguimiento láser fueran cero. La diferencia con <code>track_mode 3</code> es que el modo se inicia en la posición del TCP del robot y no en la posición del TCP del sensor. <sup>i</sup> |

- i Para el `track_mode 1, 2, o 3`, la corrección acumulada de la instrucción `CapL/CapC` anterior se conserva para Y o Z y se transmite a la siguiente instrucción `CapL/CapC`. Este es el caso durante toda la vida útil del proceso CAP. No afecta a un nuevo proceso CAP.
- ii Para el `track_mode 4, 5 o 6`, las lecturas del sensor se acumulan incluso si la corrección de Y o Z cambia a cero antes del envío al robot. Esto implica, que podría producirse un "cambio" al comienzo y al final de la instrucción `CapL/CapC`.

**Ejemplos básicos**

```
PERS capltrackdata captrack := ["laser1:",50,[1,10,1,0,0,0,0,0]]
CapL p1, v200, cd, wsd, cwd, z20, tWeldGun \Track:=captrack;
```

**Sintaxis**

```
< data object of capltrackdata >
  < 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 >
```

**Información relacionada**

|  | Descrito en:   |
|--|--|
| Tipo de dato <code>captrackdata</code> | <a href="#">captrackdata - Datos de seguimiento de CAP en la página 1635</a> |
| <i>Continuous Application Platform</i> | <i>Application manual - Continuous Application Platform</i>                  |

### 3 Tipos de datos

---

#### 3.14 capspeeddata - Datos de velocidad para CAP *Continuous Application Platform (CAP)*

#### 3.14 capspeeddata - Datos de velocidad para CAP

---

##### Utilización

capspeeddata se utiliza para definir todos los datos relativos a la velocidad para un proceso CAP - es parte de capdata y define todos los datos de velocidad y tiempos de proceso necesarios para un proceso CAP:

- la velocidad y la duración de esta velocidad que se utilizarán al inicio del proceso CAP,
- el retardo para el movimiento del robot en relación con el inicio del proceso CAP,
- velocidad del proceso CAP,

La velocidad está limitada por el rendimiento del robot. Este rendimiento es distinto según el tipo de robot y la trayectoria del movimiento.

---

##### Componentes

###### fly\_start

Tipo de dato: num

No se usa.

###### start

Tipo de dato: num

La velocidad (en mm/s) utilizada al inicio del proceso CAP.

###### startspeed\_time

Tipo de dato: num

El tiempo (en segundos) de funcionamiento a la velocidad de start.

###### startmove\_delay

Tipo de dato: num

El tiempo (en segundos) que se retarda el movimiento del robot en relación al inicio del proceso CAP.

###### main

Tipo de dato: num

La velocidad del proceso CAP principal (mm/s).

###### fly\_end

Tipo de dato: num

No se usa.

---

##### Estructura

```
< data object of capspeeddata >  
  < fly_start of num >  
  < start of num >  
  < startspeed_time of num >  
  < startmove_delay of num >  
  < main of num >
```

*Continúa en la página siguiente*

---

3.14 capspeeddata - Datos de velocidad para CAP  
*Continuous Application Platform (CAP)*  
Continuación

< fly\_end of num >

#### Información relacionada

|  | Descrito en:  |
|--|---|
| Tipo de dato capdata                   | <a href="#">capdata - Datos CAP en la página 1624</a>       |
| <i>Continuous Application Platform</i> | <i>Application manual - Continuous Application Platform</i> |

## 3 Tipos de datos

### 3.15 capstopmode - Define modos de paro para CAP *Continuous Application Platform (CAP)*

### 3.15 capstopmode - Define modos de paro para CAP

#### Utilización

`capstopmode` se usa para definir los modos de paro disponibles en CAP.

#### Ejemplos básicos

El siguiente ejemplo ilustra el tipo de dato `capstopmode`.

#### Ejemplo 1

```
CAPSetStopMode SMOOTH_STOP_ON_PATH;
```

#### Valores predefinidos

| Valor               | Descripción  |
|---------------------|--|
| SMOOTH_STOP_ON_PATH | El movimiento del robot se detiene suavemente sin abandonar la trayectoria programada.                     |
| QUICK_STOP_ON_PATH  | El movimiento del robot se detiene con la mayor rapidez posible sin abandonar la trayectoria programada.   |
| EMERGENCY_STOP      | El movimiento del robot se detiene con la mayor rapidez posible aunque abandone la trayectoria programada. |

#### Características

`capstopmode` es un tipo de dato de alias de `num` y por tanto hereda sus características.

#### Información relacionada

|   | Descrito en:   |
|---|--|
| Instrucción <code>CAPSetStopMode</code> | <a href="#">CAPSetStopMode - Establecer el modo de paro para errores de ejecución en la página 127</a> |
| <i>Continuous Application Platform</i>  | <i>Application manual - Continuous Application Platform</i>  |

## 3.16 captrackdata - Datos de seguimiento de CAP

### Utilización

captrackdata proporciona la instrucción CapL/CapC con toda la información necesaria para corregir la trayectoria con un seguimiento anticipatorio o de punto. La información se envía a la instrucción CapL/C con el argumento opcional \Track.

El componente device determina qué tipo de sensor de seguimiento se utiliza. captrackdata no puede modificarse dentro de una secuencia de instrucciones CapL/CapC. El componente device se configura mediante la primera instrucción CapL/C; si es diferente en las instrucciones CapL/C restantes de la misma secuencia de instrucciones CapL/CapC, no tiene ningún efecto.

Para poder cambiar captrackdata para que se utilice en una instrucción CapL/CapC, en primer lugar, tiene que terminarse la secuencia mediante el cambio del componente last\_inst a TRUE en capdata.

Si el \Track no está presente en la primera instrucción CapL/C y todo sigue en la misma secuencia de instrucciones CapL/CapC, no se aplica ninguna corrección.

### Componentes

#### device

*Dispositivo de sensor*

Tipo de dato: string

Define a qué dispositivo está conectado el sensor que se utiliza en las instrucciones CapL/CapC para generar las correcciones de trayectoria.

#### max\_corr

*Máxima corrección de trayectoria permitida*

Tipo de dato: num

Define la máxima corrección de trayectoria permitida.

Para seguimientos anticipatorios:

- Si el offset de TCP debido a las correcciones de trayectoria es mayor a max\_corr y se especificó \WarnMaxCorr en CapLATrSetup, el robot continúa su trayectoria, pero la corrección de trayectoria aplicada no supera max\_corr.
- Si no se especificó \WarnMaxCorr, se informa de un error de seguimiento y se detiene la ejecución del programa.

Para los seguimientos en el punto:

- Si el offset del TCP debido a las correcciones de trayectoria es mayor de max\_corr, se informa de un error de seguimiento y se detiene la ejecución del programa.

*Continúa en la página siguiente*

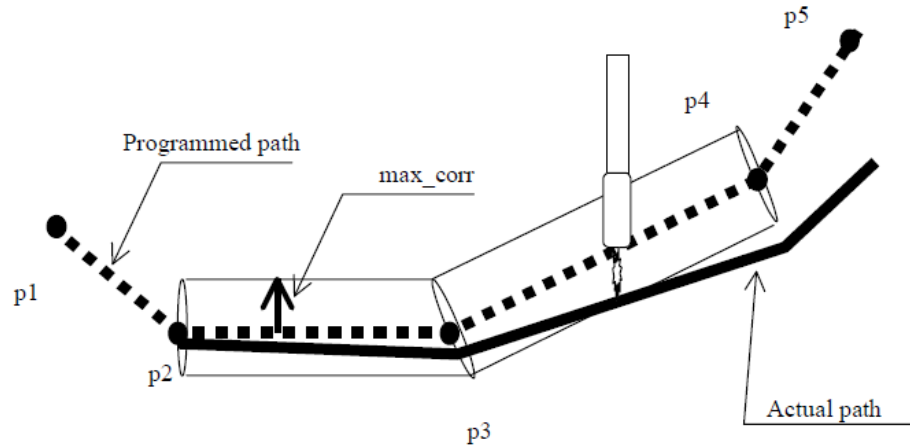
### 3 Tipos de datos

#### 3.16 captrackdata - Datos de seguimiento de CAP

##### Continuous Application Platform (CAP)

##### Continuación

La figura muestra la herramienta en una posición relativa a la trayectoria programada donde se reportaría un error de seguimiento de `max_corr`. Unidad: mm



xx1200000198

#### la\_trackdata

##### Datos de seguimiento anticipatorio

Tipo de dato: `caplatrackdata`

Define los datos de seguimiento que son específicos de seguimientos anticipatorios (por ejemplo, seguimientos de láser).

#### Ejemplos básicos

##### SIO.cfg:

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

##### Programa de RAPID:

```
PERS captrackdata captrack1 := ["digi-ip:",50,[1,10,1,0,0,0,0]];
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
```

Continúa en la página siguiente



#### Sintaxis

```
< data object of captrackdata >  
  < device of string >  
  < max_corr of num>  
  < la_trackdata of capltrackdata >
```

#### Información relacionada

|  | Descrito en:   |
|--|--|
| capltrackdata                          | <a href="#">capltrackdata - Datos de seguimiento de sensor de seguimiento anticipatorio de CAP en la página 1628</a> |
| CapAPTrSetup                           | <a href="#">CapAPTrSetup - Configuración de un seguimiento de punto en la página 83</a>                              |
| CapLATrSetup                           | <a href="#">CapLATrSetup - Configurar un sensor de seguimiento anticipatorio en la página 118</a>                    |
| <b>Continuous Application Platform</b> | <b>Application manual - Continuous Application Platform</b>  |

## 3 Tipos de datos

---

### 3.17 capweavedata - Datos de oscilación para CAP *Continuous Application Platform (CAP)*

### 3.17 capweavedata - Datos de oscilación para CAP

---

#### Utilización

capweavedata se utiliza para definir la oscilación para un proceso CAP durante su fase MAIN (consulte *Application manual - Continuous Application Platform*).

---

#### Descripción de oscilación

La oscilación se superpone en la trayectoria básica del proceso. Significa que la velocidad del proceso (definida en capspeeddata) se mantiene tal como se definió, pero la velocidad del TCP se incrementa a no ser que se alcancen las limitaciones físicas del robot.

Tipos de oscilación disponibles:

- oscilación geométrica: la forma más precisa
- oscilación de muñeca: solo se utiliza para la oscilación el eje 6 del robot
- oscilación rápida con ejes 1-3: solo se utilizan para la oscilación los ejes 1, 2 y 3 del robot
- oscilación rápida con ejes 4-6: solo se utilizan para la oscilación los ejes 4, 5 y 6 del robot

Formas de oscilación disponibles:

- Oscilación en zig zag
- Oscilación en V
- Oscilación triangular
- Oscilación circular

Todos los componentes de capweavedata se aplican a la fase MAIN.

---

#### Componentes

El sistema de coordenadas de la trayectoria se define como:

- X: trayectoria/sentido del movimiento
- Z: dirección Z de la herramienta
- Y: perpendicular tanto a X como a Z para generar un sistema de coordenadas a derechas

active

Tipo de dato: bool

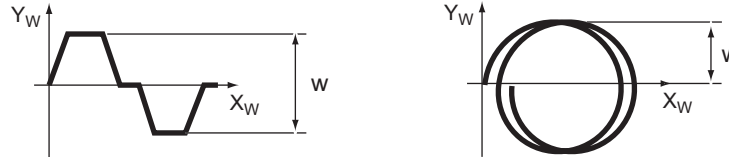
| Valor | Descripción  |
|-------|--|
| TRUE  | Realizar la oscilación durante la fase MAIN del proceso CAP    |
| FALSE | NO realizar la oscilación durante la fase MAIN del proceso CAP |

*Continúa en la página siguiente*

width

Tipo de dato: num

Para la oscilación circular, el ancho es el radio del círculo (W en la figura que aparece a continuación). Para las demás formas de oscilación, el ancho es la amplitud total del patrón de oscilación.



xx120000721

shape

Tipo de dato: num

La forma del patrón de oscilación en la fase MAIN.

| Valor | Geometría de la forma   | Resultado   |
|-------|---|---|
| 0     | Sin oscilación  |   |
| 1     | Oscilación en zig zag   | Oscilación horizontal respecto al cordón<br><br>xx1200000714                |
| 2     | Oscilación en V   | Oscilación en forma de "V", vertical respecto al cordón<br><br>xx1200000715 |
| 3     | Oscilación triangular   | Una forma triangular, vertical respecto al cordón<br><br>xx1200000716       |
| 4     | Oscilación circular<br>(Solo disponible con la oscilación geométrica, tipo de oscilación 0) | Una forma circular, vertical respecto al cordón<br><br>xx1200000717         |

type

Tipo de dato: num

Define qué ejes se utilizan para la oscilación durante la fase MAIN

| Valor especificado | Tipo de oscilación   |
|--------------------|--|
| 0                  | Oscilación geométrica. Se utilizan todos los ejes durante la oscilación. |
| 1                  | Oscilación de muñeca.  |
| 2                  | Oscilación rápida. Se utilizan los ejes 1, 2 y 3.                        |

Continúa en la página siguiente

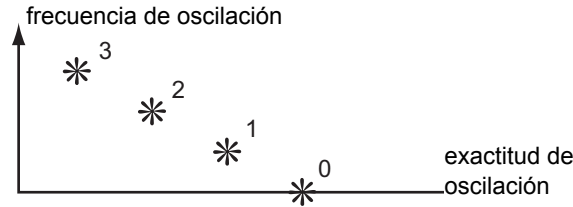
### 3 Tipos de datos

#### 3.17 capweavedata - Datos de oscilación para CAP

##### Continuous Application Platform (CAP)

##### Continuación

| Valor especificado | Tipo de oscilación                                |
|--------------------|---|
| 3                  | Oscilación rápida. Se utilizan los ejes 4, 5 y 6. |



xx1200000718



#### Nota

Todos los robots que utilizan la segunda generación de TrueMove/QuickMove tienen el comportamiento modificado para los diferentes tipos de oscilación disponibles en RW Arc y CAP, comparado con la primera generación de TrueMove/QuickMove:

- Oscilación geométrica: no hay ningún cambio.
- Oscilación de muñeca: utiliza principalmente los ejes de muñeca (4, 5 y 6) pero pueden añadirse pequeñas correcciones a los ejes principales para poder mantener el patrón en el plano deseado.
- Oscilación rápida: en la segunda generación de TrueMove/QuickMove, tanto la oscilación geométrica como la oscilación de muñeca tienen un rendimiento muy mejorado. Por lo tanto, la oscilación rápida (de ambos tipos) ya no es necesaria como un tipo de oscilación especial.

La oscilación rápida de los ejes 1, 2 y 3 es la misma que la oscilación geométrica.

La oscilación rápida de los ejes 4, 5 y 6 es la misma que la oscilación de muñeca.

Los tipos de oscilación siguen estando disponibles para compatibilidad con modelos anteriores.

El sistema utiliza la segunda generación de TrueMove/QuickMove, si hay un modificador `dyn_ipol_type 1` en `MOC.cfg`, dentro de los datos de `MOTION_PLANNER`.

length

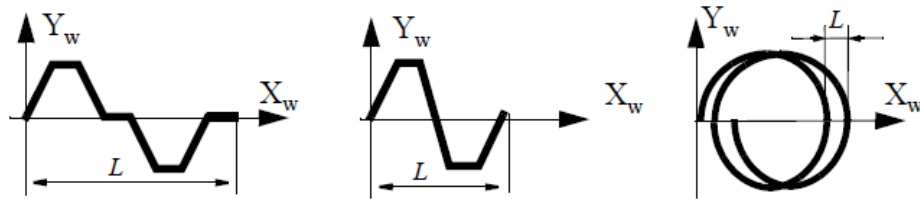
Tipo de dato: num

Define la longitud del ciclo de oscilación en la fase MAIN para la oscilación geométrica (tipo = 0) y la oscilación de muñeca (tipo = 1). El argumento de longitud no se utiliza para los otros tipos de oscilación.

Para la oscilación circular, el componente `length` define la distancia entre dos círculos sucesivos (L) si el argumento `cycle_time` cambia a 0. Si `cycle_time`

Continúa en la página siguiente

tiene un valor, la longitud puede desplazarse. El TCP gira a la izquierda con un valor de longitud positivo y a la derecha con un valor de longitud negativo.



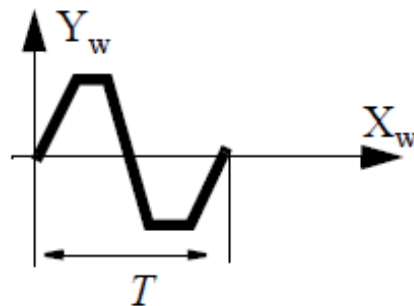
xx120000187

#### cycle\_time

Tipo de dato: num

Define la frecuencia de oscilación (en Hz) en la fase MAIN para ambos tipos de oscilación rápida (tipo = 2 o 3) y para la oscilación circular. El argumento `cycle_time` no se utiliza para los otros tipos de oscilación.

Para la oscilación circular el argumento `cycle_time` define el número de círculos por segundo. El TCP gira a la izquierda con un valor de `cycle_time` positivo y a la derecha con un valor de `cycle_time` negativo.



$T =$  Weaving cycle time

$f =$  Weaving frequency

$$f = \frac{1}{T}$$

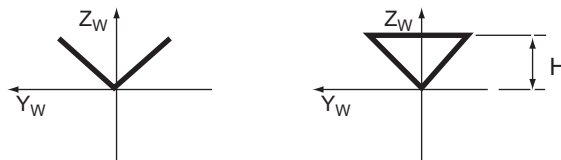
xx120000188

#### height

Tipo de dato: num

Define la altura del patrón de oscilación (en mm) durante la oscilación en V y triangular.

No disponible para la oscilación circular.



xx120000722

Continúa en la página siguiente

### 3 Tipos de datos

#### 3.17 capweavedata - Datos de oscilación para CAP

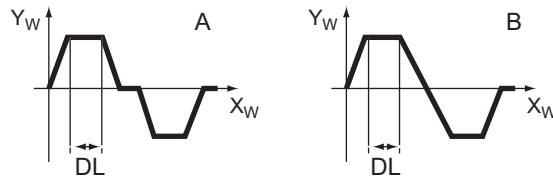
##### Continuous Application Platform (CAP)

##### Continuación

#### dwll\_left

Tipo de dato: num

La longitud de permanencia (DL) utilizada para forzar al TCP a moverse solo en el sentido del cordón a la izquierda del punto de giro de la oscilación. No disponible para la oscilación circular.



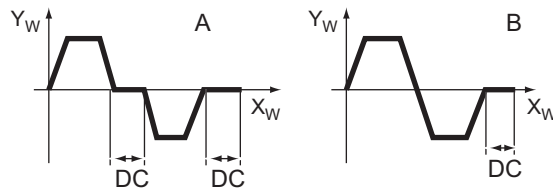
xx1200000723

|   |                              |
|---|------------------------------|
| A | Oscilación en zig zag y en V |
| B | Oscilación triangular        |

#### dwll\_center

Tipo de dato: num

La longitud de permanencia (DC) utilizada para forzar al TCP a moverse solo en el sentido del cordón en el centro del punto de la oscilación. No disponible para la oscilación circular.



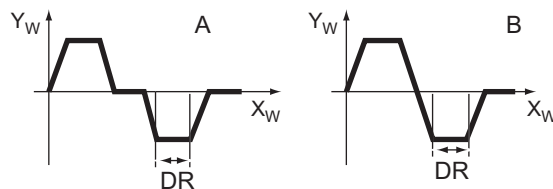
xx1200000724

|   |                              |
|---|------------------------------|
| A | Oscilación en zig zag y en V |
| B | Oscilación triangular        |

#### dwll\_right

Tipo de dato: num

La longitud de permanencia (DR) utilizada para forzar al TCP a moverse solo en el sentido del cordón a la derecha del punto de giro de la oscilación. No disponible para la oscilación circular.



xx1200000725

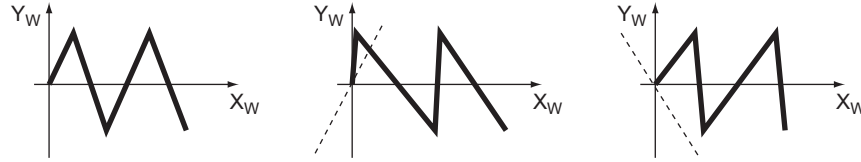
|   |                              |
|---|------------------------------|
| A | Oscilación en zig zag y en V |
| B | Oscilación triangular        |

Continúa en la página siguiente

dir

Tipo de dato: num

El ángulo de sentido de la oscilación horizontal respecto al cordón. Un ángulo de cero grados tiene como resultado una oscilación vertical respecto al cordón.

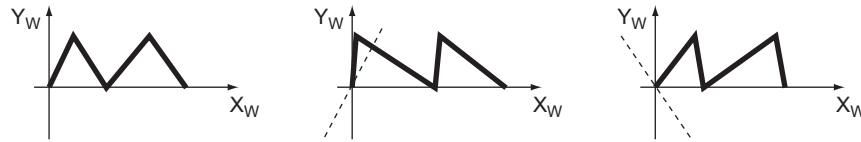


xx1200000726

tilt

Tipo de dato: num

El ángulo de inclinación de la oscilación vertical respecto al cordón. Un ángulo de cero grados tiene como resultado una oscilación vertical respecto al cordón.

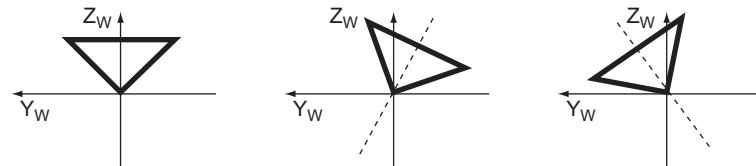


xx1200000727

rot

Tipo de dato: num

El ángulo de orientación de la oscilación, horizontal-vertical respecto al cordón. Un ángulo de cero grados tiene como resultado una oscilación simétrica.



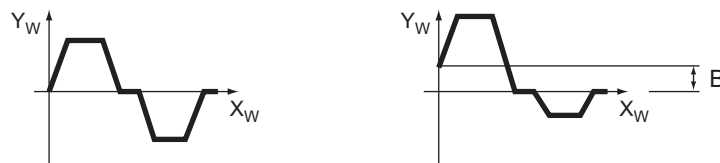
xx1200000728

bias

Tipo de dato: num

La variación horizontal respecto al patrón de oscilación. La variación solo puede especificarse para la oscilación en zig zag y no puede ser mayor a la mitad del ancho de la oscilación. No disponible para la oscilación circular.

En la figura que aparece a continuación se muestra la oscilación en zig zag con y sin variación (B).



xx1200000729

Continúa en la página siguiente

## 3 Tipos de datos

### 3.17 capweavedata - Datos de oscilación para CAP

#### Continuous Application Platform (CAP)

#### Continuación

ptrn\_sync\_on

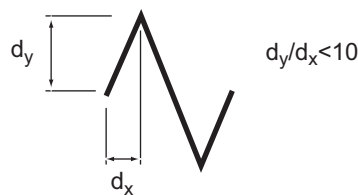
Tipo de dato: bool

| Valor | Descripción   |
|-------|---|
| TRUE  | Enviar impulsos de sincronización a los puntos de giro a la izquierda y derecha del patrón de oscilación    |
| FALSE | NO enviar impulsos de sincronización a los puntos de giro a la izquierda y derecha del patrón de oscilación |

#### Limitaciones

La frecuencia de oscilación máxima es de 2 Hz.

La inclinación del patrón de oscilación no debe superar la proporción 1:10 (84 grados). Consulte la figura que aparece a continuación.



xx1200000730

El cambio de `weave_type` en `weavedata` no es posible en puntos de zona, solo en puntos finos. Este es el comportamiento para *TrueMove* y *QuickMove*, primera y segunda generación.

Todos los robots que utilizan la segunda generación de *TrueMove* o *QuickMove* tienen el siguiente comportamiento modificado para los diferentes tipos de oscilación en RW Arc, comparado con la primera generación de *TrueMove* o *QuickMove*:

- Oscilación geométrica: no hay ningún cambio.
- Oscilación de muñeca: utiliza principalmente los ejes de muñeca (4, 5 y 6), pero pueden añadirse pequeñas correcciones a los ejes principales para poder mantener el patrón en el plano deseado.
- Oscilación rápida: en la segunda generación de *TrueMove* o *QuickMove*, tanto la oscilación geométrica como la oscilación de muñeca tienen un rendimiento muy mejorado. Por lo tanto, la oscilación rápida (ambos tipos) ya no es necesaria como un tipo de oscilación especial.

La oscilación rápida de los ejes 1, 2 y 3 es la misma que la oscilación geométrica.

La oscilación rápida de los ejes 4, 5 y 6 es la misma que la oscilación de muñeca.

Los tipos de oscilación siguen estando disponibles para compatibilidad con modelos anteriores.

El sistema utiliza la segunda generación de *TrueMove* o *QuickMove*, si hay un modificador `dyn_ipol_type 1` en `MOC.cfg` en los datos `MOTION_PLANNER` (parámetros del sistema).

Continúa en la página siguiente



#### Sintaxis

```
< 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 >
```

#### Información relacionada

|  | Descrito en:  |
|--|---|
| Tipo de dato <code>capdata</code>      | <a href="#">capdata - Datos CAP en la página 1624</a>       |
| <i>Continuous Application Platform</i> | <i>Application manual - Continuous Application Platform</i> |

## 3 Tipos de datos

### 3.18 cfgdomain - Dominio de configuración

RobotWare - OS

### 3.18 cfgdomain - Dominio de configuración

#### Utilización

`cfgdomain` (*configuration domain*) se utiliza para especificar un dominio de configuración.

#### Descripción

El uso previsto de los datos del tipo `cfgdomain` es definir el dominio de configuración que debe guardarse con la instrucción `SaveCfgData`.

#### Ejemplos básicos

El siguiente ejemplo ilustra el tipo de dato `cfgdomain`:

##### Ejemplo 1

```
SaveCfgData "SYSPAR" \File:="MYEIO.cfg", EIO_DOMAIN;
```

Guardado del dominio de la configuración de E/S con el archivo `MYEIO.cfg` en el directorio `SYSPAR`.

#### Datos predefinidos

Puede usar las constantes predefinidas siguientes para especificar un dominio de configuración.

| Nombre      | Descripción                          |
|-------------|--------------------------------------|
| EIO_DOMAIN  | Configuración del sistema de E/S     |
| MOC_DOMAIN  | Configuración de movimientos         |
| SIO_DOMAIN  | Dominio de comunicación              |
| PROC_DOMAIN | Dominio de proceso                   |
| SYS_DOMAIN  | Dominio de controlador               |
| MMC_DOMAIN  | Comunicación hombre-máquina          |
| ALL_DOMAINS | Todos los dominios enumerados arriba |

#### Características

`cfgdomain` es un tipo de dato de alias de `string` y por tanto hereda sus características.

#### Información relacionada

| Para obtener más información sobre          | Consulte   |
|---|--|
| Guardar parámetros del sistema a un archivo | <a href="#">SaveCfgData - Guardar parámetros del sistema a un archivo en la página 651</a> |
| Parámetros del sistema                      | <a href="#">Manual de referencia técnica - Parámetros del sistema</a>                      |

## 3.19 clock - Medición de tiempo

### Utilización

`clock` se utiliza para medir tiempo. Una variable `clock` funciona como un cronómetro que se usa para temporizaciones.

### Descripción

Los datos de tipo `clock` almacenan una medición de tiempo en segundos y tienen una exactitud de 0,001 segundos.

### Ejemplos básicos

El siguiente ejemplo ilustra el tipo de dato `clock`:

#### Ejemplo 1

```
VAR clock myclock;  
ClkReset myclock;
```

Se declara y pone a cero el reloj `myclock`,. Antes de usar `ClkReset`, `ClkStart`, `ClkStop` y `ClkRead`, es necesario declarar una variable de tipo `clock` en el programa.

### Limitaciones

El tiempo máximo que puede almacenar una variable de tipo `clock` es de aproximadamente 49 días (4.294.967 segundos). Las instrucciones `ClkStart`, `ClkStop` y `ClkRead` informan de los desbordamientos del reloj en el caso poco probable de que se produzcan.

Las variables `clock` deben declararse como variables `VAR`, no como variables `persistent`.

### Características

`clock` es un tipo de dato sin valor y no puede usarse en operaciones basadas en valores.

### Información relacionada

| Para obtener más información sobre                | Consulte   |
|---|--|
| Resumen de instrucciones de hora y fecha          | <i>Manual de referencia técnica - RAPID Overview, sección Resumen sobre RAPID - Sistema &amp; tiempo</i> |
| Características de los tipos de datos sin valores | <i>Manual de referencia técnica - RAPID Overview, sección Características básicas - Tipos de datos</i>   |

### 3 Tipos de datos

3.20 confdata - Datos de configuración del robot  
RobotWare - OS

#### 3.20 confdata - Datos de configuración del robot

##### Utilización

`confdata` se utiliza para definir las configuraciones de ejes del robot.

##### Descripción

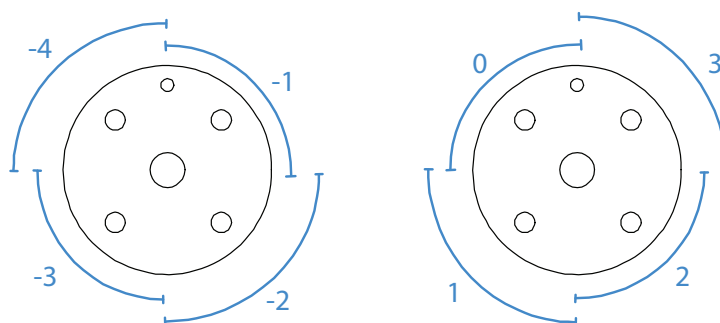
Todas las posiciones del robot se definen y almacenan mediante coordenadas rectangulares. A la hora de calcular las posiciones correspondientes de los ejes, existen con frecuencia dos o más soluciones posibles. Esto significa que el robot puede alcanzar la misma posición, es decir, situar la herramienta en la misma posición y con la misma orientación, a partir de distintas posiciones o configuraciones de los ejes del robot.

Algunos tipos de robots utilizan métodos numéricos iterativos para determinar las posiciones de los ejes. En estos casos, los parámetros de configuración puede usarse para definir valores de eje iniciales adecuados que puede usar el proceso iterativo.

Para indicar sin ambigüedades una de estas posibles configuraciones, la configuración del robot se especifica usando cuatro valores de eje. Para un eje giratorio, el valor define el cuadrante actual del eje del robot. Los cuadrantes se numeran como 0, 1, 2, y así sucesivamente (también pueden ser negativos). El número de cuadrante se conecta con el ángulo actual del eje.

Para robots de 6 ejes, el cuadrante 0 es el primer cuarto de revolución, de  $0^\circ$  a  $90^\circ$ , en sentido positivo a partir de la posición cero. El cuadrante 1 es el siguiente cuarto de revolución, de  $90^\circ$  a  $180^\circ$ , y así sucesivamente. El cuadrante -1 es el cuarto de revolución de  $0^\circ$  a  $(-90^\circ)$ , y así sucesivamente.

La figura muestra los cuadrantes de configuración para los ejes 1, 4 o 6 en un robot de 6 ejes, donde la posición cero es recta.

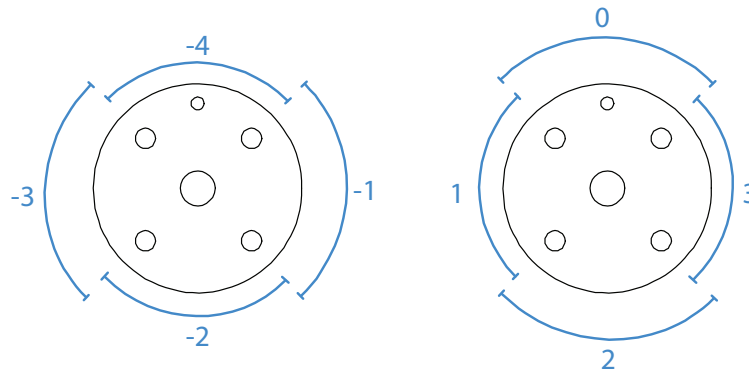


xx0500002398

Para robots de 7 ejes, el cuadrante 0 es el cuarto de revolución centrado alrededor de la posición cero,  $-45^\circ$  a  $+45^\circ$ ; cuadrante 1 es el siguiente cuarto de revolución en dirección positiva,  $45^\circ$  a  $135^\circ$ , y así sucesivamente. Cuadrante -1 es el cuarto de revolución de  $-45^\circ$  a  $-135^\circ$ , y así sucesivamente.

Continúa en la página siguiente

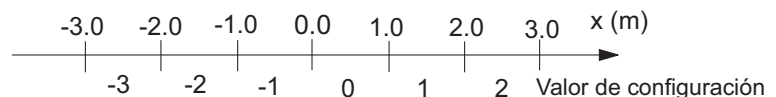
La figura muestra los cuadrantes de configuración para los ejes 1, 4 o 6 en un robot de 7 ejes, donde la posición cero es recta.



xx1700001572

En el caso de un eje lineal, el valor define el intervalo en metros del eje del robot. Para cada eje, el valor 0 significa una posición entre 0 y 1 metro y 1 significa una posición entre 1 y 2 metros. En el caso de los valores negativos, -1 significa una posición entre -1 y 0 metros, etc.

La figura siguiente muestra los valores de configuración para los ejes lineales.



xx0500002399

### Supervisión de la configuración

En algunos modelos de robot los datos de configuración (`confdata`) también se utilizan para realizar la supervisión de los puntos programados para los movimientos lineales si está configurado `ConfL\On`.

No se realiza ninguna supervisión de la configuración con `ConfJ\On`; para obtener más información, consulte [ConfJ - Controla la configuración durante el movimiento de los ejes en la página 156](#).

Antes de que se inicie un movimiento ordenado, se realiza una verificación para ver si es posible conseguir la configuración programada. Si no es posible, el programa se detiene. Cuando finaliza el movimiento (en una zona o un punto fino), también se verifica si el robot ha alcanzado la configuración programada.

La supervisión de configuración con `ConfL\On` funciona de forma diferente para los distintos robots. Consulte las siguientes secciones para obtener más detalles.

#### Robots de 6 ejes

La supervisión de la configuración comprobará que los ejes 1, 4 y 6 no se moverán más de 180 grados y que la orden de movimiento no requiere un cambio en `cfx`.

Por defecto `cfx` sólo se utiliza en los robots con enlace serie, no en los robots de barra paralela. En el caso de los robots de barra paralela, se puede habilitar `cfx` configurando el parámetro de sistema `Use cfx in robtargets for P-rod robots` como `Yes`.

*Continúa en la página siguiente*

### 3 Tipos de datos

---

#### 3.20 confdata - Datos de configuración del robot

RobotWare - OS

Continuación

Robots de 4 ejes

La supervisión de la configuración comprobará que los ejes 1 y 6 no se moverán más de 180 grados.

Robots de brazo en paralelo (robots delta)

La supervisión de la configuración comprobará que el eje 4 no se moverá más de 180 grados.

Robots SCARA

La supervisión de la configuración comprueba que los ejes 1 y 4 no se muevan más de 180 grados. También comprueba el signo del eje 2.

Robots de 7 ejes

La supervisión de la configuración comprobará que los ejes 1, 4 y 6 no se moverán más de 180 grados y que la orden de movimiento no requiere un cambio en `cfx`.

Robots de pintura

No se realiza ninguna supervisión de la configuración.

---

#### Datos de configuración el robot

Robots de 6 ejes con eslabones en serie

Existen tres singularidades dentro del rango de trabajo del robot. Para obtener más información acerca de las singularidades, consulte *Manual de referencia técnica - RAPID Overview*.

- `cf1` es el número de cuadrante del eje 1.
- `cf4` es el número de cuadrante del eje 4.
- `cf6` es el número de cuadrante del eje 6.

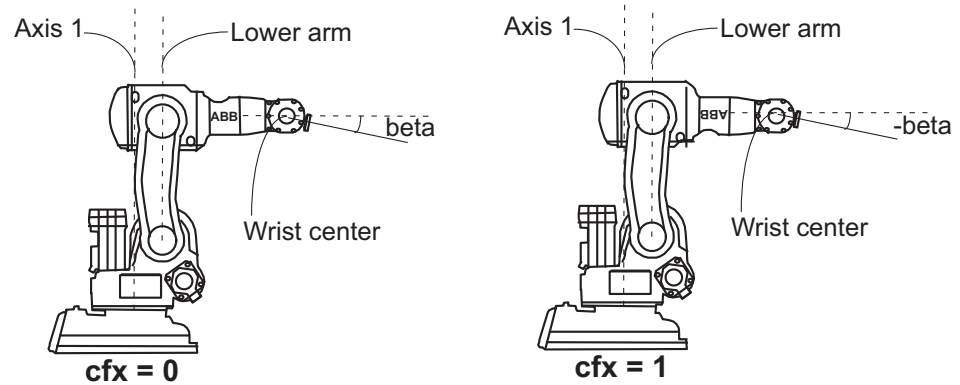
`cfx` se utiliza para seleccionar una de las ocho configuraciones posibles del robot, numeradas de 0 a 7. En la tabla que aparece a continuación se indican estas configuraciones en cuanto a cómo se posiciona el robot respecto de las tres singularidades.

| <code>cfx</code> | Centro de muñeca respecto del eje 1 | Centro de muñeca respecto del brazo interior | Ángulo del eje 5 |
|------------------|-------------------------------------|--|------------------|
| 0                | Delante de                          | Delante de                                   | Positivo         |
| 1                | Delante de                          | Delante de                                   | Negativo         |
| 2                | Delante de                          | Detrás                                       | Positivo         |
| 3                | Delante de                          | Detrás                                       | Negativo         |
| 4                | Detrás                              | Delante de                                   | Positivo         |
| 5                | Detrás                              | Delante de                                   | Negativo         |
| 6                | Detrás                              | Detrás                                       | Positivo         |
| 7                | Detrás                              | Detrás                                       | Negativo         |

En las figuras que aparecen a continuación se describen las ocho configuraciones diferentes con la misma posición y orientación de la herramienta.

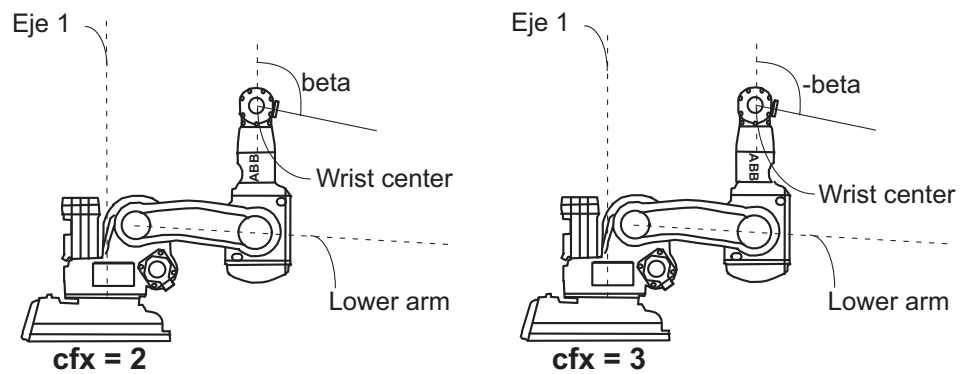
Continúa en la página siguiente

En la figura siguiente se muestra un ejemplo de las configuraciones de robot 0 y 1. Observe los distintos signos del ángulo del eje 5.



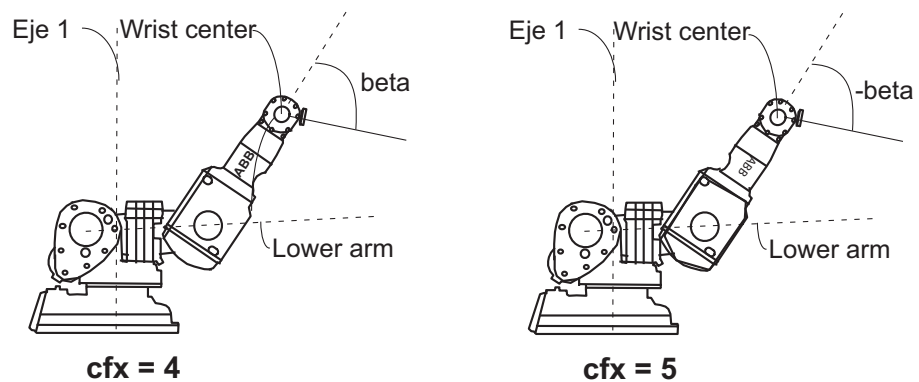
xx0500002400

En la figura siguiente se muestra un ejemplo de las configuraciones de robot 2 y 3. Observe los distintos signos del ángulo del eje 5.



xx0500002401

En la figura siguiente se muestra un ejemplo de las configuraciones de robot 4 y 5. Observe los distintos signos del ángulo del eje 5.



xx0500002402

Continúa en la página siguiente

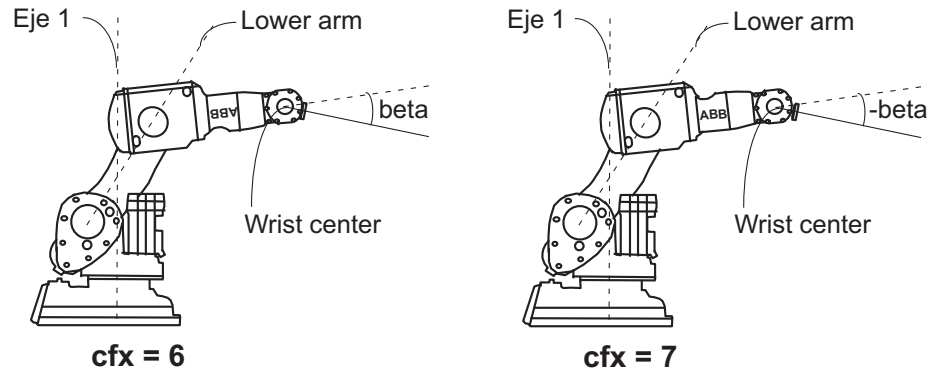
### 3 Tipos de datos

#### 3.20 confdata - Datos de configuración del robot

RobotWare - OS

Continuación

En la figura siguiente se muestra un ejemplo de las configuraciones de robot 6 y 7. Observe los distintos signos del ángulo del eje 5.



xx0500002403

#### Robots de 6 ejes con barra paralela

Se utilizan solo los tres parámetros de configuración  $cf1$ ,  $cf4$ , y  $cf6$ , a menos que el parámetro *Use cfx in robtargets for P-rod robots* esté configurado como Yes.

#### Robots de 4 ejes

Sólo se utiliza el parámetro de configuración  $cf6$ .

#### Robots de brazo paralelo

Sólo se utiliza el parámetro de configuración  $cf4$ .

#### Robots SCARA

Solo se utilizan los parámetros de configuración  $cf1$ ,  $cf4$  y  $cfx$ .

El valor  $cfx$  se utiliza para mostrar el signo del ángulo del eje 2.  $cfx$  es 1 si el ángulo del eje 2 es negativo, en caso contrario  $cfx$  es 0.

#### Robots de 7 ejes

Se utilizan los cuatro parámetros de configuración.  $cf1$ ,  $cf4$  y  $cf6$  para los ejes 1, 4 y 6, respectivamente.  $cfx$  se utiliza para seleccionar una de las 8 configuraciones de robot posibles similares al funcionamiento en otros robots.

| $cfx$ | Ángulo del eje 2 | Centro de muñeca respecto del brazo interior | Ángulo del eje 5 |
|-------|------------------|--|------------------|
| 0     | Positivo         | Delante de                                   | Positivo         |
| 1     | Positivo         | Delante de                                   | Negativo         |
| 2     | Positivo         | Detrás                                       | Positivo         |
| 3     | Positivo         | Detrás                                       | Negativo         |
| 4     | Negativo         | Delante de                                   | Positivo         |
| 5     | Negativo         | Delante de                                   | Negativo         |
| 6     | Negativo         | Detrás                                       | Positivo         |
| 7     | Negativo         | Detrás                                       | Negativo         |

Continúa en la página siguiente



#### Robots de pintura

Se utilizan los cuatro parámetros de configuración. Se utilizan `cf1`, `cf4`, `cf6` para los ejes 1, 4 y 6 respectivamente y `cfx` para el eje 5.

#### IRB 5500

Se utilizan los cuatro parámetros de configuración. Se utilizan `cf1`, `cf4`, `cf6` para los ejes 1, 4 y 6 respectivamente. El parámetro `cfx` contiene una combinación del número de cuadrante del eje 5 y las cuatro configuraciones posibles para los ejes 2 y 3.

Para obtener más información, consulte el *Product Manual - IRB 5500*.

#### IRB 5350

Este robot tiene dos ejes de rotación (brazos 1 y 2) y un eje lineal (brazo 3).

- `cf1` se utiliza para el eje de rotación 1
- `cfx` se utiliza para el eje de rotación 2
- `cf4` y `cf6` no se utilizan

---

#### Componentes

`cf1`

Tipo de dato: `num`

Eje de rotación:

El cuadrante actual del eje 1, expresado como un número entero positivo o negativo.

Eje lineal:

El intervalo actual en metros del eje 1, expresado como un número entero positivo o negativo.

`cf4`

Tipo de dato: `num`

Eje de rotación:

El cuadrante actual del eje 4, expresado como un número entero positivo o negativo.

Eje lineal:

El intervalo actual en metros del eje 4, expresado como un número entero positivo o negativo.

`cf6`

Tipo de dato: `num`

Eje de rotación:

El cuadrante actual del eje 6, expresado como un número entero positivo o negativo.

Eje lineal:

El intervalo actual en metros del eje 6, expresado como un número entero positivo o negativo.

`cfx`

Tipo de dato: `num`

Eje de rotación:

*Continúa en la página siguiente*

### 3 Tipos de datos

#### 3.20 confdata - Datos de configuración del robot

RobotWare - OS

Continuación

En los robots con eslabones en serie, la configuración actual del robot, expresada como un entero de 0 a 7.

En los robots SCARA, la configuración actual del robot, expresada como un entero de 0 a 1; consulte [Robots SCARA en la página 1652](#).

En los robots de 7 ejes, la configuración actual del robot, expresada como un entero de 0 a 7; consulte [Robots de 7 ejes en la página 1652](#).

En los robots de pintura, el cuadrante actual del eje 5, expresado como un número entero positivo o negativo. En el IRB 5500, consulte [IRB 5500 en la página 1653](#).

En otros robots, el cuadrante actual del eje 2, expresado como un número entero positivo o negativo.

Eje lineal:

El intervalo actual en metros del eje 2, expresado como un número entero positivo o negativo.

#### Ejemplos básicos

El siguiente ejemplo ilustra el tipo de dato `confdata`:

##### Ejemplo 1

```
VAR confdata conf15 := [1, -1, 0, 0]
```

Una configuración de robot `conf15` para un tipo de robot de pintura se define de la forma siguiente:

- La configuración del eje 1 del robot es el cuadrante 1, es decir, de 90 a 180°.
- La configuración del eje 4 del robot es el cuadrante -1, es decir, de -0 a (-90°).
- La configuración del eje 6 del robot es el cuadrante 0, es decir, de 0 a 90°.
- La configuración del eje 5 del robot es el cuadrante 0, es decir, de 0 a 90°.

#### Estructura

```
< dataobject of confdata >  
  < cf1 of num >  
  < cf4 of num >  
  < cf6 of num >  
  < cfx of num >
```

#### Información relacionada

| Para obtener más información sobre  | Consulte   |
|---|--|
| Sistemas de coordenadas<br>Manejo de datos de configuración<br>Singularidades | <i>Manual de referencia técnica - RAPID Overview</i>           |
| Datos de posición   | <a href="#">robtargt - Datos de posición en la página 1750</a> |
| Parámetros del sistema  | <i>Manual de referencia técnica - Parámetros del sistema</i>   |

## 3.21 corrdescr - Descriptor de generador de correcciones

### Utilización

`corrdescr` (*Correction generator descriptor*) se utiliza desde los generadores de correcciones. Un generador de correcciones añade offsets geométricos al sistema de coordenadas de la trayectoria.

### Descripción

Los datos de tipo `corrdescr` contienen una referencia a un generador de correcciones.

La conexión a un generador de correcciones se realiza mediante la instrucción `CorrCon` y el descriptor (la referencia al generador de correcciones) puede usarse para generar offsets geométricos en el sistema de coordenadas de la trayectoria, con ayuda de la instrucción `CorrWrite`.

Los offsets proporcionados anteriormente pueden ser eliminados mediante la desconexión de un generador de correcciones con la instrucción `CorrDiscon`. Es posible eliminar todos los generadores de correcciones conectados, mediante la instrucción `CorrClear`.

La función `CorrRead` devuelve la suma de todos los offsets suministrados hasta ese momento (incluye todos los generadores de correcciones que estén conectados).

### Ejemplos básicos

El siguiente ejemplo ilustra el tipo de dato `corrdescr`:

#### Ejemplo 1

```
VAR corrdescr id;
VAR pos offset;
...
CorrCon id;
offset := [1, 2 ,3];
CorrWrite id, offset;
```

La conexión al generador de correcciones se realiza con la instrucción `CorrCon`. Se hace referencia a él con el descriptor `id`. A continuación, la entrega de los offsets al generador de correcciones (con la referencia `id`) se realiza con la instrucción `CorrWrite`.

### Características

`corrdescr` es un tipo de dato sin valor.

### Información relacionada

| Para obtener más información sobre          | Consulte   |
|---|--|
| Conexión con un generador de correcciones   | <a href="#">CorrCon - Establece una conexión con un generador de correcciones en la página 177</a> |
| Desconexión de un generador de correcciones | <a href="#">CorrDiscon - Cierra la conexión con un generador de correcciones en la página 182</a>  |

*Continúa en la página siguiente*

### 3 Tipos de datos

---

#### 3.21 corrdescr - Descriptor de generador de correcciones

*Path Offset*

*Continuación*

| <b>Para obtener más información sobre</b>            | <b>Consulte</b>  |
|--|--|
| Escritura en un generador de correcciones            | <a href="#"><i>CorrWrite - Escribe en un generador de correcciones en la página 183</i></a>            |
| Lectura de los offsets totales actuales              | <a href="#"><i>CorrRead - Lee los offsets totales actuales en la página 1235</i></a>                   |
| Eliminación de todos los generadores de correcciones | <a href="#"><i>CorrClear - Elimina todos los generadores de correcciones en la página 176</i></a>      |
| Características de los tipos de datos sin valor      | <i>Manual de referencia técnica - RAPID Overview, sección Características básicas - Tipos de datos</i> |

## 3.22 datapos - Inclusión de un bloque para un objeto de datos

### Utilización

`datapos` es el bloque de inclusión de un objeto de datos (dato interno del sistema) recuperados con la función `GetNextSym`.

### Descripción

Los datos del tipo `datapos` contienen información acerca de dónde está definido un objeto determinado dentro del sistema. Se utilizan con las instrucciones `GetDataVal` y `SetDataVal`.

### Ejemplos básicos

El siguiente ejemplo ilustra el tipo de dato `datapos`:

#### Ejemplo 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
```

Esta sesión cambia a `TRUE` todos los objetos de datos locales de tipo `bool` cuyo nombre comience con `my` en el módulo `mymod`.

### Características

`datapos` es un tipo de dato sin valor.

### Información relacionada

| Para obtener más información sobre                              | Consulte  |
|---|---|
| Definición de un conjunto de símbolos en una sesión de búsqueda | <a href="#">SetDataSearch - Definir el conjunto de símbolos de una secuencia de búsqueda en la página 696</a>             |
| Obtención del siguiente símbolo coincidente                     | <a href="#">GetNextSym - Obtiene el siguiente símbolo coincidente en la página 1309</a>                                   |
| Obtención del valor de un objeto de datos                       | <a href="#">GetDataVal - Obtiene el valor de un objeto de datos en la página 228</a>                                      |
| Asignación del valor de un objeto de datos                      | <a href="#">SetDataVal - Establece el valor de un objeto de datos en la página 701</a>                                    |
| Asignación del valor de varios objetos                          | <a href="#">SetAllDataVal - Establece un valor en todos los objetos de datos de un conjunto definido en la página 691</a> |
| <i>Advanced RAPID</i>   | <i>Especificaciones del producto - Controller software IRC5</i>   |

## 3 Tipos de datos

---

### 3.23 dionum - Valores digitales (0-1)

RobotWare - OS

### 3.23 dionum - Valores digitales (0-1)

---

#### Utilización

`dionum`(*digital input output numeric*) se utiliza para valores digitales (0 ó 1).

Este tipo de dato se utiliza junto con instrucciones y funciones dedicadas a la gestión de señales digitales de entrada o salida.

---

#### Descripción

Los datos del tipo `dionum` representan los valores digitales 0 ó 1.

---

#### Ejemplos básicos

El siguiente ejemplo ilustra el tipo de dato `dionum`:

##### Ejemplo 1

```
CONST dionum close := 1;  
SetDO gripl, close;
```

Definición de la constante `close` con el valor 1. La señal `gripl` cambia al valor `close`, es decir 1.

---

#### Datos predefinidos

Las constantes `high`, `low` y `edge` están predefinidas en el sistema:

```
CONST dionum low:=0;  
CONST dionum high:=1;  
CONST dionum edge:=2;
```

Las constantes `low` y `high` se han diseñado para instrucciones de E/S.

`Edge` puede usarse junto con las instrucciones de interrupciones `ISignalDI` y `ISignalDO`.

---

#### Características

`dionum` es un tipo de dato de alias de `num` y por tanto hereda sus características.

---

#### Información relacionada

| Para obtener más información sobre           | Consulte  |
|--|---|
| Resumen de instrucciones de entrada y salida | <i>Manual de referencia técnica - RAPID Overview, sección Resumen sobre RAPID - Señales de entrada y salida</i> |
| Configuración de E/S                         | <i>Manual de referencia técnica - Parámetros del sistema</i>  |
| Tipos de datos de alias                      | <i>Manual de referencia técnica - RAPID Overview, sección Características básicas - Tipos de datos</i>          |

## 3.24 dir - Estructura de directorio de archivos

### Utilización

`dir` (*directory*) se utiliza para recorrer estructuras de directorios.

### Descripción

Los datos del tipo `dir` contienen una referencia a un directorio de un disco o una red. Pueden conectarse al directorio físico mediante la instrucción `OpenDir` y utilizarse a continuación para operaciones de lectura y escritura.

### Ejemplos básicos

El siguiente ejemplo ilustra el tipo de dato `dir`:

#### Ejemplo 1

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

Este ejemplo imprime los nombres de todos los archivos o subdirectorios que se encuentran dentro del directorio especificado.

### Características

`dir` es un tipo de dato sin valor y no puede usarse en operaciones basadas en valores.

### Información relacionada

| Para obtener más información sobre        | Consulte  |
|---|---|
| Apertura de un directorio                 | <a href="#">OpenDir - Abre un directorio en la página 494</a>                         |
| Creación de un directorio                 | <a href="#">MakeDir - Crea un nuevo directorio en la página 358</a>                   |
| Lectura de un directorio                  | <a href="#">ReadDir - Lee la siguiente entrada de un directorio en la página 1437</a> |
| Cierre de un directorio                   | <a href="#">CloseDir - Cierra un directorio en la página 153</a>                      |
| Eliminación de un directorio              | <a href="#">RemoveDir - Elimina un directorio en la página 598</a>                    |
| Eliminación de un archivo                 | <a href="#">RemoveFile - Elimina un archivo en la página 600</a>                      |
| Cambio del nombre de un archivo           | <a href="#">RenameFile - Cambia el nombre de un archivo en la página 604</a>          |
| Comprobación del tipo del archivo         | <a href="#">IsFile - Comprobar el tipo de un archivo en la página 1351</a>            |
| Gestión de archivos y dispositivos de E/S | <a href="#">Application manual - Controller software IRC5</a>                         |

## 3 Tipos de datos

---

### 3.25 dnum - Valores numéricos dobles

RobotWare - OS

### 3.25 dnum - Valores numéricos dobles

---

#### Utilización

`dnum` se utiliza para valores numéricos, por ejemplo contadores. Es capaz de contener valores enteros mayores que `num` pero sus características y funciones son las mismas que las de `num`.

---

#### Descripción

El valor del tipo de dato `dnum` puede ser:

- Un entero, por ejemplo -5
- Un número con decimales, por ejemplo 3,45

También puede escribirse de forma exponencial, por ejemplo 2E3 (=  $2 \cdot 10^3 = 2.000$ ), 2,5E-2 (= 0,025).

Los enteros entre -4503599627370496-4503599627370496 y +4503599627370496 se almacenan siempre como enteros exactos.

---

#### Ejemplos básicos

Los siguientes ejemplos ilustran el tipo de dato `dnum`:

##### Ejemplo 1

```
VAR dnum reg1;  
...  
reg1:=1000000;  
reg1 recibe el valor 1000000.
```

##### Ejemplo 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;
```

##### Ejemplo 3

```
VAR dnum a:=0;  
VAR dnum b:=0;  
a := 10 DIV 3;  
b := 10 MOD 3;
```

División entera en la que se asigna un entero `a` (=3) y se asigna a `b` el resto (=1).

---

#### Limitaciones

Los valores literales entre -4503599627370496 y 4503599627370496 asignados a una variable `dnum` se almacenan como enteros exactos.

Si un valor literal que ha sido interpretado como un valor `num` es asignado o usado como un valor `dnum`, es convertido automáticamente a `dnum`.

---

*Continúa en la página siguiente*



#### Información relacionada

| Para obtener más información sobre                      | Consulte   |
|---|--|
| Valores numéricos con el tipo de datos <code>num</code> | <a href="#">num - Valores numéricos en la página 1712</a>  |
| Expresiones numéricas                                   | <i>Manual de referencia técnica - Descripción general de RAPID, sección Programación básica en RAPID</i> |
| Operaciones con valores numéricos                       | <i>Manual de referencia técnica - Descripción general de RAPID, sección Programación básica en RAPID</i> |

## 3 Tipos de datos

### 3.26 errdomain - Dominio del error

RobotWare - OS

### 3.26 errdomain - Dominio del error

#### Utilización

`errdomain` (*error domain*) se utiliza para especificar un dominio de error.

#### Descripción

Los datos de tipo `errdomain` representan el dominio en el que se ha registrado un error, una advertencia o un cambio de estado.

#### Ejemplos básicos

El siguiente ejemplo ilustra el tipo de dato `errdomain`:

##### Ejemplo 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
```

Cuando se detecta un error con la rutina `TRAP trap_err`, el dominio, el número y el tipo del error se almacenan en las variables adecuadas.

#### Datos predefinidos

Puede usar las constantes predefinidas siguientes para especificar un dominio de error.

| Nombre       | Dominio del error                                     | Valor |
|--------------|---|-------|
| COMMON_ERR   | Todos los dominios de error y de cambios de estado    | 0     |
| OP_STATE     | Cambio de estado operativo                            | 1     |
| SYSTEM_ERR   | Errores de sistema                                    | 2     |
| HARDWARE_ERR | Errores de hardware                                   | 3     |
| PROGRAM_ERR  | Errores de programa                                   | 4     |
| MOTION_ERR   | Errores de movimiento                                 | 5     |
| OPERATOR_ERR | Errores de operador. Obsoleto, ya no se usa.          | 6     |
| IO_COM_ERR   | Errores de E/S y comunicación                         | 7     |
| USER_DEF_ERR | Errores definidos por el usuario (elevados por RAPID) | 8     |
| SAFETY_ERR   | Eventos relacionados con la seguridad                 | 9     |
| PROCESS_ERR  | Errores de proceso                                    | 11    |
| CFG_ERR      | Error de configuración                                | 12    |

Continúa en la página siguiente

#### Características

`errdomain` es un tipo de dato de alias de num y por tanto hereda sus características.

#### Información relacionada

| Para obtener más información sobre         | Consulte  |
|--|---|
| Solicitud de una interrupción para errores | <a href="#">Error - Solicita una interrupción para errores en la página 257</a> |
| Números de errores                         | <i>Manual del operador - Solución de problemas de IRC5</i>                      |
| Tipos de datos de alias                    | <i>Manual de referencia técnica - RAPID Overview</i>                            |
| <i>Advanced RAPID</i>                      | <i>Especificaciones del producto - Controller software IRC5</i>                 |

## 3 Tipos de datos

---

### 3.27 errnum - Número de error

RobotWare - OS

### 3.27 errnum - Número de error

---

#### Utilización

`errmsg` se utiliza para describir todos los errores recuperables (no fatales) que se producen durante la ejecución del programa, como por ejemplo, la división por cero.

#### Descripción

Si el robot detecta un error durante la ejecución del programa, es posible resolverlo con el gestor de errores de la rutina. Algunos ejemplos de estos errores es la existencia de valores demasiado altos y la división por cero. Por tanto, se asignan valores diferentes a la variable de sistema `ERRNO`, del tipo `errmsg`, en función de la naturaleza de un error. El gestor de errores puede ser capaz de corregir un error leyendo el contenido de esta variable, tras lo cual la ejecución del programa puede continuar de la forma adecuada.

Los errores también pueden crearse desde el propio programa mediante la instrucción `RAISE`. Este tipo concreto de error puede detectarse en el gestor de errores mediante la especificación de un número de error (de 0 a 90 o reservado mediante la instrucción `BookErrNo`) como argumento de `RAISE`.

#### Ejemplos básicos

Los siguientes ejemplos ilustran el tipo de dato `errmsg`:

##### Ejemplo 1

```
reg1 := reg2 / reg3;
...
ERROR
  IF ERRNO = ERR_DIVZERO THEN
    reg3 := 1;
  RETRY;
ENDIF
```

Si `reg3 = 0`, el robot detecta un error cuando se realiza la división. Sin embargo, es posible detectar y corregir el error mediante la asignación del valor 1 a `reg3`. A continuación, puede realizarse de nuevo la división y proseguir con la ejecución del programa.

##### Ejemplo 2

```
CONST errmsg machine_error := 1;
...
IF dil=0 RAISE machine_error;
...
ERROR
  IF ERRNO=machine_error RAISE;
```

Se produce un error en una máquina (lo cual se detecta a través de la señal de entrada `dil`). Se salta al gestor de errores de la rutina que, a su vez, llama al gestor de errores de la rutina desde la que se llamó a la rutina actual, que es posiblemente el lugar en el que se puede resolver el error. La constante `machine_error` se utiliza para informar al gestor de errores de qué tipo exacto de error se trata.

*Continúa en la página siguiente*

## Datos predefinidos

Puede usar la variable de sistema `ERRNO` para obtener el último error que se ha producido. Existen varias constantes predefinidas que puede usar para determinar qué tipo de error se ha producido.

| Nombre                         | Causa del error  |
|--------------------------------|--|
| <code>ERR_ACC_TOO_LOW</code>   | Se ha especificado una aceleración o desaceleración demasiado baja en la instrucción <code>PathAccLim</code> o <code>WorldAccLim</code> .  |
| <code>ERR_ACTIV_PROF</code>    | Error en el perfil activado.   |
| <code>ERR_ALIASCAM_DEF</code>  | La cámara en el argumento <code>CameraName</code> o el <code>cameradev</code> utilizado en el argumento <code>FromCamera</code> no está definida en la configuración <code>Communication</code> de los parámetros del sistema. O la <code>ToCamera</code> no está declarada en el programa de RAPID o ya está definida en la configuración <code>Communication</code> de los parámetros del sistema. |
| <code>ERR_ALIASIO_DEF</code>   | El <code>FromSignal</code> no está definido en la configuración de E/S ni <code>ToSignal</code> está declarado en el programa RAPID, o bien no está definido en la configuración de E/S. Instrucción <code>AliasIO</code> .  |
| <code>ERR_ALIASIO_TYPE</code>  | Los tipos de señales de los argumentos <code>FromSignal</code> y <code>ToSignal</code> no son iguales ( <code>signalx</code> ). Instrucción <code>AliasIO</code> .   |
| <code>ERR_ALRDYCNT</code>      | La variable de interrupción ya está vinculada a una rutina TRAP.   |
| <code>ERR_ALRDY_MOVING</code>  | El robot ya se está moviendo cuando se ejecuta la instrucción <code>StartMove</code> o <code>StartMoveRetry</code> .   |
| <code>ERR_AO_LIM</code>        | Valor de señal analógica fuera de límites  |
| <code>ERR_ARGDUPCND</code>     | Existe más de un argumento condicional para el mismo parámetro   |
| <code>ERR_ARGNAME</code>       | El argumento es una expresión, no presente, o del tipo <code>switch</code> cuando se ejecuta <code>ArgName</code> .  |
| <code>ERR_ARGNOTPER</code>     | El argumento no es una referencia persistente.   |
| <code>ERR_ARGNOTVAR</code>     | El argumento no es una referencia a una variable.  |
| <code>ERR_ARGVALERR</code>     | Error de valor de argumento.   |
| <code>ERR_AXIS_ACT</code>      | El eje no está activo.   |
| <code>ERR_AXIS_IND</code>      | El eje no es independiente.  |
| <code>ERR_AXIS_MOVING</code>   | El eje está en movimiento.   |
| <code>ERR_AXIS_PAR</code>      | Parámetro de eje incorrecto en la función o la instrucción.  |
| <code>ERR_BUSSTATE</code>      | Se ejecuta <code>IOEnable</code> y la red de E/S se encuentra en el estado de error o entra en el estado de error antes de que el dispositivo de E/S se active.  |
| <code>ERR_BWDLIMIT</code>      | Límite <code>StepBwdPath</code> .  |
| <code>ERR_CALC_NEG</code>      | Error de cálculo negativo de <code>StrDig</code> .   |
| <code>ERR_CALC_OVERFLOW</code> | Desbordamiento de cálculo de <code>StrDig</code> .   |
| <code>ERR_CALC_DIVZERO</code>  | División entre cero de <code>StrDig</code> .   |

Continúa en la página siguiente

### 3 Tipos de datos

#### 3.27 errnum - Número de error

RobotWare - OS

Continuación

| Nombre                 | Causa del error   |
|------------------------|---|
| ERR_CALLPROC           | Error en una llamada a un procedimiento (no es un procedimiento) en tiempo de ejecución (con enlazamiento en tiempo de ejecución).  |
| ERR_CAM_BUSY           | La cámara está ocupada con otra petición y no puede ejecutar la orden actual.   |
| ERR_CAM_COM_TIMEOUT    | Se agotó el tiempo límite de la comunicación con la cámara. La cámara no responde.  |
| ERR_CAM_GET_MISMATCH   | El parámetro capturado de la cámara con la instrucción <code>CamGetParameter</code> tiene un tipo de dato incorrecto.   |
| ERR_CAM_MAXTIME        | Tiempo límite agotado al ejecutar una instrucción <code>CamLoadJob</code> o <code>CamGetResult</code> .   |
| ERR_CAM_NO_MORE_DATA   | No es posible capturar más resultados de visión.  |
| ERR_CAM_NO_PROGMODE    | La cámara no está en el modo de programa.   |
| ERR_CAM_NO_RUNMODE     | La cámara no está en el modo de funcionamiento.   |
| ERR_CAM_NOT_ON_NETWORK | La cámara no está conectada.  |
| ERR_CAM_SET_MISMATCH   | El parámetro escrito en la cámara con la instrucción <code>CamGetParameter</code> tiene un tipo de dato incorrecto o el valor está fuera de rango.  |
| ERR_CFG_INTERNAL       | No se permite leer o escribir un parámetro interno  |
| ERR_CFG_ILL_DOMAIN     | El <code>cfgdomain</code> utilizado en la instrucción <code>SaveCfgData</code> no es válido o no está en uso.   |
| ERR_CFG_ILLTYPE        | Error de coincidencia de tipos - <code>ReadCfgData</code> , <code>WriteCfgData</code> .   |
| ERR_CFG_LIMIT          | Límite de datos - <code>WriteCfgData</code> .   |
| ERR_CFG_NOTFND         | No encontrado - <code>ReadCfgData</code> , <code>WriteCfgData</code> .  |
| ERR_CFG_OUTOFBOUNDS    | Si <code>ListNo</code> tiene el valor -1 en la entrada o mayor que el número de instancias disponibles - <code>ReadCfgData</code> , <code>WriteCfgData</code> .   |
| ERR_CFG_WRITEFILE      | El directorio no existe o el <code>FilePath</code> y el <code>File</code> utilizado es un directorio, o bien existe algún otro problema con el guardado del archivo al utilizar la instrucción <code>SaveCfgData</code> . |
| ERR_CNTNOTVAR          | El objetivo de <code>CONNECT</code> no es una referencia a una variable.  |
| ERR_CNV_NOT_ACT        | El transportador no está activado.  |
| ERR_CNV_CONNECT        | La instrucción <code>WaitWobj</code> ya está activa.  |
| ERR_CNV_DROPPED        | El objeto que estaba esperando la instrucción <code>WaitWobj</code> ha sido desechado.  |
| ERR_CNV_OBJ_LOST       | El objeto que estaba esperando la instrucción <code>WaitWobj</code> o <code>WaitSensor</code> ha pasado <code>StartwindowWidth</code> sin conectarse.   |
| ERR_COLL_STOP          | Detención del movimiento a causa de una colisión de movimiento.   |
| ERR_CONC_MAX           | Se ha superado el número de instrucciones de movimiento seguidas con el argumento <code>\Conc</code> .  |
| ERR_COMM_INIT          | No se pudo inicializar la interfaz de comunicación.   |

Continúa en la página siguiente

| Nombre           | Causa del error  |
|------------------|--|
| ERR_DEV_MAXTIME  | Tiempo límite al ejecutar una instrucción ReadBin, ReadNum, ReadStr, ReadStrBin, ReadAnyBin, o ReadRawBytes.   |
| ERR_DIPLAG_LIM   | DipLag demasiado grande en la instrucción TriggSpeed conectada al TriggL/TriggC/TriggJ/CapL/CapC actual.       |
| ERR_DIVZERO      | División entre cero.   |
| ERR_EXCRTYMAX    | Se ha sobrepasado el número máximo de reintentos. RAISE o TRYNEXT pueden utilizarse para gestionar este error. |
| ERR_EXECPHR      | Se ha intentado ejecutar una instrucción con un marcador de sustitución.                                       |
| ERR_FILEACC      | El acceso a un archivo se realiza de forma incorrecta.   |
| ERR_FILEEXIST    | El archivo ya existe.  |
| ERR_FILEOPEN     | No es posible abrir un archivo.  |
| ERR_FILNOTFND    | Archivo no encontrado.   |
| ERR_FNCNORET     | Sin valor de retorno.  |
| ERR_FRAME        | Imposible calcular la nueva base de coordenadas.   |
| ERR_GO_LIM       | Valor de señal digital de grupo fuera de límite.   |
| ERR_ILLDIM       | Dimensiones de matriz incorrectas.   |
| ERR_ILLQUAT      | Se ha intentado usar un valor no válido de orientación (cuaternio).  |
| ERR_ILLRAISE     | Número de error de RAISE fuera de rango.   |
| ERR_INDCNV_ORDER | Una instrucción requiere la ejecución de IndCnvInit antes de ejecutarse.                                       |
| ERR_INOISSAFE    | Si se intenta desactivar temporalmente una interrupción segura con ISleep.                                     |
| ERR_INOMAX       | No hay más números de interrupción disponibles.  |
| ERR_INT_NOTVAL   | Entero no válido, valor decimal.   |
| ERR_INT_MAXVAL   | Entero no válido, valor demasiado grande o demasiado pequeño.  |
| ERR_INVDIM       | Dimensiones diferentes.  |
| ERR_IODISABLE    | Se alcanzó el tiempo límite al ejecutar IODisable.   |
| ERR_IOENABLE     | Se alcanzó el tiempo límite al ejecutar IOEnable.  |
| ERR_IOERROR      | Error de E/S de instrucción Save, Load y WaitLoad.   |
| ERR_LINKREF      | Error de referencia en la tarea de programa.   |
| ERR_LOADED       | El módulo de programa ya está cargado.   |
| ERR_LOADID_FATAL | Solo para uso interno de LoadId y ManLoadIdProc.   |
| ERR_LOADID_RETRY | Solo para uso interno de LoadId.   |
| ERR_LOADNO_INUSE | La sesión de carga se está utilizando en StartLoad.  |
| ERR_LOADNO_NOUSE | La sesión de carga no se está utilizando en CancelLoad.  |
| ERR_MODULE       | Nombre de módulo incorrecto en la instrucción Save y EraseModule.  |

Continúa en la página siguiente

### 3 Tipos de datos

#### 3.27 errnum - Número de error

RobotWare - OS

Continuación

| Nombre             | Causa del error   |
|--------------------|---|
| ERR_MOD_NOT_LOADED | El módulo no existe, el símbolo no es un módulo o el nombre es demasiado largo para ser un símbolo. Error en la función <code>ModTimeDnum</code> .                        |
| ERR_NAME_INVALID   | El nombre del dispositivo de E/S no existe.   |
| ERR_NO_ALIASIO_DEF | La variable de señal es una variable declarada en RAPID. No se ha conectado a una señal E/S definida en la configuración de E/S con la instrucción <code>AliasIO</code> . |
| ERR_NORUNUNIT      | Se ha perdido el contacto con el dispositivo de E/S.  |
| ERR_NO_SGUN        | El nombre especificado de la herramienta servo no es una herramienta servo configurada.   |
| ERR_NOTARR         | El dato no es una matriz.   |
| ERR_NOTEQDIM       | La dimensión de la matriz utilizada al llamar a la rutina no coincide con sus parámetros.   |
| ERR_NOTINTVAL      | No es un valor entero.  |
| ERR_NOTPRES        | Se utiliza un parámetro, a pesar de que no se utilizó el argumento correspondiente en la llamada a la rutina.   |
| ERR_NOTSAVED       | El módulo ha cambiado desde que fue cargado en el sistema.  |
| ERR_NOT_MOVETASK   | La tarea especificada es una tarea sin movimiento.  |
| ERR_NUM_LIMIT      | El valor es superior a 3,40282347E+38 o inferior a -3,40282347E+38.   |
| ERR_ORIENT_VALUE   | Valor de orientación incorrecto en la función <code>NOrient</code> .  |
| ERR_OUTOFBND       | El índice de la matriz está fuera de los límites permitidos.  |
| ERR_OVERFLOW       | Desbordamiento de reloj.  |
| ERR_OUTSIDE_REACH  | La posición ( <code>robtargt</code> ) está fuera del área de trabajo del robot para la función <code>CalcJoint</code> .   |
| ERR_PATH           | Falta la ruta de destino en la instrucción <code>Save</code> .  |
| ERR_PATHDIST       | Distancia de recuperación demasiado larga para la instrucción <code>StartMove</code> , <code>StartMoveRetry</code> o <code>SetLeadThrough</code> .                        |
| ERR_PATH_STOP      | Detención del movimiento a causa de un error de proceso.  |
| ERR_PERSSUPSEARCH  | La variable persistente ya es TRUE al comienzo del proceso de búsqueda.   |
| ERR_PID_MOVESTOP   | Solo para uso interno de <code>LoadId</code> y <code>ManLoadIdProc</code> .   |
| ERR_PID_RAISE_PP   | Error en <code>ParIdRobValid</code> , <code>ParIdPosValid</code> , <code>LoadId</code> o <code>ManLoadIdProc</code> .   |
| ERR_PRGMEMFULL     | Memoria de programas llena.   |
| ERR_PROCSIGNAL_OFF | La señal de proceso está desactivada.   |
| ERR_PROGSTOP       | El robot ya está en el estado de programa parado cuando se ejecuta una instrucción <code>StartMove</code> , <code>StartMoveRetry</code> o <code>SetLeadThrough</code> .   |
| ERR_RANYBIN_CHK    | Error de suma de comprobación detectado en la transferencia de datos con la instrucción <code>ReadAnyBin</code> .   |
| ERR_RANYBIN_EOF    | Se ha detectado el final del archivo antes de que se lean todos los bytes en la instrucción <code>ReadAnyBin</code> o <code>ReadRawBytes</code> .                         |

Continúa en la página siguiente



| Nombre                  | Causa del error   |
|-------------------------|---|
| ERR_RCVDATA             | Se ha intentado leer datos no numéricos con ReadNum.  |
| ERR_REFUNKDAT           | Referencia a objeto de datos completo desconocido.  |
| ERR_REFUNKFUN           | Referencia a función desconocida.   |
| ERR_REFUNKPRC           | Referencia a procedimiento desconocido en el momento del enlazado o en tiempo de ejecución (enlazamiento en tiempo de ejecución).   |
| ERR_REFUNKTRP           | Referencia a rutina TRAP desconocida.   |
| ERR_RMQ_DIM             | Dimensiones incorrectas: las dimensiones de los datos indicados no son iguales a las dimensiones de los datos del mensaje.  |
| ERR_RMQ_FULL            | Cola de mensajes de destino llena.  |
| ERR_RMQ_INVALID         | Ranura de destino perdida o no válida.  |
| ERR_RMQ_INVMSG          | Mensaje no válido, probablemente enviado desde un cliente distinto de una tarea de RAPID.   |
| ERR_RMQ_MSGSIZE         | Tamaño de mensaje excesivo. Reduzca el tamaño del mensaje.  |
| ERR_RMQ_NAME            | El nombre de ranura indicado no es válido o no se encuentra.  |
| ERR_RMQ_NOMSG           | No hay ningún mensaje en la cola, probablemente como resultado de una caída de alimentación.  |
| ERR_RMQ_TIMEOUT         | Se ha alcanzado el tiempo límite esperando una respuesta en RMQSendWait o RMQReadWait.  |
| ERR_RMQ_VALUE           | La sintaxis del valor no coincide con el tipo de dato.  |
| ERR_ROBLIMIT            | La posición puede alcanzarse pero al menos uno de los ejes está fuera de los límites de los ejes o se superan los límites en al menos un eje acoplado (función CalcJoint).                            |
| ERR_SC_WRITE            | Error de envío a un PC externo.   |
| ERR_SGUN_ESTOP          | Parada de emergencia durante movimiento de herramienta servo.   |
| ERR_SGUN_MOTOFF         | La instrucción se ejecuta desde una tarea en segundo plano y el sistema se encuentra en estado de motores apagados.   |
| ERR_SGUN_NEGVAL         | El argumento PrePos se especifica con un valor menor que cero.  |
| ERR_SGUN_NOTACT         | La unidad mecánica de la herramienta servo no está activada.  |
| ERR_SGUN_NOTINIT        | La posición de la herramienta servo no está inicializada.   |
| ERR_SGUN_NOTOPEN        | La pistola no está abierta cuando se ejecuta la instrucción.  |
| ERR_SGUN_NOTSYNC        | Las puntas de la herramienta servo no están sincronizadas.  |
| ERR_SIGSUPSEARCH        | La señal ya tiene un valor positivo al comienzo del proceso de búsqueda.  |
| ERR_SIG_NOT_VALID       | La señal de E/S no está disponible. Los motivos pueden ser que el dispositivo de E/S no está en funcionamiento o que exista un error en la configuración (válido solamente para el bus de campo ICI). |
| ERR_SOCKET_ADDR_INVALID | La dirección especificada no es válida.   |

Continúa en la página siguiente

### 3 Tipos de datos

#### 3.27 errnum - Número de error

RobotWare - OS

Continuación

| Nombre                 | Causa del error  |
|------------------------|--|
| ERR_SOCKET_ADDR_INUSE  | La dirección y el puerto ya se están utilizando y no pueden utilizarse nuevamente. Utilice otro número de puerto u otra dirección en <code>SocketBind</code> .   |
| ERR_SOCKET_CLOSED      | El zócalo está cerrado o no está creado.   |
| ERR_SOCKET_IS_BOUND    | El zócalo ya ha sido vinculado a una dirección y no puede volver a vincularse.   |
| ERR_SOCKET_IS_CONN     | El zócalo está conectado.  |
| ERR_SOCKET_NET_UNREACH | No se puede acceder a la red o la conexión se pierde después de abrirse un zócalo.   |
| ERR_SOCKET_NOT_BOUND   | El zócalo no ha sido vinculado a una dirección.  |
| ERR_SOCKET_NOT_CONN    | El zócalo no está conectado  |
| ERR_SOCKET_TIMEOUT     | La conexión no se estableció dentro del tiempo límite o no se recibieron datos dentro del tiempo límite.   |
| ERR_SOCKET_UNSPEC      | Excepción no especificada de la llamada subyacente al sistema operativo.   |
| ERR_SPEED_REFRESH_LIM  | Redefinición fuera de límites en <code>SpeedRefresh</code> .   |
| ERR_SPEEDLIM_VALUE     | La velocidad utilizada en las instrucciones <code>SpeedLimAxis</code> y <code>SpeedLimCheckPoint</code> es demasiado baja.   |
| ERR_STARTMOVE          | El robot ya está en el estado en espera cuando se ejecuta una instrucción <code>StartMove</code> , <code>StartMoveRetry</code> o <code>SetLeadThrough</code> .   |
| ERR_STORE_PROF         | Error en el perfil almacenado.   |
| ERR_STRTOOLNG          | La cadena es demasiado larga.  |
| ERR_SYM_ACCESS         | Error de acceso de lectura o escritura del símbolo.  |
| ERR_SYMBOL_TYPE        | El objeto de datos y la variable utilizados en el argumento <code>Value</code> tienen tipos diferentes. Si se utilizan tipos de datos ALIAS, también se produce este ERROR, aunque los tipos tengan el mismo tipo de dato básico. Instrucciones <code>GetDataVal</code> , <code>SetDataVal</code> y <code>SetAllDataVal</code> . |
| ERR_SYNCMOVEOFF        | Tiempo límite en <code>SyncMoveOff</code> .  |
| ERR_SYNCMOVEON         | Tiempo límite en <code>SyncMoveOn</code> .   |
| ERR_SYNTAX             | Error de sintaxis en el módulo cargado.  |
| ERR_TASKNAME           | El nombre de tarea no se encuentra en el sistema.  |
| ERR_TP_DIBREAK         | Una instrucción de lectura de <code>FlexPendant</code> fue interrumpida por una entrada digital.   |
| ERR_TP_DOBREAK         | Una instrucción de lectura de <code>FlexPendant</code> fue interrumpida por una salida digital.  |
| ERR_TP_MAXTIME         | Tiempo límite al ejecutar una instrucción de lectura desde <code>FlexPendant</code> .  |
| ERR_TP_NO_CLIENT       | No hay ningún cliente con el que interactuar al utilizar una instrucción de lectura desde <code>FlexPendant</code> .   |
| ERR_TRUSTLEVEL         | No se permite deshabilitar el dispositivo de E/S.  |
| ERR_TXTNOEXIST         | Tabla o índice incorrecto en la función <code>TextGet</code> .   |
| ERR_UDPUC_COMM         | Tiempo límite de comunicación para el dispositivo <code>UdpUc</code> .   |

Continúa en la página siguiente

| Nombre           | Causa del error   |
|------------------|---|
| ERR_UI_INITVALUE | Error de valor inicial en la función UINumEntry.  |
| ERR_UI_MAXMIN    | El valor mínimo es mayor que el valor máximo en la función UINumEntry, UIDnumEntry, UINumTune o UIDnumTune.                 |
| ERR_UI_NOTINT    | El valor no es un entero cuando se especificó que debe utilizarse un entero al utilizar UINumEntry o UIDnumEntry.           |
| ERR_UISHOW_FATAL | Otro error distinto de ERR_TP_NO_CLIENT o ERR_UISHOW_FULL en la instrucción UIShow.   |
| ERR_UISHOW_FULL  | No queda espacio en FlexPendant para otra aplicación al utilizar la instrucción UIShow.                                     |
| ERR_UNIT_PAR     | El parámetro Mech_unit en TestSignDefine es incorrecto.   |
| ERR_UNKINO       | Número de interrupción desconocido al ejecutar las instrucciones IWatch o ISleep.   |
| ERR_UNKPROC      | Referencia incorrecta a la sesión de carga de una instrucción WaitLoad.   |
| ERR_UNLOAD       | Error de descarga en la instrucción UnLoad o WaitLoad.  |
| ERR_USE_PROF     | Error en el perfil usado.   |
| ERR_WAITSYNCTASK | Tiempo límite en WaitSyncTask.  |
| ERR_WAIT_MAXTIME | Tiempo límite al ejecutar una instrucción WaitDI, WaitDO, WaitAI, WaitAO, WaitGI, WaitGO, WaitUntil, WaitSensor o WaitWObj. |
| ERR_WHLSEARCH    | No hay ningún paro de búsqueda.   |
| ERR_WOBJ_MOVING  | La unidad mecánica que contiene el objeto de trabajo está moviendo CalcJointT.  |

### Características

errnum es un tipo de dato de alias de num y por tanto hereda sus características.

### Información relacionada

| Para obtener más información sobre                 | Consulte                                      |
|--|---|
| Recuperación en caso de error                      | Manual de referencia técnica - RAPID Overview |
| Tipos de datos en general, tipos de datos de alias | Manual de referencia técnica - RAPID Overview |

## 3 Tipos de datos

---

### 3.28 errstr - Cadena de error

RobotWare - OS

### 3.28 errstr - Cadena de error

---

#### Utilización

`errstr` se utiliza para escribir un texto en mensajes de error.

---

#### Ejemplos básicos

El siguiente ejemplo ilustra el tipo de dato `errstr`:

##### Ejemplo 1

```
VAR errstr arg:= "This is an example";

ErrLog 5100, \W, ERRSTR_TASK, ERRSTR_CONTEXT, arg, ERRSTR_EMPTY,
ERRSTR_UNUSED;
```

---

#### Datos predefinidos

| Nombre         | Descripción                 |
|----------------|-----------------------------|
| ERRSTR_EMPTY   | El argumento está vacío     |
| ERRSTR_UNUSED  | El argumento no se utiliza. |
| ERRSTR_TASK    | Nombre de la tarea actual   |
| ERRSTR_CONTEXT | Contexto                    |

---

#### Características

`errstr` es un tipo de dato de alias de `string` y por tanto hereda sus características.

---

#### Información relacionada

| Para obtener más información sobre                 | Consulte   |
|--|--|
| Tipos de datos en general, tipos de datos de alias | <i>Manual de referencia técnica - RAPID Overview</i> , sección <i>Características básicas - Tipos de datos</i> |

---

## 3.29 errtype - Tipo de error

### Utilización

`errtype` (*error type*) se utiliza para especificar un tipo de error.

### Descripción

Los datos de tipo `errtype` representan el tipo (cambio de estado, advertencia, error) de un mensaje de error.

### Ejemplos básicos

El siguiente ejemplo ilustra el tipo de dato `errtype`:

#### Ejemplo 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
```

Cuando se detecta un error con la rutina `TRAP trap_err`, el dominio, el número y el tipo del error se almacenan en las variables adecuadas.

### Datos predefinidos

Puede usar las constantes predefinidas siguientes para especificar un tipo de error.

| Nombre     | Tipo de error  | Valor |
|------------|--|-------|
| TYPE_ALL   | Cualquier tipo de error (cambio de estado, advertencia, error) | 0     |
| TYPE_STATE | Cambio de estado (mensaje operativo)                           | 1     |
| TYPE_WARN  | Advertencia (por ejemplo un error recuperable de RAPID)        | 2     |
| TYPE_ERR   | Error  | 3     |

### Características

`errtype` es un tipo de dato de alias de `num` y por tanto hereda sus características.

### Información relacionada

| Para obtener más información sobre         | Consulte   |
|--|--|
| Solicitud de una interrupción para errores | <a href="#">IError - Solicita una interrupción para errores en la página 257</a> |
| Números de errores                         | <i>Manual del operador - Solución de problemas de IRC5</i>                       |
| Tipos de datos de alias                    | <i>Manual de referencia técnica - RAPID Overview</i>                             |

*Continúa en la página siguiente*

### 3 Tipos de datos

---

3.29 errtype - Tipo de error

RobotWare - OS

Continuación

| Para obtener más información sobre | Consulte  |
|------------------------------------|---|
| <i>Advanced RAPID</i>              | <i>Especificaciones del producto - Controller software IRC5</i> |

### 3.30 event\_type - Tipo de rutina de evento

#### Utilización

`event_type` se utiliza para representar el tipo de la rutina de evento actual con una constante simbólica

#### Descripción

Con la función `EventType`, es posible comprobar si el código de RAPID real se ejecuta o no a causa de algún evento de sistema específico.

#### Ejemplos básicos

El siguiente ejemplo ilustra el tipo de dato `event_type`:

##### Ejemplo 1

```
VAR event_type my_type;
...
my_type := EventType( );
```

El tipo de rutina de evento que se ejecuta se almacenará en la variable `my_type`.

#### Datos predefinidos

Las siguientes constantes de tipo `event_type` están predefinidas:

| Constante de RAPID | Valor | Tipo de evento ejecutado    |
|--------------------|-------|-----------------------------|
| EVENT_NONE         | 0     | No se ejecuta ningún evento |
| EVENT_POWERON      | 1     | Evento POWER_ON             |
| EVENT_START        | 2     | Evento START                |
| EVENT_STOP         | 3     | Evento STOP                 |
| EVENT_QSTOP        | 4     | Evento QSTOP                |
| EVENT_RESTART      | 5     | Evento RESTART              |
| EVENT_RESET        | 6     | Evento RESET                |
| EVENT_STEP         | 7     | Evento STEP                 |

#### Características

`event_type` es un tipo de dato de alias de `num` y por tanto hereda sus características.

#### Información relacionada

| Para obtener más información sobre                 | Consulte   |
|--|--|
| Rutinas de evento en general                       | <i>Manual de referencia técnica - Parámetros del sistema, sección Controller - Event Routine</i>                           |
| Obtener tipo de evento                             | <a href="#">EventType</a> - <i>Obtiene el tipo de evento actual dentro de cualquier rutina de evento en la página 1280</i> |
| Tipos de datos en general, tipos de datos de alias | <i>Manual de referencia técnica - RAPID Overview, sección Características básicas - Tipos de datos</i>                     |

## 3 Tipos de datos

---

### 3.31 exec\_level - Nivel de ejecución

RobotWare - OS

### 3.31 exec\_level - Nivel de ejecución

---

#### Utilización

`exec_level` se utiliza para especificar el nivel de ejecución del programa.

---

#### Descripción

La función `ExecLevel` permite obtener el nivel de ejecución actual del código de `RAPID` que se está ejecutando en ese momento.

---

#### Datos predefinidos

Se han predefinido las constantes siguientes del tipo `exec_level`:

| Constante de RAPID         | Valor | Nivel de ejecución                             |
|----------------------------|-------|--|
| <code>LEVEL_NORMAL</code>  | 0     | Ejecutar en el nivel básico                    |
| <code>LEVEL_TRAP</code>    | 1     | Ejecutar en la rutina TRAP                     |
| <code>LEVEL_SERVICE</code> | 2     | Ejecutar en la rutina de servicio <sup>i</sup> |

<sup>i</sup> Con `LEVEL_SERVICE`, significa una rutina de evento, una rutina de servicio (incluidas las llamadas a rutinas) y una rutina de interrupción a partir de una señal de entrada del sistema.

---

#### Características

`exec_level` es un tipo de dato de alias de `num` y por tanto hereda sus características.

---

#### Información relacionada

| Para obtener más información sobre   | Consulte  |
|--------------------------------------|---|
| Obtener el nivel de ejecución actual | <a href="#">ExecLevel - Obtener el nivel de ejecución en la página 1283</a> |



---

### 3.32 extjoint - Posición de los ejes externos

---

#### Utilización

`extjoint` se utiliza para definir posiciones de eje de los ejes adicionales, posicionadores o manipuladores de piezas de trabajo.

---

#### Descripción

El robot puede controlar hasta seis ejes adicionales además de los seis ejes internos, es decir, un total de doce ejes. Los seis ejes adicionales tienen una denominación de tipo lógico: a, b, c, d, e, f. Estos ejes lógicos pueden conectarse a un eje físico y, en este caso, la conexión se define en los parámetros de sistema. Se utilizan datos de tipo `extjoint` para almacenar los valores de las posiciones de cada uno de los ejes lógicos del a al f.

Para cada eje lógico conectado a un eje físico, la posición se define de la forma siguiente:

- Para los ejes de rotación, la posición se define como la rotación en grados de la posición de calibración.
- Para los ejes lineales: la posición se define como la distancia en mm existente respecto de la posición de calibración.

Si un eje lógico no está conectado a uno físico, se utiliza el valor 9E9 como valor de posición, para indicar que el eje no está conectado. En el momento de la ejecución, se comprueban los datos de posición de los distintos ejes y se comprueba si está conectado el eje correspondiente. Si el valor de posición almacenado no cumple con la conexión actual del eje, se aplica lo siguiente:

- Si la posición no está definida en los datos de posición (su valor es 9E9), el valor no se tendrá en cuenta si el eje está conectado pero no activado. Sin embargo, si el eje está activado, se genera un error.
- Si la posición está definida en los datos de posición, a pesar de que el eje no está conectado, el valor no se tendrá en cuenta.

No se realiza ningún movimiento pero tampoco se genera ningún error en el caso de los ejes que tengan datos de posición válidos pero que no estén activados.

Si se utiliza un offset del eje externo (instrucción `EOffsOn` o `EOffsSet`), las posiciones se especifican en el sistema de coordenadas `ExtOffs`.

Si algún eje adicional está funcionando en el modo independiente y el robot y sus ejes adicionales deben realizar algún nuevo movimiento, los datos de posición de los ejes adicionales en el modo independiente no deben ser 9E9. Los datos deben ser un valor arbitrario que no esté siendo utilizado por el sistema.

---

#### Componentes

`eax_a`

*external axis a*

Tipo de dato: `num`

La posición del eje externo lógico "a", expresada en grados o mm (en función del tipo de eje).

*Continúa en la página siguiente*

## 3 Tipos de datos

---

### 3.32 extjoint - Posición de los ejes externos

RobotWare - OS

Continuación

...

eax\_f

*external axis f*

Tipo de dato: num

La posición del eje externo lógico "f", expresada en grados o mm (en función del tipo de eje).

---

### Ejemplos básicos

El siguiente ejemplo ilustra el tipo de dato `extjoint`:

#### Ejemplo 1

```
VAR extjoint axpos10 := [ 11, 12.3, 9E9, 9E9, 9E9, 9E9 ] ;
```

La posición de un posicionador externo `axpos10`, se define de la forma siguiente:

- Se cambia a 11 la posición del eje externo lógico "a", expresada en grados o mm (en función del tipo de eje).
  - Se cambia a 12.3 la posición del eje externo lógico "b", expresada en grados o mm (en función del tipo de eje).
  - Los ejes de c a f permanecen sin definir.
- 

### Estructura

```
< 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 >
```

---

### Información relacionada

| Para obtener más información sobre | Consulte  |
|------------------------------------|---|
| Datos de posición                  | <a href="#">robtargt - Datos de posición en la página 1750</a><br><a href="#">jointtargt - Datos de posición de eje en la página 1691</a> |
| Sistema de coordenadas ExtOffs     | <a href="#">EOffsOn - Activa un offset de ejes adicionales en la página 195</a>   |

### 3.33 flypointdata - Datos para inicio/fin sobre la marcha

#### Utilización

`flypointdata` se utiliza para definir todos los datos de inicio o el fin de paso para un proceso CAP. Es parte de `capdata` tanto para el inicio como el fin sobre la marcha.

#### Definiciones

`flypointdata` define datos para inicio y fin sobre la marcha:

- Esta funcionalidad solo está disponible para CAP.
- El inicio sobre la marcha se dispara mediante la combinación de *first instruction = TRUE* y el punto de zona.
- El fin sobre la marcha se dispara mediante la combinación de *last\_instr = TRUE* y el punto de zona.
- El inicio de oscilación no se tiene en cuenta.
- Si el punto sobre la marcha es un punto fino, no se realiza el inicio sobre la marcha.
- Si el punto de final es un punto fino, no se realiza el fin sobre la marcha.
- El retardo de movimiento no se tiene en cuenta.
- El reinicio tras un error funciona de la forma habitual: no hay características especiales para el inicio sobre la marcha, el inicio del limpiado está disponible, si el proceso de aplicación estaba activo cuando se produjo el error.
- Si la oscilación está activada, la transición en la zona se realiza en rampa en el patrón de oscilación empezando en la entrada a la zona hasta que se alcanza el patrón completo cuando el TCP abandona la zona.
- La supervisión está activa durante la fase START (con movimiento del TCP), la fase MAIN y la fase END\_MAIN (con movimiento del TCP).
- El retroceso en la trayectoria está limitado al retroceso a la posición 4 (consulte la figura que aparece a continuación).
- El usuario tiene que adaptar la distancia y el ángulo de aproximación y salida para el proceso de la aplicación: por ejemplo, la soldadura al arco en el punto donde se establecerá el arco (punto 4 de la figura) tiene que seleccionarse de una manera que permita la ignición.
- La distancia entre la posición 4 y 6 no debe ser = 0.
- `START process_dist` debe ser igual o menor que `START distance`.
- Si se detiene la ejecución del programa y el proceso de la aplicación está activo (entre las posiciones 3 y 6), CAP se comporta de la forma habitual, es decir, retrocediendo en la trayectoria (solo si se ha pasado la pos. 4), el inicio de oscilación, el retardo de movimiento y el tiempo límite de inicio del movimiento están disponibles.
- Si la ejecución del programa se detiene entre las posiciones 1 y 3 o entre las posiciones 7 y 10, la instrucción `CapX` se comporta como una instrucción `TrigX`.

*Continúa en la página siguiente*

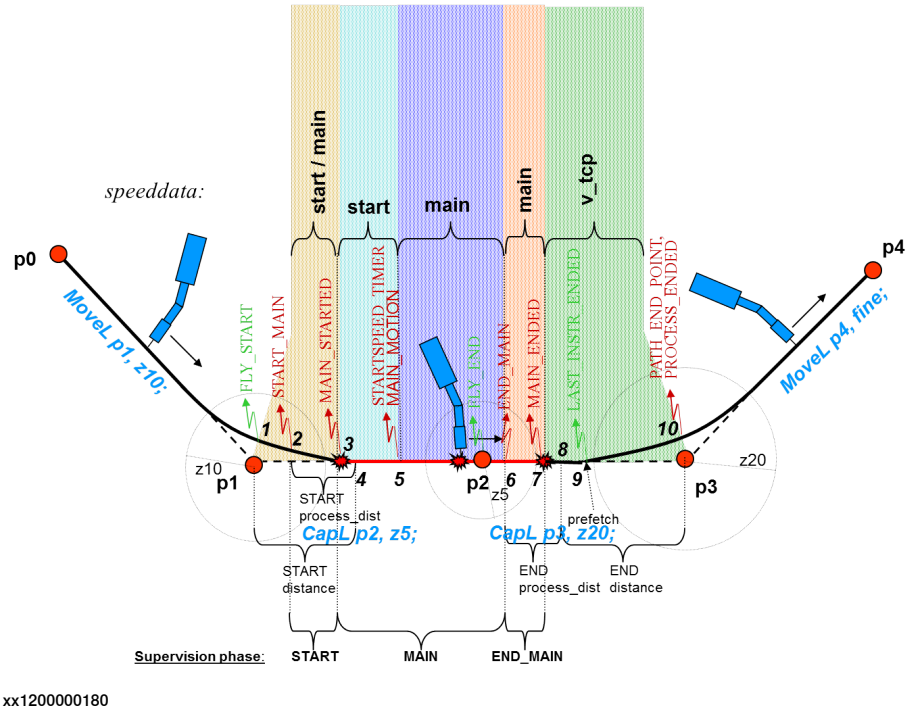
### 3 Tipos de datos

#### 3.33 flypointdata - Datos para inicio/fin sobre la marcha

##### Continuous Application Platform (CAP)

##### Continuación

- Se recomienda que el primer segmento de CAP con inicio sobre la marcha tenga al menos la longitud de `START distance`.
- Si el primer segmento es más corto que `START distance`, pero más largo que `START process_dist`, las posiciones 2 y 4 se mueve hacia la posición 1.
- Si el primer segmento es más corto o igual que `START process_dist`, las posiciones 1 y 2 coincidirán y la posición 4 está al final del segmento.
- Se recomienda que el último segmento de CAP con final sobre la marcha tenga al menos una longitud de `END distance + END process_dist`.
- Si el último segmento es más corto que `END distance + END process_dist`, pero más largo que `END process_dist`, las posiciones 7 y 9 se mueven hacia la posición 10.
- Si el último segmento es más corto o igual que `END process_dist`, las posiciones 8 y 10 coinciden y la posición 6 está al inicio del segmento.
- El tiempo límite de la fase `START` especificado en `capdata` solo se utiliza en el reinicio del proceso de aplicación.
- Si se produce un error tras la solicitud de precaptura desde que el movimiento ha llegado a la última instrucción de CAP (después de la posición 9), es decir, PGM se libera de la instrucción de CAP y puede continuar con la siguiente instrucción, se envía un mensaje de registro de error y el proceso se detiene, pero el movimiento del robot continúa.



#### Componentes

from\_start

Tipo de dato: bool

Continúa en la página siguiente

#### 3.33 flypointdata - Datos para inicio/fin sobre la marcha Continuous Application Platform (CAP) Continuación

No se usa.

process\_dist

Tipo de dato: num

La distancia (en mm) dentro de la cual se inicia el proceso (para *inicio sobre la marcha*) o finaliza (para *fin sobre la marcha*).

distance

Tipo de dato: num

Define el inicio/final de la supervisión del proceso CAP como una distancia (en mm) desde el punto de inicio/final.

#### Estructura

```
< databases of flypointdata >  
  < from_start of bool >  
  < process_dist of num >  
  < distance of num >
```

#### Información relacionada

|  | Descrito en:  |
|--|---|
| Tipo de dato capdata                   | <a href="#">capdata - Datos CAP en la página 1624</a>       |
| <i>Continuous Application Platform</i> | <i>Application manual - Continuous Application Platform</i> |

## 3 Tipos de datos

---

### 3.34 handler\_type - Tipo de gestor de ejecución

RobotWare - OS

### 3.34 handler\_type - Tipo de gestor de ejecución

---

#### Utilización

`handler_type` se utiliza para especificar el tipo de gestor de ejecución de la rutina de programa de RAPID.

---

#### Descripción

Con la función `ExecHandler`, es posible comprobar si el código de RAPID en sí se ejecuta en algún gestor de ejecución de la rutina de programa de RAPID.

---

#### Ejemplos básicos

El siguiente ejemplo ilustra el tipo de dato `handler_type`:

##### Ejemplo 1

```
VAR handler_type my_type;  
...  
my_type := ExecHandler( );
```

El tipo de gestor de ejecución en el que se encuentra el código se almacena en la variable `my_type`.

---

#### Datos predefinidos

Las siguientes constantes de tipo `handler_type` están predefinidas:

| Constante de RAPID | Valor | Tipo de gestor de ejecución      |
|--------------------|-------|----------------------------------|
| HANDLER_NONE       | 0     | No se ejecuta en ningún gestor   |
| HANDLER_BWD        | 1     | Se ejecuta en un gestor BACKWARD |
| HANDLER_ERR        | 2     | Se ejecuta en un gestor ERROR    |
| HANDLER_UNDO       | 3     | Se ejecuta en un gestor UNDO     |

---

#### Características

`handler_type` es un tipo de dato de alias de `num` y por tanto hereda sus características.

---

#### Información relacionada

| Para obtener más información sobre     | Consulte   |
|--|--|
| Obtener el tipo de gestor de ejecución | <a href="#">ExecHandler - Obtener el tipo de gestor de ejecución en la página 1282</a> |

---

### 3.35 icondata - Datos de visualización de iconos

#### Utilización

`icondata` se usa para representar los iconos estándar del dispositivo de usuario, por ejemplo del FlexPendant.

#### Descripción

Una constante de enumeración `icondata` puede ser entregada al argumento `Icon` de las instrucciones `UIMsgBox`, `UIMsgWrite`, `WaitAI`, `WaitAO`, `WaitDI`, `WaitDO`, `WaitGI`, `WaitGO` y `WaitUntil` y las funciones `UIMessageBox`, `UINumEntry`, `UIDnumEntry`, `UINumTune`, `UIDnumTune`, `UIAlphaEntry` y `UIListView`.

#### Ejemplos básicos

El siguiente ejemplo ilustra el tipo de dato `icondata`:

##### Ejemplo 1

```
VAR btnres answer;

UIMsgBox "More ?" \Buttons:=btnYesNo \Icon:=iconInfo \Result:=
  answer;
IF answer= resYes THEN
...
ELSEIF answer =ResNo THEN
...
ENDIF
```

La constante de enumeración de botón estándar `iconInfo` mostrará un icono de información en el encabezado del cuadro de mensaje de la interfaz de usuario.

#### Datos predefinidos

Se han predefinido en el sistema las constantes siguientes del tipo de dato `icondata`.

| Valor | Constante                 | Icono                |
|-------|---------------------------|----------------------|
| 0     | <code>iconNone</code>     | Ningún icono         |
| 1     | <code>iconInfo</code>     | Icono de información |
| 2     | <code>iconWarning</code>  | Icono de aviso       |
| 3     | <code>iconError</code>    | Icono de error       |
| 4     | <code>iconQuestion</code> | Icono de pregunta    |

#### Características

`icondata` es un tipo de dato de alias de `num` y por tanto hereda sus características.

#### Información relacionada

| Para obtener más información sobre              | Consulte   |
|---|--|
| Cuadro de mensaje de interacción con el usuario | <a href="#">UIMsgBox - Cuadro de mensaje de usuario de tipo básico en la página 1002</a> |

*Continúa en la página siguiente*

### 3 Tipos de datos

---

#### 3.35 icondata - Datos de visualización de iconos

RobotWare - OS

Continuación

| Para obtener más información sobre                      | Consulte   |
|---|--|
| Cuadro de mensaje de interacción con el usuario         | <a href="#">UIMessageBox - Cuadro de mensaje de usuario de tipo avanzado en la página 1577</a>         |
| Introducción de número de interacción con el usuario    | <a href="#">UINumEntry - Introducción de número de usuario en la página 1585</a>                       |
| Ajuste de número de interacción con el usuario          | <a href="#">UINumTune - Ajuste de número de usuario en la página 1592</a>                              |
| Introducción alfanumérica de interacción con el usuario | <a href="#">UIAlphaEntry - Introducción alfanumérica del usuario en la página 1545</a>                 |
| Vista de lista de interacción con el usuario            | <a href="#">UIListView - Vista de lista de usuario en la página 1568</a>                               |
| Tipos de datos en general, tipos de datos de alias      | <i>Manual de referencia técnica - RAPID Overview, sección Características básicas - Tipos de datos</i> |



### 3.36 identno - Identidad para las instrucciones de movimiento

#### Utilización

`identno` (*Identity Number*) se utiliza para controlar la sincronización de dos o más movimientos sincronizados y coordinados entre sí.

El tipo de dato `identno` sólo puede usarse en un sistema *MultiMove* que tenga la opción *Coordinated Robots* y sólo en las tareas de programa definidas como tareas de movimiento.

#### Descripción

Las instrucciones de movimiento de un sistema *MultiMove* deben programarse con el parámetro `\ID` del tipo de dato `identno` si se trata de un movimiento sincronizado y además `\ID` no se permite en ningún otro caso.

El número de `\ID` especificado debe ser el mismo en todas las tareas de programa que cooperan entre sí. El número de ID constituye una garantía de que los movimientos no se mezclen en tiempo de ejecución.

En el modo sincronizado coordinado, debe existir la misma cantidad de instrucciones de movimiento ejecutadas en todas las tareas de programa. El parámetro opcional `\ID` del tipo de dato `identno` se utiliza para comprobar que las instrucciones de movimiento asociadas se ejecuten en paralelo antes del inicio de los movimientos. El número de `\ID` debe ser el mismo en todas las instrucciones de movimiento que se ejecuten en paralelo.

El usuario no tiene que declarar ninguna variable del tipo `identno`, pero puede usar directamente un número en las instrucciones (consulte *Ejemplos básicos*).

#### Ejemplos básicos

El siguiente ejemplo ilustra el tipo de dato `identno`:

##### Ejemplo 1

```

PERS tasks task_list{2} := [{"T_ROB1"}, {"T_ROB2"}];
VAR syncident sync1;
VAR syncident sync2;

PROC procl()
...
SyncMoveOn sync1, task_list;
MoveL *\ID:=10,v100,z50,mytool;
MoveL *\ID:=20,v100,fine,mytool;
SyncMoveOff sync2;
...
ENDPROC

```

#### Características

`identno` es un tipo de dato de alias de `num` y por tanto hereda sus propiedades.

*Continúa en la página siguiente*

### 3 Tipos de datos

---

#### 3.36 identno - Identidad para las instrucciones de movimiento

##### *MultiMove - Coordinated Robots*

##### *Continuación*

---

#### Información relacionada

| Para obtener más información sobre              | Consulte   |
|---|--|
| Tipos de datos de alias                         | <i>Manual de referencia técnica - RAPID Overview, sección Características básicas - Tipos de datos</i>   |
| Inicio de movimientos sincronizados coordinados | <a href="#"><i>SyncMoveOn - Inicia los movimientos sincronizados coordinados en la página 854</i></a>    |
| Fin de movimientos sincronizados coordinados    | <a href="#"><i>SyncMoveOff - Finaliza los movimientos sincronizados coordinados en la página 848</i></a> |

### 3.37 intnum - Identidad de interrupción

#### Utilización

`intnum` (*interrupt numeric*) se usa para identificar una interrupción.

#### Descripción

Cuando se conecta una variable de tipo `intnum` a una rutina TRAP, recibe un valor específico que identifica a la interrupción. A continuación, la variable se utiliza en todas las operaciones realizadas con la interrupción, por ejemplo, al pedir o desactivar una interrupción.

Es posible conectar más de una identidad de interrupción a una misma rutina TRAP. Por tanto, la variable de sistema `INTNO` puede usarse en las rutinas TRAP para determinar el tipo de interrupción que tiene lugar.

Las variables del tipo `intnum` deben declararse siempre como globales en el módulo.

#### Ejemplos básicos

Los siguientes ejemplos ilustran el tipo de dato `intnum`:

##### Ejemplo 1

```
VAR intnum feeder_error;
...
PROC main()
  CONNECT feeder_error WITH correct_feeder;
  ISignalDI di1, 1, feeder_error;
```

Se genera una interrupción cuando la entrada `di1` cambia de valor a 1. Cuando esto ocurre, se hace una llamada a la rutina TRAP `correct_feeder`.

##### Ejemplo 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
```

*Continúa en la página siguiente*

## 3 Tipos de datos

---

### 3.37 intnum - Identidad de interrupción

RobotWare - OS

Continuación

Se genera una interrupción cuando una de las entradas, di1 o di2, cambia de valor a 1. En este caso, se realiza una llamada a la rutina TRAP `correct_feeder`. La variable de sistema `INTNO` se utiliza en la rutina TRAP para determinar qué tipo de interrupción se ha producido.

---

#### Limitaciones

El número máximo de variables activas del tipo `intnum` que pueden existir en un momento determinado (entre `CONNECT` y `IDelete`) está limitado a 100. El número máximo de interrupciones que puede albergar la cola de ejecución de rutinas TRAP en un momento determinado está limitado a 30.

---

#### Características

`Intnum` es un tipo de dato de alias de `num` y por tanto hereda sus propiedades.

---

#### Información relacionada

| Para obtener más información sobre | Consulte   |
|------------------------------------|--|
| Resumen de interrupciones          | <i>Manual de referencia técnica - RAPID Overview, sección Resumen sobre RAPID - Interrupciones</i>     |
| Tipos de datos de alias            | <i>Manual de referencia técnica - RAPID Overview, sección Características básicas - Tipos de datos</i> |
| Conexión de interrupciones         | <a href="#">CONNECT - Conecta una interrupción a una rutina TRAP en la página 161</a>                  |

### 3.38 iodev - Dispositivo de E/S

#### Utilización

`iodev` (*dispositivo de E/S*) se utiliza para dispositivos de E/S y archivos.

#### Descripción

Los datos de tipo `iodev` contienen una referencia a un archivo o dispositivos de E/S. Pueden conectarse a la unidad física mediante la instrucción `Open` y utilizarse a continuación para operaciones de lectura y escritura.

#### Ejemplos básicos

El siguiente ejemplo ilustra el tipo de dato `iodev`:

##### Ejemplo 1

```
VAR iodev file;
...
Open "HOME:/LOGDIR/INFILE.DOC", file\Read;
input := ReadNum(file);
```

Se abre el archivo `INFILE.DOC` para lectura. Al leer del archivo, se utiliza `file` como referencia en lugar del nombre del archivo.

#### Características

`iodev` es un tipo de dato sin valor.

#### Información relacionada

| Para obtener más información sobre              | Consulte   |
|---|--|
| Comunicación a través de dispositivos de E/S    | <i>Manual de referencia técnica - RAPID Overview, sección Resumen sobre RAPID - Comunicación</i>       |
| Configuración de dispositivos de E/S            | <i>Manual de referencia técnica - Parámetros del sistema</i>   |
| Características de los tipos de datos sin valor | <i>Manual de referencia técnica - RAPID Overview, sección Características básicas - Tipos de datos</i> |
| Gestión de archivos y dispositivos de E/S       | <i>Application manual - Controller software IRC5</i>   |

## 3 Tipos de datos

### 3.39 iounit\_state - Estado del dispositivo de E/S

RobotWare - OS

### 3.39 iounit\_state - Estado del dispositivo de E/S

#### Utilización

`iounit_state` se utiliza para representar el estado momentáneo de un dispositivo de E/S.

#### Descripción

Las constantes de `iounit_state` se han diseñado para usarlas al comprobar el valor de retorno de la función `IOUnitState`.

#### Ejemplos básicos

El siguiente ejemplo ilustra el tipo de dato `iounit_state`:

#### Ejemplo 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
```

Se comprueba si el dispositivo de E/S `UNIT1` funciona correctamente.

#### Datos predefinidos

Las constantes simbólicas predefinidas del tipo de dato `iounit_state` se encuentran en la función `IOUnitState`.

#### Características

`iounit_state` es un tipo de dato de alias de `num` y por tanto hereda sus características.

#### Información relacionada

| Para obtener más información sobre                   | Consulte   |
|--|--|
| Obtención del estado actual de un dispositivo de E/S | <a href="#">IOUnitState - Obtener el estado actual de un dispositivo de E/S en la página 1342</a>                |
| Instrucciones de entrada/salida                      | <i>Manual de referencia técnica - RAPID Overview, sección Resumen sobre RAPID - Señales de entrada y salida</i>  |
| Funcionalidad de entrada/salida en general           | <i>Manual de referencia técnica - RAPID Overview, sección Principios de movimiento y E/S - Principios de E/S</i> |
| Configuración de E/S                                 | <i>Manual de referencia técnica - Parámetros del sistema</i>   |

## 3.40 jointtarget - Datos de posición de eje

### Utilización

`jointtarget` se utiliza para definir la posición a la que se moverán los ejes del robot y los ejes externos al ejecutar una instrucción `MoveAbsJ`.

### Descripción

`jointtarget` define las posiciones individuales de los distintos ejes, tanto de los del robot como de los externos.

### Componentes

`robax`

*robot axes*

Tipo de dato: `robjoint`

Posiciones de eje de los ejes del robot, en grados.

La posición del eje se define como la rotación en grados del eje (brazo) correspondiente en sentido positivo o negativo a partir de la posición de calibración del eje.

`extax`

*external axes*

Tipo de dato: `extjoint`

La posición de los ejes externos.

La posición se define de la forma siguiente para cada eje independiente (`eax_a`, `eax_b` ... `eax_f`):

- Para los ejes de rotación, la posición se define como la rotación en grados de la posición de calibración.
- Para los ejes lineales, la posición se define como la distancia en mm existente respecto de la posición de calibración.

Los ejes externos `eax_a` ... son ejes lógicos. La relación existente entre el número lógico del eje y el número físico del eje se define en los parámetros del sistema.

El valor 9E9 se utiliza para definir los ejes que no están conectados. Si los ejes definidos en los datos de posición son distintos de los ejes que están conectados realmente en el momento de la ejecución del programa, ocurre lo siguiente:

- Si la posición no está definida en los datos de posición (su valor es 9E9), el valor no se tendrá en cuenta si el eje está conectado pero no activado. Sin embargo, si el eje está activado, se genera un error.
- Si la posición está definida en los datos de posición, a pesar de que el eje no está conectado, el valor no se tiene en cuenta.

No se realiza ningún movimiento pero tampoco se genera ningún error en el caso de los ejes que tengan datos de posición válidos pero que no estén activados.

Si algún eje externo está funcionando en el modo independiente y el robot y sus ejes externos deben realizar algún nuevo movimiento, los datos de posición del

*Continúa en la página siguiente*

## 3 Tipos de datos

### 3.40 jointtarget - Datos de posición de eje

RobotWare - OS

Continuación

eje externo en el modo independiente no deben ser 9E9 sino algún valor arbitrario (no utilizado por el sistema).

#### Ejemplos básicos

El siguiente ejemplo ilustra el tipo de dato `jointtarget`:

##### Ejemplo 1

```
CONST jointtarget calib_pos := [ [ 0, 0, 0, 0, 0, 0 ], [ 0, 9E9,
9E9, 9E9, 9E9, 9E9 ] ];
```

En el caso del sistema IRB2400, la posición de calibración normal se define en `calib_pos` con el tipo de dato `jointtarget`. La posición 0 (grados o mm) de calibración normal se define también para el eje externo lógico a. Los ejes externos del b al f permanecen sin definir.

#### Estructura

```
< 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 >
```

#### Información relacionada

| Para obtener más información sobre        | Consulte  |
|---|---|
| Desplazamiento hacia una posición de ejes | <a href="#">MoveAbsJ - Mueve el robot a una posición de ejes absoluta en la página 382</a><br><a href="#">MoveExtJ - Mueve una o varias unidades mecánicas sin TCP en la página 417</a> |
| Instrucciones de posicionamiento          | <i>Manual de referencia técnica - RAPID Overview, sección Resumen sobre RAPID - Movimiento</i>  |
| Configuración de ejes externos            | <i>Application manual - Additional axes and stand alone controller</i>  |



### 3.41 listitem - Estructura de datos de elementos de lista

#### Utilización

`listitem` se utiliza para definir líneas de menú que contienen texto con pequeños iconos opcionales en el dispositivo de usuario, por ejemplo el FlexPendant.

#### Descripción

Los datos del tipo `listitem` permiten al usuario definir líneas de menú para la función `UIListView`.

#### Ejemplo básico

El siguiente ejemplo ilustra el tipo de dato `listitem` :

#### Ejemplo 1

```
CONST listitem list {3}:=[[stEmpty, "Item1"], [stEmpty, "Item2"],
[stEmpty, "Item3"]];
```

Una lista de menú con Item1....Item3 para su uso en la función `UIListView`.

#### Componentes

El tipo de dato contiene los componentes siguientes:

`image`

Tipo de dato: `string`

La ruta, incluido el nombre de archivo, de la imagen de icono a mostrar (no implementado en esta versión de software).

Utilice una cadena vacía " " o `stEmpty` si no desea mostrar ningún icono.

`text`

Tipo de dato: `string`

El texto de la línea de menú a mostrar.

#### Estructura

```
<dataobject of listitem>
  <image of string>
  <text of string>
```

#### Información relacionada

| Para obtener más información sobre                   | Consulte   |
|--|--|
| Interacción con el usuario con <code>Listview</code> | <a href="#">UIListView - Vista de lista de usuario en la página 1568</a> |

## 3 Tipos de datos

### 3.42 loaddata - Datos de carga RobotWare - OS

### 3.42 loaddata - Datos de carga

#### Utilización

`loaddata` se utiliza para describir las cargas fijadas a la interfaz mecánica del robot (la brida de montaje del robot).

Los datos de carga suelen definir la carga útil o la carga de la pinza (ajustada con la instrucción `GripLoad` o `MechUnitLoad` en el caso de los posicionadores) del robot, es decir, la carga presente en la pinza del robot. `loaddata` también se usa como parte del dato `tooldata` para describir la carga de la herramienta.

#### Descripción

Se utilizan cargas especificadas para configurar un modelo dinámico del robot, de forma que sus movimientos puedan ser controlados de la mejor forma posible.



#### ¡AVISO!

Es importante definir siempre la carga real de la herramienta y, si se usa, la carga útil del robot (por ejemplo, una pieza sujeta por una pinza). Una definición incorrecta de los datos de carga puede dar lugar a la sobrecarga de la estructura mecánica del robot. Existe también el riesgo de que pueda superarse la velocidad en el modo manual a velocidad reducida.

Cuando se especifican datos de carga incorrectos, este hecho suele tener las consecuencias siguientes:

- El robot no puede funcionar a su capacidad máxima.
- Peor exactitud de la trayectoria, con riesgo de sobrepasar posiciones.
- Riesgo de sobrecarga de la estructura mecánica.

El controlador monitoriza continuamente la carga y escribe un registro de eventos si la carga es más elevada que la prevista. Este registro de eventos se guarda y registra en la memoria del controlador.

#### Componentes



#### Nota

En esta descripción, `loaddata` solo se describe tal y como se usa para una carga útil. Para su uso como una carga de la herramienta, consulte [tooldata - Datos de herramienta en la página 1795](#).

`mass`

Tipo de dato: `num`

La masa (el peso) de la carga en kilos.

`cog`

*center of gravity*

Tipo de dato: `pos`

*Continúa en la página siguiente*

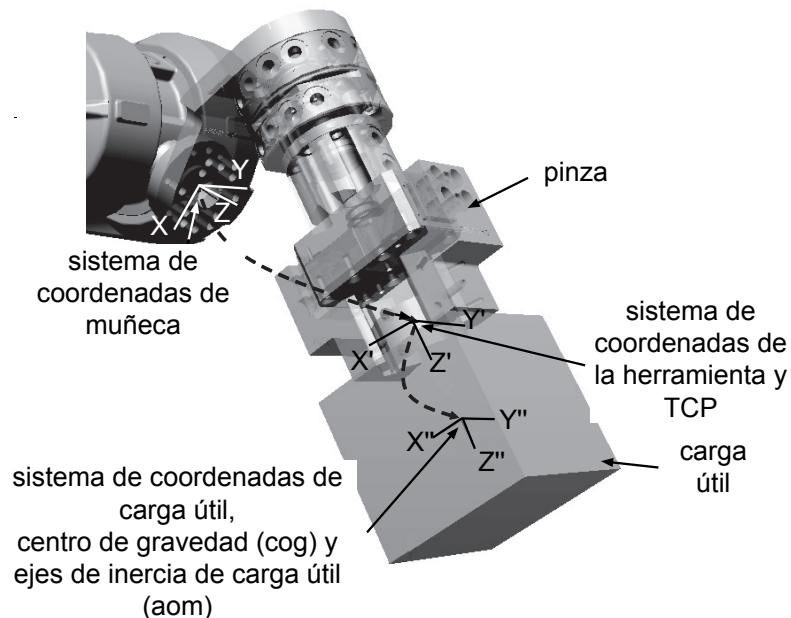
El centro de gravedad de la carga útil expresada en mm en el sistema de coordenadas de la herramienta si el robot es el que sujeta la herramienta. Si se utiliza una herramienta estacionaria, el centro de gravedad de la carga útil sostenida por la pinza se expresa en la base de coordenadas de objeto del sistema de coordenadas del objeto de trabajo movido por el robot.

aom

**axes of moment**Tipo de dato: `orient`

La orientación de los ejes de momento. Se trata de los ejes principales del momento de inercia de la carga útil, con origen en `cog`. Si el robot es el que sujeta la herramienta, los ejes de momento se expresan en el sistema de coordenadas de la herramienta.

La figura muestra el centro de gravedad y los ejes de inercia de la carga útil.



xx110000515

Figure 3.1: Herramienta sostenida por el robot

**Nota**

Si se utiliza *PayloadsInWristCoords*, los ejes del momento para la carga útil de la herramienta sostenida por el robot se expresan en el sistema de coordenadas de la muñeca. Para obtener más información, consulte *Manual de referencia técnica - Parámetros del sistema*, sección *PayloadsInWristCoords*.

Continúa en la página siguiente

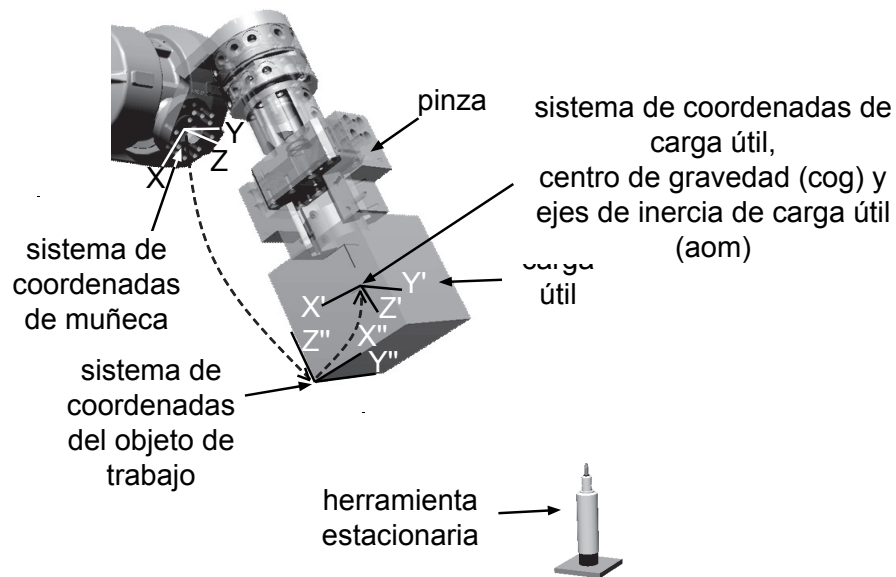
### 3 Tipos de datos

#### 3.42 loaddata - Datos de carga

RobotWare - OS

Continuación

Los ejes de momento se expresan en el sistema de coordenadas del objeto si se utiliza una herramienta estacionaria.



xx1100000516

Figure 3.2: Herramienta estacionaria



#### Nota

Si se utiliza *PayloadsInWristCoords* o *StationaryPayLoadMode*, los ejes del momento para la carga útil de la herramienta estacionaria se expresan en el sistema de coordenadas de la muñeca. Consulte *Manual de referencia técnica - Parámetros del sistema*, secciones *PayloadsInWristCoords* y *StationaryPayLoadMode*.

ix

#### inertia x

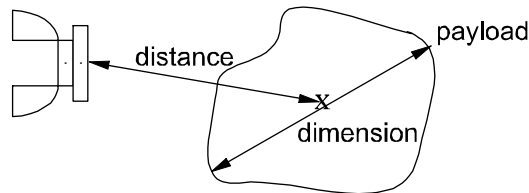
Tipo de dato: num

El momento de inercia de la carga alrededor del eje x del momento, expresado en  $\text{kgm}^2$ .

La definición correcta de los momentos de inercia permitirá una utilización óptima del planificador de trayectorias y un mejor control de los ejes. Esto puede resultar especialmente importante a la hora de manejar grandes planchas de metal, etc. Todos los momentos de inercia  $i_x$ ,  $i_y$  e  $i_z$  iguales a  $0 \text{ kgm}^2$  implican una masa puntual.

Continúa en la página siguiente

Normalmente, sólo es necesario definir momentos de inercia cuando la distancia existente entre la brida de montaje y el centro de gravedad es menor que el tamaño máximo de la carga (consulte la figura que aparece a continuación).



xx0500002372

iy

*inertia y*

Tipo de dato: num

El momento de inercia de la carga alrededor del eje Y, expresado en  $\text{kgm}^2$ .

Para obtener más información, consulte ix.

iz

*inertia z*

Tipo de dato: num

El momento de inercia de la carga alrededor del eje Z, expresado en  $\text{kgm}^2$ .

Para obtener más información, consulte ix.

### Ejemplos básicos

Los siguientes ejemplos ilustran el tipo de dato loaddata:

#### Ejemplo 1

```
PERS loaddata piece1 := [ 5, [50, 0, 50], [1, 0, 0, 0], 0, 0, 0];
```

La carga útil movida por una herramienta sostenida por el robot en la figura [Herramienta sostenida por el robot en la página 1695](#) se describe utilizando los valores siguientes:

- Peso 5 kg.
- El centro de gravedad es  $x = 50$ ,  $y = 0$  y  $z = 50$  mm en el sistema de coordenadas de la herramienta.
- La carga útil es una masa puntual.

#### Ejemplo 2

```
Set gripper;  
WaitTime 0.3;  
GripLoad piece1;
```

Conexión de la carga útil, piece1, especificada en el momento en que el robot sujeta la carga.

Continúa en la página siguiente

## 3 Tipos de datos

---

### 3.42 loaddata - Datos de carga

RobotWare - OS

Continuación

#### Ejemplo 3

```
Reset gripper;  
WaitTime 0.3;  
GripLoad load0;
```

Desconexión de la carga útil, especificada en el momento en que el robot suelta una carga útil.

#### Ejemplo 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] ] ];
```

La carga útil movida de acuerdo con la herramienta estacionaria de la figura [Herramienta estacionaria en la página 1696](#) se describe con los siguientes valores para loaddata:

- Peso 5 kg
- El centro de gravedad es  $x = 50$ ,  $y = 50$  y  $z = 50$  mm en el sistema de coordenadas del objeto de trabajo wobj2.
- El sistema de coordenadas/ejes de movimiento de la carga útil se gira  $180^\circ$  alrededor de Y" de acuerdo con el sistema de coordenadas del objeto.
- La carga útil es una masa puntual.

Los siguientes valores se usan para wobjdata:

- El robot sostiene el objeto de trabajo.
- Se utiliza el sistema fijo de coordenadas del usuario, es decir, el sistema de coordenadas del usuario es el mismo que el sistema de coordenadas de la muñeca.
- El sistema de coordenadas del objeto se gira  $-120^\circ$  alrededor de Y y las coordenadas de su origen son  $x = 50$ ,  $y = -50$  y  $z = 200$  mm en el sistema de coordenadas del usuario.

---

### Limitaciones

La carga útil sólo debe definirse mediante variables persistentes (PERS) y no desde dentro de una rutina. De esta forma, los valores se guardan al guardar el programa y se recuperan al cargarlo.

Los argumentos de tipo loaddata de la instrucción GripLoad y MechUnitLoad deben ser sólo del tipo persistente completo (ni elementos de matriz ni componentes de registro).

---

### Datos predefinidos

La carga load0 define una carga útil cuya masa es igual a 0 kg, es decir, ninguna carga. Esta carga se utiliza como argumento en las instrucciones GripLoad y MechUnitLoad para desconectar la carga útil.

La carga load0 está siempre disponible desde el programa, pero no puede ser modificada (está almacenada en el módulo de sistema BASE).

```
PERS loaddata load0 := [ 0.001, [0, 0, 0.001], [1, 0, 0, 0], 0, 0  
, 0 ];
```

Continúa en la página siguiente

#### Estructura

```

<dataobject 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>
  
```

#### Información relacionada

| Para obtener más información sobre  | Consulte  |
|---|---|
| Sistemas de coordenadas   | <i>Manual de referencia técnica - RAPID Overview</i>  |
| Definición de cargas de herramienta                                       | <a href="#">tooldata - Datos de herramienta en la página 1795</a>                                   |
| Definición de una carga útil para robots                                  | <a href="#">GripLoad - Define la carga útil de un robot en la página 244</a>                        |
| Definición de una carga útil para unidades mecánicas                      | <a href="#">MechUnitLoad - Define una carga útil para una unidad mecánica en la página 372</a>      |
| Identificación de la carga de la herramienta, carga útil o carga de brazo | <i>Manual del operador - IRC5 con FlexPendant</i>   |
| Definición de cargas de brazo   | <i>Manual de referencia técnica - Parámetros del sistema, consulte Cómo definir cargas de brazo</i> |
| Definición de datos de objetos de trabajo                                 | <a href="#">wobjdata - Datos del objeto de trabajo en la página 1823</a>                            |
| PayloadsInWristCoords<br>StationaryPayLoadMode                            | <i>Manual de referencia técnica - Parámetros del sistema, tema Controller, tipo General Rapid</i>   |

## 3 Tipos de datos

### 3.43 loadidnum - Tipo de identificación de carga

RobotWare - OS

### 3.43 loadidnum - Tipo de identificación de carga

#### Utilización

loadidnum se utiliza para representar un entero con una constante simbólica.

#### Descripción

Las constantes loadidnum se han diseñado para ser usadas durante la identificación de cargas de herramientas o cargas útiles, como argumentos de la instrucción LoadId. Consulte el ejemplo que aparece a continuación.

#### Ejemplos básicos

El siguiente ejemplo ilustra el tipo de dato loadidnum:

##### Ejemplo 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;
```

Identificación de la carga de la herramienta gun1 con identificación de la masa con los movimientos del eje 3 del robot, a través del uso de la constante MASS\_WITH\_AX3 del tipo de dato loadidnum.

#### Datos predefinidos

Las siguientes constantes simbólicas para el tipo de dato loadidnum están predefinidas y se utilizan como argumentos de la instrucción LoadId.

| Valor | Constante simbólica | Comentario  |
|-------|---------------------|---|
| 1     | MASS_KNOWN          | Masa conocida de la herramienta o de la carga útil respectivamente.   |
| 2     | MASS_WITH_AX3       | Masa de la herramienta o de la carga útil desconocida. La identificación de la masa se realiza con los movimientos del eje 3. |

#### Características

loadidnum es un tipo de dato de alias de num y por tanto hereda sus características.

#### Información relacionada

| Para obtener más información sobre                 | Consulte   |
|--|--|
| Identificación de carga de un programa predefinido | <i>Manual del operador - IRC5 con FlexPendant, sección Programación y testing - Rutinas de servicio - LoadIdentify, rutina de servicio de identificación de cargas</i> |
| Tipo de robot válido                               | <a href="#">ParIdRobValid - Tipo de robot válido para la identificación de parámetros en la página 1403</a>  |
| Posición de robot válida                           | <a href="#">ParIdPosValid - Posición de robot válida para la identificación de parámetros en la página 1400</a>  |

Continúa en la página siguiente



#### 3.43 loadidnum - Tipo de identificación de carga

*RobotWare - OS*

*Continuación*

| Para obtener más información sobre               | Consulte  |
|--|---|
| Identificación de cargas con un ejemplo completo | <a href="#">LoadId - Identificación de carga de la herramienta o la carga útil en la página 352</a> |

### 3 Tipos de datos

---

3.44 loadsession - Sesión de carga de programa  
RobotWare - OS

#### 3.44 loadsession - Sesión de carga de programa

---

##### Utilización

loadsession se utiliza para definir distintas sesiones de carga de módulos de programa de RAPID.

##### Descripción

Los datos de tipo loadsession se utilizan en las instrucciones StartLoad y WaitLoad para identificar la sesión de carga. loadsession sólo contiene una referencia a la sesión de carga.

##### Características

loadsession es un tipo de dato sin valor y no puede usarse en operaciones basadas en valores.

##### Información relacionada

| Para obtener más información sobre                | Consulte   |
|---|--|
| Carga de módulos de programa durante la ejecución | <a href="#">StartLoad - Carga de programa durante la ejecución en la página 792</a><br><a href="#">WaitLoad - Conectar un módulo cargado a una tarea en la página 1064</a> |
| Características de los tipos de datos sin valor   | <a href="#">Manual de referencia técnica - RAPID Overview, sección Características básicas - Tipos de datos</a>  |

### 3.45 mecunit - Unidad mecánica

#### Utilización

`mecunit` se utiliza para definir las distintas unidades mecánicas que pueden controlarse y utilizarse desde el programa.

Los nombres de las unidades mecánicas se definen en los parámetros del sistema y, por tanto, no debe definirlos en el programa.

#### Descripción

Los datos del tipo `mecunit` sólo contienen una referencia a la unidad mecánica.

#### Limitaciones

No debe definir los datos de tipo `mecunit` en el programa. Sin embargo, si lo hace, aparecerá un mensaje de error tan pronto como se ejecute la instrucción o función que hace referencia a este dato `mecunit`. Sin embargo, sí es posible utilizarlos como parámetros al declarar una rutina.

#### Datos predefinidos

Todas las unidades mecánicas definidas en los parámetros del sistema están predefinidas en cada tarea de programa. Sin embargo, sólo las unidades mecánicas controladas por la tarea de programa actual (definida en los parámetros de sistema *Controller/Task/Use Mechanical Unit Group*) se usan para realizar cualquier operación de control.

Además, la variable predefinida `ROB_ID` del tipo de dato `mecunit` está disponible en cada tarea de programa. Si la tarea de programa actual controla un robot, la variable de alias `ROB_ID` contiene una referencia a uno de los robots de `ROB_1` a `ROB_6`, que puede usarse para controlar el funcionamiento del robot. La variable `ROB_ID` no es válida si la tarea de programa actual no controla ningún robot.

#### Ejemplos básicos

El siguiente ejemplo ilustra el tipo de dato `mecunit`:

##### Ejemplo 1

```
IF TaskRunRob() THEN
  IndReset ROB_ID, 6;
ENDIF
```

Si la tarea de programa actual controla un robot, restablece el eje 6 del robot.

#### Características

`mecunit` es un tipo de dato *sin valor*. Esto significa que los datos de este tipo no son compatibles con operaciones basadas en valores.

#### Información relacionada

| Para obtener más información sobre               | Consulte   |
|--|--|
| Comprobación de si la tarea controla algún robot | <a href="#">TaskRunRob - Comprueba si una tarea controla algún robot en la página 1516</a> |

*Continúa en la página siguiente*

### 3 Tipos de datos

---

#### 3.45 mecunit - Unidad mecánica

RobotWare - OS

Continuación

| Para obtener más información sobre                          | Consulte   |
|---|--|
| Comprobación de si la tarea controla alguna unidad mecánica | <a href="#">TaskRunMec - Comprueba si una tarea controla alguna unidad mecánica en la página 1515</a>  |
| Obtención del nombre de las unidades mecánicas del sistema  | <a href="#">GetNextMechUnit - Obtener el nombre y los datos de las unidades mecánicas en la página 1306</a>  |
| Activación y desactivación de unidades mecánicas            | <a href="#">ActUnit - Activa una unidad mecánica en la página 30</a><br><a href="#">DeactUnit - Desactiva una unidad mecánica en la página 187</a> |
| Configuración de unidades mecánicas                         | <a href="#">Manual de referencia técnica - Parámetros del sistema</a>  |
| Características de los tipos de datos sin valor             | <a href="#">Manual de referencia técnica - RAPID Overview, sección Características básicas - Tipos de datos</a>                                    |

## 3.46 motsetdata - Datos de parámetros de movimiento

### Utilización

`motsetdata` se utiliza para definir un conjunto de parámetros de movimiento que afectan a todas las instrucciones de movimiento del programa:

- Velocidad máxima y ajuste de velocidad
- Datos de aceleración
- Comportamiento cerca de puntos singulares
- Gestión de distintas configuraciones de robot
- Ajuste de la resolución de las trayectorias
- Supervisión del movimiento
- Limitación de la aceleración y deceleración
- Reorientación de la herramienta durante trayectorias circulares
- Activación y desactivación de búfer de eventos

Normalmente no es necesario utilizar este tipo de dato dado que estos valores sólo pueden establecerse con las instrucciones `VelSet`, `AccSet`, `SingArea`, `ConfJ`, `ConfL`, `PathResol`, `MotionSup`, `PathAccLim`, `CirPathMode`, `WorldAccLim`, `ActEventBuffer`, `DeactEventBuffer` y `CornerPathWarning`.

Los valores actuales de estos parámetros de movimiento están disponibles a través de la variable de sistema `C_MOTSET`.

### Descripción

Los parámetros de movimiento actuales (almacenados en la variable de sistema `C_MOTSET`) afectan a todos los movimientos.

### Componentes



#### Nota

Algunos componentes están preparados en la estructura pero actualmente no están implementados en las instrucciones correspondientes.

`vel.oride`

Tipo de dato: `veldata/num`

Ajuste de velocidad como porcentaje de la velocidad programada.

`vel.max`

Tipo de dato: `veldata/num`

Velocidad máxima en mm/seg.

`acc.acc`

Tipo de dato: `accddata/num`

Aceleración y deceleración como porcentaje de los valores normales.

`acc.ramp`

Tipo de dato: `accddata/num`

*Continúa en la página siguiente*

### 3 Tipos de datos

---

#### 3.46 motsetdata - Datos de parámetros de movimiento

RobotWare - OS

Continuación

La proporción en que la aceleración y deceleración aumentan como porcentaje de los valores normales.

acc.finepramp

Tipo de dato: accdata/num

La velocidad a la que la deceleración disminuye como porcentaje de los valores normales cuando el robot decelera hacia un punto fino.

sing.wrist

Tipo de dato: singdata/bool

La orientación hasta la cual se permite que la herramienta se desvíe levemente, para prevenir una singularidad de muñeca.

sing.lockaxis4

Tipo de dato: singdata/bool

Bloqueo del eje 4 de un robot de seis ejes en la posición 0 o en  $\pm 180$  grados para evitar problemas de singularidad cuando el eje 5 se aproxime a 0.

sing.arm

Tipo de dato: singdata/bool

La orientación hasta la cual se permite que la herramienta se desvíe levemente, para prevenir una singularidad de brazo (no implementada).

sing.base

Tipo de dato: singdata/bool

La orientación de la que no debe desviarse la herramienta.

conf.jsup

Tipo de dato: confsupdata/bool

Durante el movimiento de los ejes, el robot alcanzará la configuración de robot programada.

conf.lsup

Tipo de dato: confsupdata/bool

La supervisión de la configuración de los ejes está activa durante el movimiento lineal y circular.

conf.ax1

Tipo de dato: confsupdata/num

Desviación máxima permitida en grados para el eje 1 (no implementada).

conf.ax4

Tipo de dato: confsupdata/num

Desviación máxima permitida en grados para el eje 4 (no implementada).

conf.ax6

Tipo de dato: confsupdata/num

Desviación máxima permitida en grados para el eje 6 (no implementada).

*Continúa en la página siguiente*

pathresol

Tipo de dato: num

El ajuste actual como porcentaje de la resolución de trayectoria configurada.

motionsup

Tipo de dato: bool

Conmutar el estado de la función de supervisión de movimientos de RAPID (**TRUE** = On y **FALSE** = Off).

tunevalue

Tipo de dato: num

Ajuste actual de RAPID como porcentaje del valor de ajuste configurado para la función de supervisión de movimientos.

backoffaftercoll

Tipo de dato: bool

Duplicar el estado de retroceso de RAPID para eliminar cualquier fuerza residual en una colisión de movimiento:

**TRUE** = Retrocederá para eliminar cualquier fuerza residual en una colisión por movimiento

**FALSE** = No habrá retroceso en una colisión por movimiento

acclim

Tipo de dato: bool

Limitación de aceleración de la herramienta a lo largo de la trayectoria. (**TRUE** = con limitación y **FALSE** = sin limitación).

accmax

Tipo de dato: num

Limitación de aceleración del TCP en  $m/s^2$ . Si **acclim** tiene el valor **FALSE**, el valor es siempre -1.

decellim

Tipo de dato: bool

Limitación de deceleración de la herramienta a lo largo de la trayectoria. (**TRUE** = con limitación y **FALSE** = sin limitación).

decelmax

Tipo de dato: num

Limitación de deceleración del TCP en  $m/s^2$ . Si **decellim** tiene el valor **FALSE**, el valor es siempre -1.

cirpathreori

Tipo de dato: num

Reorientación de la herramienta durante trayectorias circulares:

0 = Método estándar con interpolación en la base de coordenadas de la trayectoria

1 = Método modificado con interpolación en la base de coordenadas del objeto

*Continúa en la página siguiente*

## 3 Tipos de datos

---

### 3.46 motsetdata - Datos de parámetros de movimiento

RobotWare - OS

Continuación

2 = Método modificado con orientación programada de la herramienta en CirPoint

worldacclim

Tipo de dato: bool

Limitación de aceleración en el sistema de coordenadas mundo. (TRUE = con limitación y FALSE = sin limitación).

worldaccmax

Tipo de dato: num

Limitación de aceleración en el sistema de coordenadas mundo en  $\text{m/s}^2$ . Si worldacclim tiene el valor FALSE, el valor es siempre -1.

evtbufferact

Tipo de dato: bool

Búfer de eventos activo o no activo. (TRUE = búfer de eventos activo y FALSE = búfer de eventos no activo).

corner\_path\_warn\_suppress

Tipo de dato: bool

Se informa o no del aviso de trayectoria de esquina. TRUE = se ha eliminado el aviso de trayectoria de esquina, FALSE = no se ha eliminado el aviso de trayectoria de esquina.

---

### Limitaciones

Sólo y sólo uno de los componentes `sing.wrist`, `sing.arm` o `sing.base` puede tener un valor igual a TRUE.

---

### Ejemplos básicos

El siguiente ejemplo ilustra el tipo de dato `motsetdata`:

#### Ejemplo 1

```
IF C_MOTSET.vel.oride > 50 THEN
  ...
ELSE
  ...
ENDIF
```

Se ejecutan partes distintas del programa en función del ajuste de velocidad actual.

---

### Datos predefinidos

`C_MOTSET` describe los parámetros de movimiento actuales del robot y está siempre disponible desde el programa. Por otro lado, `C_MOTSET` sólo puede modificarse mediante un conjunto de instrucciones, no mediante asignación.

Los valores predeterminados siguientes para los parámetros de movimiento se establecen:

- cuando se utiliza el modo de reinicio **Restablecer RAPID**
- al cargar un nuevo programa o un nuevo módulo
- al iniciar la ejecución del programa desde el principio

Continúa en la página siguiente



- al mover el puntero del programa a `main`
- al mover el puntero del programa a una rutina
- al mover el puntero de programa de una forma que se pierde el orden de la ejecución.

```
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
  TRUE,-> backoffaftercoll
  FALSE,-> acclim
  -1,-> accmax
  FALSE,-> decellim
  -1,-> decelmax
  0,-> cirpathreori
  FALSE,-> worldacclim
  -1,-> worldaccmax
  TRUE,-> evtbufferact
  FALSE];-> corner_path_warn_suppress
```



#### Nota

La velocidad máxima de TCP para el tipo de robot utilizado puede cambiarse en los parámetros *Motion* de configuración del sistema, escriba *Motion Planner* y el atributo *Linear Max Speed*. La función de RAPID `MaxRobSpeed` devuelve el mismo valor.

#### Estructura

|                            |   |
|----------------------------|---|
| <dataobject of motsetdata> |   |
| <vel of veldata>           | Afectado por la instrucción <code>VelSet</code>   |
| <oride of num>             |   |
| <max of num>               |   |
| <acc of accdata>           | Afectado por la instrucción <code>AccSet</code>   |
| <acc of num>               |   |
| <ramp of num>              |   |
| <finepramp of num>         |   |
| <sing of singdata>         | Afectado por la instrucción <code>SingArea</code> |
| <wrist of bool>            |   |
| <lockaxis4 of bool>        |   |
| <arm of bool>              |   |
| <base of bool>             |   |

Continúa en la página siguiente

### 3 Tipos de datos

#### 3.46 motsetdata - Datos de parámetros de movimiento

RobotWare - OS

Continuación

|  |  |
|--|--|
| <code>&lt;conf of confsupdata&gt;</code>               | Afectado por las instrucciones <code>ConfJ</code> y <code>ConfL</code>                     |
| <code>&lt;jsup of bool&gt;</code>                      |  |
| <code>&lt;lsup of bool&gt;</code>                      |  |
| <code>&lt;ax1 of num&gt;</code>                        |  |
| <code>&lt;ax4 of num&gt;</code>                        |  |
| <code>&lt;ax6 of num&gt;</code>                        |  |
| <code>&lt;pathresol of num&gt;</code>                  | Afectado por la instrucción <code>PathResol</code>   |
| <code>&lt;motionsup of bool&gt;</code>                 | Afectado por la instrucción <code>MotionSup</code>   |
| <code>&lt;tunevalue of num&gt;</code>                  | Afectado por la instrucción <code>MotionSup</code>   |
| <code>&lt;backoffaftercoll of bool&gt;</code>          | Afectado por la instrucción <code>MotionSup</code>   |
| <code>&lt;acclim of bool&gt;</code>                    | Afectado por la instrucción <code>PathAccLim</code>  |
| <code>&lt;accmax of num&gt;</code>                     | Afectado por la instrucción <code>PathAccLim</code>  |
| <code>&lt;decellim of bool&gt;</code>                  | Afectado por la instrucción <code>PathAccLim</code>  |
| <code>&lt;decelmax of num&gt;</code>                   | Afectado por la instrucción <code>PathAccLim</code>  |
| <code>&lt;cirpathreori of num&gt;</code>               | Afectado por la instrucción <code>CirPathMode</code>                                       |
| <code>&lt;worldacclim of bool&gt;</code>               | Afectado por la instrucción <code>WorldAccLim</code>                                       |
| <code>&lt;worldaccmax of num&gt;</code>                | Afectado por la instrucción <code>WorldAccLim</code>                                       |
| <code>&lt;evtbufferact of bool&gt;</code>              | Afectado por las instrucciones <code>ActEventBuffer</code> y <code>DeactEventBuffer</code> |
| <code>&lt;corner_path_warn_suppress of bool&gt;</code> | Afectado por la instrucción <code>CornerPathWarning</code>                                 |

#### Información relacionada

| Para obtener más información sobre                                      | Consulte   |
|---|--|
| Reducción de la aceleración   | <a href="#">AccSet - Reduce la aceleración en la página 25</a>   |
| Activación de un búfer de eventos                                       | <a href="#">ActEventBuffer - Activación de búfer de eventos en la página 28</a>                                |
| Reorientación de la herramienta durante trayectorias circulares         | <a href="#">CirPathMode - Reorientación de la herramienta durante trayectorias circulares en la página 133</a> |
| Configuración del robot durante el movimiento de ejes                   | <a href="#">ConfJ - Controla la configuración durante el movimiento de los ejes en la página 156</a>           |
| Configuración del robot durante el movimiento lineal                    | <a href="#">ConfL - Monitoriza la configuración durante el movimiento lineal en la página 158</a>              |
| Eliminar los avisos de trayectoria de esquina                           | <a href="#">CornerPathWarning - Mostrar u ocultar avisos de trayectoria de esquina en la página 174</a>        |
| Desactivación de búfer de eventos                                       | <a href="#">DeactEventBuffer - Desactivación de búfer de eventos en la página 185</a>                          |
| Activa o desactiva la función de supervisión de movimiento              | <a href="#">MotionSup - Desactiva/activa la supervisión del movimiento en la página 379</a>                    |
| Limitación de aceleración a lo largo de la trayectoria                  | <a href="#">PathAccLim - Reduce la aceleración del TCP a lo largo de la trayectoria en la página 504</a>       |
| Redefine el tiempo de muestreo de la trayectoria geométrica configurada | <a href="#">PathResol - Ajusta la resolución de la trayectoria en la página 530</a>                            |

Continúa en la página siguiente

| Para obtener más información sobre                                    | Consulte  |
|---|---|
| Definición del método de interpolación alrededor de puntos singulares | <a href="#">SingArea - Define el método de interpolación alrededor de puntos singulares en la página 730</a>          |
| Definición de la velocidad máxima                                     | <a href="#">VelSet - Cambia la velocidad programada en la página 1029</a>   |
| Control de aceleración en el sistema de coordenadas mundo             | <a href="#">WorldAcCLim - Control de aceleración en el sistema de coordenadas mundo en la página 1096</a>             |
| Instrucciones para el establecimiento de parámetros de movimiento     | <a href="#">Manual de referencia técnica - RAPID Overview, sección Resumen sobre RAPID - Parámetros de movimiento</a> |

## 3 Tipos de datos

---

### 3.47 num - Valores numéricos

RobotWare - OS

### 3.47 num - Valores numéricos

---

#### Utilización

`num` se utiliza con valores numéricos, como por ejemplo, contadores.

---

#### Descripción

El valor del tipo de dato `num` puede ser:

- Un entero, por ejemplo -5
- Un número con decimales, por ejemplo 3,45

También puede escribirse de forma exponencial, por ejemplo 2E3 (=  $2 \cdot 10^3 = 2.000$ ), 2,5E-2 (= 0,025).

Los enteros entre -8388607 y +8388608 se almacenan siempre como enteros exactos.

Los números con decimales son sólo números aproximados y, por tanto, no deben utilizarse en comparaciones de tipo *igual a* ni *distinto de*. En el caso de las divisiones y las operaciones que utilizan números con decimales, el resultado también será un número con decimales, es decir, no un entero exacto. Por ejemplo:

```
a := 10;  
b := 5;  
IF a/b=2 THEN
```

...

Dado que el resultado de  $a/b$  no es un entero, esta condición no tiene por qué cumplirse necesariamente.

---

#### Ejemplos básicos

Los siguientes ejemplos ilustran el tipo de dato `num`:

##### Ejemplo 1

```
VAR num reg1;  
...  
reg1 := 3;  
reg1 recibe el valor 3.
```

##### Ejemplo 2

```
a := 10 DIV 3;  
b := 10 MOD 3;
```

División entera en la que se asigna un entero  $a (=3)$  y se asigna a  $b$  el resto ( $=1$ ).

---

#### Datos predefinidos

El sistema cuenta con ciertos datos predefinidos. Por ejemplo, se define la constante  $\pi$ .

```
CONST num pi := 3.1415926;
```

---

#### Limitaciones

Los valores literales entre -8388607 y 8388608 asignados a una variable `num` se almacenan siempre como enteros exactos.

---

*Continúa en la página siguiente*

Si un valor literal que ha sido interpretado como un valor `dnum` es asignado o usado como un valor `num`, es convertido automáticamente a `num`.

#### Información relacionada

| Para obtener más información sobre                       | Consulte   |
|--|--|
| Valores numéricos con el tipo de datos <code>dnum</code> | <a href="#">dnum - Valores numéricos dobles en la página 1660</a>  |
| Expresiones numéricas                                    | Manual de referencia técnica - <i>RAPID Overview</i> , sección <i>Programación básica en RAPID - Expresiones</i> |
| Operaciones con valores numéricos                        | Manual de referencia técnica - <i>RAPID Overview</i> , sección <i>Programación básica en RAPID - Expresiones</i> |

## 3 Tipos de datos

3.48 opcalc - Operador aritmético  
RobotWare - OS

### 3.48 opcalc - Operador aritmético

#### Utilización

opcalc se utiliza para representar un operador aritmético en argumentos entregados a las funciones o instrucciones de RAPID.

#### Descripción

El uso previsto de las constantes opcalc es decir el tipo de operación aritmética.

#### Ejemplos

El siguiente ejemplo ilustra el tipo de dato opcalc:

##### Ejemplo 1

```
res := StrDigCalc(str1, OpAdd, str2);
```

Se asigna a `res` el resultado de la operación de suma de los valores representados por las cadenas `str1` y `str2`. `OpAdd` es del tipo de dato opcalc.

#### Datos predefinidos

Las siguientes constantes simbólicas para el tipo de dato opcalc están predefinidas y se usan para definir el tipo de operación aritmética utilizado, por ejemplo en la función `StrDigCalc`.

| Constante | Valor | Comentario        |
|-----------|-------|-------------------|
| OpAdd     | 1     | Suma (+)          |
| OpSub     | 2     | Resta (-)         |
| OpMult    | 3     | Multipliación (*) |
| OpDiv     | 4     | División (/)      |
| OpMod     | 5     | Módulo (%)        |

#### Características

opcalc es un tipo de dato de alias de num y por tanto hereda sus características.

#### Información relacionada

| Para obtener más información sobre                 | Consulte   |
|--|--|
| Tipos de datos en general, tipos de datos de alias | <i>Manual de referencia técnica - RAPID Overview</i>   |
| Operaciones aritméticas con cadenas digitales.     | <a href="#">StrDigCalc - Operaciones aritméticas con el tipo de dato stringdig en la página 1491</a> |

### 3.49 opnum - Operador de comparación

#### Utilización

`opnum` se utiliza para representar un operador de comparación en argumentos entregados a las funciones o instrucciones de RAPID.

#### Descripción

La constante `opnum` se ha diseñado para definir el tipo de comparación al comprobar valores en las instrucciones genéricas.

#### Ejemplos básicos

El siguiente ejemplo ilustra el tipo de dato `opnum`:

##### Ejemplo 1

```
TriggCheckIO checkgrip, 100, airok, EQ, 1, intnol;
```

#### Datos predefinidos

Las siguientes constantes simbólicas para el tipo de dato `opnum` están predefinidas y se usan para definir el tipo de operación aritmética utilizado, por ejemplo en la función `TriggCheckIO`.

| Valor | Constante simbólica | Comentario          |
|-------|---------------------|---------------------|
| 1     | LT                  | Menor que           |
| 2     | LTEQ                | Menor que o igual a |
| 3     | EQ                  | Igual a             |
| 4     | NOTEQ               | Distinto de         |
| 5     | GTEQ                | Mayor que o igual a |
| 6     | GT                  | Mayor que           |

#### Características

`opnum` es un tipo de dato de alias de `num` y por tanto hereda sus características.

#### Información relacionada

| Para obtener más información sobre                         | Consulte   |
|--|--|
| Tipos de datos en general, tipos de datos de alias         | <i>Manual de referencia técnica - RAPID Overview</i> , sección <i>Características básicas - Tipos de datos</i> |
| Definición de una comprobación de E/S en una posición fija | <a href="#">TriggCheckIO - Define una comprobación de E/S en una posición fija en la página 904</a>            |

## 3 Tipos de datos

---

3.50 orient - Orientación  
RobotWare - OS

### 3.50 orient - Orientación

---

#### Utilización

`orient` se utiliza para orientaciones (por ejemplo la orientación de una herramienta) y rotaciones (por ejemplo la rotación de un sistema de coordenadas).

---

#### Descripción

La orientación se describe en forma de un cuaternio compuesto por cuatro componentes:  $q_1$ ,  $q_2$ ,  $q_3$  y  $q_4$ .

---

#### Componentes

El tipo de dato `orient` contiene los componentes siguientes:

$q_1$

Tipo de dato: num  
Cuaternio 1.

$q_2$

Tipo de dato: num  
Cuaternio 2.

$q_3$

Tipo de dato: num  
Cuaternio 3.

$q_4$

Tipo de dato: num  
Cuaternio 4.

---

#### Ejemplos básicos

El siguiente ejemplo ilustra el tipo de dato `orient`:

##### Ejemplo 1

```
VAR orient orient1;  
.orient1 := [1, 0, 0, 0];
```

Se asigna a la orientación `orient1` el valor  $q_1=1$ ,  $q_2=q_3=q_4=0$ . Esto equivale a ninguna rotación.

---

#### Limitaciones

La orientación debe estar normalizada, es decir, la suma de los cuadrados debe ser igual a 1:

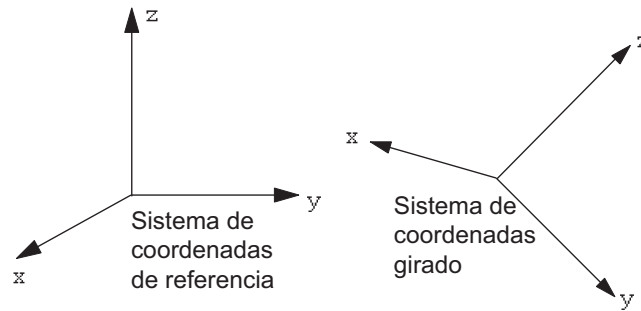
$$q_1^2 + q_2^2 + q_3^2 + q_4^2 = 1$$

*Continúa en la página siguiente*



#### ¿Qué es un cuaternio?

La orientación de un sistema de coordenadas (por ejemplo el de una herramienta) se describe mediante una matriz de rotación que describe la dirección de los ejes del sistema de coordenadas respecto de un sistema de referencia (consulte la figura que aparece a continuación).



xx0500002376

Los ejes del sistema de coordenadas girado (x, y, z) son vectores que pueden expresarse en el sistema de coordenadas de referencia de la forma siguiente:

$$\mathbf{x} = (x_1, x_2, x_3)$$

$$\mathbf{y} = (y_1, y_2, y_3)$$

$$\mathbf{z} = (z_1, z_2, z_3)$$

Esto significa que el componente x del vector x del sistema de coordenadas de referencia será x<sub>1</sub>, el componente y será x<sub>2</sub>, etc.

Estos tres vectores pueden reunirse en una matriz (una matriz de rotación) en la que cada uno de los vectores compone una de las columnas:

$$\begin{bmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \end{bmatrix}$$

xx0500002381

Un cuaternio es sólo una forma más concisa de referirse a esta matriz de rotación. Los cuaternios se calculan partiendo de los elementos de la matriz de rotación:

|  |   |
|--|---|
| $q_1 = \frac{\sqrt{x_1 + y_2 + z_3 + 1}}{2}$ |   |
| $q_2 = \frac{\sqrt{x_1 - y_2 - z_3 + 1}}{2}$ | sign q2 = sign (y <sub>3</sub> - z <sub>2</sub> ) |
| $q_3 = \frac{\sqrt{y_2 - x_1 - z_3 + 1}}{2}$ | sign q3 = sign (z <sub>1</sub> - x <sub>3</sub> ) |
| $q_4 = \frac{\sqrt{z_3 - x_1 - y_2 + 1}}{2}$ | sign q4 = sign (x <sub>2</sub> - y <sub>1</sub> ) |

Continúa en la página siguiente

### 3 Tipos de datos

#### 3.50 orient - Orientación

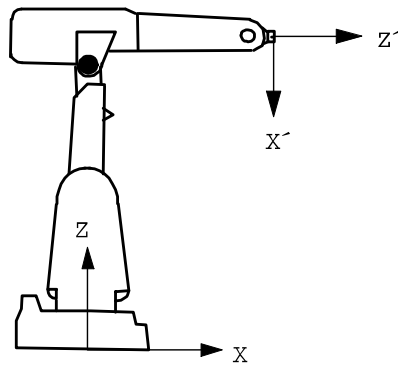
RobotWare - OS

Continuación

#### Ejemplo 1

Una herramienta se orienta de forma que su eje Z apunta directamente hacia delante (en la misma dirección que el eje X del sistema de coordenadas de la base). El eje Y de la herramienta se corresponde con el eje Y del sistema de coordenadas de la base (consulte la figura que aparece a continuación). ¿Cómo se define la orientación de la herramienta en los datos de posición (`robtarget`)?

La orientación de la herramienta en una posición programada suele estar relacionada con el sistema de coordenadas del objeto de trabajo utilizado. En este ejemplo no se utiliza ningún objeto de trabajo y el sistema de coordenadas de la base es igual al sistema de coordenadas mundo. Por tanto, la orientación está relacionada con el sistema de coordenadas de la base.



xx0500002377

En este caso, los ejes estarán relacionados de la forma siguiente:

$$x' = -z = (0, 0, -1)$$

$$y' = y = (0, 1, 0)$$

$$z' = x = (1, 0, 0)$$

Esto corresponde a la matriz de rotación siguiente:

$$\begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{bmatrix}$$

xx0500002388

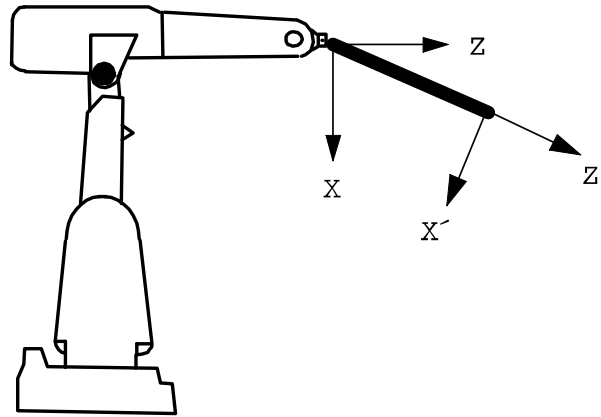
La matriz de rotación proporciona el cuaternio correspondiente:

|  |                          |
|--|--------------------------|
| $q1 = \frac{\sqrt{0+1+0+1}}{2} = \frac{\sqrt{2}}{2} = 0,707$ |                          |
| $q2 = \frac{\sqrt{0-1-0+1}}{2} = 0$                          |                          |
| $q3 = \frac{\sqrt{1-0-0+1}}{2} = \frac{\sqrt{2}}{2} = 0,707$ | sign q3 = sign (1+1) = + |
| $q4 = \frac{\sqrt{0-0-1+1}}{2} = 0$                          |                          |

Continúa en la página siguiente

#### Ejemplo 2

La dirección de la herramienta gira 30° alrededor de los ejes X' y Z' respecto del sistema de coordenadas de la muñeca (consulte la figura que aparece a continuación). ¿Cómo se define la orientación de la herramienta en los datos de la herramienta?



xx0500002378

En este caso, los ejes estarán relacionados de la forma siguiente:

$$x' = (\cos 30^\circ, 0, -\sin 30^\circ)$$

$$y' = (0, 1, 0)$$

$$z' = (\sin 30^\circ, 0, \cos 30^\circ)$$

Esto corresponde a la matriz de rotación siguiente:

$$\begin{bmatrix} \cos 30^\circ & 0 & \sin 30^\circ \\ 0 & 1 & 0 \\ -\sin 30^\circ & 0 & \cos 30^\circ \end{bmatrix}$$

xx0500002393

La matriz de rotación proporciona el cuaternio correspondiente:

|  |  |
|--|--|
| $q1 = \frac{\sqrt{\cos 30^\circ + 1 + \cos 30^\circ + 1}}{2} = 0,965926$ |  |
| $q2 = \frac{\sqrt{\cos 30^\circ - 1 - \cos 30^\circ + 1}}{2} = 0$        |  |
| $q3 = \frac{\sqrt{1 - \cos 30^\circ - \cos 30^\circ + 1}}{2} = 0,258819$ | sign q3 = sign (sin 30° + sin 30°) = + |
| $q4 = \frac{\sqrt{\cos 30^\circ - \cos 30^\circ - 1 + 1}}{2} = 0$        |  |

#### Estructura

```
< dataobject of orient >
  < q1 of num >
  < q2 of num >
  < q3 of num >
  < q4 of num >
```

Continúa en la página siguiente

### 3 Tipos de datos

---

3.50 orient - Orientación

*RobotWare - OS*

*Continuación*

---

#### Información relacionada

| Para obtener más información sobre | Consulte  |
|------------------------------------|---|
| Operaciones con orientaciones      | <i>Manual de referencia técnica - RAPID Overview, sección Características básicas - Expresiones</i> |

### 3.51 paridnum - Tipo de identificación de parámetro

#### Utilización

`paridnum` se utiliza para representar un entero con una constante simbólica.

#### Descripción

Las constantes `paridnum` se han diseñado para ser usadas en la identificación de parámetros, por ejemplo, identificaciones de carga de herramientas y cargas útiles o de la carga de un manipulador externo.

#### Ejemplos básicos

El siguiente ejemplo ilustra el tipo de dato `paridnum`:

##### Ejemplo 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
```

Se utiliza la constante predefinida `TOOL_LOAD_ID` del tipo de dato `paridnum`.

#### Datos predefinidos

Las siguientes constantes simbólicas para el tipo de dato `paridnum` están predefinidas y se utilizan como argumentos de las siguientes instrucciones: `ParIdRobValid`, `ParIdPosValid`, `LoadId` y `ManLoadIdProc`.

| Valor | Constante simbólica | Comentario   |
|-------|---------------------|--|
| 1     | TOOL_LOAD_ID        | Identificación de la carga de la herramienta                               |
| 2     | PAY_LOAD_ID         | Identificar la carga útil (consulte la instrucción <code>GripLoad</code> ) |
| 3     | IRBP_K              | Identificación de la carga del manipulador externo IRBP K                  |
| 4     | IRBP_L              | Identificación de la carga del manipulador externo IRBP L                  |
| 4     | IRBP_C              | Identificación de la carga del manipulador externo IRBP C                  |
| 4     | IRBP_C_INDEX        | Identificación de la carga del manipulador externo IRBP C_INDEX            |
| 4     | IRBP_T              | Identificación de la carga del manipulador externo IRBP T                  |

*Continúa en la página siguiente*

## 3 Tipos de datos

### 3.51 paridnum - Tipo de identificación de parámetro

RobotWare - OS

Continuación

| Valor | Constante simbólica | Comentario  |
|-------|---------------------|---|
| 5     | IRBP_R              | Identificación de la carga del manipulador externo IRBP R |
| 6     | IRBP_A              | Identificación de la carga del manipulador externo IRBP A |
| 6     | IRBP_B              | Identificación de la carga del manipulador externo IRBP B |
| 6     | IRBP_D              | Identificación de la carga del manipulador externo IRBP D |



#### Nota

Sólo `TOOL_LOAD_ID` y `PAY_LOAD_ID` se usan en los programas de RAPID definidos por el usuario para la identificación de cargas de la herramienta o la carga útil del robot.

#### Características

`paridnum` es un tipo de dato de alias de `num` y por tanto hereda sus características.

#### Información relacionada

| Para obtener más información sobre                | Consulte  |
|---|---|
| Programa predefinido Load Identify                | <i>Manual del operador - IRC5 con FlexPendant</i>   |
| Tipo de robot válido                              | <a href="#">ParldRobValid</a> - Tipo de robot válido para la identificación de parámetros en la página 1403     |
| Posición de robot válida                          | <a href="#">ParldPosValid</a> - Posición de robot válida para la identificación de parámetros en la página 1400 |
| Identificación de cargas con un ejemplo completo  | <a href="#">LoadId</a> - Identificación de carga de la herramienta o la carga útil en la página 352             |
| Identificación de carga de manipuladores externos | <a href="#">ManLoadIdProc</a> - Identificación de carga de los manipuladores IRBP en la página 360              |

### 3.52 paridvalidnum - Resultado de ParIdRobValid

#### Utilización

paridvalidnum se utiliza para representar un entero con una constante simbólica.

#### Descripción

Las constantes paridvalidnum se han diseñado para ser usadas en la identificación de parámetros, por ejemplo identificaciones de carga de herramientas y cargas útiles, a la hora de comprobar el valor de retorno de la función ParIdRobValid.

#### Ejemplos básicos

Los siguientes ejemplos ilustran el tipo de dato 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
```

Se utilizan las constantes predefinidas ROB\_LOAD\_VAL, ROB\_LM1\_LOAD\_VAL y ROB\_NOT\_LOAD\_VAL del tipo de dato paridvalidnum.

#### Datos predefinidos

Las siguientes constantes simbólicas para el tipo de dato paridvalidnum están predefinidas y se utilizan para comprobar el valor de retorno de la función ParIdRobValid.

| Valor | Constante simbólica | Comentario   |
|-------|---------------------|--|
| 10    | ROB_LOAD_VAL        | Tipo de robot válido para la identificación actual de parámetros   |
| 11    | ROB_NOT_LOAD_VAL    | No es ningún tipo de robot válido para la identificación actual de parámetros                              |
| 12    | ROB_LM1_LOAD_VAL    | Tipo de robot válido IRB 6400FHD para la identificación actual de parámetros, si la carga real es < 200 kg |

#### Características

paridvalidnum es un tipo de dato de alias de num y por tanto hereda sus características.

*Continúa en la página siguiente*

### 3 Tipos de datos

---

3.52 paridvalidnum - Resultado de ParIdRobValid

RobotWare - OS

Continuación

---

#### Información relacionada

| Para obtener más información sobre               | Consulte   |
|--|--|
| Programa predefinido Load Identify               | <i>Manual del operador - IRC5 con FlexPendant</i>  |
| Tipo de robot válido                             | <i><a href="#">ParIdRobValid - Tipo de robot válido para la identificación de parámetros en la página 1403</a></i>     |
| Posición de robot válida                         | <i><a href="#">ParIdPosValid - Posición de robot válida para la identificación de parámetros en la página 1400</a></i> |
| Identificación de cargas con un ejemplo completo | <i><a href="#">LoadId - Identificación de carga de la herramienta o la carga útil en la página 352</a></i>             |



### 3.53 pathrecid - Identificador de grabadora de trayectorias

#### Utilización

`pathrecid` se utiliza para identificar un punto de ruptura para la grabadora de trayectorias.

#### Descripción

La grabadora de trayectorias es una función del sistema que permite grabar la trayectoria ejecutada por el robot. Los datos de tipo `pathrecid` pueden ser vinculados a una ubicación de trayectoria determinada mediante la instrucción `PathRecStart`. A continuación, el usuario puede solicitar a la grabadora que realice un movimiento de vuelta hasta el identificador de la trayectoria, mediante la instrucción `PathRecMoveBwd`.

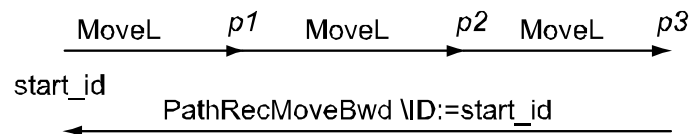
#### Ejemplos básicos

El siguiente ejemplo ilustra el tipo de dato `pathrecid`:

##### Ejemplo 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

En el ejemplo anterior, se inicia la grabadora de trayectorias y se marca el punto inicial con el identificador de trayectoria `start_id`. A partir de ese momento, el robot avanzará con las instrucciones de movimiento tradicionales y volverá a la posición inicial mediante la trayectoria grabada. Para poder ejecutar instrucciones de movimiento con la grabadora de trayectorias, es necesario cambiar el nivel de la trayectoria con `StorePath`.

#### Características

`pathrecid` es un tipo de dato sin valor

*Continúa en la página siguiente*

### 3 Tipos de datos

#### 3.53 pathrecid - Identificador de grabadora de trayectorias

*Path Recovery*

*Continuación*

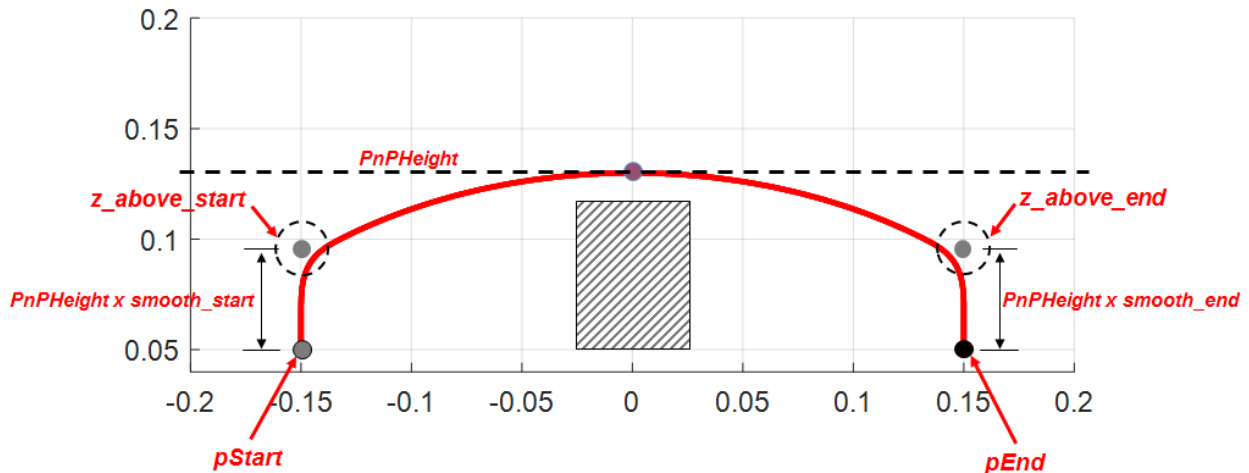
#### Información relacionada

| Para obtener más información sobre                         | Consulte  |
|--|---|
| Inicio y detención de la grabadora de trayectorias         | <a href="#">PathRecStart - Inicia la grabadora de trayectorias en la página 524</a><br><a href="#">PathRecStop - Detiene la grabadora de trayectorias en la página 527</a>  |
| Comprobación de que la trayectoria grabada es correcta     | <a href="#">PathRecValidBwd - Comprueba si existe una trayectoria de retroceso válida guardada en la página 1410</a><br><a href="#">PathRecValidFwd - Comprueba si existe una trayectoria de avance válida guardada en la página 1413</a> |
| Reproducción de la grabación de trayectorias hacia atrás   | <a href="#">PathRecMoveBwd - Hace retroceder la grabadora de trayectorias en la página 514</a>  |
| Reproducción de la grabación de trayectorias hacia delante | <a href="#">PathRecMoveFwd - Hace avanzar la grabadora de trayectorias en la página 521</a>   |
| Características de los tipos de datos sin valor            | <i>Manual de referencia técnica - RAPID Overview, sección Características básicas - Tipos de datos</i>  |

## 3.54 pnpdata: Configurar las trayectorias de elección y colocación

## Utilización

`pnpdata` se utiliza para configurar las trayectorias de elección y colocación.



xx1700001194

## Descripción

Datos del tipo `pnpdata` describen las alturas de los movimientos verticales como un porcentaje de la altura total de la trayectoria, y las zonas de las trayectorias de esquina generadas.

## Componentes

El tipo de dato contiene los componentes siguientes:

`smooth_start`

Tipo de dato: `num`

Porcentaje de `PnPHeight` para describir la altura del movimiento vertical sobre el punto de inicio.

Un valor inferior implica una velocidad considerablemente mayor en el tiempo de ciclo.

El valor predeterminado es 100.

`smooth_end`

Tipo de dato: `num`

Porcentaje de `PnPHeight` para describir la altura del movimiento vertical sobre el punto final.

Un valor inferior implica una velocidad considerablemente mayor en el tiempo de ciclo.

El valor predeterminado es 100.

`z_above_start`

Tipo de dato: `zonedata`

*Continúa en la página siguiente*

### 3 Tipos de datos

---

#### 3.54 pnpdata: Configurar las trayectorias de elección y colocación

##### SCARA robots

##### Continuación

Datos de zona del movimiento. `z_above_start` describe el tamaño de las trayectorias de esquina generadas al final del movimiento vertical sobre el punto de inicio.

El valor predeterminado es `z100`.

`z_above_end`

Tipo de dato: `zonedata`

Datos de zona del movimiento. `z_above_end` describe el tamaño de las trayectorias de esquina generadas al principio del movimiento vertical sobre el punto final.

El valor predeterminado es `z100`.

---

#### Ejemplos básicos

Consulte [MovePnP: Mueve el robot a lo largo de una trayectoria de elección y colocación. en la página 473](#).

---

#### Estructura

```
<dataobject of pnpdata>  
  <smooth_start of num>  
  <smooth_end of num>  
  <z_above_start of zonedata>  
  <z_above_end of zonedata>
```

---

#### Información relacionada

| Para obtener más información sobre  | Consulte   |
|---|--|
| Desplazamiento del robot a lo largo de una trayectoria de elección y colocación | <a href="#">MovePnP: Mueve el robot a lo largo de una trayectoria de elección y colocación. en la página 473</a> |
|   | <i>Manual de referencia técnica - RAPID Overview</i>   |

### 3.55 pos - Posiciones (sólo X, Y y Z)

---

#### Utilización

`pos` se utiliza para posiciones (sólo para X, Y y Z).

El tipo de dato `robtarget` se utiliza con las posiciones del robot, incluida la orientación de la herramienta y la configuración de los ejes.

---

#### Descripción

Los datos de tipo `pos` describen las coordenadas de una posición: X, Y y Z.

---

#### Componentes

El tipo de dato `pos` contiene los componentes siguientes:

x

Tipo de dato: `num`  
El valor X de la posición.

y

Tipo de dato: `num`  
El valor Y de la posición.

z

Tipo de dato: `num`  
El valor Z de la posición.

---

#### Ejemplos básicos

Los siguientes ejemplos ilustran el tipo de dato `pos`:

##### Ejemplo 1

```
VAR pos pos1;  
...  
pos1 := [500, 0, 940];
```

Se asigna a la posición `pos1` el valor: X=500 mm, Y=0 mm, Z=940 mm.

##### Ejemplo 2

```
pos1.x := pos1.x + 50;
```

Se traslada la posición de `pos1` 50 mm en la dirección X.

---

#### Estructura

```
< dataobject of pos >  
< x of num >  
< y of num >  
< z of num >
```

*Continúa en la página siguiente*

### 3 Tipos de datos

---

3.55 pos - Posiciones (sólo X, Y y Z)

RobotWare - OS

Continuación

---

#### Información relacionada

| Para obtener más información sobre         | Consulte  |
|--|---|
| Operaciones con posiciones                 | <i>Manual de referencia técnica - RAPID Overview, sección Características básicas - Expresiones</i> |
| Posición del robot incluida su orientación | <a href="#">robtaret - Datos de posición en la página 1750</a>                                      |

## 3.56 pose - Transformaciones de coordenadas

### Utilización

`pose` se utiliza para cambiar de un sistema de coordenadas a otro.

### Descripción

Los datos de tipo `pose` describen cómo se desplaza y gira un sistema de coordenadas alrededor de otro sistema de coordenadas. Por ejemplo, estos datos pueden describir cómo está situado y orientado el sistema de coordenadas de la herramienta respecto del sistema de coordenadas de la muñeca.

### Componentes

El tipo de dato contiene los componentes siguientes:

#### `trans` (Pantallas)

*translation*

Tipo de dato: `pos`

El desplazamiento de posición (x, y, z) del sistema de coordenadas.

#### `rot`

*rotation*

Tipo de dato: `orient`

La rotación del sistema de coordenadas.

### Ejemplos básicos

Los siguientes ejemplos ilustran el tipo de dato `pose`:

```
VAR pose frame1;
...
frame1.trans := [50, 0, 40];
frame1.rot := [1, 0, 0, 0];
```

Se asigna a la transformación de coordenadas `frame1` un valor que corresponde a un desplazamiento en su posición, donde X=50 mm, Y=0 mm, Z=40 mm. Sin embargo, no hay ninguna rotación.

### Estructura

```
< dataobject of pose >
  < trans of pos >
  < rot of orient >
```

### Información relacionada

| Para obtener más información sobre | Consulte   |
|------------------------------------|--|
| ¿Qué es un cuaternio?              | <a href="#">orient - Orientación en la página 1716</a> |

## 3 Tipos de datos

---

### 3.57 processtimes - Tiempos de proceso *Continuous Application Platform (CAP)*

#### 3.57 processtimes - Tiempos de proceso

---

##### Utilización

`processtimes` se utiliza para definir los tiempos de duración para todas las fases de supervisión de estado en CAP, excepto la fase MAIN, que se define mediante el movimiento del robot (consulte la sección *Supervisión en Application manual - Continuous Application Platform*).

`processtimes` es un componente de `capdata` y define los tiempos límite para las siguientes fases de supervisión de estado en CAP:

- PRE\_START
- POST1
- POST2

El tiempo límite especificado tiene que ser mayor de cero, si debe utilizarse la supervisión durante la fase de supervisión de estado correspondiente en CAP (consulte la sección *Supervisión y fases de proceso en Application manual - Continuous Application Platform*).

---

##### Componentes

###### pre

Tipo de dato: `num`

Define la duración de la fase PRE\_START en segundos. Durante ese tiempo deben cumplirse todas las condiciones definidas para esa fase.

###### post1

Tipo de dato: `num`

Define la duración de la fase POST1 en segundos. Durante ese tiempo deben cumplirse todas las condiciones definidas para esa fase.

###### post2

Tipo de dato: `num`

Define la duración de la fase POST2 en segundos. Durante ese tiempo deben cumplirse todas las condiciones definidas para esa fase.

---

##### Sintaxis

```
< data object of processtimes >  
  < pre of num >  
  < post1 of num >  
  < post2 of num >
```

---

##### Información relacionada

|  | Descrito en:  |
|--|---|
| Tipo de dato <code>capdata</code>      | <a href="#">capdata - Datos CAP en la página 1624</a>       |
| <i>Continuous Application Platform</i> | <i>Application manual - Continuous Application Platform</i> |

---



## 3.58 progdisp - Desplazamiento de programa

### Utilización

`progdisp` se utiliza para almacenar el desplazamiento de programa actual de los ejes del robot y los ejes externos.

Normalmente no es necesario utilizar este tipo de dato dado que los datos se establecen con las instrucciones `PDispSet`, `PDispOn`, `PDispOff`, `EOffsSet`, `EOffsOn` y `EOffsOff`. Sólo se utiliza para almacenar temporalmente el valor actual para un uso posterior.

### Descripción

Los valores actuales del desplazamiento de programa están disponibles a través de la variable de sistema `C_PROGDISP`.

Para obtener más información, consulte las instrucciones `PDispSet`, `PDispOn`, `EOffsSet` y `EOffsOn`.

### Componentes

`pdisp`

*program displacement*

Tipo de dato: `pose`

El desplazamiento de programa del robot, expresado mediante una traslación y una orientación. La traslación se expresa en mm.

`eoffs`

*external offset*

Tipo de dato: `extjoint`

El offset de cada uno de los ejes externos. Si el eje es lineal, el valor se expresa en mm. Si es de rotación, el valor se expresa en grados.

### Ejemplos básicos

El siguiente ejemplo ilustra el tipo de dato `progdisp`:

#### Ejemplo 1

```
VAR progdisp progdispl;
...
SearchL sen1, psearch, p10, v100, tool1;
PDispOn \ExeP:=psearch, *, tool1;
EOffsOn \ExeP:=psearch, *;
...
progdispl:=C_PROGDISP;
PDispOff;
EOffsOff;
...
PDispSet progdispl.pdisp;
EOffsSet progdispl.eoffs;
```

*Continúa en la página siguiente*

## 3 Tipos de datos

---

### 3.58 progdisp - Desplazamiento de programa

RobotWare - OS

Continuación

En primer lugar, se activa un desplazamiento de programa desde una posición buscada. A continuación, los valores de desplazamiento de programa actuales se almacenan temporalmente en la variable `progdisp1` y el desplazamiento de programa se desactiva. Más adelante, la reactivación se realiza utilizando las instrucciones `PDispSet` y `EOffsSet`.

---

#### Datos predefinidos

La variable de sistema `C_PROGDISP` describe el desplazamiento de programa actual de los ejes del robot y los ejes externos y está siempre disponible desde el programa. Por otro lado, sólo puede modificarse mediante un conjunto de instrucciones, no mediante asignación.

Los siguientes valores predeterminados de desplazamiento de programa se establecen:

- cuando se utiliza el modo de reinicio **Restablecer RAPID**
- al cargar un nuevo programa o un nuevo módulo
- al iniciar la ejecución del programa desde el principio
- al mover el puntero del programa a `main`
- al mover el puntero del programa a una rutina
- al mover el puntero de programa de una forma que se pierde el orden de la ejecución.

```
VAR progdisp C_PROGDISP :=  
  [ [ [ 0, 0, 0], [1, 0, 0, 0]],-> posedata  
    [ 0, 0, 0, 0, 0, 0]];-> extjointdata
```

---

#### Estructura

```
< 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 >
```

Continúa en la página siguiente

#### Información relacionada

| Para obtener más información sobre                              | Consulte   |
|---|--|
| Instrucciones para la definición de desplazamientos de programa | <i>Manual de referencia técnica - RAPID Overview, sección Resumen sobre RAPID - Parámetros de movimiento</i>           |
| Sistemas de coordenadas   | <i>Manual de referencia técnica - RAPID Overview, sección Principios de movimiento y E/S - Sistemas de coordenadas</i> |

## 3 Tipos de datos

---

### 3.59 rawbytes - Datos sin formato

RobotWare - OS

### 3.59 rawbytes - Datos sin formato

---

#### Utilización

`rawbytes` se utiliza como un contenedor genérico de datos. Puede usarse para la comunicación con los dispositivos de E/S.

---

#### Descripción

Los datos `rawbytes` pueden contener cualquier tipo de dato (`num`, `byte`, `string`) mediante las instrucciones y funciones de soporte. En cualquier variable de tipo `rawbytes`, el sistema almacena también la longitud actual de los bytes válidos.

---

#### Ejemplos básicos

El siguiente ejemplo ilustra el tipo de dato `rawbytes`:

##### Ejemplo 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;
```

En este ejemplo, la variable `raw_data` del tipo `rawbytes` es borrada en primer lugar. Es decir, todos sus bytes cambian a 0 (el valor predeterminado tras la declaración). A continuación, se guarda el valor de `integer` en los 2 primeros bytes y el valor de `float` en los 4 bytes siguientes.

---

#### Limitaciones

Las variables de tipo `rawbytes` pueden contener de 0 a 1.024 bytes.

---

#### Estructura

`rawbytes` es un tipo de dato sin valor.

Tras la declaración de la variable de tipo `rawbytes`, todos los bytes de `rawbytes` quedan definidos como 0 y la longitud actual de los bytes válidos de la variable es también 0.

---

#### Información relacionada

| Para obtener más información sobre   | Consulte   |
|--|--|
| Obtención de la longitud de un dato <code>rawbytes</code>                    | <a href="#">RawBytesLen - Obtiene la longitud de un dato de tipo rawbytes en la página 1432</a>        |
| Borrado del contenido de un dato de tipo <code>rawbytes</code>               | <a href="#">ClearRawBytes - Borra el contenido de un dato de tipo rawbytes en la página 146</a>        |
| Copiado del contenido de un dato de tipo <code>rawbytes</code>               | <a href="#">CopyRawBytes - Copia el contenido de un dato de tipo rawbytes en la página 171</a>         |
| Empaquetamiento de un encabezado de DeviceNet en datos <code>rawbytes</code> | <a href="#">PackDNHeader - Empaqueta un encabezado de DeviceNet en datos rawbytes en la página 496</a> |

Continúa en la página siguiente

| Para obtener más información sobre              | Consulte  |
|---|---|
| Empaquetamiento de datos en datos rawbytes      | <a href="#">PackRawBytes - Empaqueta datos en un dato de tipo rawbytes en la página 499</a>       |
| Escritura de un dato rawbytes                   | <a href="#">WriteRawBytes - Escribe un dato de tipo rawbytes en la página 1111</a>                |
| Lectura de un dato rawbytes                     | <a href="#">ReadRawBytes - Lee datos de tipo rawbytes en la página 589</a>                        |
| Desempaquetamiento de datos de un dato rawbytes | <a href="#">UnpackRawBytes - Desempaqueta datos de un dato de tipo rawbytes en la página 1024</a> |
| Gestión de archivos y dispositivos de E/S       | <a href="#">Application manual - Controller software IRC5</a>                                     |

## 3 Tipos de datos

---

### 3.60 restartblkdata - bloque de datos para el reinicio *Continuous Application Platform (CAP)*

### 3.60 restartblkdata - bloque de datos para el reinicio

---

#### Utilización

restartblkdata se utiliza para definir el comportamiento de un proceso CAP en el reinicio.

restartblkdata es un componente de capdata y define lo siguiente para un proceso CAP en el reinicio, si:

- El robot debe ejecutar/bloquear la oscilación estacionaria durante el reinicio del proceso (weave\_start).
- El reinicio del movimiento del robot debe retardarse o no hay reinicio de proceso relativo (motion\_delay).
- La fase PRE y la fase PRE\_START deben ejecutarse/bloquearse (pre\_phase).
- Si debe utilizarse o no una velocidad diferente de la velocidad principal durante el inicio del proceso (startspeed\_phase).
- La fase START\_POST1 y la fase POST1 deben ejecutarse/bloquearse (post1\_phase).
- La fase START\_POST2 y la fase POST2 deben ejecutarse/bloquearse (post2\_phase).

---

#### Componentes

##### weave\_start

Tipo de dato: bool

| Valor | Descripción   |
|-------|---|
| FALSE | Oscilación estacionaria en el reinicio hasta que se inicia el proceso     |
| TRUE  | Sin oscilación estacionaria en el reinicio hasta que se inicia el proceso |

##### motion\_delay

Tipo de dato: bool

| Valor | Descripción   |
|-------|---|
| FALSE | Retardo del movimiento del robot en el reinicio tras iniciarse el proceso     |
| TRUE  | Sin retardo del movimiento del robot en el reinicio tras iniciarse el proceso |

##### pre\_phase

Tipo de dato: bool

| Valor | Descripción   |
|-------|---|
| FALSE | Ejecutar la fase PRE y la fase PRE_START en el reinicio     |
| TRUE  | NO ejecutar la fase PRE ni la fase PRE_START en el reinicio |

*Continúa en la página siguiente*

#### startspeed\_phase

Tipo de dato: bool

| Valor | Descripción   |
|-------|---|
| FALSE | Mover el robot con la velocidad de inicio al comienzo del reinicio  |
| TRUE  | NO mover el robot con la velocidad de inicio al comienzo del reinicio; utilizar directamente la velocidad principal |

#### post1\_phase

Tipo de dato: bool

| Valor | Descripción   |
|-------|---|
| FALSE | Ejecutar la fase START_POST1 y la fase POST1 en el reinicio     |
| TRUE  | NO ejecutar la fase START_POST1 ni la fase POST1 en el reinicio |

#### post2\_phase

Tipo de dato: bool

| Valor | Descripción   |
|-------|---|
| FALSE | Ejecutar la fase START_POST2 y la fase POST2 en el reinicio     |
| TRUE  | NO ejecutar la fase START_POST2 ni la fase POST2 en el reinicio |

#### Sintaxis

```
< 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 >
```

#### Información relacionada

|  | Descrito en:  |
|--|---|
| Tipo de dato capdata                   | <a href="#">capdata - Datos CAP en la página 1624</a>       |
| <i>Continuous Application Platform</i> | <i>Application manual - Continuous Application Platform</i> |

## 3 Tipos de datos

---

### 3.61 restartdata - Datos de reinicio de señales de disparo

RobotWare - OS

### 3.61 restartdata - Datos de reinicio de señales de disparo

---

#### Utilización

`restartdata` refleja los valores previos y posteriores de las señales de E/S especificadas (señales de proceso) en la secuencia de paro de los movimientos del robot. Las señales de E/S que deben supervisarse se especifican en la instrucción `TriggStopProc`.

`TriggStopProc` y `restartdata` han sido creadas para ser usadas en el reinicio tras un paro de programa (STOP) o un paro de emergencia (QSTOP) de las propias instrucciones de proceso definidas en RAPID (rutinas NOSTEPIN).

---

#### Definición

En la tabla se muestra la definición del momento de lectura de los valores previos y posteriores de las señales de I/O.

| Tipo de paro               | Tiempo de lectura del valor previo de la señal de E/S | Tiempo de lectura del valor posterior de la señal de E/S |
|----------------------------|---|--|
| STOP dentro de trayectoria | Cuando todos los ejes del robot están parados         | Aproximadamente 400 ms tras el tiempo previo             |
| QSTOP fuera de trayectoria | Lo antes posible                                      | Aproximadamente 400 ms tras el tiempo previo             |

---

#### Descripción

`restartdata` refleja los datos siguientes tras la detención de la ejecución del programa:

- Datos válidos de reinicio
  - Indicación de si el robot se ha detenido dentro o fuera de la trayectoria
  - Valor previo de las señales de E/S
  - Valor posterior de las señales de E/S
  - Número de flancos, entre el tiempo previo y el tiempo posterior, de la señal correspondiente al proceso en curso
- 

#### Componentes

`restartstop`

*valid restartdata after stop*

Tipo de dato: `bool`

TRUE = Refleja el último paro STOP o QSTOP.

FALSE = Datos de reinicio no válidos. Todos los valores de las señales de E/S cambian a -1.

`stoponpath`

*stop on path*

Tipo de dato: `bool`

TRUE = El robot se ha detenido dentro de la trayectoria (STOP).

FALSE = El robot se ha detenido, pero fuera de la trayectoria (QSTOP).

---

*Continúa en la página siguiente*



predo1val

***pre do1 value***

Tipo de dato: dionum

El valor previo de la señal digital "do1" especificada en el argumento DO1 de la instrucción TriggStopProc.

postdo1val

***post do1 value***

Tipo de dato: dionum

El valor posterior de la señal digital "do1" especificada en el argumento DO1 de la instrucción TriggStopProc.

prego1val

***pre go1 value***

Tipo de dato: num

El valor previo de la señal digital de grupo "go1" especificada en el argumento GO1 de la instrucción TriggStopProc.

postgo1val

***post go1 value***

Tipo de dato: num

El valor posterior de la señal digital de grupo "go1" especificada en el argumento GO1 de la instrucción TriggStopProc.

prego2val

***pre go2 value***

Tipo de dato: num

El valor previo de la señal digital de grupo "go2" especificada en el argumento GO2 de la instrucción TriggStopProc.

postgo2val

***post go2 value***

Tipo de dato: num

El valor posterior de la señal digital de grupo "go2" especificada en el argumento GO2 de la instrucción TriggStopProc.

prego3val

***pre go3 value***

Tipo de dato: num

El valor previo de la señal digital de grupo "go3" especificada en el argumento GO3 de la instrucción TriggStopProc.

postgo3val

***post go3 value***

Tipo de dato: num

*Continúa en la página siguiente*

### 3 Tipos de datos

---

#### 3.61 restartdata - Datos de reinicio de señales de disparo

RobotWare - OS

Continuación

El valor posterior de la señal digital de grupo "go3" especificada en el argumento G03 de la instrucción TriggStopProc.

prego4val

*pre go4 value*

Tipo de dato: num

El valor previo de la señal digital de grupo "go4" especificada en el argumento G04 de la instrucción TriggStopProc.

postgo4val

*post go4 value*

Tipo de dato: num

El valor posterior de la señal digital de grupo "go4" especificada en el argumento G04 de la instrucción TriggStopProc.

preshadowval

*pre shadow value*

Tipo de dato: dionum

El valor previo de la señal digital "shadow" especificada en el argumento ShadowDO de la instrucción TriggStopProc.

shadowflanks

*number of shadow flanks*

Tipo de dato: num

El número de transiciones de valor (flancos) de la señal digital "shadow" entre el tiempo previo y el tiempo posterior. La señal "shadow" se especifica en el argumento ShadowDO de la instrucción TriggStopProc.

postshadowval

*post shadow value*

Tipo de dato: dionum

El valor posterior de la señal digital "shadow" especificada en el argumento ShadowDO de la instrucción TriggStopProc.

---

#### Estructura

```
< dataobject of restartdata >
  < restartstop of bool >
  < stoponpath of bool >
  < predolval of dionum >
  < postdolval 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 >
```

Continúa en la página siguiente

```
< preshadowval of dionum >
< shadowflanks of dionum >
< postshadowval of dionum >
```

#### Información relacionada

| Para obtener más información sobre             | Consulte   |
|--|--|
| Instrucciones de procesos predefinidos         | <a href="#">TriggL - Movimiento lineal del robot con eventos en la página 940</a><br><a href="#">TriggC - Movimiento circular del robot con eventos en la página 895</a> |
| Configuración de reflejos de datos de reinicio | <a href="#">TriggStopProc - Genera datos de reinicio para las señales de disparo ante paros en la página 984</a>   |
| Retroceso por la trayectoria                   | <a href="#">StepBwdPath - Retrocede un paso a lo largo de la trayectoria en la página 811</a>  |
| <i>Advanced RAPID</i>                          | <i>Especificaciones del producto - Controller software IRC5</i>  |

## 3 Tipos de datos

---

### 3.62 rmqheader - Encabezado de mensaje de RAPID Message Queue *FlexPendant Interface, PC Interface, or Multitasking*

#### 3.62 rmqheader - Encabezado de mensaje de RAPID Message Queue

---

##### Utilización

`rmqheader` (*RAPID Message Queue Header*) se usa para leer la estructura de los datos contenidos en un mensaje de tipo `rmqmessage`.

---

##### Descripción

La parte de encabezado de un tipo de dato sin valor `rmqmessage` convertido al tipo de dato de valor `rmqheader`.

---

##### Componentes

`datatype`

Tipo de dato: `string`

El nombre del tipo de dato utilizado, por ejemplo `num`, `string` o algún otro tipo de dato con valor.

`ndim`

*Number of Dimensions*

Tipo de dato: `num`

Número de dimensiones de matriz.

`dim1`

*Size of first dimension*

Tipo de dato: `num`

El tamaño de la primera dimensión. 0 si no se usa.

`dim2`

*Size of second dimension*

Tipo de dato: `num`

El tamaño de la segunda dimensión. 0 si no se usa.

`dim3`

*Size of third dimension*

Tipo de dato: `num`

El tamaño de la tercera dimensión. 0 si no se usa.

---

##### Ejemplos

A continuación aparecen algunos ejemplos básicos del tipo de dato `rmqheader`.

##### Ejemplo 1

```
VAR rmqmessage message;  
VAR rmqheader header;  
...  
RMQGetMessage message;  
RMQGetMsgHeader message \Header:=header;
```

Copiar y convertir la información de `rmqheader` desde un mensaje `rmqmessage`.

*Continúa en la página siguiente*

---

### 3.62 rmqheader - Encabezado de mensaje de RAPID Message Queue FlexPendant Interface, PC Interface, or Multitasking Continuación

#### Estructura

```

<dataobject of rmqheader>
  <datatype of string>
  <ndim of num>
  <dim1 of num>
  <dim2 of num>
  <dim3 of num>

```

#### Información relacionada

| Para obtener más información sobre                     | Consulte  |
|--|---|
| Descripción de la funcionalidad de RAPID Message Queue | <i>Application manual - Controller software IRC5, sección RAPID Message Queue.</i>                        |
| Extraer los datos de encabezado de un rmqmessage       | <a href="#">RMQGetMsgHeader - Obtener información de encabezado de un mensaje de RMQ en la página 632</a> |
| RMQ Message  | <a href="#">rmqmessage - Mensaje de RAPID Message Queue en la página 1746</a>                             |

## 3 Tipos de datos

### 3.63 rmqmessage - Mensaje de RAPID Message Queue *FlexPendant Interface, PC Interface, or Multitasking*

### 3.63 rmqmessage - Mensaje de RAPID Message Queue

#### Utilización

`rmqmessage` (*RAPID Message Queue Message*) se usa para el almacenamiento temporal de los datos de comunicación.

#### Descripción

El tipo de dato `rmqmessage` es el mensaje utilizado para almacenar los datos durante la comunicación entre distintas tareas de RAPID o distintos clientes de Robot Application Builder con la funcionalidad RMQ. Contiene información acerca del tipo de datos enviados, las dimensiones de los datos, la identidad del remitente y los datos en sí.

Un `rmqmessage` es un tipo de dato de gran tamaño (de aproximadamente 3.000 bytes) y se recomienda reutilizar la variable para ahorrar memoria de RAPID.

#### Ejemplos básicos

El siguiente ejemplo ilustra el tipo de dato `rmqmessage`:

##### Ejemplo 1

```
VAR rmqmessage rmqmessage1;  
VAR string myreodata;  
...  
RMQGetMsgData rmqmessage1, myreodata;
```

La variable `rmqmessage1` se define y puede usarse en un comando RMQ (RAPID Message Queue). En este ejemplo, la parte de datos contenida en el `rmqmessage1` se copia a la variable `myreodata`.

#### Características

`rmqmessage` es un tipo de dato sin valor y no puede usarse en operaciones basadas en valores.

#### Información relacionada

| Para obtener más información sobre  | Consulte  |
|---|---|
| Descripción de la funcionalidad de RAPID Message Queue  | <a href="#">Application manual - Controller software IRC5, sección RAPID Message Queue.</a>               |
| RMQ Header  | <a href="#">rmqheader - Encabezado de mensaje de RAPID Message Queue en la página 1744</a>                |
| Extraer los datos de encabezado de un <code>rmqmessage</code>   | <a href="#">RMQGetMsgHeader - Obtener información de encabezado de un mensaje de RMQ en la página 632</a> |
| Ordenar y habilitar interrupciones para un tipo de dato en concreto   | <a href="#">IRMQMessage - Ordenar interrupciones de RMQ para un tipo de dato en la página 305</a>         |
| Obtener el primer mensaje de una cola de RAPID Message Queue.   | <a href="#">RMQGetMessage - Obtener un mensaje de RMQ en la página 626</a>                                |
| Enviar datos a la cola de una tarea de RAPID o un cliente de Robot Application Builder y esperar una respuesta del cliente. | <a href="#">RMQSendWait - Enviar un mensaje de datos de RMQ y esperar una respuesta en la página 642</a>  |

*Continúa en la página siguiente*

#### 3.63 rmqmessage - Mensaje de RAPID Message Queue *FlexPendant Interface, PC Interface, or Multitasking* Continuación

| Para obtener más información sobre              | Consulte  |
|---|---|
| Extraer los datos de un <code>rmqmessage</code> | <a href="#">RMQGetMsgData - Obtener la parte de datos de un mensaje de RMQ en la página 629</a> |

## 3 Tipos de datos

### 3.64 rmqslot - Número de identidad de un cliente de RMQ

*FlexPendant Interface, PC Interface, or Multitasking*

### 3.64 rmqslot - Número de identidad de un cliente de RMQ

#### Utilización

`rmqslot` (*RAPID Message Queue Slot*) se utiliza al comunicarse con un RMQ o un cliente de Robot Application Builder.

#### Descripción

El `rmqslot` es un número de identidad de una cola de RAPID Message Queue configurada para una tarea de RAPID o el número de identidad de un cliente de Robot Application Builder.

#### Ejemplos básicos

El siguiente ejemplo ilustra el tipo de dato `rmqslot`:

##### Ejemplo 1

```
VAR rmqslot rmqslot1;  
RMQFindSlot rmqslot1, "RMQ_T_ROB1";  
...
```

La variable `rmqslot1` se define y puede usarse en la instrucción `RMQFindSlot` para obtener el número de identidad de la cola de RAPID Message Queue "RMQ\_T\_ROB1" configurada para la tarea de RAPID "T\_ROB1".

#### Características

`rmqslot` es un tipo de dato sin valor y no puede usarse en operaciones basadas en valores.

#### Información relacionada

| Para obtener más información sobre   | Consulte   |
|--|--|
| Descripción de la funcionalidad de RAPID Message Queue   | <i>Application manual - Controller software IRC5</i> , sección <i>RAPID Message Queue</i> .              |
| Encontrar el número de identidad de una tarea de RAPID Message Queue o de un cliente de Robot Application Builder. | <a href="#">RMQFindSlot - Buscar una identidad de ranura para el nombre de ranura en la página 624</a>   |
| Enviar datos a la cola de una tarea de RAPID o de un cliente de Robot Application Builder.                         | <a href="#">RMQSendMessage - Enviar un mensaje de datos de RMQ en la página 638</a>                      |
| Enviar datos a un cliente y esperar una respuesta del cliente.   | <a href="#">RMQSendWait - Enviar un mensaje de datos de RMQ y esperar una respuesta en la página 642</a> |
| Obtener el nombre de ranura de una identidad de ranura especificada.   | <a href="#">RMQGetSlotName - Obtener el nombre de un cliente de RMQ en la página 1456</a>                |



### 3.65 robjoint - Posición de eje de los ejes del robot

#### Utilización

`robjoint` se utiliza para definir la posición en grados de los ejes del robot.

#### Descripción

Los datos de tipo `robjoint` se utilizan para almacenar posiciones de eje en grados para los ejes de robot del 1 al 6. La posición de un eje se define como la rotación en grados del eje (brazo) correspondiente, en sentido positivo o negativo respecto de la posición de calibración del eje.

#### Componentes

`rax_1`

*robot axis 1*

Tipo de dato: `num`

La posición del eje 1 del robot, en grados, respecto de la posición de calibración.

...

`rax_6`

*robot axis 6*

Tipo de dato: `num`

La posición del eje 6 del robot, en grados, respecto de la posición de calibración.

#### Estructura

```
< 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 >
```

#### Información relacionada

| Para obtener más información sobre        | Consulte   |
|---|--|
| Datos de posición de ejes                 | <a href="#">jointtarget - Datos de posición de eje en la página 1691</a>                   |
| Desplazamiento hacia una posición de ejes | <a href="#">MoveAbsJ - Mueve el robot a una posición de ejes absoluta en la página 382</a> |

## 3 Tipos de datos

---

3.66 robtarget - Datos de posición  
RobotWare - OS

### 3.66 robtarget - Datos de posición

---

#### Utilización

`robtarget` (*robot target*) se utiliza para definir la posición del robot y de los ejes adicionales.

#### Descripción

Los datos de posición se utilizan en las instrucciones de movimiento para indicar la posición hacia la que deben desplazarse los ejes del robot y los ejes adicionales. Debido a que el robot puede alcanzar una misma posición con métodos diferentes, también se especifica la configuración de los ejes. De esta forma, se definen los valores de los ejes si por algún motivo resultan ambiguos, por ejemplo en los casos siguientes:

- Si el robot se encuentra en una posición avanzada o retrasada
- Si el eje 4 está orientado hacia abajo o hacia arriba
- Si el eje 6 se encuentra en una revolución negativa o positiva



#### ¡AVISO!

La posición se define partiendo del sistema de coordenadas del objeto de trabajo, incluidos los posibles desplazamientos de programa. Si la posición se programa con un objeto de trabajo distinto del utilizado en la instrucción, el robot no se moverá de la forma esperada. Asegúrese de usar el mismo objeto de trabajo que el utilizado al programar las instrucciones de movimiento. Un uso incorrecto puede causar accidentes o daños en el robot o en otros equipos.

#### Componentes

`trans`

##### *translation*

Tipo de dato: `pos`

La posición (x, y, z) del punto central de la herramienta, expresado en mm.

La posición se especifica respecto del sistema de coordenadas del objeto actual, incluido el desplazamiento de programa. Si no se ha especificado ningún objeto de trabajo, se utiliza el sistema de coordenadas mundo.

`rot`

##### *rotation*

Tipo de dato: `orient`

La orientación de la herramienta, expresada en forma de un cuaternio (q1, q2, q3 y q4).

La orientación se especifica respecto del sistema de coordenadas del objeto actual, incluido el desplazamiento de programa. Si no se ha especificado ningún objeto de trabajo, se utiliza el sistema de coordenadas mundo.

*Continúa en la página siguiente*

robconf

### *robot configuration*

Tipo de dato: `confdata`

La configuración de ejes del robot (`cf1`, `cf4`, `cf6` y `cfx`). Esto se define en forma del cuarto de revolución actual de los ejes 1, 4 y 6. El primer cuarto de revolución positivo, de 0 a 90° se define como 0. El significado del componente `cfx` depende del tipo de robot.

Para obtener más información, consulte el tipo de dato `confdata`.

extax

### *external axes*

Tipo de dato: `extjoint`

La posición del eje adicional.

La posición se define de la forma siguiente para cada eje independiente (`eax_a`, `eax_b`...`eax_f`):

- Para los ejes de rotación, la posición se define como la rotación en grados de la posición de calibración.
- Para los ejes lineales, la posición se define como la distancia en mm existente respecto de la posición de calibración.

Los ejes adicionales `eax_a` ... son ejes lógicos. La relación existente entre el número lógico del eje y el número físico del eje se define en los parámetros del sistema.

El valor 9E9 se utiliza para definir los ejes que no están conectados. Si los ejes definidos en los datos de posición son distintos de los ejes que están conectados realmente en el momento de la ejecución del programa, ocurre lo siguiente:

- Si la posición no está definida en los datos de posición (su valor es 9E9), el valor no se tendrá en cuenta si el eje está conectado pero no activado. Sin embargo, si el eje está activado, se genera un error.
- Si la posición está definida en los datos de posición, a pesar de que el eje no está conectado, el valor no se tiene en cuenta.

No se realiza ningún movimiento pero tampoco se genera ningún error en el caso de los ejes que tengan datos de posición válidos pero que no estén activados.

Si algún eje adicional está funcionando en el modo independiente y el robot y sus ejes adicionales deben realizar algún nuevo movimiento, los datos de posición de los ejes adicionales en el modo independiente no deben ser 9E9. Los datos deben ser un valor arbitrario que no esté siendo utilizado por el sistema.

---

## Ejemplos básicos

Los siguientes ejemplos ilustran el tipo de dato `robtargt`:

### Ejemplo 1

```
CONST robtargt p15 := [ [600, 500, 225.3], [1, 0, 0, 0], [1, 1, 0, 0], [ 11, 12.3, 9E9, 9E9, 9E9, 9E9 ] ];
```

*Continúa en la página siguiente*

### 3 Tipos de datos

---

#### 3.66 robtarget - Datos de posición

RobotWare - OS

Continuación

Se define la posición p15 de la forma siguiente:

- La posición del robot:  $x = 600$ ,  $y = 500$  y  $z = 225,3$  mm en el sistema de coordenadas de objeto.
- La orientación de la herramienta en la misma dirección que el sistema de coordenadas del objeto.
- La configuración de ejes del robot es la siguiente: ejes 1 y 4 en la posición de  $90$  a  $180^\circ$ , eje 6 en la posición de  $0$  a  $90^\circ$ .
- La posición de los ejes adicionales lógicos a y b, expresada en grados o mm (en función del tipo de eje). Los ejes de c a f permanecen sin definir.

#### Ejemplo 2

```
VAR robtarget p20;  
...  
p20 := CRobT(\Tool:=tool\wobj:=wobj0);  
p20 := Offs(p20,10,0,0);
```

Se establece la posición p20 en la misma posición que la posición actual del robot, mediante una llamada a la función CRobT. A continuación, se mueve la posición 10 mm en la dirección x.

---

#### Estructura

```
< 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 >  
  < 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 >
```

---

#### Información relacionada

| Para obtener más información sobre | Consulte   |
|------------------------------------|--|
| Instrucciones de movimiento        | <i>Manual de referencia técnica - RAPID Overview, sección Resumen sobre RAPID - Movimiento</i> |

Continúa en la página siguiente

| Para obtener más información sobre    | Consulte   |
|---------------------------------------|--|
| Sistemas de coordenadas               | <i>Manual de referencia técnica - RAPID Overview, sección Principios de movimiento y E/S - Sistemas de coordenadas</i> |
| Manejo de datos de configuración      | <i>Manual de referencia técnica - RAPID Overview, sección Principios de movimiento y E/S - Configuración del robot</i> |
| Configuración de los ejes adicionales | <i>Application manual - Additional axes and stand alone controller</i>   |
| ¿Qué es un cuaternio?                 | <a href="#">orient - Orientación en la página 1716</a>   |

## 3 Tipos de datos

---

### 3.67 sensor - Descriptor de dispositivo externo *Robor Reference Interface*

### 3.67 sensor - Descriptor de dispositivo externo

---

#### Utilización

`sensor` es un descriptor del dispositivo externo al que se desea establecer la conexión.

---

#### Descripción

El descriptor de un dispositivo en el nivel RAPID está encapsulado en el tipo de dato de registro `sensor`. Contiene información acerca del dispositivo de sensor, por ejemplo ID, código de error y estado de comunicación del sensor.

---

#### Componentes

`id`

Tipo de dato: `num`

El identificador interno del dispositivo, que se establecerá en la primera operación con el dispositivo desde el nivel RAPID (aún no implementado).

`error`

Tipo de dato: `num`

El parámetro `error` se establece cuando se cambia el valor del parámetro `state` a `STATE_ERROR`. Cuando `state` pasa de `STATE_ERROR` a `STATE_CONNECTED`, el valor del parámetro `error` cambia a 0.

| Número de error | Error   |
|-----------------|---|
| 0               | Sin errores.  |
| 112600          | Fallo de inicialización de la interfaz de comunicación. |
| 112602          | Error de interfaz de comunicación.                      |

`state`

Tipo de dato: `sensorstate`

Refleja el estado de comunicación actual del dispositivo.

---

#### Ejemplos

A continuación se muestra un ejemplo del tipo de dato `sensor`.

#### Ejemplo 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;
```

*Continúa en la página siguiente*

---

ENDPROC

Al llamar a la rutina `RRI_Open`, en primer lugar se abre una conexión al dispositivo `AnyDevice`. A continuación, se inicia una transmisión cíclica con una frecuencia de `SampleRate`.

#### Estructura

```
<dataobject of sensor>
  <id of num>
  <error of num>
  <state of sensorstate>
```

#### Información relacionada

| Para obtener más información sobre                    | Consulte  |
|---|---|
| Establecimiento de una conexión a un sistema externo. | <a href="#">SiConnect - Conexión a Sensor Interface en la página 722.</a>                           |
| Cierre de una conexión a un sistema externo.          | <a href="#">SiClose - Cierre de Sensor Interface en la página 725.</a>                              |
| Registro de datos para transmisión cíclica.           | <a href="#">SiSetCyclic - Establecimiento de modo cíclico de Sensor Interface en la página 733.</a> |
| Suscripción a la transmisión cíclica de datos.        | <a href="#">SiGetCyclic - Obtención de modo cíclico de Sensor Interface en la página 727</a>        |
| Estado de comunicación de un dispositivo.             | <a href="#">sensorstate - Estado de comunicación del dispositivo en la página 1756.</a>             |
| <i>Robot Reference Interface</i>                      | <i>Application manual - Controller software IRC5</i>  |

### 3 Tipos de datos

#### 3.68 sensorstate - Estado de comunicación del dispositivo

*Robot Reference Interface*

#### 3.68 sensorstate - Estado de comunicación del dispositivo

##### Utilización

`sensorstate` se utiliza para representar un estado de comunicación actual de un dispositivo.

##### Descripción

Se utiliza una constante `sensorstate` para reflejar el estado de comunicación actual de un dispositivo. Puede usarse desde RAPID para evaluar el estado de la conexión con el sensor.

##### Datos predefinidos

Se han predefinido las siguientes constantes simbólicas del tipo de dato `sensorstate`, que pueden usarse para evaluar en qué estado de comunicación se encuentra el dispositivo.

| Constante       | Valor |
|-----------------|-------|
| STATE_ERROR     | -1    |
| STATE_UNDEFINED | 0     |
| STATE_CONNECTED | 1     |
| STATE_OPERATING | 2     |
| STATE_CLOSED    | 3     |

##### Características

`sensorstate` es un tipo de dato de alias de `num` y por tanto hereda sus características.

##### Información relacionada

| Para obtener más información sobre                    | Consulte  |
|---|---|
| Establecimiento de una conexión a un sistema externo. | <a href="#">SiConnect - Conexión a Sensor Interface en la página 722.</a>                           |
| Cierre de una conexión a un sistema externo.          | <a href="#">SiClose - Cierre de Sensor Interface en la página 725.</a>                              |
| Registro de datos para transmisión cíclica.           | <a href="#">SiSetCyclic - Establecimiento de modo cíclico de Sensor Interface en la página 733.</a> |
| Suscripción a la transmisión cíclica de datos.        | <a href="#">SiGetCyclic - Obtención de modo cíclico de Sensor Interface en la página 727</a>        |
| Descriptor del dispositivo externo.                   | <a href="#">sensor - Descriptor de dispositivo externo en la página 1754.</a>                       |
| <i>Robot Reference Interface</i>                      | <i>Application manual - Controller software IRC5</i>  |



**3.69 sensorvardata - Configuración de múltiples variables de datos para la interfaz de sensores****Utilización**

`sensorvardata` Se utiliza para configurar la información necesaria para los diferentes puntos de datos que son gestionados por los comandos `ReadVarArr` y `WriteVarArr`.

**Componentes**

El tipo de dato contiene los componentes siguientes:

`varnumber`

Tipo de dato: `num`

Define el número de variables que se leerán/escribirán.

`sensordatatype`

Tipo de dato: `num`

El tipo de datos que representarán a dicho valor. El valor se convertirá a un tipo concreto antes de ser enviado o tras ser recibido por el enlace de comunicación con el sensor.

Los valores válidos son:

| Valor | Descripción |
|-------|-------------|
| 0     | INT16       |
| 1     | UINT16      |
| 2     | DOUBLE      |

Cuando se utilice un sensor de ServoRobot® a través de un enlace EtherNet, se utilizará siempre `sensordatatype 2` (doble). Para el resto de sensores y comunicaciones, se usará `sensordatatype 0 o 1`

`raw`

Tipo de dato: `bool`

Si este indicador es verdadero no se realizará ninguna modificación interna de los datos antes ni después de que se envíen a través del protocolo a/desde el dispositivo. El valor enviado/recibido tendrá la misma representación de bits que en el enlace de comunicación.

`raw` No está disponible para `sensordatatype 2`.

`scale`

Tipo de dato: `num`

Configura el factor de escala para escalar valores de datos. Los valores válidos son 1, 10 o 100. Un valor enviado a un dispositivo por medio de `WriteVarArr` se multiplicará por el factor de escala antes de ser enviado y un valor obtenido de un dispositivo por medio de `ReadVarArr` se dividirá por el factor de escala antes de ser devuelto.

`scale` No está disponible para `sensordatatype 2`.

*Continúa en la página siguiente*

### 3 Tipos de datos

---

#### 3.69 sensorvardata - Configuración de múltiples variables de datos para la interfaz de sensores

##### *Sensor Interface*

##### *Continuación*

value

Tipo de dato: dnum

El valor que habrá que leer/escribir en el dispositivo.

---

#### Estructura

```
< data object of sensorvardata >  
  < varnumber of num >  
  < sensordatatype of num >  
  < raw of bool >  
  < scale of num >  
  < value of dnum >
```

---

#### Información relacionada

| Para obtener más información sobre             | Consulte   |
|--|--|
| Leer múltiples variables de un dispositivo     | <a href="#">ReadVarArr - Lee múltiples variables de un dispositivo sensor en la página 592</a>       |
| Escribir múltiples variables en un dispositivo | <a href="#">WriteVarArr - Escribe múltiples variables en un dispositivo sensor en la página 1119</a> |
| Configuración de la comunicación del sensor    | <i>Manual de referencia técnica - RAPID Overview</i>   |
| Configuración de la comunicación del sensor    | <i>Manual de referencia técnica - Parámetros del sistema</i>   |

## 3.70 shapedata - Datos de forma de zonas mundo

### Utilización

shapedata se utiliza para describir la geometría de una zona mundo.

### Descripción

Es posible definir zonas mundo con 4 formas geométricas diferentes.

- Un prisma cuyos lados son paralelos al sistema de coordenadas mundo y que se definen mediante una instrucción `WZBoxDef`
- Una esfera, definida mediante una instrucción `WZSphDef`
- Un cilindro paralelo al eje z del sistema de coordenadas mundo y definido mediante una instrucción `WZCylDef`
- Un área de espacio de ejes para los ejes del robot y/o los externos, definidos por la instrucción `WZHomeJointDef` o `WZLimJointDef`

La geometría de una zona mundo se define mediante una de las instrucciones indicadas anteriormente y la acción de una zona mundo definida mediante la instrucción `WZLimSup` o `WZDOSet`.

### Ejemplos básicos

El siguiente ejemplo ilustra el tipo de dato shapedata:

#### Ejemplo 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
```

Una `conveyor` se define como un prisma y se activa la supervisión de esta área. Se define un `pole` como un cilindro y se activa la supervisión de la zona. Si el robot alcanza una de estas áreas, se detiene el movimiento.

### Características

shapedata es un tipo de dato sin valor.

### Información relacionada

| Para obtener más información sobre           | Consulte   |
|--|--|
| World Zones                                  | <i>Manual de referencia técnica - RAPID Overview, sección Resumen sobre RAPID - Parámetros de movimiento</i> |
| Definición de zonas mundo en forma de prisma | <a href="#">WZBoxDef - Define una zona mundo con forma de prisma en la página 1121</a>                       |

*Continúa en la página siguiente*

### 3 Tipos de datos

---

#### 3.70 shapedata - Datos de forma de zonas mundo

##### *World Zones*

##### *Continuación*

| <b>Para obtener más información sobre</b>                              | <b>Consulte</b>  |
|--|--|
| Definición de zonas mundo esféricas                                    | <a href="#">WZSphDef - Define una zona mundo con forma esférica en la página 1148</a>                              |
| Definición de zonas mundo cilíndricas                                  | <a href="#">WZCylDef - Define una zona mundo con forma cilíndrica en la página 1123</a>                            |
| Definición de una zona mundo para las posiciones iniciales de los ejes | <a href="#">WZHomeJointDef - Define una zona mundo para las posiciones iniciales de los ejes en la página 1137</a> |
| Definición de una zona mundo para las posiciones límite de los ejes    | <a href="#">WZLimJointDef - Define una zona mundo para la limitación de los ejes en la página 1141</a>             |
| Activación de la supervisión de límites de las zonas mundo             | <a href="#">WZLimSup - Activa la supervisión de límites de las zonas mundo en la página 1145</a>                   |
| Activación de salidas digitales basadas en zonas mundo                 | <a href="#">WZDSet - Activación de salidas digitales basadas en zonas mundo en la página 1128</a>                  |

### 3.71 signalorigin - Describe el origen de la señal de E/S

#### Utilización

signalorigin se utiliza para representar un entero con una constante simbólica.

#### Descripción

Las constantes simbólicas predefinidas del tipo signalorigin pueden utilizarse para comprobar el origen de la señal de E/S. Se han diseñado para usarlas al comprobar el valor de retorno de la función GetSignalOrigin.

#### Ejemplos básicos

El siguiente ejemplo ilustra el tipo de dato signalorigin:

##### Ejemplo 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

```

El origen de la señal se almacenará en la variable sigorig.

#### Datos predefinidos

Las siguientes constantes de tipo signalorigin están predefinidas:

| Valor de retorno | Constante simbólica | Comentario   |
|------------------|---------------------|--|
| 0                | SIGORIG_NONE        | La variable de señal de E/S es declarada en RAPID y no tiene ningún alias asociado.  |
| 1                | SIGORIG_CFG         | La señal se configura en la configuración de E/S.  |
| 2                | SIGORIG_ALIAS       | La variable de señal de E/S es declarada en RAPID y tiene un alias asociado a una señal de E/S configurada en la configuración de E/S. |

#### Características

signalorigin es un tipo de dato de alias de num y por tanto hereda sus propiedades.

*Continúa en la página siguiente*

### 3 Tipos de datos

---

3.71 signalorigin - Describe el origen de la señal de E/S

RobotWare - OS

Continuación

---

#### Información relacionada

| Para obtener más información sobre                             | Consulte   |
|--|--|
| Obtención de información acerca del origen de una señal de E/S | <a href="#">GetSignalOrigin - Obtención de información acerca del origen de una señal de E/S en la página 1314</a> |

## 3.72 signalxx - Señales digitales y analógicas

## Utilización

Los tipos de datos denominados como `signalxx` se utilizan con las señales digitales y analógicas de entrada y salida.

Los nombres de las señales se definen en los parámetros del sistema y, por tanto, no es necesario definirlos en el programa.

## Descripción

| Tipo de dato          | Se usa para                            |
|-----------------------|--|
| <code>signalai</code> | Señales analógicas de entrada          |
| <code>signalao</code> | Señales analógicas de salida           |
| <code>signaldi</code> | Señales digitales de entrada           |
| <code>signaldo</code> | Señales digitales de salida            |
| <code>signalgi</code> | Grupos de señales digitales de entrada |
| <code>signalgo</code> | Grupos de señales digitales de salida  |

Las variables del tipo `signalxo` sólo contienen una referencia a la señal. El valor se establece mediante las instrucciones `SetDO`, `SetGO` y `SetAO`.

Las variables del tipo `signalxi` contienen una referencia a una señal, así como la posibilidad de obtener directamente el valor desde el programa, si se utiliza en un contexto de valor.

El valor de una señal de entrada puede leerse directamente desde el programa, como en los ejemplos siguientes:

```
! Digital input
IF dil = 1 THEN ...

! Digital group input
IF gil = 5 THEN ...

! Analog input
IF ail > 5.2 THEN ...
```

También puede usarse en asignaciones, por ejemplo:

```
VAR num current_value;

! Digital input
current_value := dil;

! Digital group input
current_value := gil;

! Analog input
current_value := ail;
```

*Continúa en la página siguiente*

## 3 Tipos de datos

### 3.72 signalxx - Señales digitales y analógicas

RobotWare - OS

Continuación

#### Limitaciones

No debe definir el tipo de dato `signalxx` en el programa. Sin embargo, si se llega a hacer, aparecerá un mensaje de error tan pronto como se ejecute una instrucción o una función que haga referencia a esta señal. Sin embargo, sí es posible utilizarlos como parámetros al declarar una rutina.

#### Datos predefinidos

Las señales definidas en los parámetros del sistema están siempre disponibles desde el programa mediante las variables de señal predefinidas (datos instalados). Sin embargo, debe tener en cuenta que si se define otro dato con el mismo nombre, se pierde la posibilidad de usar la señal.

#### Características

`signalxx` es un tipo de dato de semivalor que permite operaciones orientadas a valores.

#### Gestión de errores

Se generan los siguientes errores recuperables, que pueden gestionarse con un gestor de errores. La variable del sistema `ERRNO` cambiará a:

| Nombre                          | Causa del error   |
|---------------------------------|---|
| <code>ERR_NO_ALIASIO_DEF</code> | La variable de señal es una variable declarada en <code>RAPID</code> . No se ha conectado a una señal E/S definida en la configuración de E/S con la instrucción <code>AliasIO</code> . |
| <code>ERR_NORUNUNIT</code>      | Se ha perdido el contacto con el dispositivo de E/S.  |

#### Información relacionada

| Para obtener más información sobre              | Consulte   |
|---|--|
| Resumen de instrucciones de entrada y salida    | <i>Manual de referencia técnica - RAPID Overview, sección Resumen sobre RAPID - Señales de entrada y salida</i>  |
| Funcionalidad de entrada/salida en general      | <i>Manual de referencia técnica - RAPID Overview, sección Principios de movimiento y E/S - Principios de E/S</i> |
| Configuración de E/S                            | <i>Manual de referencia técnica - Parámetros del sistema</i>   |
| Características de los tipos de datos sin valor | <i>Manual de referencia técnica - RAPID Overview, sección Características básicas - Tipos de datos</i>           |



### 3.73 socketdev - Dispositivo de zócalo

#### Utilización

socketdev (*socket device*) se usa para comunicarse con otros ordenadores en una red o entre tareas de RAPID.

#### Descripción

El dispositivo de zócalo es el manejador de un enlace de comunicaciones con otro ordenador de una red.

#### Ejemplos básicos

El siguiente ejemplo ilustra el tipo de dato socketdev:

##### Ejemplo 1

```
VAR socketdev socket1;
```

Se define la variable `socket1`, que puede usarse en un comando de zócalo, por ejemplo `SocketCreate`.

#### Limitaciones

Es posible declarar tantos zócalos como se desee, pero sólo es posible utilizar 32 de ellos al mismo tiempo.

#### Características

socketdev es un tipo de dato sin valor.

#### Información relacionada

| Para obtener más información sobre              | Consulte   |
|---|--|
| Comunicación con zócalos en general             | <i>Application manual - Controller software IRC5</i>   |
| Creación de un nuevo zócalo                     | <a href="#">SocketCreate - Crea un nuevo zócalo en la página 747</a>                                   |
| Características de los tipos de datos sin valor | <i>Manual de referencia técnica - RAPID Overview, sección Características básicas - Tipos de datos</i> |

## 3 Tipos de datos

### 3.74 socketstatus - Estado de comunicación de zócalo

#### Socket Messaging

### 3.74 socketstatus - Estado de comunicación de zócalo

#### Utilización

`socketstatus` se usa para representar el estado de la comunicación con zócalos.

#### Descripción

El estado del zócalo se captura con la función `SocketGetStatus` y puede usarse en tareas como el control del flujo de un programa o la depuración.

#### Ejemplos básicos

El siguiente ejemplo ilustra el tipo de dato `socketstatus`:

##### Ejemplo 1

```
VAR socketdev socket1;  
VAR socketstatus state;  
...  
SocketCreate socket1;  
state := SocketGetStatus( socket1 );
```

El estado de zócalo `SOCKET_CREATED` se almacena en la variable `state`.

#### Datos predefinidos

Las siguientes constantes de tipo `socketstatus` están predefinidas:

| Constante de RAPID            | Valor | Estado de zócalo...                                   |
|-------------------------------|-------|---|
| <code>SOCKET_CREATED</code>   | 1     | Creado  |
| <code>SOCKET_CONNECTED</code> | 2     | Cliente conectado a un host remoto                    |
| <code>SOCKET_BOUND</code>     | 3     | Servidor enlazado a una dirección y un puerto locales |
| <code>SOCKET_LISTENING</code> | 4     | Servidor a la escucha de conexiones entrantes         |
| <code>SOCKET_CLOSED</code>    | 5     | Cerrado   |

#### Características

`socketstatus` es un tipo de dato de alias de `num` y por tanto hereda sus características.

#### Información relacionada

| Para obtener más información sobre                 | Consulte   |
|--|--|
| Comunicación con zócalos en general                | <i>Application manual - Controller software IRC5</i>   |
| Obtención del estado de zócalo                     | <a href="#">SocketGetStatus - Obtiene el estado actual de un zócalo en la página 1473</a>              |
| Tipos de datos en general, tipos de datos de alias | <i>Manual de referencia técnica - RAPID Overview, sección Características básicas - Tipos de datos</i> |

### 3.75 speeddata - Datos de velocidad

---

#### Utilización

`speeddata` se utiliza para especificar la velocidad a la que deben moverse los ejes, tanto los del robot como los ejes externos.

---

#### Description

Los datos de velocidad se utilizan para definir las velocidades siguientes:

- Velocidad a la que se mueve el punto central de la herramienta
- Velocidad de reorientación de la herramienta
- Velocidad a la que se mueven los ejes lineales o de rotación.

Cuando se combinan varios tipos de movimiento, una de las velocidades suele limitar todos los movimientos. La velocidad de los demás movimientos se reduce de forma que todos los movimientos terminen de ejecutarse al mismo tiempo.

La velocidad también está limitada por el rendimiento del robot. Este rendimiento es distinto según el tipo de robot y la trayectoria del movimiento.

---

#### Componentes

`v_tcp`

*velocity tcp*

Tipo de dato: `num`

La velocidad del punto central de la herramienta (TCP) en mm/s.

Si se utiliza una herramienta estacionaria o ejes externos coordinados, la velocidad se especifica respecto del objeto de trabajo.

`v_ori`

*velocity orientation*

Tipo de dato: `num`

La velocidad de reorientación alrededor del TCP, expresada en grados/s.

Si se utiliza una herramienta estacionaria o ejes externos coordinados, la velocidad se especifica respecto del objeto de trabajo.

`v_leax`

*velocity linear external axes*

Tipo de dato: `num`

La velocidad de los ejes externos lineales, en mm/s.

`v_reax`

*velocity rotational external axes*

Tipo de dato: `num`

La velocidad de los ejes externos de rotación, en grados/s.

*Continúa en la página siguiente*

### 3 Tipos de datos

#### 3.75 speeddata - Datos de velocidad

RobotWare - OS

Continuación

#### Ejemplos básicos

El siguiente ejemplo ilustra el tipo de dato `speeddata`:

##### Ejemplo 1

```
VAR speeddata vmedium := [ 1000, 30, 200, 15 ];
```

Se definen los datos de velocidad `vmedium` con las velocidades siguientes:

- 1.000 mm/s para el TCP.
- 30 grados/s para la reorientación de la herramienta.
- 200 mm/s para los ejes externos lineales.
- 15 grados/s para los ejes externos de rotación.

```
vmedium.v_tcp := 900;
```

Se cambia la velocidad del TCP a 900 mm/s.

#### Limitaciones

Con un movimiento muy lento, cada movimiento debe ser lo suficientemente corto para generar un tiempo de interpolación inferior a los 240 segundos.

#### Datos predefinidos

Existen varios datos de velocidad ya definidos en el sistema.

Datos de velocidad predefinidos para su uso en los movimientos del robot y de los ejes externos:

| Nombre | Velocidad del TCP | Orientación | Eje externo lineal | Eje externo de rotación |
|--------|-------------------|-------------|--------------------|-------------------------|
| 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                 |

Continúa en la página siguiente

| Nombre | Velocidad del TCP | Orientación | Eje externo lineal | Eje externo de rotación |
|--------|-------------------|-------------|--------------------|-------------------------|
| 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   | i                 | ii          | iii                | iv                      |

- i Velocidad de TCP máx. para el tipo de robot utilizado y valores de TCP prácticos normales, especificados por el parámetro de sistema *TCP Linear Max Speed (m/s)*. La función de RAPID `MaxRobSpeed` devuelve este valor. Si se utilizaran valores de TCP extremadamente altos en la base de coordenadas de la herramienta, puede crear su propio `speeddata` con una velocidad de TCP mayor que la devuelta por `MaxRobSpeed` y usar `VelSet` para permitir mayor velocidad.
- ii Velocidad de reorientación máx. para el tipo de robot usado, especificada por el parámetro de sistema *TCP Reorient Max Speed (deg/s)*. La función de RAPID `MaxRobReorientSpeed` devuelve este valor.
- iii Velocidad lineal máx. para ejes adicionales, especificada por el parámetro de sistema *Ext. Axis Linear Max Speed (m/s)*. La función de RAPID `MaxExtLinearSpeed` devuelve este valor.
- iv Velocidad de giro máx. para ejes adicionales, especificada por el parámetro de sistema *Ext. Axis Rotational Max Speed (deg/s)*. La función de RAPID `MaxExtReorientSpeed` devuelve este valor.

Datos de velocidad predefinidos para su uso en los movimientos de ejes externos de rotación con la instrucción `MoveExtJ`.

| Nombre  | Velocidad del TCP | Orientación | Eje externo lineal | Eje externo de rotación |
|---------|-------------------|-------------|--------------------|-------------------------|
| 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                  |

Datos de velocidad predefinidos para su uso en los movimientos de ejes externos lineales con la instrucción `MoveExtJ`.

| Nombre   | Velocidad del TCP | Orientación | Eje externo lineal | Eje externo de rotación |
|----------|-------------------|-------------|--------------------|-------------------------|
| 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                    |

Continúa en la página siguiente

### 3 Tipos de datos

---

#### 3.75 speeddata - Datos de velocidad

RobotWare - OS

Continuación

---

#### Estructura

```
< dataobject of speeddata >  
  < v_tcp of num >  
  < v_ori of num >  
  < v_leax of num >  
  < v_reax of num >
```

---

#### Información relacionada

| Para obtener más información sobre            | Consulte   |
|---|--|
| Instrucciones de posicionamiento              | <i>Manual de referencia técnica - RAPID Overview, sección Resumen sobre RAPID - Movimiento</i>   |
| Movimiento y velocidad en general             | <i>Manual de referencia técnica - RAPID Overview, sección Principios de movimiento y E/S - Posicionamiento durante la ejecución del programa</i> |
| Definición de la velocidad máxima             | <a href="#">VelSet - Cambia la velocidad programada en la página 1029</a>  |
| Velocidad máxima del TCP para el robot actual | <a href="#">MaxRobSpeed - Velocidad máxima del robot en la página 1373</a>   |

### 3.76 stoppointdata - Datos de punto de paro

#### Utilización

`stoppointdata` se utiliza para especificar cómo debe terminar una posición, es decir, a qué distancia de la posición programada deben encontrarse los ejes antes de iniciar un movimiento hasta la posición siguiente.

#### Descripción

Una posición puede terminar en forma de un punto de paso o un punto de paro. El punto de paso significa que la posición programada no llega a alcanzarse nunca. La instrucción especifica una zona para el movimiento, definiendo una trayectoria de esquina. En lugar de dirigirse hacia la posición programada, la dirección del movimiento toma la forma de una trayectoria de esquina antes de alcanzar la posición. Consulte Tipos de datos - `zonedata`.

Un punto de paro significa que los ejes del robot y los ejes externos deben alcanzar la posición especificada antes de que continúen con el movimiento siguiente. Se considera que el robot ha alcanzado un punto de paro cuando se satisfacen los criterios de convergencia del punto. Los criterios de convergencia son la velocidad y la posición. También es posible especificar criterios de temporización. Para el punto de parada `fine`, consulte también el tipo de dato `zonedata`.

Es posible definir tres tipos de puntos de paro mediante `stoppointdata`.

- El tipo de punto de paro en **posición** se define como un porcentaje de los criterios de convergencia (posición y velocidad) para el punto de paro `fine` predefinido. El tipo “en posición” también utiliza un tiempo mínimo y un tiempo máximo. El robot espera el cumplimiento de los criterios de posición y velocidad, durante al menos el tiempo mínimo y como mucho el tiempo máximo.
- En los puntos de paro de **tiempo de paro**, se espera siempre el tiempo especificado sin dejar el punto de paro.
- Los puntos de paro de **tiempo de seguimiento** son puntos de paro de un tipo especial que se utilizan para coordinar los movimientos del robot con un transportador.

Los datos de tipo `stoppointdata` también determinan cómo deben sincronizarse los movimientos con la ejecución de RAPID. Si el movimiento se sincroniza, la ejecución de RAPID espera a un evento de tipo “en posición” cuando el robot está en posición. Si el movimiento no está sincronizado, la ejecución de RAPID recibe un evento de “precaptura” casi medio segundo antes de que el robot físico alcance la posición programada. Cuando la ejecución del programa recibe un evento “en posición” o “precaptura”, continúa con la instrucción siguiente. Cuando llega el evento “precaptura”, el robot sigue teniendo un margen de maniobra amplio. Cuando llega el evento “en posición”, el robot está cerca de la posición programada. Con los tipos **tiempo de paro** y **tiempo de seguimiento**, la instrucción siguiente empieza a ejecutarse al mismo tiempo que empieza la cuenta atrás del tiempo de parada y del tiempo de seguimiento, respectivamente. Sin embargo, para el tipo

*Continúa en la página siguiente*

### 3 Tipos de datos

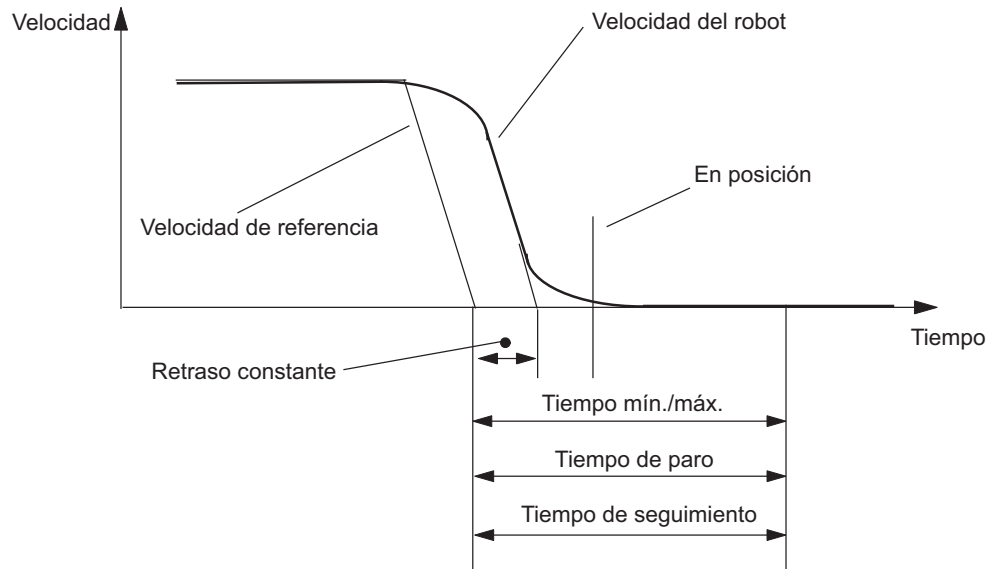
#### 3.76 stoppointdata - Datos de punto de paro

RobotWare - OS

Continuación

en posición, la siguiente instrucción se inicia cuando se cumplen los criterios de convergencia.

Si utiliza instrucciones de movimiento con el argumento \Conc, no se realiza ninguna sincronización, de modo que la ejecución real de la instrucción de movimiento estará preparada inmediatamente.



xx0500002374

La figura anterior representa la terminación de los puntos de paro. La velocidad del robot no reduce la velocidad lineal. El servo del robot siempre va por delante del robot físico. Esto se representa como el retraso constante en la figura anterior. El retraso constante es de aproximadamente 0,1 segundos. Los elementos de temporización de stoppointdata utilizan como disparador una velocidad de referencia. La medición de tiempo comienza cuando la velocidad de referencia es cero. Por tanto, el tiempo de los elementos de temporización siempre incluyen el retraso constante. La consecuencia es que no tiene sentido utilizar valores inferiores al retraso constante.

#### Componentes

type

*type of stop point*

Tipo de dato: stoppoint

La tabla siguiente define el tipo de stoppoint.

|              |   |
|--------------|---|
| 1 (inpos)    | El movimiento termina como un tipo de punto de parada "en posición". Activa el elemento inpos de stoppointdata. No se utilizan los datos de zona de la instrucción, sino fine o z0.       |
| 2 (stoptime) | El movimiento termina con un tipo de punto de paro de "tiempo de paro". Activa el elemento stoptime de stoppointdata. No se utilizan los datos de zona de la instrucción, sino fine o z0. |

Continúa en la página siguiente



|                |  |
|----------------|--|
| 3 (followtime) | El movimiento termina con un tipo de tiempo exacto de seguimiento del transportador. Los datos de zona de la instrucción se utilizan cuando el robot abandona el transportador. Activa el elemento <code>followtime</code> de <code>stoppointdata</code> . |
|----------------|--|

El tipo de dato `stoppoint` es un tipo de dato de alias de `num`. Se utiliza para elegir el tipo de punto de paro y qué elementos de datos de `stoppointdata` deben utilizarse. Sus constantes predefinidas son:

| Valor | Constante simbólica   | Comentario                              |
|-------|-----------------------|---|
| 1     | <code>inpos</code>    | Número de tipo en posición              |
| 2     | <code>stoptime</code> | Número de tipo de tiempo de paro        |
| 3     | <code>flwtime</code>  | Número de tipo de tiempo de seguimiento |

`progsynch`

#### *program synchronization*

Tipo de dato: `bool`

Sincronización con la ejecución del programa de RAPID.

- **TRUE:** El movimiento está sincronizado con la ejecución de RAPID. El programa no empieza a ejecutar la instrucción siguiente hasta que se ha alcanzado el punto de paro.
- **FALSE:** El movimiento no está sincronizado con la ejecución de RAPID. El programa empieza a ejecutar la instrucción siguiente antes de alcanzar el punto de paro.

Si utiliza instrucciones de movimiento con el argumento `\Conc`, no se realiza ninguna sincronización de forma independiente de los datos de `progsynch`, de modo que la instrucción de movimiento real estará siempre preparada inmediatamente.

`inpos.position`

#### *position condition for TCP*

Tipo de dato: `num`

La condición de posición (el radio) del TCP en porcentaje de un punto de paro `fine normal`.

`inpos.speed`

#### *speed condition for TCP*

Tipo de dato: `num`

La condición de velocidad del TCP en porcentaje de un punto de paro `fine normal`.

`inpos.mintime`

#### *minimum wait time*

Tipo de dato: `num`

El tiempo de espera mínimo, en segundos, antes de alcanzar la posición. Se utiliza para hacer que el robot espere en el punto al menos el tiempo especificado. El valor máximo es de 20,0 segundos.

*Continúa en la página siguiente*

## 3 Tipos de datos

---

### 3.76 stoppointdata - Datos de punto de paro

RobotWare - OS

Continuación

`inpos.maxtime`

*maximum wait time*

Tipo de dato: `num`

El tiempo máximo, en segundos, que se espera a que se cumplan los criterios de convergencia. Se utilizan para garantizar que el robot no se quede parado en el punto si se han establecido condiciones de velocidad y posición demasiado estrictas. El valor máximo es de 20,0 segundos.

`stoptime`

*stop time*

Tipo de dato: `num`

El tiempo, en segundos, que el TCP permanece parado en la posición antes de empezar el movimiento siguiente. Rango válido de 0 a 20 s, resolución 0,001 s.

`followtime`

*follow time*

Tipo de dato: `num`

El tiempo, en segundos, que el TCP sigue al transportador. Rango válido de 0 a 20 s, resolución 0,001 s.

`signal`

Tipo de dato: `string`

Reservado para un uso futuro.

`relation`

Tipo de dato: `opnum`

Reservado para un uso futuro.

`checkvalue`

Tipo de dato: `num`

Reservado para un uso futuro.

---

### Ejemplos básicos

Los siguientes ejemplos ilustran el tipo de dato `stoppointdata`:

`Inpos`

```
VAR stoppointdata my_inpos := [ inpos, TRUE, [ 25, 40, 0.1, 5], 0,
                                0, "", 0, 0];
MoveL *, v1000, fine \Inpos:=my_inpos, grip4;
```

Se definen los datos del punto de paro `my_inpos` con las características siguientes:

- El punto de paro es de tipo "en posición", `inpos`.
- El punto de paro estará sincronizado con la ejecución del programa de **RAPID**, `TRUE`.
- El criterio de distancia del punto de paro es del 25% de la distancia definida para el punto de paro `fine`, 25.
- El criterio de velocidad del punto de paro es del 40% de la velocidad definida para el punto de paro `fine`, 40.

Continúa en la página siguiente

- El tiempo mínimo de espera antes de la convergencia es 0,1 s, 0.1.
- El tiempo máximo que se espera a la convergencia es 5 s, 5.

El robot se mueve hacia la posición programada hasta que se cumple uno de los criterios de posición o velocidad.

```
my_inpos.inpos.position := 40;  
MoveL *, v1000, fine \Inpos:=my_inpos, grip4;
```

Se ajusta el criterio de distancia del punto de paro al 40%.

#### Stoptime

```
VAR stoppointdata my_stoptime := [ stoptime, FALSE, [ 0, 0, 0, 0 ],  
1.45, 0, "", 0, 0];  
MoveL *, v1000, fine \Inpos:=my_stoptime, grip4;
```

Se definen los datos del punto de paro my\_stoptime con las características siguientes:

- El punto de paro es de tipo de tiempo de espera, stoptime.
- El punto de paro no estará sincronizado con la ejecución del programa de RAPID, FALSE.
- El tiempo de espera en posición es de 1,45 s.

El robot se mueve hacia la posición programada hasta que llega el evento de precaptura. Se ejecuta la siguiente instrucción de RAPID. Si es una instrucción de movimiento, el robot se detiene durante 1,45 segundos antes de empezar el movimiento siguiente.

```
my_stoptime.stoptime := 6.66;  
MoveL *, v1000, fine \Inpos:=my_stoptime, grip4;
```

Se ajusta el tiempo de paro del punto de paro a 6,66 s. Si la siguiente instrucción de RAPID es una instrucción de movimiento, el robot se detiene durante 6,66 s.

#### Followtime

```
VAR stoppointdata my_followtime := [ fllwtime, TRUE, [ 0, 0, 0,  
0 ], 0, 0.5, "", 0, 0];  
MoveL *, v1000, z10 \Inpos:=my_followtime, grip6\wobj:=conveyor1;
```

Se definen los datos del punto de paro my\_followtime con las características siguientes:

- El punto de paro es de tipo de tiempo de seguimiento, fllwtime.
- El punto de paro estará sincronizado con la ejecución del programa de RAPID, TRUE.
- El tiempo de seguimiento del punto de paro es de 0,5 s, 0.5.

El robot sigue al transportador durante 0,5 s antes de dejarlo, con una zona de 10 mm, z10.

```
my_followtime.followtime := 0.4;
```

Se ajusta el tiempo de seguimiento del punto de paro a 0,4 s.

*Continúa en la página siguiente*

### 3 Tipos de datos

#### 3.76 stoppointdata - Datos de punto de paro

RobotWare - OS

Continuación

#### Datos predefinidos

Existen varios datos de punto de paro ya definidos en el sistema.

#### Puntos de paro en posición

| Nombre   | Progsynch | Posición | Velocidad | Mintime | Maxtime | Stoptime | Followtime |
|----------|-----------|----------|-----------|---------|---------|----------|------------|
| 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 tiene los mismos criterios de convergencia que el punto de paro fine)

#### Puntos de paro de tiempo de paro

| Nombre      | Progsynch | Posición | Velocidad | Mintime | Maxtime | Stoptime | Followtime |
|-------------|-----------|----------|-----------|---------|---------|----------|------------|
| stoptime0_5 | FALSE     | -        | -         | -       | -       | 0.5 s    | -          |
| stoptime1_0 | FALSE     | -        | -         | -       | -       | 1.0 s    | -          |
| stoptime1_5 | FALSE     | -        | -         | -       | -       | 1.5 s    | -          |

#### Puntos de paro de tiempo de seguimiento

| Nombre     | Progsynch | Posición | Velocidad | Mintime | Maxtime | Stoptime | Followtime |
|------------|-----------|----------|-----------|---------|---------|----------|------------|
| flwtime0_5 | TRUE      | -        | -         | -       | -       | -        | 0.5 s      |
| flwtime1_0 | TRUE      | -        | -         | -       | -       | -        | 1.0 s      |
| flwtime1_5 | TRUE      | -        | -         | -       | -       | -        | 1.5 s      |

#### Estructura

```
< 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 >  
< checkvalue of num >
```

#### Información relacionada

| Para obtener más información sobre | Consulte  |
|------------------------------------|---|
| Instrucciones de posicionamiento   | Manual de referencia técnica - RAPID Overview, sección Resumen sobre RAPID - Movimiento |

Continúa en la página siguiente

| Para obtener más información sobre  | Consulte   |
|-------------------------------------|--|
| Movimientos/trayectorias en general | <i>Manual de referencia técnica - RAPID Overview, sección Principios de movimiento y E/S - Posicionamiento durante la ejecución del programa</i> |
| Puntos de paro o de paso            | <a href="#">zonedata - Datos de zonas en la página 1831</a>  |

## 3 Tipos de datos

3.77 string - Cadenas  
RobotWare - OS

### 3.77 string - Cadenas

#### Utilización

`string` se utiliza con cadenas de caracteres.

#### Descripción

Las cadenas de caracteres están compuestas por varios caracteres (un máximo de 80) encerrados entre comillas (""), por ejemplo "Esto es una cadena de caracteres".

Si desea incluir comillas dentro de la cadena de caracteres, debe escribirlas dos veces, por ejemplo "Esta cadena contiene un carácter "" de comilla doble".

Si desea incluir una barra invertida dentro de la cadena de caracteres, debe escribirla dos veces, por ejemplo "Esta cadena contiene un carácter \\".

#### Ejemplos básicos

El siguiente ejemplo ilustra el tipo de dato `string`:

##### Ejemplo 1

```
VAR string text;  
...  
text := "start welding pipe 1";  
TPWrite text;
```

El texto `start welding pipe 1` se escribe en el FlexPendant.

#### Limitaciones

Una cadena puede tener de 0 a 80 caracteres, incluidas las comillas o las barras invertidas que se utilicen.

Las cadenas de caracteres pueden contener cualquiera de los caracteres especificados en la norma ISO 8859-1 (Latin-1), además de caracteres de control (caracteres ajenos al estándar ISO 8859-1 (Latin-1) y con códigos numéricos de 0 a 255).

#### Datos predefinidos

El sistema dispone de varias constantes de cadena de caracteres predefinidas, que pueden usarse junto con las funciones para cadenas de caracteres. Consulte el ejemplo `StrMemb`.

| Nombre    | Conjunto de caracteres   |
|-----------|--|
| STR_DIGIT | <digit> ::=<br>0   1   2   3   4   5   6   7   8   9   |
| STR_UPPER | <upper case letter> ::=<br>A   B   C   D   E   F   G   H   I   J<br>  K   L   M   N   O   P   Q   R   S   T<br>  U   V   W   X   Y   Z   À   Á   Â   Ã<br>  Ä   Å   Æ   Ç   È   É   Ê   Ë   Ì   Í<br>  Î   Ï   1) Ñ   Ò   Ó   Ô   Õ   Ö   Ø<br>  Ù   Ú   Û   Ü   2)   3) |

Continúa en la página siguiente

| Nombre    | Conjunto de caracteres  |
|-----------|---|
| STR_LOWER | <lower case letter> ::=<br>a   b   c   d   e   f   g   h   i   j<br>  k   l   m   n   o   p   q   r   s   t<br>  u   v   w   x   y   z   à   á   â   ã<br>  ä   å   æ   ç   è   é   ê   ë   ì   í<br>  î   ï   1)   ñ   ò   ó   ô   õ   ö   ø<br>  ù   ú   û   ü   2)   3)   ß   ÿ- |
| STR_WHITE | <blank character> ::=   |

- 1) Letra "eth" islandesa.
- 2) Letra Y con acento agudo.
- 3) Letra "thorn" islandesa.

Las constantes siguientes ya están definidas en el sistema:

```

CONST string diskhome := "HOME:";

! For old programs from S4C system
CONST string ramldisk := "HOME:";

CONST string disktemp := "TEMP:";

CONST string flp1 := "flp1:";

CONST string stSpace := " ";

CONST string stEmpty := "";
    
```

#### Información relacionada

| Para obtener más información sobre              | Consulte  |
|---|---|
| Operaciones con cadenas de caracteres           | <i>Manual de referencia técnica - RAPID Overview, sección Características básicas - Expresiones</i>       |
| Valores de cadena de caracteres                 | <i>Manual de referencia técnica - RAPID Overview, sección Características básicas - Elementos básicos</i> |
| Instrucciones que utilizan juegos de caracteres | <a href="#">StrMemb - Comprueba si un carácter pertenece a un conjunto en la página 1503</a>              |

## 3 Tipos de datos

### 3.78 stringdig - Cadena de caracteres con sólo dos dígitos

RobotWare - OS

### 3.78 stringdig - Cadena de caracteres con sólo dos dígitos

#### Utilización

`stringdig` se utiliza para representar enteros positivos grandes en una cadena, usando únicamente dígitos.

Este tipo de dato ha sido introducido dado que el tipo de dato `num` no es capaz de manejar enteros positivos superiores a 8.388.608 con una representación exacta.

#### Descripción

Un dato `stringdig` sólo puede contener los dígitos del 0 al 9 encerrados entre guiones (""), por ejemplo "0123456789".

El tipo de dato `stringdig` puede manejar enteros positivos hasta 4.294.967.295.

#### Ejemplos básicos

El siguiente ejemplo ilustra el tipo de dato `stringdig`:

##### Ejemplo 1

```
VAR stringdig digits1;  
VAR stringdig digits2;  
VAR bool flag1;  
...  
digits1 ="09000000";  
digits2 = "9000001";  
flag1 := StrDigCmp (digits1, LT, digits2);
```

El dato `flag1` cambia a `TRUE` porque `09000000` es inferior a `9000001`.

#### Características

`stringdig` es un tipo de dato de alias de `string` y por tanto hereda la mayor parte de sus características.

#### Información relacionada

| Para obtener más información sobre          | Consulte  |
|---|---|
| Valores de cadena de caracteres             | <i>Manual de referencia técnica - RAPID Overview</i> , sección <i>Características básicas - Elementos básicos</i>   |
| Cadenas                                     | <a href="#">string - Cadenas en la página 1778</a>  |
| Valores numéricos                           | <a href="#">num - Valores numéricos en la página 1712</a>   |
| Operador de comparación                     | <a href="#">opnum - Operador de comparación en la página 1715</a> <a href="#">StrDigCmp - Comparar dos cadenas que sólo contienen dígitos en la página 1494</a> |
| Comparar cadenas que sólo contienen dígitos | <a href="#">StrDigCmp - Comparar dos cadenas que sólo contienen dígitos en la página 1494</a>   |



#### 3.79 supervtimeouts - Tiempos límite de supervisión de intercambio

##### Utilización

`supervtimeouts` se utiliza para definir los tiempos límite para la supervisión del intercambio en CAP.

`supervtimeouts` es un componente de `capdata` y define los tiempos límite para las siguientes fases de la supervisión de intercambio en CAP:

- PRE
- END\_PRE y START
- END MAIN y START\_POST1
- END\_POST1 y START\_POST2
- END\_POST2

Si el parámetro tiene el valor 0, no se establece ningún tiempo límite.

##### Componentes

###### pre\_cond

Tipo de dato: `num`

Tiempo límite (en segundos) para que se cumplan las condiciones de la fase PRE.

###### start\_cond

Tipo de dato: `num`

Tiempo límite (en segundos) para que se cumplan las condiciones de las fases END\_PRE y START.

###### end\_main\_cond

Tipo de dato: `num`

Tiempo límite (en segundos) para que se cumplan las condiciones de las fases END\_MAIN y START\_POST1.

###### end\_post1\_cond

Tipo de dato: `num`

Tiempo límite (en segundos) para que se cumplan las condiciones de las fases END\_POST1 y START\_POST2.

###### end\_post2\_cond

Tipo de dato: `num`

Tiempo límite (en segundos) para que se cumplan las condiciones de la fase END\_POST2.

##### Sintaxis

```
< 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 >
```

*Continúa en la página siguiente*

### 3 Tipos de datos

---

3.79 supervtimeouts - Tiempos límite de supervisión de intercambio

*Continuous Application Platform (CAP)*

*Continuación*

---

#### Información relacionada

|  | Descrito en:  |
|--|---|
| Tipo de dato <code>capdata</code>      | <a href="#">capdata - Datos CAP en la página 1624</a>       |
| <i>Continuous Application Platform</i> | <i>Application manual - Continuous Application Platform</i> |

## 3.80 switch - Parámetros opcionales

### Utilización

`switch` se utiliza para los parámetros opcionales.

### Descripción

El tipo especial `switch` (sólo) puede ser asignado a parámetros opcionales y proporciona una forma de usar argumentos modificadores, es decir, argumentos que sólo se especifican por nombre (no por valor). No es posible transmitir un valor a un parámetro modificador. La única forma de usar un parámetro modificador es comprobar su presencia mediante la función predefinida `Present`.

### Ejemplos básicos

El siguiente ejemplo ilustra el tipo de dato `switch`:

#### Ejemplo 1

```
PROC my_routine(\switch on | switch off)
....
IF Present (off) THEN
....
ENDIF
ENDPROC
```

En función de los argumentos utilizados por la rutina que llama a `my_routine`, el flujo del programa puede ser controlado.

### Características

`switch` es un tipo de dato sin valor y no puede usarse en operaciones basadas en valores.

### Información relacionada

| Para obtener más información sobre                    | Consulte  |
|---|---|
| Parámetros  | <i>Manual de referencia técnica - RAPID Overview</i> , sección <i>Características básicas - Rutinas</i> . |
| Cómo comprobar si está presente un parámetro opcional | <a href="#">Present - Comprueba si se está usando un parámetro opcional en la página 1427</a>             |

## 3 Tipos de datos

### 3.81 symnum - Número simbólico

RobotWare - OS

### 3.81 symnum - Número simbólico

#### Utilización

`symnum` (*Symbolic Number*) se utiliza para representar un entero con una constante simbólica.

#### Descripción

Las constantes de `symnum` se han diseñado para usarlas al comprobar el valor de retorno de las funciones `OpMode` y `RunMode`.

#### Ejemplos básicos

El siguiente ejemplo ilustra el tipo de dato `symnum`:

#### Ejemplo 1

```
IF RunMode() = RUN_CONT_CYCLE THEN
..
ELSE
..
ENDIF
```

#### Datos predefinidos

Se han definido las constantes simbólicas siguientes del tipo de dato `symnum`. Puede usarlas a la hora de comprobar valores de retorno de las funciones `OpMode` y `RunMode`.

| Valor | Constante simbólica | Comentario   |
|-------|---------------------|--|
| 0     | RUN_UNDEF           | Modo de ejecución no definido  |
| 1     | RUN_CONT_CYCLE      | Modo de ejecución continuo o en modo ciclo   |
| 2     | RUN_INSTR_FWD       | Modo de ejecución de avance de instrucciones   |
| 3     | RUN_INSTR_BWD       | Modo de ejecución hacia atrás  |
| 4     | RUN_SIM             | Modo de ejecución simulado   |
| 5     | RUN_STEP_MOVE       | Instrucciones de movimiento en ejecución hacia delante e instrucciones lógicas en modo de ejecución continuo |

| Valor | Constante simbólica | Comentario   |
|-------|---------------------|--|
| 0     | OP_UNDEF            | Modo de funcionamiento no definido                     |
| 1     | OP_AUTO             | Modo de funcionamiento automático                      |
| 2     | OP_MAN_PROG         | Modo de funcionamiento manual a 250 mm/seg como máximo |
| 3     | OP_MAN_TEST         | Modo de funcionamiento manual a máxima velocidad, 100% |

#### Características

`Symnum` es un tipo de dato de alias de `num` y por tanto hereda sus características.

Continúa en la página siguiente

#### Información relacionada

| Para obtener más información sobre                 | Consulte   |
|--|--|
| Tipos de datos en general, tipos de datos de alias | <i>Manual de referencia técnica - RAPID Overview, sección Características básicas - Tipos de datos</i> |

## 3 Tipos de datos

### 3.82 syncident - Identidad de punto de sincronización

#### Multitasking

## 3.82 syncident - Identidad de punto de sincronización

### Utilización

`syncident` (*synchronization identity*) se utiliza para especificar el nombre de un punto de sincronización. El nombre del punto de sincronización tendrá el mismo nombre (identidad) de los datos declarados con el tipo `syncident`.

### Descripción

`syncident` se utiliza para identificar un punto de programa en el que la tarea de programa actual esperará a que las tareas de programa cooperantes alcancen el mismo punto de sincronización.

El nombre (la identidad) del dato del tipo `syncident` debe ser el mismo en todas las tareas de programa cooperantes.

Los datos del tipo `syncident` se utilizan en las instrucciones `WaitSyncTask`, `SyncMoveOn` y `SyncMoveOff`.

### Ejemplos básicos

El siguiente ejemplo ilustra el tipo de dato `syncident`:

#### Ejemplo 1

Ejemplo de programa de la tarea de programa `ROB1`

```
PERS tasks task_list{3} := [ ["STN1"], ["ROB1"], ["ROB2"] ];  
VAR syncident sync1;
```

```
WaitSyncTask sync1, task_list;
```

En el momento de la ejecución de la instrucción `WaitSyncTask` en la tarea de programa `ROB1`, la ejecución de dicha tarea de programa esperará a que las demás tareas de programa, `STN1` y `ROB2`, hayan alcanzado su instrucción `WaitSyncTask` correspondiente con el mismo punto de sincronización (reunión), `sync1`.

### Estructura

`syncident` es un tipo de dato sin valor.

### Información relacionada

| Para obtener más información sobre                    | Consulte  |
|---|---|
| Especificación de tareas de programa cooperativas     | <a href="#">tasks - Tareas de programa RAPID en la página 1791</a>  |
| Espera de un punto de sincronización con otras tareas | <a href="#">WaitSyncTask - Esperar en un punto de sincronización con otras tareas de programa en la página 1075</a> |
| Inicio de movimientos sincronizados coordinados       | <a href="#">SyncMoveOn - Inicia los movimientos sincronizados coordinados en la página 854</a>                      |
| Fin de movimientos sincronizados coordinados          | <a href="#">SyncMoveOff - Finaliza los movimientos sincronizados coordinados en la página 848</a>                   |

## 3.83 System data - Ajustes de datos del sistema RAPID actual

## Utilización

`System data` refleja los valores actuales de los datos del sistema de RAPID, el número actual de recuperación en caso de error `ERRNO`, el número actual de interrupción `INTNO`, etc.

Estos datos pueden ser utilizados y leídos por el programa. Pueden usarse para leer el estado actual, por ejemplo, el desplazamiento actual del programa.

## C\_MOTSET

La variable `C_MOTSET` del tipo de dato `motsetdata` refleja los valores de movimiento actuales:

| Descripción   | Tipo de dato            | Cambiado por                             | Consulte también  |
|---|-------------------------|--|---|
| Parámetros actuales de movimiento, es decir:  | <code>motsetdata</code> | Instrucciones                            | <a href="#">motsetdata - Datos de parámetros de movimiento en la página 1705</a>  |
| Ajuste de velocidad y velocidad máxima  |                         | <code>VelSet</code>                      | <a href="#">VelSet - Cambia la velocidad programada en la página 1029</a>   |
| Ajuste de aceleración   |                         | <code>AccSet</code>                      | <a href="#">AccSet - Reduce la aceleración en la página 25</a>  |
| Movimientos cerca de puntos singulares  |                         | <code>SingArea</code>                    | <a href="#">SingArea - Define el método de interpolación alrededor de puntos singulares en la página 730</a>  |
| Control de configuración lineal<br>Control de configuración de ejes                           |                         | <code>ConfL</code><br><code>ConfJ</code> | <a href="#">ConfL - Monitoriza la configuración durante el movimiento lineal en la página 158</a><br><a href="#">ConfJ - Controla la configuración durante el movimiento de los ejes en la página 156</a> |
| Resolución de la trayectoria  |                         | <code>PathResol</code>                   | <a href="#">PathResol - Ajusta la resolución de la trayectoria en la página 530</a>   |
| Ajuste de la supervisión de movimiento  |                         | <code>MotionSup</code>                   | <a href="#">MotionSup - Desactiva/activa la supervisión del movimiento en la página 379</a>   |
| Reducción de la aceleración y deceleración del TCP a lo largo de la trayectoria de movimiento |                         | <code>PathAccLim</code>                  | <a href="#">PathAccLim - Reduce la aceleración del TCP a lo largo de la trayectoria en la página 504</a>  |
| Modificación de la orientación de la herramienta durante la interpolación circular            |                         | <code>CirPathMode</code>                 | <a href="#">CirPathMode - Reorientación de la herramienta durante trayectorias circulares en la página 133</a>  |
| Reducción de la aceleración de la carga útil en el sistema de coordenadas mundo               |                         | <code>WorldAccLim</code>                 | <a href="#">WorldAccLim - Control de aceleración en el sistema de coordenadas mundo en la página 1096</a>   |

Continúa en la página siguiente

### 3 Tipos de datos

#### 3.83 System data - Ajustes de datos del sistema RAPID actual

RobotWare - OS

Continuación

C\_PROGDISP

La variable `C_PROGDISP` del tipo de dato `progdisp` refleja el desplazamiento actual del programa y el offset de los ejes externos:

| Descripción   | Tipo de dato | Cambiado por   | Consulte también  |
|---|--------------|----------------|---|
| Desplazamiento de programa actual para los ejes del robot | progdisp     | Instrucciones: | <a href="#">progdisp - Desplazamiento de programa en la página 1733</a>   |
|   |              | PDispSet       | <a href="#">PDispSet - Activa un desplazamiento de programa a partir de una base de coordenadas conocida en la página 538</a> |
|   |              | PDispOn        | <a href="#">PDispOn - Activa el desplazamiento de programa en la página 533</a>   |
|   |              | PDispOff       | <a href="#">PDispOff - Desactiva el desplazamiento de programa en la página 532</a>   |
| Offset actual de los ejes externos                        |              | EOfsSet        | <a href="#">EOfsSet - Activa un offset de ejes adicionales a partir de valores conocidos en la página 197</a>                 |
|   |              | EOfsOn         | <a href="#">EOfsOn - Activa un offset de ejes adicionales en la página 195</a>  |
|   |              | EOfsOff        | <a href="#">EOfsOff - Desactiva un offset de ejes adicionales en la página 194</a>  |

ERRNO

La variable `ERRNO` del tipo de dato `errnum` refleja el número actual de recuperación en caso de error:

| Descripción                          | Tipo de dato | Cambiado por | Consulte también   |
|--------------------------------------|--------------|--------------|--|
| El último error que ha tenido lugar. | errnum       | El sistema   | <a href="#">Manual de referencia técnica - RAPID Overview, sección Resumen sobre RAPID - Recuperación en caso de error</a><br><a href="#">intnum - Identidad de interrupción en la página 1687</a> |

INTNO

La variable `INTNO` del tipo de dato `intnum` refleja el número actual de interrupción:

| Descripción                                 | Tipo de dato | Cambiado por | Consulte también  |
|---|--------------|--------------|---|
| La última interrupción que ha tenido lugar. | intnum       | El sistema   | <a href="#">Manual de referencia técnica - RAPID Overview, sección Resumen sobre RAPID - Interrupciones</a><br><a href="#">intnum - Identidad de interrupción en la página 1687</a> |

Continúa en la página siguiente



ROB\_ID

La variable `ROB_ID` del tipo de dato `mecunit` contiene una referencia al robot con TCP (si lo hay) de la tarea de programa actual.

| Descripción  | Tipo de dato         | Cambiado por | Consulte también  |
|--|----------------------|--------------|---|
| Referencia al robot (si corresponde) en la tarea de programa actual. Compruébelo antes de usar con <code>TaskRunRob ( )</code> | <code>mecunit</code> | El sistema   | <a href="#">mecunit - Unidad mecánica en la página 1703</a> |

## 3 Tipos de datos

---

### 3.84 taskid - Identificación de tarea

#### *Multitasking*

### 3.84 taskid - Identificación de tarea

---

#### Utilización

`taskid` se utiliza para identificar tareas de programa disponibles en el sistema. Los nombres de las tareas de programa se definen en los parámetros del sistema y, por tanto, no es necesario definirlos en el programa.

---

#### Descripción

Los datos del tipo `taskid` sólo contienen una referencia a la tarea de programa.

---

#### Limitaciones

No debe definir los datos de tipo `taskid` en el programa. Sin embargo, sí es posible utilizarlos como parámetros al declarar una rutina.

---

#### Datos predefinidos

Las tareas de programa definidas en los parámetros del sistema están siempre disponibles desde el programa (datos instalados). Existen variables predefinidas con el tipo de dato `taskid` para todas las tareas de programa del sistema. La identificación de la variable será "nombre\_tarea"+"ID". Por ejemplo, para la tarea `T_ROB1` la identificación de la tarea es `T_ROB1Id`, `T_ROB2` - `T_ROB2Id` etc.

---

#### Características

`taskid` es un tipo de dato sin valor. Esto significa que los datos de este tipo no son compatibles con operaciones basadas en valores.

---

#### Información relacionada

| Para obtener más información sobre              | Consulte   |
|---|--|
| Guardado de módulos de programa                 | <a href="#">Save - Guarda un módulo de programa en la página 648</a>                                   |
| Configuración de tareas de programa             | <i>Manual de referencia técnica - Parámetros del sistema</i>   |
| Características de los tipos de datos sin valor | <i>Manual de referencia técnica - RAPID Overview, sección Características básicas - Tipos de datos</i> |

## 3.85 tasks - Tareas de programa RAPID

### Utilización

`tasks` se utiliza para especificar varias tareas de programa de RAPID.

### Descripción

Para especificar varias tareas de programa de RAPID, es posible indicar el nombre de cada tarea como una cadena. En este caso, una matriz del tipo de dato `tasks` puede contener todos los nombres de las tareas.

A continuación, esta lista de tareas puede ser utilizada en las instrucciones `WaitSyncTask` y `SyncMoveOn`.



#### Nota

Estas instrucciones exigen que los datos estén definidos como variables `PERS` globales del sistema y disponibles en todas las tareas cooperantes.

### Componentes

El tipo de dato contiene los componentes siguientes.

`taskname`

Tipo de dato: `string`

El nombre de una tarea de programa de RAPID, especificado en una cadena.

### Ejemplos básicos

El siguiente ejemplo ilustra el tipo de dato `tasks`:

#### Ejemplo 1

Ejemplo de programa de la tarea de programa `T_ROB1`

```
PERS tasks task_list{3} := [ ["T_STN1"], ["T_ROB1"], ["T_ROB2"] ];
VAR syncident sync1;
```

```
WaitSyncTask sync1, task_list;
```

En el momento de la ejecución de la instrucción `WaitSyncTask` en la tarea de programa `T_ROB1`, la ejecución de dicha tarea de programa esperará a que las demás tareas de programa, `T_STN1` y `T_ROB2`, hayan alcanzado su instrucción `WaitSyncTask` correspondiente con el mismo punto de sincronización (reunión), `sync1`.

### Estructura

```
<dataobject of tasks>
  <taskname of string>
```

*Continúa en la página siguiente*

### 3 Tipos de datos

---

#### 3.85 tasks - Tareas de programa RAPID

*Multitasking*

*Continuación*

---

#### Información relacionada

| Para obtener más información sobre                    | Consulte  |
|---|---|
| Identidad para punto de sincronización                | <a href="#">syncident - Identidad de punto de sincronización en la página 1786</a>                                  |
| Espera de un punto de sincronización con otras tareas | <a href="#">WaitSyncTask - Esperar en un punto de sincronización con otras tareas de programa en la página 1075</a> |
| Inicio de movimientos sincronizados coordinados       | <a href="#">SyncMoveOn - Inicia los movimientos sincronizados coordinados en la página 854</a>                      |
| Fin de movimientos sincronizados coordinados          | <a href="#">SyncMoveOff - Finaliza los movimientos sincronizados coordinados en la página 848</a>                   |

### 3.86 testsignal - Señal de prueba

#### Utilización

El tipo de dato `testsignal` se utiliza cuando se realiza un test del sistema de movimiento del robot.

#### Descripción

El sistema de robot cuenta con varias señales de test predefinidas. El tipo de dato `testsignal` puede usarse para simplificar la programación de la instrucción `TestSignDefine`.

#### Ejemplos básicos

Los siguientes ejemplos ilustran el tipo de datos `testsignal`:

##### Ejemplo 1

```
TestSignDefine 2, speed, Orbit, 2, 0;
```

La constante predefinida `speed` se usa para leer la velocidad actual del eje 2 en el manipulador `orbit`.

##### Ejemplo 2

```
TestSignDefine 4, 4001, ROB_1, 2, 0;
```

La velocidad de la señal de prueba se usa para leer la velocidad real del eje 2 del robot.

#### Datos predefinidos

Las siguientes señales de prueba están disponibles para ejes adicionales y están predeterminadas en el sistema. Todos los datos se indican en unidades SI y se miden en el lado motor del eje.

| Constante simbólica         | Valor | Unidad |
|-----------------------------|-------|--------|
| <code>speed</code>          | 6     | rad/s  |
| <code>torque_ref</code>     | 9     | Nm     |
| <code>resolver_angle</code> | 1     | rad    |
| <code>speed_ref</code>      | 4     | rad/s  |
| <code>dig_input1</code>     | 102   | 0 ó 1  |
| <code>dig_input2</code>     | 103   | 0 ó 1  |

Las siguientes señales de prueba están disponibles tanto para el robot y como para ejes adicionales y están predeterminadas en el sistema. Todos los datos se miden en el lado motor del eje.

| Señal de prueba           | Valor | Unidad                             |
|---------------------------|-------|------------------------------------|
| Posición                  | 4000  | grados o milímetros <sup>i</sup>   |
| Velocidad                 | 4001  | grados/segundo o mm/s <sup>i</sup> |
| Par                       | 4002  | Nm                                 |
| Par externo <sup>ii</sup> | 4003  | Nm                                 |

<sup>i</sup> La unidad depende de si el eje es rotacional o lineal.

*Continúa en la página siguiente*

### 3 Tipos de datos

---

#### 3.86 testsignal - Señal de prueba

RobotWare - OS

Continuación

- ii Devuelve un par aplicado estimado externamente (por contacto con el entorno). En un eje adicional el par externo puede no ser válido.

---

#### Características

`testsignal` es un tipo de dato de alias de `num` y por tanto hereda sus características.

---

#### Información relacionada

| Para obtener más información sobre | Consulte  |
|------------------------------------|---|
| Definición de una señal de test    | <a href="#">TestSignDefine - Define una señal de prueba en la página 872</a>                            |
| Lectura de una señal de prueba     | <a href="#">TestSignRead - Obtiene el valor de una señal de test en la página 1528</a>                  |
| Puesta a cero de señales de prueba | <a href="#">TestSignReset - Restablece todas las definiciones de señales de prueba en la página 874</a> |

### 3.87 tooldata - Datos de herramienta

#### Utilización

`tooldata` se utiliza para describir las características de una herramienta, por ejemplo, una pistola de soldadura o una pinza. Estas características son la posición y orientación del TCP (punto central de la herramienta) y las características físicas de la carga de la herramienta.

Si la herramienta está fija en el espacio (una herramienta estacionaria), los datos de la herramienta definen primero la posición y orientación de esta herramienta en el espacio, TCP. A continuación, describe la carga de la pinza movida por el robot.

#### Descripción

Los datos de la herramienta afectan de las formas siguientes a los movimientos del robot:

- El punto central de la herramienta (TCP) se refiere a un punto que seguirá la trayectoria especificada y el rendimiento de velocidad deseado. Si se reorienta la herramienta o se utilizan los ejes externos coordinados, sólo este punto seguirá la trayectoria deseada a la velocidad programada.
- Si se utiliza una herramienta estacionaria, la velocidad y la trayectoria programadas serán las del objeto de trabajo sostenido por el robot.
- Las posiciones programadas se refieren a la posición del TCP actual y la orientación en relación con el sistema de coordenadas de la herramienta. Esto significa que si por ejemplo, se reemplaza una herramienta porque está dañada, sigue siendo posible utilizar el programa anterior con sólo redefinir el sistema de coordenadas de la herramienta.

Los datos de la herramienta también se usan en los movimientos del robot para:

- Definir el TCP que no debe moverse cuando se reorienta el robot.
- Definir el sistema de coordenadas de la herramienta para facilitar el acercamiento o la rotación de las direcciones de coordenadas de la herramienta.

*Continúa en la página siguiente*

## 3 Tipos de datos

---

3.87 tooldata - Datos de herramienta

RobotWare - OS

Continuación



### ¡AVISO!

Es importante definir siempre la carga real de la herramienta y, si se usa, la carga útil del robot (por ejemplo, una pieza sujeta por una pinza). Una definición incorrecta de los datos de carga puede dar lugar a la sobrecarga de la estructura mecánica del robot. Existe también el riesgo de que pueda superarse la velocidad en el modo manual a velocidad reducida.

Cuando se especifican datos de carga incorrectos, este hecho suele tener las consecuencias siguientes:

- El robot no puede funcionar a su capacidad máxima.
- Peor exactitud de la trayectoria, con riesgo de sobrepasar posiciones.
- Riesgo de sobrecarga de la estructura mecánica.

El controlador monitoriza continuamente la carga y escribe un registro de eventos si la carga es más elevada que la prevista. Este registro de eventos se guarda y registra en la memoria del controlador.

---

### Componentes

robhold

#### *robot hold*

Tipo de dato: `bool`

Define si el robot es el que está sosteniendo la herramienta:

- `TRUE`: El robot sostiene la herramienta.
- `FALSE`: El robot no sostiene la herramienta, sino que se trata de una herramienta estacionaria.

tframe

#### *tool frame*

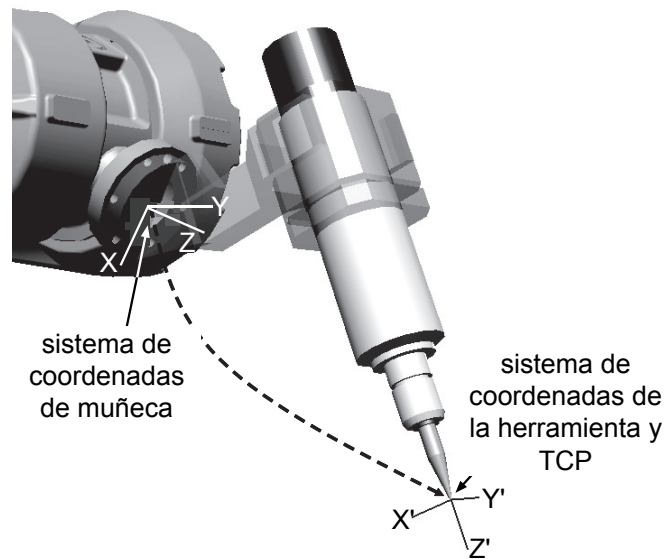
Tipo de dato: `pose`

El sistema de coordenadas de la herramienta, es decir:

- La posición del TCP (x, y, z) en mm, expresada en el sistema de coordenadas de la muñeca (tool0) (consulte la figura siguiente).
- La orientación del sistema de coordenadas de la herramienta, expresado en el sistema de coordenadas de la muñeca (consulte la figura siguiente).

*Continúa en la página siguiente*





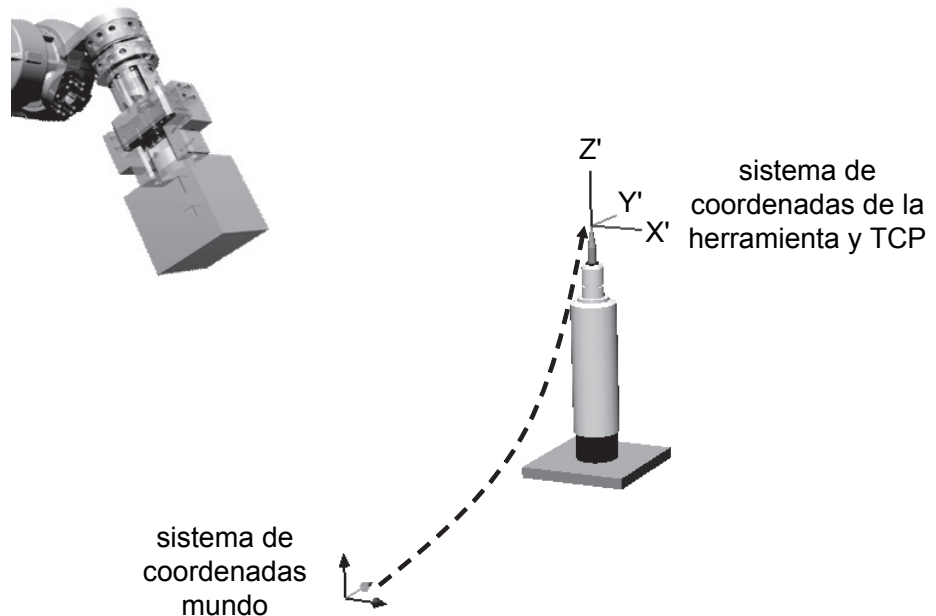
xx110000517

Figure 3.3: Herramienta sostenida por el robot



**Nota**

Si se utiliza una herramienta estacionaria, la base de coordenadas de la herramienta se define respecto del sistema de coordenadas mundo.



xx110000518

Figure 3.4: Herramienta estacionaria

Continúa en la página siguiente

### 3 Tipos de datos

3.87 tooldata - Datos de herramienta

RobotWare - OS

Continuación

tload

*tool load*

Tipo de dato: loaddata



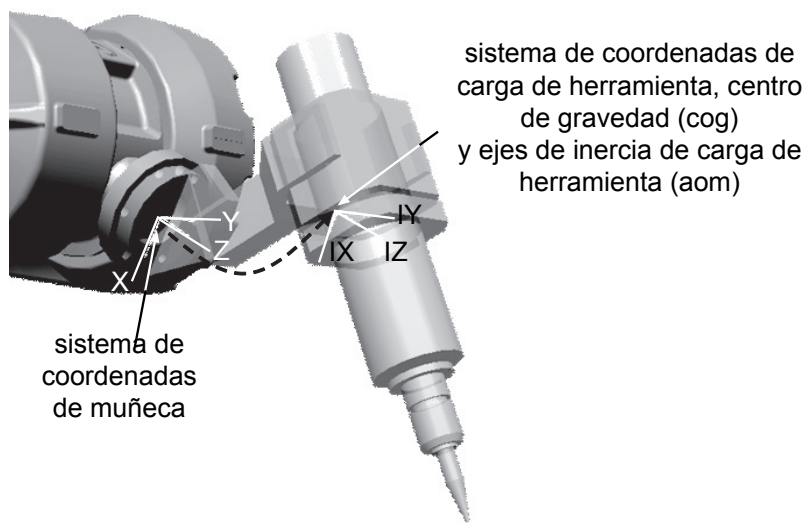
#### Nota

Los datos se utilizan tanto para la herramienta sostenida por el robot como para la herramienta estacionaria. En el caso de la herramienta sostenida por el robot los datos describen la carga de la herramienta. En el caso de la herramienta estacionaria, los datos describen la carga de la pinza sostenida por el robot.

#### Herramienta sostenida por el robot:

La carga de la herramienta, es decir:

- La masa (peso) de la herramienta en kg.
- El centro de gravedad de la carga de la herramienta (x, y, z) en mm, expresado en el sistema de coordenadas de la muñeca.
- La orientación de los ejes de inercia principales de momento de la herramienta expresada en el sistema de coordenadas de la muñeca.
- Los momentos de inercia alrededor de los ejes internos del momento en  $\text{kgm}^2$ . Si todos los componentes de inercia están definidos con el valor 0  $\text{kgm}^2$ , la herramienta se gestiona como una masa puntual.



xx110000519

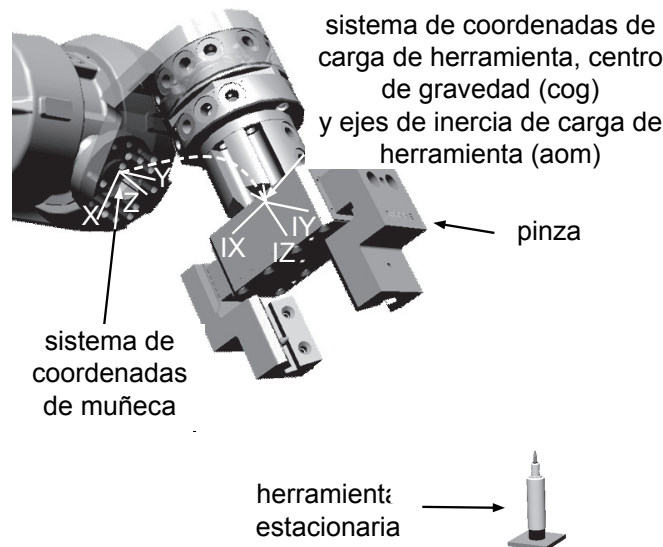
#### Herramienta estacionaria:

La carga de la pinza que sostiene el objeto de trabajo:

- La masa (peso) de la pinza movida en kg.
- El centro de gravedad de la pinza movida (x, y, z) en mm, expresado en el sistema de coordenadas de la muñeca.

Continúa en la página siguiente

- La orientación de los ejes de inercia principales de momento de la pinza movida expresada en el sistema de coordenadas de la muñeca.
- Los momentos de inercia alrededor de los ejes internos del momento en  $\text{kgm}^2$ . Si todos los componentes de inercia están definidos con el valor 0  $\text{kgm}^2$ , la pinza se gestiona como una masa puntual.



xx110000520



#### Nota

Sólo debe especificar la carga de la herramienta/pinza en `tooldata`. La carga útil manejada por la pinza se conecta y desconecta mediante la instrucción `GripLoad` y se define con un `loaddata`

En lugar de utilizar la instrucción `GripLoad` es posible definir y utilizar `tooldata` diferentes para la *pinza con la pieza de trabajo pinzada* y la *pinza sin pieza de trabajo*.

#### Resumen

La posición y orientación de TCP en `tooldata` se definen en el sistema de coordenadas de muñeca en una herramienta sostenida por el robot.

La posición y orientación de TCP en `tooldata` se definen en el sistema de coordenadas mundo en una herramienta sostenida por el robot.

La parte `loaddata` en `tooldata` está en todos los casos relacionada en el sistema de coordenadas de muñeca, independientemente de si se utiliza una herramienta sostenida por el robot (para describir la herramienta) o una herramienta estacionaria (para describir la pinza).

Continúa en la página siguiente

## 3 Tipos de datos

---

### 3.87 tooldata - Datos de herramienta

RobotWare - OS

Continuación

---

#### Ejemplos básicos

Los siguientes ejemplos ilustran el tipo de dato `tooldata`:

##### Ejemplo 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]];
```

La herramienta se describe utilizando los valores siguientes:

- El robot sostiene la herramienta.
- El TCP está situado en un punto a 223,1 mm en línea recta de la brida de montaje y a 97,4 mm a lo largo del eje X del sistema de coordenadas de la muñeca.
- Las direcciones X y Z de la herramienta se giran 45° en relación con la dirección Y en el sistema de coordenadas de la muñeca.
- La masa de la herramienta es de 5 kg.
- El centro de gravedad está situado en un punto a 75 mm en línea recta de la brida de montaje y 23 mm a lo largo del eje X del sistema de coordenadas de la muñeca.
- Es posible considerar la carga como una masa puntual, es decir, sin ningún momento de inercia.

##### Ejemplo 2

```
gripper.tframe.trans.z := 225.2;
```

El TCP de la herramienta, `gripper` se ajusta a 225.2 en la dirección Z.

---

#### Limitaciones

Los datos de la herramienta deben definirse como variables persistentes (`PERS`) y no deben definirse desde dentro de una rutina. De esta forma, los valores se guardan al guardar el programa y se recuperan al cargarlo.

Los argumentos de datos de herramienta de cualquier instrucción de movimiento deben ser sólo del tipo persistente completo (ni elementos de matriz ni componentes de registro).

---

#### Datos predefinidos

La herramienta `tool0` define el sistema de coordenadas de la muñeca, cuyo origen es el centro de la brida de montaje. `tool0` está siempre disponible desde el programa, pero no puede ser modificada en ningún momento (está almacenada en el módulo de sistema `BASE`).

```
PERS tooldata tool0 := [ TRUE, [ [0, 0, 0], [1, 0, 0 ,0] ], [0.001,
[0, 0, 0.001], [1, 0, 0, 0], 0, 0, 0] ];
```

---

#### Estructura

```
< dataobject of tooldata >
  < robhold of bool >
  < tframe of pose >
    < trans of pos >
      < x of num >
      < y of num >
      < z of num >
```

Continúa en la página siguiente

---

```

< 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 >

```

#### Información relacionada

| Para obtener más información sobre        | Consulte   |
|---|--|
| Instrucciones de posicionamiento          | <i>Manual de referencia técnica - RAPID Overview, sección Resumen sobre RAPID - Movimiento</i>                         |
| Sistemas de coordenadas                   | <i>Manual de referencia técnica - RAPID Overview, sección Principios de movimiento y E/S - Sistemas de coordenadas</i> |
| Definición de una carga útil para robots  | <a href="#">GripLoad - Define la carga útil de un robot en la página 244</a>   |
| Definición de datos de carga              | <a href="#">loaddata - Datos de carga en la página 1694</a>  |
| Definición de datos de objetos de trabajo | <a href="#">wobjdata - Datos del objeto de trabajo en la página 1823</a>   |

### 3 Tipos de datos

---

3.88 tpnum - Número de ventana del FlexPendant  
RobotWare - OS

#### 3.88 tpnum - Número de ventana del FlexPendant

---

##### Utilización

tpnum se utiliza para representar la ventana del FlexPendant con una constante simbólica.

##### Descripción

La constante tpnum se ha diseñado para su uso con la instrucción TPShow.

##### Ejemplos básicos

El siguiente ejemplo ilustra el tipo de dato tpnum:

##### Ejemplo 1

```
TPShow TP_LATEST;
```

La última ventana usada en el FlexPendant antes de la ventana actual del FlexPendant será la que se active tras la ejecución de esta instrucción.

##### Datos predefinidos

Se ha predefinido la constante simbólica siguiente para el tipo de dato tpnum. Puede usarla con la instrucción TPShow.

| Valor | Constante simbólica | Comentario                             |
|-------|---------------------|--|
| 2     | TP_LATEST           | Última ventana usada en el FlexPendant |

##### Características

tpnum es un tipo de dato de alias de num y por tanto hereda sus características.

##### Información relacionada

| Para obtener más información sobre                 | Consulte   |
|--|--|
| Tipos de datos en general, tipos de datos de alias | <i>Manual de referencia técnica - RAPID Overview, sección Características básicas - Tipos de datos</i> |
| Comunicación a través del FlexPendant              | <i>Manual de referencia técnica - RAPID Overview, sección Resumen sobre RAPID - Comunicación</i>       |
| Cambio de ventana en el FlexPendant                | <a href="#">TPShow - Cambia de ventana en el FlexPendant en la página 891</a>                          |

## 3.89 trapdata - Datos de interrupción para la rutina TRAP actual

### Utilización

trapdata (*trap data*) se utiliza para contener los datos de interrupción que provocan la ejecución de la rutina TRAP actual.

Debe utilizarse en las rutinas TRAP generadas por la instrucción IError, antes del uso de la instrucción ReadErrData.

### Descripción

Los datos del tipo trapdata representan información interna relacionada con la interrupción que provocó la ejecución de la rutina TRAP actual. Su contenido depende del tipo de interrupción.

### Ejemplos básicos

El siguiente ejemplo ilustra el tipo de dato trapdata:

#### Ejemplo 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
```

Cuando se detecta un error con la rutina TRAP trap\_err, el dominio, el número y el tipo del error se almacenan en las variables adecuadas sin valor, del tipo trapdata.

### Características

trapdata es un tipo de dato sin valor.

### Información relacionada

| Para obtener más información sobre                            | Consulte  |
|---|---|
| Resumen de interrupciones                                     | <i>Manual de referencia técnica - RAPID Overview</i>  |
| Más información sobre la gestión de interrupciones            | <i>Manual de referencia técnica - RAPID Overview</i>  |
| Tipos de datos sin valor                                      | <i>Manual de referencia técnica - RAPID Overview</i>  |
| Solicitud de una interrupción para errores                    | <a href="#">IError - Solicita una interrupción para errores en la página 257</a>                        |
| Obtención de datos de interrupción para la rutina TRAP actual | <a href="#">GetTrapData - Obtiene datos de interrupción para la rutina TRAP actual en la página 240</a> |
| Obtención de información sobre un error                       | <a href="#">ReadErrData - Obtiene información sobre un error en la página 586</a>                       |

Continúa en la página siguiente

### 3 Tipos de datos

---

3.89 trapdata - Datos de interrupción para la rutina TRAP actual

RobotWare - OS

Continuación

| Para obtener más información sobre | Consulte  |
|------------------------------------|---|
| <i>Advanced RAPID</i>              | <i>Especificaciones del producto - Controller software IRC5</i> |



### 3.90 triggdata - Eventos de posicionamiento, trigg

#### Utilización

`triggdata` se utiliza para almacenar datos acerca de un evento de posicionamiento durante un movimiento del robot.

Un evento de posicionamiento puede tener la forma de un parámetro en una señal de salida o puede significar la ejecución de una rutina de interrupción en una posición específica a lo largo de la trayectoria de movimiento del robot.

#### Descripción

Para definir las condiciones de las mediciones respectivas de un evento de posicionamiento, se utilizan variables del tipo `triggdata`. Los datos contenidos en la variable se forman en el programa mediante una instrucción `TriggIO`, `TriggEquip`, `TriggCheckIO`, `TriggInt`, `TriggSpeed` o `TriggRampAO` y se utilizan desde una instrucción `TriggL`, `TriggC` o `TriggJ`.

#### Ejemplos básicos

El siguiente ejemplo ilustra el tipo de dato `triggdata` :

##### Ejemplo 1

```
VAR triggdata gunoff;
TriggIO gunoff, 0,5 \DOP:=gun, 0;
TriggL p1, v500, gunoff, fine, gun1;
```

La señal digital de salida `gun` cambia al valor 0 cuando el TCP se encuentra en una posición a 0,5 mm del punto `p1`.

#### Características

`triggdata` es un tipo de dato sin valor.

#### Información relacionada

| Para obtener más información sobre              | Consulte  |
|---|---|
| Definición de disparos                          | <p><a href="#">TriggIO</a> - Define un evento de E/S de posición o tiempo fijos cerca de un punto de paro en la página 926</p> <p><a href="#">TriggEquip</a> - Define un evento de E/S basado en la posición y el tiempo en la trayectoria en la página 914</p> <p><a href="#">TriggCheckIO</a> - Define una comprobación de E/S en una posición fija en la página 904</p> <p><a href="#">TriggInt</a> - Define una interrupción dependiente de una posición en la página 921</p> |
| Utilización de disparos                         | <p><a href="#">TriggL</a> - Movimiento lineal del robot con eventos en la página 940</p> <p><a href="#">TriggC</a> - Movimiento circular del robot con eventos en la página 895</p> <p><a href="#">TriggJ</a> - Movimientos de ejes del robot a partir de eventos en la página 932</p>  |
| Características de los tipos de datos sin valor | Manual de referencia técnica - RAPID Overview, sección Características básicas - Tipos de datos   |

## 3 Tipos de datos

---

3.91 triggios - Eventos de posicionamiento, trigg  
RobotWare - OS

### 3.91 triggios - Eventos de posicionamiento, trigg

---

#### Utilización

Se utiliza `triggios` para almacenar datos acerca de un evento de posicionamiento durante un movimiento del robot. Si el evento de posicionamiento está distribuido en una posición específica de la trayectoria, se otorga un valor específico a una señal de salida.

#### Descripción

`triggios` se utiliza para definir condiciones y acciones para el establecimiento de una señal digital de salida, un grupo de señales digitales de salida o una señal analógica de salida en una posición fija a lo largo de la trayectoria de movimiento del robot.

#### Componentes

`used`

Tipo de dato: `bool`

Define si el elemento de la matriz debe utilizarse o no.

`distance`

Tipo de dato: `num`

Define la posición de la trayectoria en la que debe producirse el evento de E/S. Se especifica como la distancia en mm (valor positivo) desde el punto final de la trayectoria de movimiento si el componente `start` tiene el valor `FALSE`.

`start`

Tipo de dato: `bool`

Cambia a `TRUE` si la distancia comienza en el punto inicial del movimiento en lugar del punto final.

`equiplag`

*Equipment Lag*

Tipo de dato: `num`

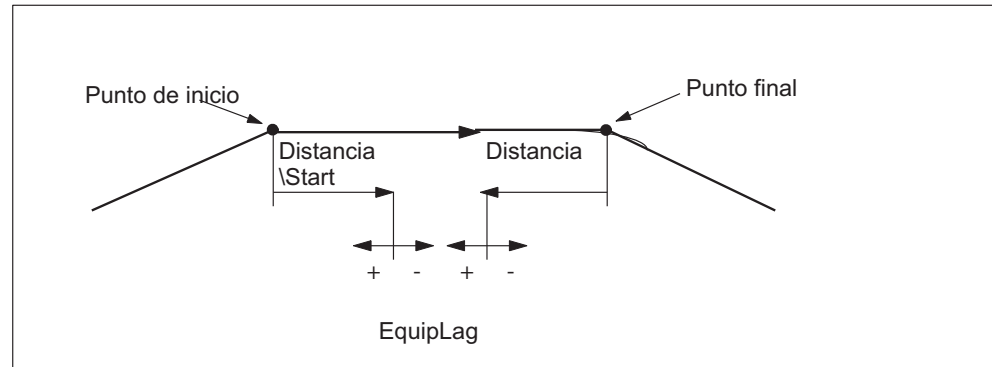
Especifica el retardo del equipo externo, en segundos.

Para la compensación del retardo de los equipos externos, utilice un valor de argumento positivo. Un valor positivo significa que la señal de E/S es activada por el sistema de robot en el momento especificado, antes de que el TCP alcance físicamente la distancia especificada respecto del punto de inicio o final del movimiento.

Un valor negativo significa que la señal de E/S es activada por el sistema de robot en el momento especificado, después de que el TCP físico haya sobrepasado la distancia especificada respecto del punto de inicio o final del movimiento.

*Continúa en la página siguiente*

En la figura se muestra el uso del componente `equiplag`.



xx080000173

signalname

Tipo de dato: string

El nombre de la señal que debe cambiar. Debe ser una señal digital de salida, un grupo de señales digitales de salida o una señal analógica de salida.

setvalue

Tipo de dato: num

El valor deseado para la señal de salida (dentro del rango permitido para la señal actual).

xxx

Tipo de dato: num

Este componente no se utiliza en este momento. Ha sido añadido con el fin de poder añadir funcionalidad en versiones futuras sin perder la compatibilidad.

## Ejemplos

El siguiente ejemplo ilustra el tipo de dato `triggios`:

### Ejemplo 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;
```

La señal `gun` se activa cuando el TCP está 3 mm después del punto `p1`.

Continúa en la página siguiente

### 3 Tipos de datos

---

#### 3.91 triggios - Eventos de posicionamiento, trigg

RobotWare - OS

Continuación

---

#### Estructura

```
<dataobject of triggios>  
  <used of bool>  
  <distance of num>  
  <start of bool>  
  <equiplag of num>  
  <signalname of string>  
  <setvalue of num>  
  <xxx of num>
```

---

#### Información relacionada

| Para obtener más información sobre                | Consulte   |
|---|--|
| Eventos de posicionamiento, trigg                 | <a href="#">triggiosdnum - Eventos de posicionamiento, trigg en la página 1809</a>             |
| Movimientos lineales del robot con eventos de E/S | <a href="#">TriggLIOs - Movimientos lineales del robot con eventos de E/S en la página 957</a> |

---

### 3.92 triggiosdnum - Eventos de posicionamiento, trigg

---

#### Utilización

`triggiosdnum` se utiliza para almacenar datos acerca de un evento de posicionamiento durante un movimiento del robot. Si el evento de posicionamiento está distribuido en una posición específica de la trayectoria, una señal de salida cambia a un valor específico.

---

#### Descripción

`triggiosdnum` se utiliza para definir condiciones y acciones para el establecimiento de una señal digital de salida, un grupo de señales digitales de salida o una señal analógica de salida en una posición fija a lo largo de la trayectoria de movimiento del robot.

---

#### Componentes

`used`

Tipo de dato: `bool`

Define si el elemento de la matriz debe utilizarse o no.

`distance`

Tipo de dato: `num`

Define la posición de la trayectoria en la que debe producirse el evento de E/S. Se especifica como la distancia en mm (valor positivo) desde el punto final de la trayectoria de movimiento si el componente `start` tiene el valor FALSE.

`start`

Tipo de dato: `bool`

Cambia a TRUE si la distancia comienza en el punto inicial del movimiento en lugar del punto final.

`equiplag`

*Equipment Lag*

Tipo de dato: `num`

Especifica el retardo del equipo externo, en segundos.

Para la compensación del retardo de los equipos externos, utilice un valor de argumento positivo. Un valor positivo significa que la señal de E/S es activada por el sistema de robot en el momento especificado, antes de que el TCP alcance físicamente la distancia especificada respecto del punto de inicio o final del movimiento.

Un valor negativo significa que la señal de E/S es activada por el sistema de robot en el momento especificado, después de que el TCP físico haya sobrepasado la distancia especificada respecto del punto de inicio o final del movimiento.

`signalname`

Tipo de dato: `string`

*Continúa en la página siguiente*

### 3 Tipos de datos

---

#### 3.92 triggiosdnum - Eventos de posicionamiento, trigg

RobotWare - OS

Continuación

El nombre de la señal que debe cambiar. Debe ser una señal digital de salida, un grupo de señales digitales de salida o una señal analógica de salida.

setvalue

Tipo de dato: dnum

El valor deseado para la señal de salida (dentro del rango permitido para la señal actual).

xxx

Tipo de dato: num

Este componente no se utiliza en este momento. Ha sido añadido con el fin de poder añadir funcionalidad en versiones futuras sin perder la compatibilidad.

---

#### Ejemplos

El siguiente ejemplo ilustra el tipo de dato triggiosdnum:

##### Ejemplo 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;
```

La señal go\_gun se activa cuando el TCP está 3 mm después del punto p1.

---

#### Estructura

```
<dataobject of triggiosdnum>
  <used of bool>
  <distance of num>
  <start of bool>
  <equiplag of num>
  <signalname of string>
  <setvalue of dnum>
  <xxx of num>
```

---

#### Información relacionada

| Para obtener más información sobre                | Consulte   |
|---|--|
| Eventos de posicionamiento, trigg                 | <a href="#">triggios - Eventos de posicionamiento, trigg en la página 1806</a>                 |
| Movimientos lineales del robot con eventos de E/S | <a href="#">TriggLIOs - Movimientos lineales del robot con eventos de E/S en la página 957</a> |

### 3.93 triggmode - Disparar modo de acción

#### Utilización

`triggmode` se utiliza para especificar diferentes modos de acción al definir disparadores.

#### Descripción

El uso previsto de una constante `triggmode` es definir el modo para las instrucciones utilizadas para definir los disparadores.

#### Ejemplos básicos

Los siguientes ejemplos ilustran el tipo de dato `triggmode`:

##### Ejemplo 1

```
CONNECT intnol WITH trap1;
TriggInt trigg1, Distance:=17, intnol \Inhib:=inhibit
  \Mode:=TRIGG_MODE1;
TriggL p1, v500, trigg1, z50, gun1;
```

La rutina de interrupción `trap1` se ejecuta cuando TCP se encuentra en una posición 17 mm anterior al punto `p1` si el indicador de la variable persistente `inhibit` fuera `TRUE` (el modo `TRIGG_MODE1` invierte el valor leído del indicador `inhibit`).

##### Ejemplo 2

```
TriggEquip trigg1, 17, 0 \GOp:=go1, SetValue:=5 \Inhib:=inhibit
  \Mode:=TRIGG_MODE2;
TriggL p1, v500, trigg1, z50, gun1;
```

Si el indicador persistente `inhibit` es `FALSE` cuando el TCP está en una posición de 17 mm antes del punto `p1`, la señal de E/S `go1` cambia al valor especificado en `SetValue`. Si la variable persistente `inhibit` es `TRUE`, no se realiza ninguna acción (se mantiene el valor de la señal de E/S `go1`).

##### Ejemplo 3

```
TriggEquip trigg1, 17, 0 \GOp:=go1, SetValue:=0 \Inhib:=inhibit
  \InhibSetValue:=setDnum \Mode:=TRIGG_MODE3;
TriggL p1, v500, trigg1, z50, gun1;
```

Si el indicador persistente `inhibit` es `TRUE` cuando el TCP está en una posición de 17 mm antes del punto `p1`, la señal de E/S `go1` cambia al valor leído de `dnum` de la variable persistente `setDnum`. Si `inhibit` es `FALSE`, no se realiza ninguna acción (se mantiene el valor de la señal de E/S `go1`).

##### Ejemplo 4

```
TriggEquip trigg1, 17, 0 \GOp:=go1, SetValue:=5 \Inhib:=inhibit
  \Mode:=TRIGG_MODE3;
TriggL p1, v500, trigg1, z50, gun1;
```

Si el indicador persistente `inhibit` fuera `TRUE` cuando TCP se encuentra en una posición de 17 mm antes del punto `p1`, la señal de E/S `go1` se establece en el valor especificado en `SetValue` (5 pulgadas en este ejemplo). Si `inhibit` fuera `FALSE`, no se realizará ninguna acción (mantenga el valor de la señal de E/S `go1`).

*Continúa en la página siguiente*

### 3 Tipos de datos

#### 3.93 triggmode - Disparar modo de acción

RobotWare - OS

Continuación

#### Datos predefinidos

Se han definido las constantes simbólicas siguientes para el tipo de dato `triggmode`. Puede usarlas para especificar diferentes modos de acción al definir disparadores.

| Valor | Constante simbólica | Comentario   |
|-------|---------------------|--|
| 1     | TRIGG_MODE1         | <p>Puede usarse en las instrucciones <code>TriggCheckIO</code>, <code>TriggEquip</code>, <code>TriggIO</code>, <code>TriggInt</code>, <code>TriggSpeed</code> y <code>TriggRampAO</code>.</p> <p>Invertir el valor leído de la variable persistente utilizada en el argumento opcional <code>Inhib</code>.</p>   |
| 2     | TRIGG_MODE2         | <p>Puede usarse en las instrucciones <code>TriggEquip</code>, <code>TriggIO</code>, <code>TriggSpeed</code> y <code>TriggRampAO</code> si utilizan el argumento opcional <code>Inhib</code>.</p> <p>Si el valor real del indicador especificado utilizado en <code>Inhib</code> es <code>TRUE</code> en la posición y el tiempo para definir la señal, la señal de E/S especificada no se actualiza (ninguna acción).</p>  |
| 3     | TRIGG_MODE3         | <p>El modo solo puede utilizarse junto con el argumento opcional <code>Inhib</code> en las instrucciones <code>TriggEquip</code> y <code>TriggIO</code>.</p> <p>Si el valor real del indicador especificado utilizado en <code>Inhib</code> es <code>FALSE</code>, la señal de E/S no se actualiza (ninguna acción).</p> <p>Si se utiliza solo con <code>Inhib</code> y el indicador especificado utilizado en <code>Inhib</code> fuera <code>TRUE</code> en la posición, la señal de E/S se establece en el valor especificado en el argumento <code>SetValue</code> o <code>SetDvalue</code>.</p> <p>Si se utiliza con <code>Inhib</code> e <code>InhibSetValue</code>: Los argumentos <code>SetValue</code> y <code>SetDvalue</code> ya no vuelven a considerarse. Si el valor real del indicador especificado utilizado en <code>Inhib</code> fuera <code>TRUE</code>, entonces la señal de E/S se establece en el valor que la variable persistente utilizada en el argumento opcional <code>InhibSetValue</code> tenga en la posición.</p> |

#### Características

`triggmode` es un tipo de dato de alias de `num` y por tanto hereda sus características.

Continúa en la página siguiente



#### Información relacionada

| Para obtener más información sobre                         | Consulte   |
|--|--|
| Utilización de disparadores                                | <p><a href="#">TriggL - Movimiento lineal del robot con eventos en la página 940</a></p> <p><a href="#">TriggC - Movimiento circular del robot con eventos en la página 895</a></p> <p><a href="#">TriggJ - Movimientos de ejes del robot a partir de eventos en la página 932</a></p>   |
| Definición de disparadores                                 | <p><a href="#">TriggEquip - Define un evento de E/S basado en la posición y el tiempo en la trayectoria en la página 914</a></p> <p><a href="#">TriggInt - Define una interrupción dependiente de una posición en la página 921</a></p> <p><a href="#">TriggIO - Define un evento de E/S de posición o tiempo fijos cerca de un punto de paro en la página 926</a></p> <p><a href="#">TriggRampAO - Define un evento AO de rampa de posición fija en la trayectoria en la página 965</a></p> <p><a href="#">TriggSpeed - Define la velocidad del TCP en proporción a una salida analógica con un evento de escala fija de posición-tiempo en la página 973</a></p> |
| Definición de una comprobación de E/S en una posición fija | <p><a href="#">TriggCheckIO - Define una comprobación de E/S en una posición fija en la página 904</a></p>   |
| Almacenamiento de datos de disparo                         | <p><a href="#">triggdata - Eventos de posicionamiento, trigg en la página 1805</a></p>   |
| Tipos de datos en general, tipos de datos de alias         | <p><i>Manual de referencia técnica - RAPID Overview, sección Características básicas - Tipos de datos</i></p>  |

## 3 Tipos de datos

---

### 3.94 triggstrgo - Positioning events, trigg RobotWare - OS

#### 3.94 triggstrgo - Positioning events, trigg

---

##### Utilización

`triggstrgo`(*trigg stringdig group output*) se utiliza para almacenar datos acerca de un evento de posicionamiento durante un movimiento del robot. Si el evento de posicionamiento está distribuido en una posición específica de la trayectoria, un grupo de señales digitales de salida cambia a un valor específico.

---

##### Descripción

`triggstrgo` se utiliza para definir condiciones y acciones para el establecimiento de señales digitales de salida en una posición fija a lo largo de la trayectoria de movimiento del robot.

---

##### Componentes

`used`

Tipo de dato: `bool`

Define si el elemento de la matriz debe utilizarse o no.

`distance`

Tipo de dato: `num`

Define la posición de la trayectoria en la que debe producirse el evento de E/S. Se especifica como la distancia en mm (valor positivo) desde el punto final de la trayectoria de movimiento si el componente `start` tiene el valor `FALSE`.

`start`

Tipo de dato: `bool`

Cambia a `TRUE` si la distancia comienza en el punto inicial del movimiento en lugar del punto final.

`equiplag`

*Equipment Lag*

Tipo de dato: `num`

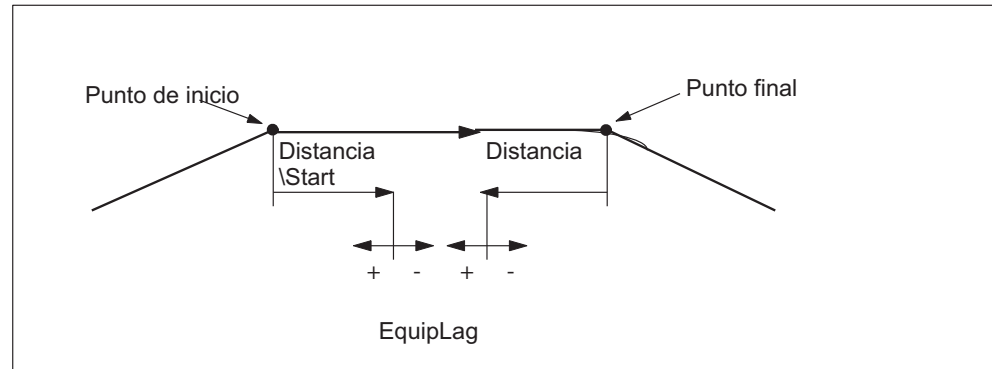
Especifica el retardo del equipo externo, en segundos.

Para la compensación del retardo de los equipos externos, utilice un valor de argumento positivo. Un valor positivo significa que la señal de E/S es activada por el sistema de robot en el momento especificado, antes de que el TCP alcance físicamente la distancia especificada respecto del punto de inicio o final del movimiento.

Un valor negativo significa que la señal de E/S es activada por el sistema de robot en el momento especificado, después de que el TCP físico haya sobrepasado la distancia especificada respecto del punto de inicio o final del movimiento.

*Continúa en la página siguiente*

En la figura se muestra el uso del componente `equiplag`.



xx080000173

signalname

Tipo de dato: `string`

El nombre de la señal que debe cambiar. Debe ser un nombre de señal de salida de grupo.

setvalue

Tipo de dato: `stringdig`

El valor deseado para la señal de salida (dentro del rango permitido para la salida digital de grupo actual). El uso del tipo de dato `stringdig` hace posible utilizar valores hasta el 4.294.967.295, que es el valor máximo que puede tener un grupo de señales digitales (el sistema admite como máximo 32 señales en una señal de grupo).

xxx

Tipo de dato: `num`

Este componente no se utiliza en este momento. Ha sido añadido con el fin de poder añadir funcionalidad en versiones futuras sin perder la compatibilidad.

## Ejemplos

El siguiente ejemplo ilustra el tipo de dato `triggstrgo`:

### Ejemplo 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;
```

*Continúa en la página siguiente*

### 3 Tipos de datos

---

#### 3.94 triggstrgo - Positioning events, trigg

RobotWare - OS

Continuación

La señal `gun` recibe el valor 4.294.967.295 cuando el TCP está 3 mm después del punto `p1`.

---

#### Estructura

```
<dataobject of triggstrgo>  
<used of bool>  
<distance of num>  
<start of bool>  
<equiplag of num>  
<signalname of string>  
<setvalue of stringdig>  
<xxx of num>
```

---

#### Información relacionada

| Para obtener más información sobre                   | Consulte   |
|--|--|
| Movimientos lineales del robot con eventos de E/S    | <a href="#">TriggLIOs - Movimientos lineales del robot con eventos de E/S en la página 957</a>       |
| Comparar dos cadenas que sólo contienen dígitos      | <a href="#">StrDigCmp - Comparar dos cadenas que sólo contienen dígitos en la página 1494</a>        |
| Operaciones aritméticas con tipos de datos stringdig | <a href="#">StrDigCalc - Operaciones aritméticas con el tipo de dato stringdig en la página 1491</a> |

## 3.95 tsp\_status: Estado de panel de selección de tareas

## Utilización

tsp\_status se utiliza para reflejar el estado del panel de selección de tareas Panel de selección de tareas de FlexPendant.

## Descripción

Con las funciones TaskIsActive y GetTSPStatuses posible leer el estado actual del Panel de selección de tareas de FlexPendant.

## Ejemplos básicos

El siguiente ejemplo ilustra el tipo de dato tsp\_status:

## Ejemplo 1

```
VAR tsp_status tspstatus;
...
tspstatus:=GetTSPStatus("MYTASK");
IF tspstatus >= TSP_NORMAL_UNCHECKED AND tspstatus <=
  TSP_SEMISTATIC_UNCHECKED THEN
  TPWrite "Task MYTASK is unchecked in the Task Selection Panel";
ELSEIF tspstatus >= TSP_NORMAL_CHECKED THEN
  TPWrite "Task MYTASK is checked in the Task Selection Panel";
ELSE
  TPWrite "Task MYTASK is unchecked in TSP due to execution in
    service routine";
ENDIF
```

Este ejemplo de programa investiga si la tarea MYTASK del programa está activada o desactivada en el Panel de selección de tareas de FlexPendant.

## Datos predefinidos

Las siguientes constantes de tipo tsp\_status están predefinidas:

| Constante de RAPID          | Valor | Descripción   |
|-----------------------------|-------|---|
| TSP_STATUS_NOT_NORMAL_TASK  | 0     | TaskIsActive. La tarea es semiestática o estática, no es una tarea normal.  |
| TSP_STATUS_DEACT            | 1     | TaskIsActive. La tarea normal está desactivada en el Panel de selección de tareas.  |
| TSP_STATUS_DEACT_SERV_ROUT  | 2     | TaskIsActive. La tarea normal está desactivada en el Panel de selección de tareas porque otra tarea está ejecutando una rutina de servicio. |
| TSP_STATUS_ACT              | 3     | TaskIsActive. La tarea normal está activa en el Panel de selección de tareas.   |
| TSP_UNCHECKED_RUN_SERV_ROUT | 10    | GetTSPStatus. La tarea normal no está marcada en el Panel de selección de tareas porque otra tarea está ejecutando una rutina de servicio.  |
| TSP_NORMAL_UNCHECKED        | 11    | GetTSPStatus. La tarea normal no está marcada en el Panel de selección de tareas.   |

*Continúa en la página siguiente*

## 3 Tipos de datos

### 3.95 tsp\_status: Estado de panel de selección de tareas

RobotWare - OS

Continuación

| Constante de RAPID       | Valor | Descripción   |
|--------------------------|-------|---|
| TSP_STATIC_UNCHECKED     | 12    | GetTSPStatus. La tarea estática no está marcada en el Panel de selección de tareas.     |
| TSP_SEMISTATIC_UNCHECKED | 13    | GetTSPStatus. La tarea semiestática no está marcada en el Panel de selección de tareas. |
| TSP_NORMAL_CHECKED       | 14    | GetTSPStatus. La tarea normal está marcada en el Panel de selección de tareas.          |
| TSP_STATIC_CHECKED       | 15    | GetTSPStatus. La tarea estática está marcada en el Panel de selección de tareas.        |
| TSP_SEMISTATIC_CHECKED   | 16    | GetTSPStatus. La tarea semiestática está marcada en el Panel de selección de tareas.    |

#### Características

tsp\_status es un tipo de dato de alias de num y por tanto hereda sus características.

#### Información relacionada

| Para obtener más información sobre                          | Consulte  |
|---|---|
| Obtener el estado del panel de selección de tareas actuales | <a href="#">GetTSPStatus: Obtener el estado del panel de selección de tareas actuales en la página 1323</a> |
| Comprobar si una tarea normal está activa                   | <a href="#">TaskIsActive: Comprobar si una tarea normal está activa en la página 1519</a>                   |

### 3.96 tunetype - Tipo de ajuste de servo

#### Utilización

`tunetype` se utiliza para representar un entero con una constante simbólica para tipos distintos de ajuste de servo.

#### Descripción

Las constantes `tunetype` se han diseñado para su uso como argumento de la instrucción `TuneServo`.

#### Ejemplos básicos

El siguiente ejemplo ilustra el tipo de dato `tunetype`:

##### Ejemplo 1

```
TuneServo MHA160R1, 1, 110 \Type:= TUNE_KP;
```

#### Datos predefinidos

Se han definido las constantes simbólicas siguientes para el tipo de dato `tunetype`. Puede usarlas como argumento de la instrucción `TuneServo`.

| Valor | Constante simbólica | Comentario   |
|-------|---------------------|--|
| 0     | TUNE_DF             | Reduce el riesgo de sobrepasar posiciones                |
| 1     | TUNE_KP             | Afecta a la ganancia del control de posición             |
| 2     | TUNE_KV             | Afecta a la ganancia del control de velocidad            |
| 3     | TUNE_TI             | Afecta al tiempo de integración del control de velocidad |
| 4     | TUNE_FRIC_LEV       | Afecta al nivel de compensación de fricción              |
| 5     | TUNE_FRIC_RAMP      | Afecta a la pendiente de compensación de fricción        |
| 6     | TUNE_DG             | Reduce el riesgo de sobrepasar posiciones                |
| 7     | TUNE_DH             | Reduce las vibraciones con cargas pesadas                |
| 8     | TUNE_DI             | Reduce los errores de trayectoria                        |
| 9     | TUNE_DK             | Sólo para uso interno de ABB                             |
| 10    | TUNE_DL             | Sólo para uso interno de ABB                             |

#### Características

`tunetype` es un tipo de dato de alias de `num` y por tanto hereda sus características.

#### Información relacionada

| Para obtener más información sobre                 | Consulte   |
|--|--|
| Tipos de datos en general, tipos de datos de alias | <i>Manual de referencia técnica - RAPID Overview, sección Características básicas - Tipos de datos</i> |
| Uso del tipo de dato <code>tunetype</code>         | <a href="#">TuneServo - Ajuste de servos en la página 994</a>  |

## 3 Tipos de datos

---

### 3.97 uishownum - ID de instancia para UIShow

RobotWare - OS

### 3.97 uishownum - ID de instancia para UIShow

---

#### Utilización

`uishownum` es el tipo de dato utilizado para el parámetro `InstanceId` de la instrucción `UIShow`. Se usa para identificar una vista en el `FlexPendant`.

---

#### Descripción

Cuando una variable persistente del tipo `uishownum` se usa con la instrucción `UIShow`, recibe un valor específico que identifica la vista iniciada en el `FlexPendant`. A continuación, la variable persistente se utiliza en todas las operaciones realizadas con esa vista, como iniciar de nuevo la vista, modificar la vista, etc.

---

#### Ejemplos

El siguiente ejemplo ilustra el tipo de dato `uishownum`:

##### Ejemplo 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;
```

El código de la parte superior iniciará una vista de la aplicación `MyAppl` con el comando de inicialización `Cmd1`. El token utilizado para identificar la vista se guarda en el parámetro `myinstance`.

---

#### Características

`uishownum` es un tipo de dato de alias de `num` y por tanto hereda sus propiedades.

---

#### Información relacionada

| Para obtener más información sobre | Consulte  |
|------------------------------------|---|
| UIShow                             | <a href="#">UIShow - Visualización de interfaz de usuario en la página 1017</a> |



### 3.98 weavestartdata - Datos de inicio de oscilación

#### Utilización

weavestartdata se utiliza para controlar la oscilación estacionaria durante el inicio y reinicio de un proceso en CAP.

weavestartdata es un componente de capdata y define las propiedades de la oscilación estacionaria al inicio o al reinicio de un proceso CAP:

- si habrá oscilación estacionaria al inicio (`active`)
- anchura de la oscilación estacionaria (`width`)
- dirección de la trayectoria en relación con la dirección (`dir`)
- frecuencia de la oscilación estacionaria (`cycle_time`)

La oscilación estacionaria siempre utiliza oscilación geométrica con un patrón de zig zag; consulte [capweavedata - Datos de oscilación para CAP en la página 1638](#).

#### Componentes

##### active

Tipo de dato: `bool`

| Valor | Descripción  |
|-------|--|
| TRUE  | Realizar la oscilación estacionaria al inicio de un proceso CAP    |
| FALSE | NO realizar la oscilación estacionaria al inicio de un proceso CAP |

##### width

Tipo de dato: `num`

Define la amplitud de la oscilación estacionaria (mm).

##### dir

Tipo de dato: `num`

Define la dirección de la oscilación estacionaria en relación a la dirección de trayectoria (grados). Cero grados indica que la oscilación es perpendicular tanto respecto a la trayectoria como a la coordenada Z de la herramienta.

##### cycle\_time

Tipo de dato: `num`

Establece el tiempo total (en segundos) para un ciclo completo de oscilación estacionaria, es decir, determina la frecuencia de oscilación. La oscilación estacionaria durará hasta que se inicie el proceso, es decir, que se cumplan los criterios de supervisión de la fase START\_MAIN.

#### Sintaxis

```
< data object of weavestartdata >
  < active of bool >
  < width of num >
  < dir of num >
  < cycle_time of num >
```

*Continúa en la página siguiente*

### 3 Tipos de datos

---

3.98 weavestartdata - Datos de inicio de oscilación

*Continuous Application Platform (CAP)*

*Continuación*

---

#### Información relacionada

|  | Descrito en:  |
|--|---|
| Tipo de dato <code>capdata</code>      | <a href="#">capdata - Datos CAP en la página 1624</a>       |
| <i>Continuous Application Platform</i> | <i>Application manual - Continuous Application Platform</i> |

### 3.99 wobjdata - Datos del objeto de trabajo

---

#### Utilización

wobjdata se utiliza para describir el objeto de trabajo que el robot está soldando, procesando, moviendo por sí solo, etc.

---

#### Descripción

Si los objetos de trabajo están definidos en una instrucción de posicionamiento, la posición se basará en las coordenadas del objeto de trabajo. Las ventajas de hacerlo son las siguientes:

- Si se introducen manualmente los datos de posición, por ejemplo, mediante la programación fuera de línea, suele ser posible tomar los valores de un plano.
- Los programas pueden reutilizarse rápidamente después de cualquier cambio en la instalación del robot. Por ejemplo, si se cambia de posición un útil, sólo es necesario redefinir el sistema de coordenadas del usuario.
- También es posible compensar las variaciones en la forma en que el objeto de trabajo está fijado. Sin embargo, para estos fines se requerirá algún tipo de sensor para posicionar el objeto de trabajo.

Si se utiliza una herramienta estacionaria o ejes externos coordinados, es necesario definir el objeto de trabajo, ya que en este caso la trayectoria y la velocidad estarían relacionadas con el objeto de trabajo en lugar del TCP.

Los datos del objeto de trabajo pueden usarse también para los movimientos.

- El robot puede desplazarse en las direcciones del objeto de trabajo.
- La posición actual mostrada se basa en el sistema de coordenadas del objeto de trabajo.

---

#### Componentes

robhold

*robot hold*

Tipo de dato: bool

Define si el robot de la tarea de programa actual es el que está sosteniendo el objeto de trabajo:

- TRUE: El robot sostiene el objeto de trabajo, es decir, se está utilizando una herramienta estacionaria.
- FALSE: El robot no está sosteniendo el objeto de trabajo, es decir, el robot está sosteniendo la herramienta.

ufprog

*user frame programmed*

Tipo de dato: bool

Define si se está utilizando un sistema fijo de coordenadas del usuario:

- TRUE: Sistema de coordenadas del usuario fijo

*Continúa en la página siguiente*

## 3 Tipos de datos

---

### 3.99 wobdata - Datos del objeto de trabajo

RobotWare - OS

Continuación

- **FALSE**: Sistema móvil de coordenadas del usuario, es decir, se utilizan ejes externos coordinados. También debe usarse en los sistemas MultiMove que se utilizan en el modo semicoordinado o sincronizado coordinado.

ufmec

*user frame mechanical unit*

Tipo de dato: `string`

La unidad mecánica con la que se coordinan los movimientos del robot. Sólo se especifican en el caso de los sistemas móviles de coordenadas del usuario (`ufprog` es **FALSE**).

Especifica el nombre de la unidad mecánica definido en los parámetros del sistema, por ejemplo `orbit_a`.

uframe

*user frame*

Tipo de dato: `pose`

El sistema de coordenadas del usuario, es decir, la posición de la superficie o del útil de trabajo actual (consulte la figura siguiente):

- La posición del origen del sistema de coordenadas (x, y, z) en mm.
- La rotación del sistema de coordenadas, expresada como un cuaternio (q1, q2, q3, q4).

Si el robot es el que sostiene la herramienta, el sistema de coordenadas del usuario se define en el sistema de coordenadas mundo (en el sistema de coordenadas de la muñeca si se utiliza una herramienta estacionaria).

En el caso de una base de coordenadas de usuario móvil (`ufprog` con el valor **FALSE**), la base de coordenadas del usuario es definida continuamente por el sistema.

oframe

*object frame*

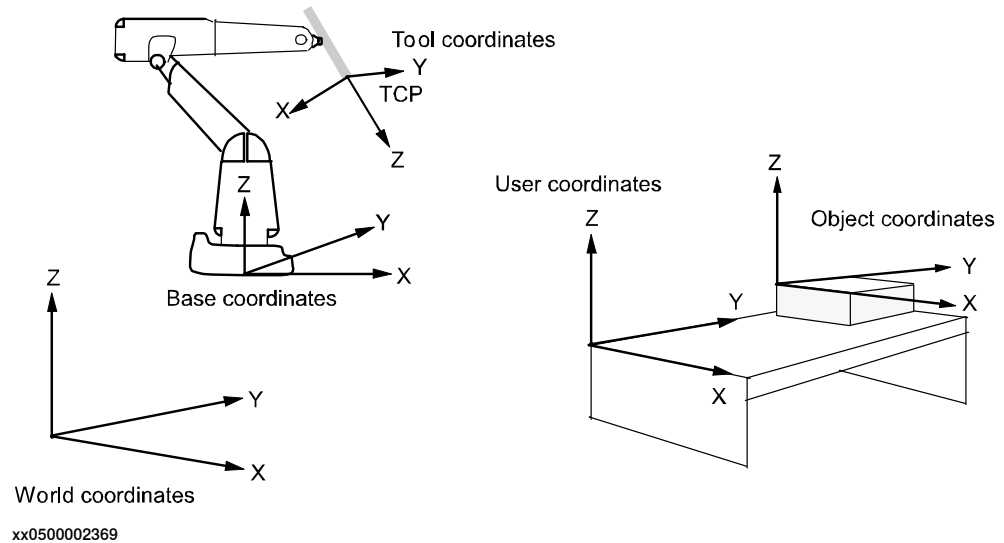
Tipo de dato: `pose`

El sistema de coordenadas del objeto, es decir, la posición del objeto de trabajo actual (consulte la figura siguiente):

- La posición del origen del sistema de coordenadas (x, y, z) en mm.
- La rotación del sistema de coordenadas, expresada como un cuaternio (q1, q2, q3, q4).

*Continúa en la página siguiente*

El sistema de coordenadas del objeto se define en el sistema de coordenadas del usuario.



#### Ejemplos básicos

El siguiente ejemplo ilustra el tipo de dato `wobjdata`:

##### Ejemplo 1

```
PERS wobjdata wobj2 :=[ FALSE, TRUE, "", [ [300, 600, 200], [1, 0, 0, 0], [0, 200, 30], [1, 0, 0, 0] ] ];
```

El objeto de trabajo de la figura anterior se describe utilizando los valores siguientes:

- El robot no sostiene el objeto de trabajo.
- Se utiliza el sistema fijo de coordenadas del usuario.
- El sistema de coordenadas del usuario no se gira y las coordenadas de su origen son  $x = 300$ ,  $y = 600$  y  $z = 200$  mm en el sistema de coordenadas mundo.
- El sistema de coordenadas del objeto no se gira y las coordenadas de su origen son  $x = 0$ ,  $y = 200$  y  $z = 30$  mm en el sistema de coordenadas del usuario.

```
wobj2.oframe.trans.z := 38.3;
```

- Se ajusta la posición del objeto de trabajo `wobj2` a 38,3 mm en la dirección Z.

#### Limitaciones

Los datos del objeto de trabajo deben definirse como variables persistentes (`PERS`) y no deben definirse desde dentro de una rutina. De esta forma, los valores se guardan al guardar el programa y se recuperan al cargarlo.

Los argumentos de datos de objeto de trabajo de cualquier instrucción de movimiento deben ser sólo del tipo persistente completo (ni elementos de matriz ni componentes de registro).

Continúa en la página siguiente

### 3 Tipos de datos

#### 3.99 wobjdata - Datos del objeto de trabajo

RobotWare - OS

Continuación

#### Datos predefinidos

Los datos de objeto de trabajo `wobj0` se definen de forma que el sistema de coordenadas del objeto coincida con el sistema de coordenadas mundo. El robot es el que sostiene el objeto de trabajo.

`wobj0` está siempre disponible desde el programa, pero no puede ser modificada en ningún momento (está almacenada en el módulo de sistema BASE).

```
PERS wobjdata wobj0 := [ FALSE, TRUE, "", [ [0, 0, 0], [1, 0, 0, 0], [0, 0, 0], [1, 0, 0, 0] ] ];
```

#### Estructura

```
< 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 >
```

#### Información relacionada

| Para obtener más información sobre  | Consulte   |
|-------------------------------------|--|
| Instrucciones de posicionamiento    | <i>Manual de referencia técnica - RAPID Overview, sección Resumen sobre RAPID - Movimiento</i>                         |
| Sistemas de coordenadas             | <i>Manual de referencia técnica - RAPID Overview, sección Principios de movimiento y E/S - Sistemas de coordenadas</i> |
| Ejes externos coordinados           | <i>Manual de referencia técnica - RAPID Overview, sección Principios de movimiento y E/S - Sistemas de coordenadas</i> |
| Calibración de los ejes coordinados | <i>Application manual - Additional axes and stand alone controller</i><br><i>Manual de aplicaciones - MultiMove</i>    |

### 3.100 wzstationary - Datos de zona mundo estacionaria

#### Utilización

`wzstationary` (*world zone stationary*) se usa para identificar una zona mundo estacionaria y sólo puede usarse en una rutina de evento conectada al evento POWER ON.

Las zonas mundo se supervisan durante los movimientos del robot, tanto durante la ejecución del programa como durante los movimientos. Si el TCP del robot alcanza la zona mundo o si los ejes del robot o los ejes externos alcanzan la zona mundo, el movimiento se detiene o se activa o desactiva una señal digital de salida.

#### Descripción

Las zonas mundo `wzstationary` se definen y activan con una instrucción `WZLimSup` o una instrucción `WZDOSet`.

`WZLimSup` o `WZDOSet` asignan un valor numérico a variables o variables persistentes con el tipo de dato `wzstationary`. El valor identifica una zona mundo.

Las zonas mundo estacionarias siempre están activas en el estado Motors ON y sólo se eliminan con un Reinicio. No es posible desactivar, activar ni eliminar una zona mundo estacionaria a través de instrucciones de RAPID.

Las zonas mundo estacionarias deben estar activadas desde la puesta en marcha y deben estar definidas en la rutina de evento POWER ON o en una tarea semiestática.

#### Ejemplos básicos

El siguiente ejemplo ilustra el tipo de dato `wzstationary`:

##### Ejemplo 1

```
VAR wzstationary conveyor;
...
PROC ...
  VAR shapedata volume;
  ...
  WZBoxDef \Inside, volume, p_corner1, p_corner2;
  WZLimSup \Stat, conveyor, volume;
ENDPROC
```

Un `conveyor` se define como un prisma de ángulos rectos (el volumen que queda debajo de la cinta). Si el robot alcanza este volumen, se detiene el movimiento.

#### Limitaciones

Los datos `wzstationary` deben definirse como variables (VAR) o variables persistentes (PERS). Pueden ser globales en la tarea o locales dentro del módulo, pero no locales dentro de una rutina.

Los argumentos de tipo `wzstationary` deben ser sólo datos completos (ni elementos de matriz ni componentes de registro).

Los valores de inicialización de los datos del tipo `wzstationary` no se utilizan en el sistema de control. Cuando sea necesario utilizar una variable persistente

*Continúa en la página siguiente*

### 3 Tipos de datos

---

#### 3.100 wzstationary - Datos de zona mundo estacionaria

##### World Zones

##### Continuación

en un sistema multitarea, cambie el valor inicial a 0 en las dos tareas, por ejemplo  
PERS wzstationary share\_workarea := [0];

---

#### Más ejemplos

Para ver un ejemplo completo, consulte la instrucción `WZLimSup`.

---

#### Características

`wzstationary` es un tipo de dato de alias de `wztemporary` y hereda sus características.

---

#### Información relacionada

| Para obtener más información sobre                         | Consulte   |
|--|--|
| World Zones  | <i>Manual de referencia técnica - RAPID Overview</i> , sección <i>Principios de movimiento y E/S - Zonas mundo</i> |
| Forma de las zonas mundo                                   | <a href="#">shapedata - Datos de forma de zonas mundo en la página 1759</a>  |
| Zona mundo temporal  | <a href="#">wztemporary - Datos de zona mundo temporal en la página 1829</a>                                       |
| Activación de la supervisión de límites de las zonas mundo | <a href="#">WZLimSup - Activa la supervisión de límites de las zonas mundo en la página 1145</a>                   |
| Activación de salidas digitales basadas en zonas mundo     | <a href="#">WZDOSet - Activación de salidas digitales basadas en zonas mundo en la página 1128</a>                 |



### 3.101 wztemporary - Datos de zona mundo temporal

#### Utilización

`wztemporary` (*world zone temporary*) se utiliza para identificar una zona mundo temporal y puede usarse en cualquier parte del programa RAPID para cualquier tarea de movimiento.

Las zonas mundo se supervisan durante los movimientos del robot, tanto durante la ejecución del programa como durante los movimientos. Si el TCP del robot alcanza la zona mundo o si los ejes del robot o los ejes externos alcanzan la zona mundo, el movimiento se detiene o se activa o desactiva una señal digital de salida.

#### Descripción

Las zonas mundo `wztemporary` se definen y activan con una instrucción `WZLimSup` o una instrucción `WZDOSet`.

`WZLimSup` o `WZDOSet` asignan un valor numérico a variables o variables persistentes con el tipo de dato `wztemporary`. El valor identifica una zona mundo. Una vez definidas y activadas, las zonas mundo temporales pueden desactivarse con `WZDisable`, activarse de nuevo con `WZEnable` y eliminarse con `WZFree`.

Todas las zonas mundo temporales de la tarea de movimiento se eliminan automáticamente y todos los objetos de datos del tipo `wztemporary` de la tarea de movimiento cambian a 0:

- Cuando se carga un nuevo programa en la tarea motion
- Cuando se inicia la ejecución del programa desde el principio en la tarea motion

#### Ejemplos básicos

El siguiente ejemplo ilustra el tipo de dato `wztemporary`:

##### Ejemplo 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
```

Se define una variable `wztemporary`, `roll`, con forma de cilindro. Si el robot alcanza este volumen, se detiene el movimiento.

#### Limitaciones

Los datos `wztemporary` deben definirse como variables (VAR) o variables persistentes (PERS). Pueden ser globales en la tarea o locales dentro del módulo, pero no locales dentro de una rutina.

*Continúa en la página siguiente*

### 3 Tipos de datos

#### 3.101 wztemporary - Datos de zona mundo temporal

RobotWare - OS

Continuación

Los argumentos de tipo `wztemporary` deben ser sólo datos completos (ni elementos de matriz ni componentes de registro).

Las zonas mundo temporales sólo deben definirse (`WZLimSup` o `WZDOSet`) y liberarse (`WZFree`) desde la tarea MAIN. Las definiciones de zonas mundo temporales en cualquier segundo plano no se permiten porque afectarían a la ejecución del programa en la tarea de movimiento relacionada. Las instrucciones `WZDisable` y `WZEnable` pueden usarse en la tarea en segundo plano. Cuando sea necesario utilizar una variable persistente en un sistema multitarea, cambie el valor inicial a 0 en las dos tareas, por ejemplo `PERS wztemporary share_workarea := [0];`

#### Más ejemplos

Para ver un ejemplo completo, consulte la instrucción `WZDOSet`.

#### Estructura

```
< dataobject of wztemporary >  
< wz of num >
```

#### Información relacionada

| Para obtener más información sobre                         | Consulte   |
|--|--|
| World Zones  | <i>Manual de referencia técnica - RAPID Overview, sección Principios de movimiento y E/S - Zonas mundo</i> |
| Forma de las zonas mundo                                   | <a href="#">shapedata - Datos de forma de zonas mundo en la página 1759</a>                                |
| Zona mundo estacionaria                                    | <a href="#">wzstationary - Datos de zona mundo estacionaria en la página 1827</a>                          |
| Activación de la supervisión de límites de las zonas mundo | <a href="#">WZLimSup - Activa la supervisión de límites de las zonas mundo en la página 1145</a>           |
| Activación de salidas digitales basadas en zonas mundo     | <a href="#">WZDOSet - Activación de salidas digitales basadas en zonas mundo en la página 1128</a>         |
| Desactivación de zonas mundo                               | <a href="#">WZDisable - Desactiva la supervisión de las zonas mundo temporales en la página 1126</a>       |
| Activación de zonas mundo                                  | <a href="#">WZEnable - Activa la supervisión de las zonas mundo temporales en la página 1133</a>           |
| Eliminación de zonas mundo                                 | <a href="#">WZFree - Elimina la supervisión de las zonas mundo temporales en la página 1135</a>            |

## 3.102 zonedata - Datos de zonas

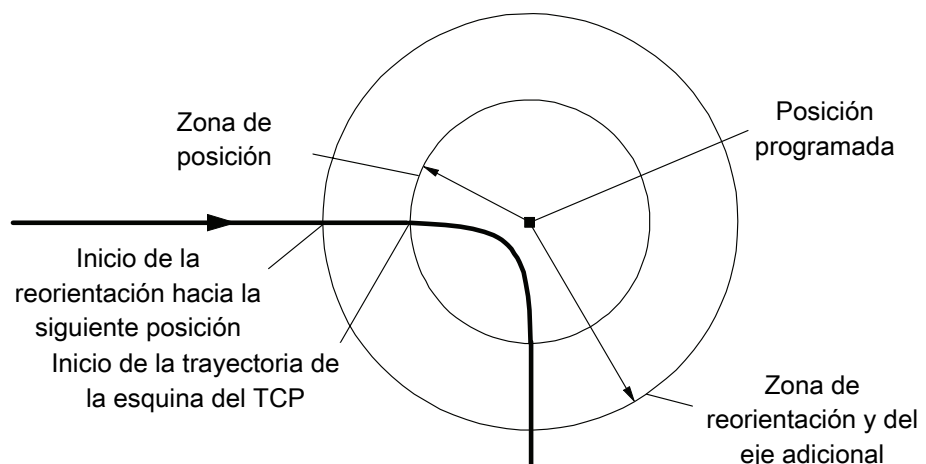
## Utilización

`zonedata` se utiliza para especificar cómo debe terminar una posición, es decir, a qué distancia de la posición programada deben encontrarse los ejes antes de iniciar un movimiento hasta la posición siguiente.

## Descripción

Una posición puede terminar en forma de un punto de paso o un punto de paro. Un punto de paro significa que los ejes del robot y los ejes externos deben alcanzar la posición especificada (deteniéndose) antes de que la ejecución del programa continúe en la instrucción siguiente. También es posible definir puntos de paro distintos del tipo predefinido `fine`. Los criterios de paro que indican si se considera que el robot ha alcanzado el punto, pueden manipularse mediante `stoppointdata`. El punto de paso significa que la posición programada no llega a alcanzarse nunca. En su lugar, la dirección del movimiento cambia antes de que se alcance la posición. Es posible definir dos zonas (rangos) diferentes para cada posición:

- La zona de posición para la trayectoria del TCP
- La zona de reorientación y del eje adicional.



xx180000945

## La zona para la trayectoria del TCP

Se genera una trayectoria de esquina tan pronto como se alcanza el borde de la zona de esquina (consulte la figura anterior).

*Continúa en la página siguiente*

### 3 Tipos de datos

#### 3.102 zonedata - Datos de zonas

RobotWare - OS

Continuación

#### Cálculo de la zona de reorientación y del eje adicional

El tipo de datos `zonedata` contiene un componente que determina la zona de posición, `pzone_tcp`. No obstante, la zona de reorientación y del eje adicional puede verse afectada por todos los siguientes componentes `zonedata`.

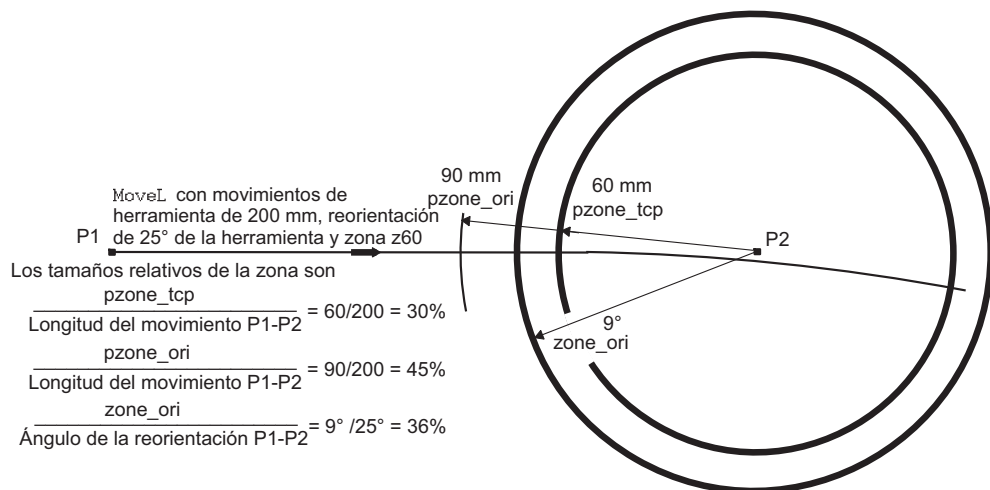
- `pzone_ori` - radio de zona, en movimiento TCP en mm para reorientar la herramienta
- `pzone_eax` - radio de zona, en movimiento TCP en mm para mover el eje adicional
- `zone_ori` - zona de ángulo, en grados de la reorientación de herramienta
- `zone_leax` - tamaño de zona, en mm de movimiento del eje lineal adicional
- `zone_reax` - zona de ángulo, en grados de la reorientación del eje giratorio adicional

El tamaño de la zona de reorientación y del eje adicional suele estar limitado por la zona más pequeña generada desde los componentes aplicables anteriores. La zona se define como el tamaño relativo más pequeño de la zona, basado en los componentes de la misma y en el movimiento programado.

Si los cálculos ofrecen una zona de reorientación y del eje adicional más pequeña que la zona de posición, la zona de reorientación y del eje adicional se establecerá con el mismo tamaño que la zona de posición. La excepción es si no hay (o casi no hay) movimiento de de posición TCP. Si la reorientación es grande y el movimiento de posición es pequeño, la zona de posición puede reducirse al tamaño de la zona de reorientación y del eje adicional.

#### Zona de reorientación y del eje adicional limitada por `zone_ori`

En la figura siguiente se muestra un ejemplo de la zona de reorientación y del eje adicional reducida al 36% del movimiento previsto para `zone_ori`.



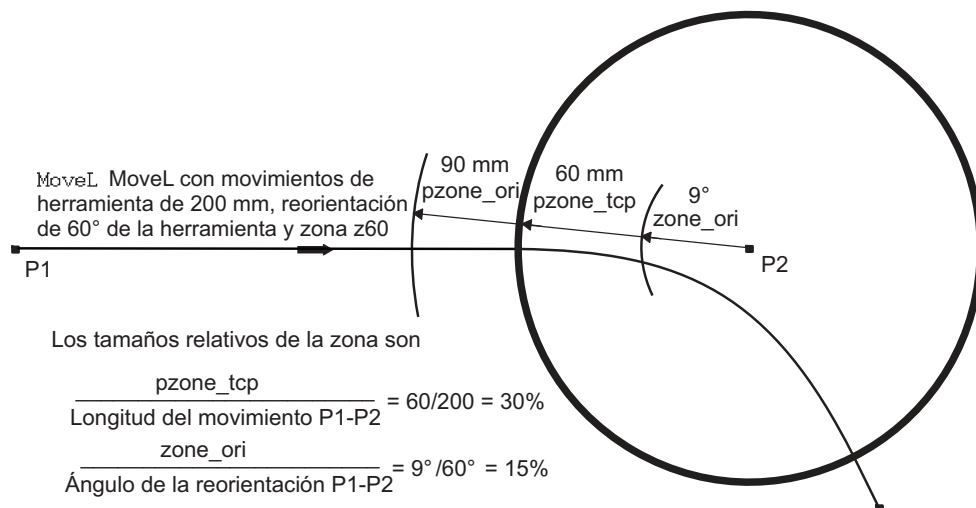
xx0500002362

#### Aumenta la zona de reorientación y del eje adicional a la zona de posición.

El tamaño de la zona de reorientación y del eje adicional nunca debe ser más pequeño que el de la zona de posición. Por lo tanto, si por ejemplo, `zone_ori` tiene un tamaño más pequeño que `pzone_tcp`, el tamaño de la zona de reorientación y del eje adicional aumenta al de la zona de posición.

Continúa en la página siguiente

En la figura siguiente se muestra un ejemplo en el que `zone_ori` indica una zona de reorientación y del eje adicional del 15% del movimiento, pero se aumenta al 30% del movimiento para coincidir con la zona de posición.



xx0500002363

#### Fórmulas de cálculo de la zona de reorientación y del eje adicional

Por lo general, no se aplican todos los componentes `zonedata`. Por ejemplo, para una reorientación del eje giratorio adicional sin movimiento del robot, solamente se aplica `zone_reax`.

Para todos los componentes `zonedata` aplicables, las siguientes relaciones más pequeñas son las que determinan el tamaño de la zona de reorientación y del eje adicional (siempre que sea superior o equivalente a la zona de posición).

$$\frac{pzone\_ori}{\text{longitud del movimiento P1 - P2}}$$

$$\frac{zone\_ori}{\text{ángulo de reorientación de la herramienta P1 - P2}}$$

$$\frac{pzone\_eax}{\text{longitud del movimiento P1 - P2}}$$

$$\frac{pzone\_leax}{\text{longitud del movimiento lineal máx. del eje adic. P1 - P2}}$$

$$\frac{pzone\_reax}{\text{ángulo de reorientación máx. del eje adic. giratorio P1 - P2}}$$

xx1800000781

#### Zonas de esquina reducidas

Si las posiciones programadas están cerca unas de otras y las zonas de esquina son grandes, se pueden reducir las zonas de esquina desde el tamaño programado. Consulte la sección *Interpolación de trayectorias de esquina* in *Manual de referencia técnica - RAPID Overview*.

Continúa en la página siguiente

### 3 Tipos de datos

---

#### 3.102 zonedata - Datos de zonas

RobotWare - OS

Continuación

---

#### Componentes

finep

##### *fine point*

Tipo de dato: bool

Define si el movimiento debe terminar como un punto de paro (punto fino *fine*) o como punto de paso.

- **TRUE**: El movimiento termina como un punto de paro y la ejecución del programa no continúa hasta que el robot llega hasta el punto de paro. Los demás componentes de los datos de la zona no se utilizan.
- **FALSE**: El movimiento determina un punto de paso y la ejecución del programa continúa en marcha cuando se han cumplido las condiciones de precaptura (consulte el parámetro del sistema *Prefetch Time*).

pzone\_tcp

##### *path zone TCP*

Tipo de dato: num

El tamaño (el radio) de la zona del TCP en mm.

pzone\_ori

##### *path zone orientation*

Tipo de dato: num

El tamaño de la zona (el radio) para la reorientación de la herramienta. El tamaño se define como la distancia en mm del TCP respecto del punto programado.

El tamaño debe ser mayor que el valor correspondiente de *pzone\_tcp*. Si se especifica un valor menor, el tamaño aumenta automáticamente para que sea igual que el *pzone\_tcp*.

pzone\_eax

##### *path zone external axes*

Tipo de dato: num

El tamaño de la zona (el radio) para los ejes externos. El tamaño se define como la distancia en mm del TCP respecto del punto programado.

El tamaño debe ser mayor que el valor correspondiente de *pzone\_tcp*. Si se especifica un valor menor, el tamaño aumenta automáticamente para que sea igual que el *pzone\_tcp*.

zone\_ori

##### *zone orientation*

Tipo de dato: num

El tamaño de la zona para la reorientación de la herramienta, en grados. Si el robot está sosteniendo el objeto de trabajo, esto implica un ángulo de rotación para el objeto de trabajo.

zone\_leax

##### *zone linear external axes*

Continúa en la página siguiente

Tipo de dato: num

El tamaño de zona para los ejes externos lineales, en mm.

zone\_reax

*zone rotational external axes*

Tipo de dato: num

El tamaño de zona para los ejes externos de rotación, en grados.

#### Ejemplos básicos

El siguiente ejemplo ilustra el tipo de dato zonedata:

##### Ejemplo 1

```
VAR zonedata path := [ FALSE, 25, 40, 40, 10, 35, 5 ];
```

Se definen los datos de zona path con las características siguientes:

- El tamaño de la zona para la trayectoria del TCP es de 25 mm.
- El tamaño de la zona para la reorientación de la herramienta es de 40 mm (movimiento del TCP).
- El tamaño de la zona para los ejes externos es de 40 mm (movimiento del TCP).

Si el TCP está detenido, se produce una gran reorientación o existe un gran movimiento de los ejes externos respecto de la zona, se aplica lo siguiente:

- El tamaño de la zona para la reorientación de la herramienta es de 10 grados.
- El tamaño de la zona para los ejes externos lineales es de 35 mm.
- El tamaño de la zona para los ejes externos de rotación es de 5 grados.

#### Datos predefinidos

Existen varios datos de zona ya definidos en el sistema.

##### Puntos de paro

Use zonedata con nombre fine.

##### Puntos de paso

| Zona de trayectoria |             |             |             | Zona        |            |                 |
|---------------------|-------------|-------------|-------------|-------------|------------|-----------------|
| Nombre              | Trayec. TCP | Orientación | Eje externo | Orientación | Eje lineal | Eje de rotación |
| 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°            |

*Continúa en la página siguiente*

### 3 Tipos de datos

#### 3.102 zonedata - Datos de zonas

RobotWare - OS

Continuación

| Zona de trayectoria |             |             |             | Zona        |            |                 |
|---------------------|-------------|-------------|-------------|-------------|------------|-----------------|
| Nombre              | Trayec. TCP | Orientación | Eje externo | Orientación | Eje lineal | Eje de rotación |
| 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°             |

#### Estructura

```
<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>
```

#### Información relacionada

| Para obtener más información sobre  | Consulte   |
|-------------------------------------|--|
| Instrucciones de posicionamiento    | <i>Manual de referencia técnica - RAPID Overview</i>                     |
| Movimientos/trayectorias en general | <i>Manual de referencia técnica - RAPID Overview</i>                     |
| Configuración de ejes externos      | <i>Application manual - Additional axes and stand alone controller</i>   |
| Otros puntos de paro                | <a href="#">stoppointdata - Datos de punto de paro en la página 1771</a> |



## 4 Ejemplos de tipos de programación

### 4.1 Gestor de ERROR con movimientos

#### Utilización

Estos ejemplos de tipos describen cómo usar instrucciones de movimiento en un gestor de ERROR después de producirse un proceso elevado asincrónicamente o un error de movimiento.

Esta función sólo puede usarse en la tarea main T\_ROB1 o, si se cuenta con un sistema MultiMove, en las tareas de movimiento.

#### Descripción

El gestor de ERROR puede iniciar un nuevo movimiento temporal y por último reanudar el movimiento original interrumpido y detenido. Por ejemplo, puede usarse para ir a una posición de servicio o para limpiar la pistola tras producirse un proceso elevado asíncronamente o un error de movimiento.

Para alcanzar esta funcionalidad, deben usarse las instrucciones StorePath - RestoPath en el gestor de ERROR. Para reanudar el movimiento y continuar con la ejecución del programa existen varias instrucciones RAPID.

#### Ejemplos de tipo

A continuación aparecen algunos ejemplos de tipo de esta funcionalidad.

#### Principio

```
...
ERROR
  IF ERRNO = ERR_PATH_STOP THEN
    StorePath;
    ! Move away and back to the interrupted position
    ...
    RestoPath;
    StartMoveRetry;
  ENDIF
ENDPROC
```

En la ejecución de StartMoveRetry, el robot reanuda su movimiento, se reanuda cualquier proceso activo y el programa reintenta su ejecución. StartMoveRetry hace lo mismo que StartMove más RETRY en una operación indivisible.

#### Reinicio automático de la ejecución

```
CONST robtargt service_pos := [...];
VAR robtargt 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;
```

*Continúa en la página siguiente*

## 4 Ejemplos de tipos de programación

---

### 4.1 Gestor de ERROR con movimientos

#### *Path Recovery*

#### *Continuación*

```
! 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;
! 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
```

Éste es un ejemplo de tipo de cómo usar la recuperación asíncrona de errores automática tras algún tipo de proceso durante los movimientos del robot.

#### Reinicio manual de la ejecución

```
...
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
```

Éste es un ejemplo de tipo de cómo usar el manejo manual de la recuperación asíncrona de errores tras algún tipo de proceso durante los movimientos del robot.

Una vez que el gestor de `ERROR` anterior se ha ejecutado hasta el final, la ejecución del programa se detiene y el puntero de programa del principio de la instrucción con el error de proceso (también al principio de cualquier rutina `NOSTEPIN` utilizada). El siguiente inicio de programa reanuda el programa y el movimiento desde la posición en la que se produjo el error de proceso original.

---

### Ejecución de programas

#### Comportamiento de ejecución:

- Al inicio de la ejecución del gestor de `ERROR`, el programa abandona su nivel de ejecución base.
- En la ejecución de `StorePath`, el sistema de movimiento abandona su nivel de ejecución base.
- En la ejecución de `RestoPath`, el sistema de movimiento retorna a su nivel de ejecución base.
- En la ejecución de `StartMoveRetry`, el programa regresa a su nivel de ejecución base.

*Continúa en la página siguiente*

#### Limitaciones

Las siguientes instrucciones de RAPID deben usarse en el gestor de ERROR con instrucciones de movimiento para ponerlo a funcionar con la recuperación automática de errores tras un error de proceso generado asincrónicamente o un error de trayectoria:

| Instrucción    | Descripción  |
|----------------|--|
| StorePath      | Entrar en el nuevo nivel de trayectoria de movimiento  |
| RestoPath      | Retornar al nivel base de trayectoria de movimientos   |
| StartMoveRetry | Reanudar los movimientos interrumpidos en el nivel base de trayectoria de movimientos. También se reanuda el proceso y se reintenta la ejecución de programas.<br>Misma funcionalidad que StartMove + RETRY. |

La siguiente instrucción RAPID debe usarse en el gestor de ERROR para ponerlo a funcionar con la recuperación manual de errores tras un error de proceso generado asincrónicamente o un error de trayectoria:

| Instrucción   | Descripción   |
|---------------|---|
| StopMoveReset | Entrar en el nuevo nivel de trayectoria de movimiento |

#### Información relacionada

| Para obtener más información sobre  | Consulte  |
|---|---|
| Para entrar en el nuevo nivel de trayectoria de movimiento                                      | <a href="#">StorePath - Almacena la trayectoria cuando se produce una interrupción en la página 835</a> |
| Para retornar al nivel base de trayectoria de movimientos                                       | <a href="#">RestoPath - Restablece la trayectoria después de una interrupción en la página 615</a>      |
| Para reanudar la ejecución del movimiento interrumpido, el proceso y el reintento del programa. | <a href="#">StartMoveRetry - Reanuda el movimiento y la ejecución del robot en la página 799</a>        |

## 4 Ejemplos de tipos de programación

---

### 4.2 Rutinas de servicio con o sin movimientos

#### *Path recovery*

### 4.2 Rutinas de servicio con o sin movimientos

---

#### Utilización

Estos ejemplos de tipo describen cómo usar las instrucciones de movimiento en una rutina de servicio. El mismo principio aplicable a `StopMove`, `StartMove` y `StopMoveReset` es válido también para las rutinas de servicio válidas sin movimientos (sólo instrucciones lógicas).

Ambas rutinas de servicio u otras rutinas (procedimientos) sin parámetros pueden iniciarse manualmente y realizar movimientos de acuerdo con estos ejemplos de tipo.

Esta funcionalidad sólo puede usarse en la tarea principal `T_ROB1` o, si se cuenta con un sistema `MultiMove`, en las tareas de movimiento en el modo independiente o semicoordinado.

#### Descripción

La rutina de servicio puede iniciar un nuevo movimiento temporal y, en un inicio posterior del programa, reanudar el movimiento original. Por ejemplo, puede usarse para ir a una posición de servicio o iniciar manualmente la limpieza de la pistola. Para alcanzar esta funcionalidad, deben usarse las instrucciones `StorePath` - `RestoPath` y `StopMoveReset` en la rutina de servicio.

#### Ejemplos de tipo

A continuación aparecen algunos ejemplos de tipo de esta funcionalidad.

#### Principio

```
PROC xxxx()  
  StopMove;  
  StorePath;  
  ! Move away and back to the interrupted position  
  ...  
  RestoPath;  
  StopMoveReset;  
ENDPROC
```

`StopMove` es necesario con el fin de asegurarse de que el movimiento detenido originalmente no se reanude en caso de una secuencia manual "parar programa-reiniciar programa" durante la ejecución de la rutina de servicio.

#### Paro dentro de la trayectoria

```
VAR robtargt service_pos := [...];  
...  
PROC proc_stop_on_path()  
  VAR robtargt 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
```

*Continúa en la página siguiente*

## 4 Ejemplos de tipos de programación

### 4.2 Rutinas de servicio con o sin movimientos

*Path recovery*

*Continuación*

```
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;
! Reset the stop move state for the interrupted movement
! on motion base path level
StopMoveReset;
ENDPROC
```

**En este ejemplo de tipo, los movimientos de la rutina de servicio comienzan y terminan en la posición de la trayectoria en la que se detuvo el programa.**

**Recuerde también que la herramienta y el objeto de trabajo utilizados se conocen en el momento de la programación.**

**Pero en el siguiente punto de paro**

```
TASK PERS tooldata used_tool := [...];
TASK PERS wobjdata used_wobj := [...];
...
PROC proc_stop_in_stop_point()
VAR robtarg 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
```

**En este ejemplo de tipo, los movimientos de la rutina de servicio continúan y terminan en el punto ToPoint de las instrucciones de movimiento interrumpidas, antes de que se complete la instrucción StorePath.**

**Recuerde también que la herramienta y el objeto de trabajo utilizados no se conocen en el momento de la programación.**

*Continúa en la página siguiente*

## 4 Ejemplos de tipos de programación

### 4.2 Rutinas de servicio con o sin movimientos

#### Path recovery

#### Continuación

#### Ejecución de programas

##### Comportamiento de ejecución:

- Al inicio de la ejecución de la rutina de servicio, el programa abandona su nivel de ejecución base.
- En la ejecución de `StorePath`, el sistema de movimiento abandona su nivel de ejecución base.
- En la ejecución de `RestoPath`, el sistema de movimiento retorna a su nivel de ejecución base.
- En la ejecución de `ENDPROC`, el programa regresa a su nivel de ejecución base.

#### Limitaciones

Las siguientes instrucciones de `RAPID` deben usarse en la rutina de servicio junto con las instrucciones de movimiento para que funcione:

| Instrucción                | Descripción  |
|----------------------------|--|
| <code>StorePath</code>     | Entrar en el nuevo nivel de trayectoria de movimiento  |
| <code>RestoPath</code>     | Retornar al nivel base de trayectoria de movimientos   |
| <code>StopMoveReset</code> | Restablecer el estado de paro de movimiento del movimiento interrumpido en el nivel base de la trayectoria |

#### Información relacionada

| Para obtener más información sobre   | Consulte  |
|--|---|
| Ninguna reanudación del movimiento ya detenido en el nivel base de trayectoria                             | <a href="#">StopMove - Detiene el movimiento del robot en la página 828</a>                             |
| Reanudación del movimiento ya detenido en el nivel base de trayectoria                                     | <a href="#">StopMove - Detiene el movimiento del robot en la página 828</a>                             |
| Para entrar en el nuevo nivel de trayectoria de movimiento   | <a href="#">StorePath - Almacena la trayectoria cuando se produce una interrupción en la página 835</a> |
| Para retornar al nivel base de trayectoria   | <a href="#">RestoPath - Restablece la trayectoria después de una interrupción en la página 615</a>      |
| Restablecer el estado de paro de movimiento del movimiento interrumpido en el nivel base de la trayectoria | <a href="#">StopMoveReset - Restablece el estado de movimiento de paro de sistema en la página 832</a>  |

#### 4.3 Interrupciones de E/S del sistema con o sin movimiento

##### Utilización

Estos ejemplos de tipo describen cómo usar las instrucciones de movimiento en una rutina de interrupción de E/S de sistema. El mismo principio aplicable a `StopMove`, `StartMove` y `StopMoveReset` es válido también para las interrupciones de E/S del sistema sin movimientos (sólo instrucciones lógicas).

Esta funcionalidad sólo puede usarse en la tarea principal `T_ROB1` o, si se cuenta con un sistema `MultiMove`, en las tareas de movimiento en el modo independiente o semicoordinado.

##### Descripción

La interrupción de E/S del sistema puede iniciar un nuevo movimiento temporal y, en un inicio posterior del programa, reanudar el movimiento original. Por ejemplo, puede usarse para ir a una posición de servicio o limpiar una pistola cuando se produce una interrupción.

Para alcanzar esta funcionalidad, deben usarse las instrucciones `StorePath` - `RestoPath` y `StopMoveReset` en la rutina de interrupción de E/S del sistema.

##### Ejemplos de tipo

A continuación aparecen algunos ejemplos de tipo de esta funcionalidad.

##### Principio

```
PROC xxxx()  
  StopMove;  
  StorePath;  
  ! Move away and back to the interrupted position  
  ...  
  RestoPath;  
  StopMoveReset;  
ENDPROC
```

`StopMove` es necesario para poder garantizar que el movimiento detenido originalmente no se reinicie al principio de la rutina de interrupción de E/S.

Si no se usa `StopMove` con `StartMove`, el movimiento de la rutina de interrupción de E/S continuará inmediatamente y finalizará en `ToPoint` en la instrucción de movimiento interrumpida.

##### Paro dentro de la trayectoria

```
VAR robtargt service_pos := [...];  
...  
PROC proc_stop_on_path()  
  VAR robtargt stop_pos;  
  ! Current stopped movements on motion base path level is not  
    restarted in the system I/O routine.  
  StopMove;  
  ! New motion path level for new movements in the system I/O  
    routine.  
  StorePath;
```

*Continúa en la página siguiente*

## 4 Ejemplos de tipos de programación

---

### 4.3 Interrupciones de E/S del sistema con o sin movimiento

#### *Path recovery*

#### *Continuación*

```
! 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;
! Reset the stop move state for the interrupted movement on motion
  base path level
StopMoveReset;
ENDPROC
```

En este ejemplo de tipo, los movimientos interrumpidos se detienen inmediatamente y se reinician al iniciarse el programa una vez finalizada la rutina de interrupción de E/S del sistema.

Recuerde también que la herramienta y el objeto de trabajo utilizados se conocen en el momento de la programación.

#### **Pero en el siguiente punto de paro**

```
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
```

En este ejemplo de tipo, los movimientos de la rutina de E/S del sistema continúan inmediatamente y terminan en el punto `ToPoint` de las instrucciones de movimiento interrumpidas.

*Continúa en la página siguiente*



## 4 Ejemplos de tipos de programación

### 4.3 Interrupciones de E/S del sistema con o sin movimiento

*Path recovery*

*Continuación*

Recuerde también que la herramienta y el objeto de trabajo utilizados no se conocen en el momento de la programación.

#### Ejecución de programas

Comportamiento de ejecución:

- Al inicio de la ejecución de la rutina de E/S del sistema, el programa abandona su nivel de ejecución base.
- En la ejecución de `StorePath`, el sistema de movimiento abandona su nivel de ejecución base.
- En la ejecución de `RestoPath`, el sistema de movimiento retorna a su nivel de ejecución base.
- En la ejecución de `ENDPROC`, el programa regresa a su nivel de ejecución base.

#### Limitaciones

Las siguientes instrucciones de `RAPID` deben utilizarse en la rutina de E/S del sistema junto con las instrucciones de movimiento para que funcione:

| Instrucción                | Descripción  |
|----------------------------|--|
| <code>StorePath</code>     | Entrar en el nuevo nivel de trayectoria de movimiento  |
| <code>RestoPath</code>     | Retornar al nivel base de trayectoria de movimientos   |
| <code>StopMoveReset</code> | Restablecer el estado de paro de movimiento del movimiento interrumpido en el nivel base de la trayectoria |

#### Información relacionada

| Para obtener más información sobre   | Consulte  |
|--|---|
| Ninguna reanudación del movimiento ya detenido en el nivel base de trayectoria                             | <a href="#">StopMove - Detiene el movimiento del robot en la página 828</a>                             |
| Reanudación del movimiento ya detenido en el nivel base de trayectoria                                     | <a href="#">StartMove - Reanuda el movimiento del robot en la página 796</a>                            |
| Para entrar en el nuevo nivel de trayectoria de movimiento   | <a href="#">StorePath - Almacena la trayectoria cuando se produce una interrupción en la página 835</a> |
| Para retornar al nivel base de trayectoria   | <a href="#">RestoPath - Restablece la trayectoria después de una interrupción en la página 615</a>      |
| Restablecer el estado de paro de movimiento del movimiento interrumpido en el nivel base de la trayectoria | <a href="#">StopMoveReset - Restablece el estado de movimiento de paro de sistema en la página 832</a>  |

## 4 Ejemplos de tipos de programación

---

### 4.4 Rutinas TRAP con movimientos

#### *Path Recovery*

### 4.4 Rutinas TRAP con movimientos

---

#### Utilización

Estos ejemplos de tipo describen cómo usar las instrucciones de movimiento en una rutina TRAP tras producirse una interrupción.

Esta funcionalidad sólo puede usarse en la tarea principal T\_ROB1 o, si se cuenta con un sistema MultiMove, en las tareas de movimiento.

#### Descripción

La rutina TRAP puede iniciar un nuevo movimiento temporal y por último reanudar el movimiento original. Por ejemplo, puede usarse para ir a una posición de servicio o limpiar una pistola cuando se produce una interrupción.

Para alcanzar esta funcionalidad, deben usarse las instrucciones StorePath - RestoPath y StartMove en la rutina TRAP.

#### Ejemplos de tipo

A continuación aparecen algunos ejemplos de tipo de esta funcionalidad.

#### Principio

```
TRAP xxxx
  StopMove;
  StorePath;
  ! Move away and back to the interrupted position
  ...
  RestoPath;
  StartMove;
ENDTRAP
```

Si se utiliza StopMove, el movimiento se detiene inmediatamente dentro de la trayectoria en curso. De lo contrario, el movimiento continúa hacia el punto ToPoint de la instrucción de movimiento actual.

#### Pero en el siguiente punto de paro

```
VAR robtargt service_pos := [...];
...
TRAP trap_in_stop_point
  VAR robtargt 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;
```

*Continúa en la página siguiente*

```
! Restart the interrupted movements on motion base path level
StartMove;
ENDTRAP
```

En este ejemplo de tipo, los movimientos de la rutina TRAP se inician y terminan en el punto `ToPoint` de las instrucciones de movimiento interrumpidas. Recuerde también que la herramienta y el objeto de trabajo se conocen en el momento de la programación.

#### Paro inmediato dentro de la trayectoria

```
TASK PERS tooldata used_tool := [...];
TASK PERS wobjdata used_wobj := [...];
...
TRAP trap_stop_at_once
  VAR robtarg 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 interrupted movements on motion base path level
  StartMove;
ENDTRAP
```

En este ejemplo de tipo, los movimientos de la rutina TRAP comienzan y terminan en la posición de la trayectoria en la que se detuvo la instrucción de movimiento interrumpida. Recuerde también que la herramienta y el objeto de trabajo utilizados no se conocen en el momento de la programación.

---

### Ejecución de programas

#### Comportamiento de ejecución:

- Al inicio de la ejecución de la rutina `TRAP`, el programa abandona su nivel de ejecución base.
- En la ejecución de `StorePath`, el sistema de movimiento abandona su nivel de ejecución base.
- En la ejecución de `RestoPath`, el sistema de movimiento retorna a su nivel de ejecución base.
- En la ejecución de `ENDTRAP`, el programa regresa a su nivel de ejecución base.

*Continúa en la página siguiente*

## 4 Ejemplos de tipos de programación

---

### 4.4 Rutinas TRAP con movimientos

#### *Path Recovery*

#### *Continuación*

---

#### Limitaciones

Las siguientes instrucciones de `RAPID` deben usarse en la rutina `TRAP` junto con las instrucciones de movimiento para que funcione:

| Instrucción            | Descripción   |
|------------------------|---|
| <code>StorePath</code> | Entrar en el nuevo nivel de trayectoria de movimiento                                 |
| <code>RestoPath</code> | Retornar al nivel base de trayectoria de movimientos                                  |
| <code>StartMove</code> | Reanudar los movimientos interrumpidos en el nivel base de trayectoria de movimientos |

---

#### Información relacionada

| Para obtener más información sobre                         | Consulte   |
|--|--|
| Para detener inmediatamente el movimiento actual           | <a href="#"><i>StopMove - Detiene el movimiento del robot en la página 828</i></a>                             |
| Para entrar en el nuevo nivel de trayectoria de movimiento | <a href="#"><i>StorePath - Almacena la trayectoria cuando se produce una interrupción en la página 835</i></a> |
| Para retornar al nivel base de trayectoria                 | <a href="#"><i>RestoPath - Restablece la trayectoria después de una interrupción en la página 615</i></a>      |
| Para reanudar el movimiento interrumpido                   | <a href="#"><i>StartMove - Reanuda el movimiento del robot en la página 796</i></a>                            |

# Índice

## A

Abs, 1151  
 AbsDnum, 1153  
 AccSet, 25  
 ACos, 1155  
 ACosDnum, 1156  
 ActEventBuffer, 28  
 ActUnit, 30  
 Add, 32  
 AInput, 1157  
 aiotrigger, 1607  
 ALIAS, 1609  
 AliasCamera, 34  
 AliasIO, 37  
 AliasIOReset, 40  
 AND, 1159  
 AOutput, 1161  
 ArgName, 1163  
 ASin, 1167  
 ASinDnum, 1168  
 Assignment  
   =, 42  
 ATan, 1169  
 ATan2, 1171  
 ATan2Dnum, 1172  
 ATanDnum, 1170

## B

BitAnd, 1173  
 BitAndDnum, 1175  
 BitCheck, 1177  
 BitCheckDnum, 1179  
 BitClear, 44  
 BitLSh, 1181  
 BitLShDnum, 1183  
 BitNeg, 1186  
 BitNegDnum, 1188  
 BitOr, 1190  
 BitOrDnum, 1192  
 BitRSh, 1194  
 BitRShDnum, 1196  
 BitSet, 47  
 BitXOr, 1198  
 BitXOrDnum, 1200  
 BookErrNo, 50  
 bool, 1610  
 Break, 52  
 btnres, 1611  
 busstate, 1613  
 buttontdata, 1614  
 byte, 1616  
 ByteToStr, 1202

## C

CalcJointT, 1204  
 CalcRobT, 1209  
 CalcRotAxFrameZ, 1211  
 CalcRotAxisFrame, 1216  
 CallByVar, 53  
 cameradev, 1617  
 camerastatus, 1618  
 cameratarget, 1619  
 CamFlush, 55  
 CamGetExposure, 1220

CamGetLoadedJob, 1222  
 CamGetMode, 1224  
 CamGetName, 1225  
 CamGetParameter, 56  
 CamGetResult, 58  
 CamLoadJob, 60  
 CamNumberOfResults, 1226  
 CamReqImage, 62  
 CamSetExposure, 64  
 CamSetParameter, 66  
 CamSetProgramMode, 68  
 CamSetRunMode, 70  
 CamStartLoadJob, 72  
 CamStartSetParameter, 74  
 CamWaitLoadJob, 77  
 CamWaitSetParameter, 79  
 CancelLoad, 81  
 capaptrreferencedata, 1622  
 CapAPTrSetup, 83  
 CapAPTrSetupAI, 86  
 CapAPTrSetupAO, 89  
 CapAPTrSetupPERS, 92  
 CapC, 95  
 CapCondSetDO, 105  
 capdata, 1624  
 CapEquiDist, 107  
 CapGetFailSigs, 1228  
 CapL, 109  
 caplatrackdata, 1628  
 CapLATrSetup, 118  
 CapNoProcess, 123  
 CapRefresh, 125  
 CAPSetStopMode, 127  
 capspeeddata, 1632  
 capstopmode, 1634  
 captrackdata, 1635  
 capweavedata, 1638  
 CapWeaveSync, 128  
 CASE, 870  
 CDate, 1230  
 cfgdomain, 1646  
 CheckProgRef, 131  
 CirPathMode, 133  
 CJointT, 1231  
 Clear, 139  
 ClearIOBuff, 140  
 ClearPath, 142  
 ClearRawBytes, 146  
 ClkRead, 1233  
 ClkReset, 148  
 ClkStart, 149  
 ClkStop, 151  
 clock, 1647  
 Close, 152  
 CloseDir, 153  
 comment, 154  
 CompactIF, 155  
 confdata, 1648  
 ConfJ, 156  
 ConfL, 158  
 CONNECT, 161  
 ContactL, 163  
 CopyFile, 169  
 CopyRawBytes, 171  
 CornerPathWarning, 174  
 CorrClear, 176

CorrCon, 177  
corrdescr, 1655  
CorrDiscon, 182  
CorrRead, 1235  
CorrWrite, 183  
Cos, 1236  
CosDnum, 1237  
CPos, 1238  
CRobT, 1240  
CrossProd, 1243  
CSpeedOverride, 1246  
CTime, 1248  
CTool, 1249  
CWObj, 1251

## D

datapos, 1657  
DeactEventBuffer, 185  
DeactUnit, 187  
Decr, 189  
DecToHex, 1253  
DefAccFrame, 1254  
DefDFrame, 1257  
DefFrame, 1260  
Dim, 1263  
DInput, 1265  
dionum, 1658  
dir, 1659  
Distance, 1267  
DIV, 1269  
dnum, 1660  
DnumToNum, 1270  
DnumToStr, 1272  
DotProd, 1274  
DOutput, 1276  
DropSensor, 191  
DropWObj, 193

## E

ELSE, 260  
ELSEIF, 260  
ENDIF, 260  
EOF\_NUM, 1442  
EOffsOff, 194  
EOffsOn, 195  
EOffsSet, 197  
EraseModule, 199  
errdomain, 1662  
ErrLog, 201  
errnum, 1664  
ErrRaise, 205  
errstr, 1672  
errtype, 1673  
ErrWrite, 210  
etiqueta, 346  
EulerZYX, 1278  
event\_type, 1675  
EventType, 1280  
exec\_level, 1676  
ExecHandler, 1282  
ExecLevel, 1283  
EXIT, 212  
ExitCycle, 213  
Exp, 1284  
extjoint, 1677

## F

FileSize, 1285  
FileTimeDnum, 1288  
FitCircle, 215  
flypointdata, 1679  
FOR, 219  
FricIdEvaluate, 223  
FricIdInit, 222  
FricIdSetFricLevels, 226  
FSSize, 1291

## G

gestor de ERROR, 1837  
GetAxisDistance, 1294  
GetAxisMoveTime, 1296  
GetDataVal, 228  
GetGroupSignalInfo, 231  
GetJointData, 233  
GetMaxNumberOfCyclicBool, 1298  
GetMecUnitName, 1299  
GetModalPayloadMode, 1300  
GetMotorTorque, 1301  
GetNextCyclicBool, 1304  
GetNextMechUnit, 1306  
GetNextSym, 1309  
GetNumberOfCyclicBool, 1311  
GetServiceInfo, 1312  
GetSignalOrigin, 1314  
GetSysData, 235  
GetSysInfo, 1316  
GetTaskName, 1319  
GetTime, 1321  
GetTrapData, 240  
GetTSPStatus, 1323  
GetUASUserName, 1325  
GInput, 1327  
GInputDnum, 1329  
GOTO, 242  
GOutput, 1332  
GOutputDnum, 1334  
GripLoad, 244

## H

handler\_type, 1682  
HexToDec, 1337  
HollowWristReset, 246

## I

ICap, 248  
icondata, 1683  
IDelete, 254  
identno, 1685  
IDisable, 255  
IEnable, 256  
IError, 257  
IF, 260  
Incr, 262  
IndAMove, 264  
IndCMove, 268  
IndDMove, 272  
IndInpos, 1338  
IndReset, 276  
IndRMove, 281  
IndSpeed, 1340  
InitSuperv, 286  
instruction name, 238, 614  
intnum, 1687

InvertDO, 287  
 IOBusStart, 289  
 IOBusState, 290  
 iodev, 1689  
 IODisable, 293  
 IOEnable, 296  
 IOEventMessage, 299  
 iounit\_state, 1690  
 IOUnitState, 1342  
 IPathPos, 301  
 IPers, 303  
 IRMQMessage, 305  
 IsBrakeCheckActive, 1345  
 IsCollFree, 1346  
 IsCyclicBool, 1348  
 IsFile, 1351  
 ISignalAI, 309  
 ISignalAO, 319  
 ISignalDI, 323  
 ISignalDO, 326  
 ISignalGI, 329  
 ISignalGO, 332  
 IsLeadThrough, 1355  
 ISleep, 335  
 IsMechUnitActive, 1357  
 IsPers, 1358  
 IsStopMoveAct, 1360  
 IsStopStateEvent, 1362  
 IsSyncMoveOn, 1364  
 IsSysId, 1366  
 IsVar, 1367  
 ITimer, 337  
 IVarValue, 340  
 IWatch, 344

**J**

jointtarget, 1691

**L**

leer bloque, 580  
 listitem, 1693  
 Load, 347  
 loaddata, 1694  
 LoadId, 352  
 loadidnum, 1700  
 loadsession, 1702

**M**

MakeDir, 358  
 ManLoadIdProc, 360  
 MatrixSolve, 364  
 MatrixSolveQR, 367  
 MatrixSVD, 369  
 Max, 1369  
 MaxExtLinearSpeed, 1370  
 MaxExtReorientSpeed, 1371  
 MaxRobReorientSpeed, 1372  
 MaxRobSpeed, 1373  
 MechUnitLoad, 372  
 mecunit, 1703  
 Min, 1374  
 MirPos, 1375  
 MOD, 1377  
 ModExist, 1378  
 ModTimeDnum, 1379  
 MotionPlannerNo, 1381  
 MotionProcessModeSet, 377

MotionSup, 379  
 motsetdata, 1705  
 MoveAbsJ, 382  
 MoveC, 389  
 MoveCAO, 397  
 MoveCDO, 402  
 MoveCGO, 407  
 MoveCSync, 412  
 MoveExtJ, 417  
 MoveJ, 421  
 MoveJAO, 427  
 MoveJDO, 432  
 MoveJGO, 437  
 MoveJSync, 442  
 MoveL, 447  
 MoveLAO, 453  
 MoveLDO, 458  
 MoveLGO, 463  
 MoveLSync, 468  
 MovePnP, 473  
 MToolRotCalib, 483  
 MToolTCPCalib, 486

**N**

NonMotionMode, 1383  
 NOrient, 1386  
 NOT, 1385  
 num, 1712  
 NumToDnum, 1388  
 NumToStr, 1389

**O**

Offs, 1391  
 opcalc, 1714  
 Open, 489  
 OpenDir, 494  
 OpMode, 1393  
 opnum, 1715  
 OR, 1394  
 orient, 1716  
 OrientZYX, 1396  
 ORobT, 1398

**P**

PackDNHeader, 496  
 PackRawBytes, 499  
 paridnum, 1721  
 ParIdPosValid, 1400  
 ParIdRobValid, 1403  
 paridvalidnum, 1723  
 PathAccLim, 504  
 PathLengthGet, 1406  
 PathLengthReset, 508  
 PathLengthStart, 510  
 PathLengthStop, 512  
 PathLevel, 1408  
 pathrecid, 1725  
 PathRecMoveBwd, 514  
 PathRecMoveFwd, 521  
 PathRecStart, 524  
 PathRecStop, 527  
 PathRecValidBwd, 1410  
 PathRecValidFwd, 1413  
 PathResol, 530  
 PDispOff, 532  
 PDispOn, 533  
 PDispSet, 538

PFRestart, 1417  
pnpdata, 1727  
pos, 1729  
pose, 1731  
PoseInv, 1418  
PoseMult, 1420  
PoseVect, 1422  
Pow, 1424  
PowDnum, 1425  
PPMovedInManMode, 1426  
Present, 1427  
ProcCall, 541  
ProcerrRecovery, 543  
processtimes, 1732  
progdisp, 1733  
ProgMemFree, 1429  
PrxActivAndStoreRecord, 549  
PrxActivRecord, 551  
PrxDbgStoreRecord, 553  
PrxDeactRecord, 554  
PrxGetMaxRecordpos, 1430  
PrxResetPos, 555  
PrxResetRecords, 556  
PrxSetPosOffset, 557  
PrxSetRecordSampleTime, 558  
PrxSetSynccalarm, 559  
PrxStartRecord, 561  
PrxStopRecord, 563  
PrxStoreRecord, 564  
PrxUseFileRecord, 566  
PulseDO, 568

## R

RAISE, 571  
RaiseToUser, 574  
Rand, 1431  
rawbytes, 1736  
RawBytesLen, 1432  
ReadAnyBin, 577  
ReadBin, 1434  
ReadBlock, 580  
ReadCfgData, 582  
ReadDir, 1437  
ReadErrData, 586  
ReadMotor, 1440  
ReadNum, 1442  
ReadRawBytes, 589  
ReadStr, 1445  
ReadStrBin, 1449  
ReadVar, 1451  
ReadVarArr, 592  
RelTool, 1453  
RemainingRetries, 1455  
RemoveAllCyclicBool, 594  
RemoveCyclicBool, 596  
RemoveDir, 598  
RemoveFile, 600  
RemoveSuperv, 602  
RenameFile, 604  
Reset, 606  
ResetAxisDistance, 608  
ResetAxisMoveTime, 610  
ResetPPMoved, 612  
ResetRetryCount, 613  
restartblkdata, 1738  
restartdata, 1740  
RestoPath, 615

RETRY, 617  
RETURN, 619  
Rewind, 621  
RMQEmptyQueue, 622  
RMQFindSlot, 624  
RMQGetMessage, 626  
RMQGetMsgData, 629  
RMQGetMsgHeader, 632  
RMQGetSlotName, 1456  
rmqheader, 1744  
rmqmessage, 1746  
RMQReadWait, 635  
RMQSendMessage, 638  
RMQSendWait, 642  
rmqslot, 1748  
robjoint, 1749  
RobName, 1458  
RobOS, 1460  
robtarg, 1750  
Round, 1461  
RoundDnum, 1463  
RunMode, 1465  
rutinas de interrupción de E/S, 1843  
rutinas de servicio, 1840  
Rutinas TRAP, 1846

## S

SafetyControllerGetChecksum, 1467  
SafetyControllerGetOpModePinCode, 1468  
SafetyControllerGetSWVersion, 1469  
SafetyControllerGetUserChecksum, 1470  
SafetyControllerSyncRequest, 647  
Save, 648  
SaveCfgData, 651  
SCWrite, 653  
SearchC, 656  
SearchExtJ, 667  
SearchL, 675  
SenDevice, 687  
sensor, 1754  
sensorstate, 1756  
sensorvardata, 1757  
Set, 689  
SetAllDataVal, 691  
SetAO, 694  
SetDataSearch, 696  
SetDataVal, 701  
SetDO, 704  
SetGO, 707  
SetLeadThrough, 711  
SetSysData, 714  
SetupCyclicBool, 716  
SetupSuperv, 719  
shapedata, 1759  
SiClose, 725  
SiConnect, 722  
SiGetCyclic, 727  
signalorigin, 1761  
signalxx, 1763  
SimCollision, 729  
Sin, 1471  
SinDnum, 1472  
SingArea, 730  
SiSetCyclic, 733  
SkipWarn, 735  
SocketAccept, 736



- SocketBind, 739
  - SocketClose, 742
  - SocketConnect, 744
  - SocketCreate, 747
  - socketdev, 1765
  - SocketGetStatus, 1473
  - SocketListen, 749
  - SocketPeek, 1476
  - SocketReceive, 751
  - SocketReceiveFrom, 756
  - SocketSend, 761
  - SocketSendTo, 765
  - socketstatus, 1766
  - SoftAct, 770
  - SoftDeact, 772
  - SoftElbow, 773
  - speeddata, 1767
  - SpeedLimAxis, 775
  - SpeedLimCheckPoint, 779
  - SpeedRefresh, 785
  - SpyStart, 788
  - SpyStop, 791
  - Sqrt, 1478
  - SqrtDnum, 1479
  - StartLoad, 792
  - StartMove, 796
  - StartMoveRetry, 799
  - STCalcForce, 1480
  - STCalcTorque, 1482
  - STCalib, 802
  - STClose, 807
  - StepBwdPath, 811
  - STIndGun, 24, 813
  - STIndGunReset, 24, 815
  - STIsCalib, 1484
  - STIsClosed, 1486
  - STIsIndGun, 1488
  - STIsOpen, 1489
  - SToolRotCalib, 816
  - SToolTCPCalib, 819
  - Stop, 822
  - STOpen, 825
  - StopMove, 828
  - StopMoveReset, 832
  - stoppointdata, 1771
  - StorePath, 835
  - StrDigCalc, 1491
  - StrDigCmp, 1494
  - StrFind, 1496
  - StrFormat, 838
  - string, 1778
  - stringdig, 1780
  - StrLen, 1498
  - StrMap, 1499
  - StrMatch, 1501
  - StrMemb, 1503
  - StrOrder, 1505
  - StrPart, 1507
  - StrToByte, 1509
  - StrToVal, 1511
  - STTune, 840
  - STTuneReset, 844
  - supervtimeouts, 1781
  - SupSyncSensorOff, 845
  - SupSyncSensorOn, 846
  - switch, 1783
  - symnum, 1784
  - syncident, 1786
  - SyncMoveOff, 848
  - SyncMoveOn, 854
  - SyncMoveResume, 860
  - SyncMoveSuspend, 862
  - SyncMoveUndo, 864
  - SyncToSensor, 866
  - system data, 1787
  - SystemStopAction, 868
- T**
- Tan, 1513
  - TanDnum, 1514
  - taskid, 1790
  - TasksActive, 1519
  - TasksExecuting, 1521
  - TaskRunMec, 1515
  - TaskRunRob, 1516
  - tasks, 1791
  - TasksInSync, 1517
  - TEST, 870
  - TestAndSet, 1523
  - TestDI, 1526
  - testsignal, 1793
  - TestSignDefine, 872
  - TestSignRead, 1528
  - TestSignReset, 874
  - TextGet, 1530
  - TextTabFreeToUse, 1533
  - TextTabGet, 1535
  - TextTabInstall, 875
  - tooldata, 1795
  - TPERase, 877
  - tpnum, 1802
  - TPReadDnum, 878
  - TPReadFK, 882
  - TPReadNum, 887
  - TPShow, 891
  - TPWrite, 892
  - trapdata, 1803
  - TriggC, 895
  - TriggCheckIO, 904
  - triggdata, 1805
  - TriggDataCopy, 910
  - TriggDataReset, 912
  - TriggDataValid, 1537
  - TriggEquip, 914
  - TriggInt, 921
  - TriggIO, 926
  - triggios, 1806
  - triggiosdnum, 1809
  - TriggJ, 932
  - TriggJIOs, 949
  - TriggL, 940
  - TriggLIOs, 957
  - triggmode, 1811
  - TriggRampAO, 965
  - TriggSpeed, 973
  - TriggStopProc, 984
  - triggstrgo, 1814
  - Trunc, 1539
  - TruncDnum, 1541
  - TryInt, 990
  - TRYNEXT, 992
  - tsp\_status, 1817
  - TuneReset, 993

TuneServo, 994  
tunetype, 1819  
Type, 1543

## U

UIAlphaEntry, 1545  
UIClientExist, 1552  
UIDnumEntry, 1554  
UIDnumTune, 1561  
UIListView, 1568  
UIMessageBox, 1577  
UIMsgBox, 1002  
UIMsgWrite, 1011  
UIMsgWriteAbort, 1016  
UINumEntry, 1585  
UINumTune, 1592  
UIShow, 1017  
uishownum, 1820  
UnLoad, 1021  
UnpackRawBytes, 1024

## V

ValidIO, 1599  
ValToStr, 1601  
VectMagn, 1603  
VelSet, 1029

## W

WaitAI, 1032  
WaitAO, 1037  
WaitDI, 1042  
WaitDO, 1047  
WaitGI, 1052  
WaitGO, 1058  
WaitLoad, 1064  
WaitRob, 1069  
WaitSensor, 1071

WaitSyncTask, 1075  
WaitTestAndSet, 1079  
WaitTime, 1082  
WaitUntil, 1084  
WaitWObj, 1090  
WarmStart, 1093  
weavestartdata, 1821  
WHILE, 1094  
wobjdata, 1823  
WorldAccLim, 1096  
Write, 1098  
WriteAnyBin, 1101  
WriteBin, 1103  
WriteBlock, 1105  
WriteCfgData, 1107  
WriteRawBytes, 1111  
WriteStrBin, 1114  
WriteVar, 1116  
WriteVarArr, 1119  
WZBoxDef, 1121  
WZCylDef, 1123  
WZDisable, 1126  
WZDOSet, 1128  
WZEnable, 1133  
WZFree, 1135  
WZHomeJointDef, 1137  
WZLimJointDef, 1141  
WZLimSup, 1145  
WZSphDef, 1148  
wzstationary, 1827  
wztemporary, 1829

## X

XOR, 1605

## Z

zonedata, 1831





**ABB AB**

**Robotics & Discrete Automation**

S-721 68 VÄSTERÅS, Sweden

Telephone +46 (0) 21 344 400

**ABB AS**

**Robotics & Discrete Automation**

Nordlysvegen 7, N-4340 BRYNE, Norway

Box 265, N-4349 BRYNE, Norway

Telephone: +47 22 87 2000

**ABB Engineering (Shanghai) Ltd.**

Robotics & Discrete Automation

No. 4528 Kangxin Highway

PuDong District

SHANGHAI 201319, China

Telephone: +86 21 6105 6666

**ABB Inc.**

**Robotics & Discrete Automation**

1250 Brown Road

Auburn Hills, MI 48326

USA

Telephone: +1 248 391 9000

**[abb.com/robotics](http://abb.com/robotics)**